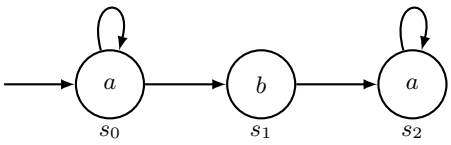
# VU Programm- und Systemverifikation
## Homework: Temporal Logic and Automated Reasoning

**Due date:** June 4, 2020, 1pm

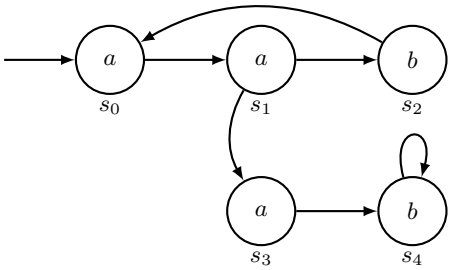
**Task 1 (5 points):** Consider the following Kripke Structure:




For each formula, give the states of the Kripke structure for which the formula holds. In other words, for each of the states from the set $\{s_0, s_1, s_2\}$, consider the computation trees starting at that state, and for each tree, check whether the given formula holds on it or not.

1. $\mathbf{A}\,\mathbf{X}\,\mathbf{X}\,a$ $\quad \{s_1, s_2\}$
2. $\mathbf{A}\,(\mathbf{F}\,\mathbf{G}\,a)$ $\quad \{s_0, s_1, s_2\}$
3. $\mathbf{A}\,(\mathbf{G}\,\mathbf{F}\,a)$ $\quad \{s_0, s_1, s_2\}$
4. $\mathbf{A}\,(\mathbf{G}\,\mathbf{F}\,b)$ $\quad \emptyset$
5. $\mathbf{A}\,\mathbf{F}\,\mathbf{A}\,\mathbf{G}\,a$ $\quad \{s_1, s_2\}$
6. $\mathbf{A}\,\mathbf{F}\,\mathbf{E}\mathbf{G}\,a$ $\quad \{s_0, s_1, s_2\}$
7. $\mathbf{A}\,(a \wedge \mathbf{X}\,a)$ $\quad \{s_2\}$
8. $\mathbf{E}(b\,\mathbf{U}\,a)$ $\quad \{s_0, s_1, s_2\}$
9. $\mathbf{A}\,(a\,\mathbf{U}\,b)$ $\quad \{s1\}$
10. $\mathbf{E}(a\,\mathbf{U}\,b)$ $\quad \{s_0, s_1\}$

**Task 2 (3 points):** Consider the following Kripke Structure with initial state $s_0$:




1. Does the LTL formula $\mathbf{A}\,\mathbf{F}\,\mathbf{X}\,b$ hold in the initial state $s_0$?

   ✓ yes  □ no

   Justify your answer!

   LTL reasons about entire paths. $\mathbf{X}\,b$ hold in state $s_1$ along the path $s_0, s_1, s_2, \ldots$ and in $s_3$ and $s_4$ along the path $s_0, s_1, s_3, s_4, \ldots$.

2. Does the CTL formula $\mathbf{A}\,\mathbf{F}\,\mathbf{A}\,\mathbf{X}\,b$ hold in the initial state $s_0$?

   □ yes  ✓ no

   Justify your answer!

   This CTL formula requires that we eventually reach a state $s$ such that on *all* successor states of that state $b$ holds. It can easily be verified that this doesn't hold for any of the states we necessarily have to visit (in particular, it doesn't hold for $s_1$, since $s_3$ isn't labelled $b$), and while $\mathbf{A}\,\mathbf{X}\,b$ holds for $s_3$, we can avoid visiting $s_3$ by always choosing $s_2$ as successor state of $s_1$.

3. Do the formulas (i) and (ii) above express the same property?

   □ yes  ✓ no

   If not, explain why.

   If they did, both should either hold or not hold for the Kripke structure above; but $\mathbf{A}\,\mathbf{F}\,\mathbf{X}\,b$ holds and $\mathbf{A}\,\mathbf{F}\,\mathbf{A}\,\mathbf{X}\,b$ doesn't, so they can be equivalent.

**Task 3 (2 points):** Encode the following statements in LTL:

(Note: Different solutions are possible. Also, the $\mathbf{A}$ at the beginning is redundant.)

1. The sequence *abab* occurs at some point in all traces.

   $\mathbf{A}\,\mathbf{F}\,(a \wedge (\mathbf{X}\,b) \wedge (\mathbf{X}\,\mathbf{X}\,a) \wedge (\mathbf{X}\,\mathbf{X}\,\mathbf{X}\,b)$

2. As long as a pot is *watched* it never *boils*.

   $\mathbf{A}\,\mathbf{G}\,(watched \Rightarrow \neg boils)$

   Other possible solutions:

   - $\mathbf{A}\,(\neg boils\,\mathbf{U}\,\neg watched)$ (if we assume the pot will boil eventually)
   - $\mathbf{A}\,(\neg boils\,\mathbf{U}\,\neg watched) \vee (\mathbf{G}\,\neg boils)$

3. Once you see $c$, you can't "unsee $c$" (i.e., from then on, it holds indefinitely).

   $\mathbf{A}\,\mathbf{G}\,(c \Rightarrow \mathbf{G}\,c)$

4. On each path, either you encounter (at least) two $b$s, or (at least) two $(\neg b)$s.

   $\mathbf{A}\,(((\mathbf{F}\,(b \wedge \mathbf{X}\,\mathbf{F}\,b)) \vee (\mathbf{F}\,(\neg b \wedge \mathbf{X}\,\mathbf{F}\,\neg b)))$

**Task 4 (5 points):** Let $F$ be a formula in *conjunctive normal form* (CNF), i.e., a conjunction of clauses

$$\bigwedge_i \bigvee_j \ell_{i,j}, \quad \text{where } \ell_{i,j} \in \{P, \neg P \mid P \text{ is an atom}\}$$

Note that every clause $C_i$ with $m$ literals can be seen as a "nested" disjunction, i.e.,

$$C_i = (\ell_{i,1} \vee (\ell_{i,2} \vee (\ldots \vee (\ell_{i,m-1} \vee \ell_{i,m}))))$$

Explain how you can use Tseitin's transformation to convert any arbitrary formula in CNF into an equi-satisfiable formula in CNF whose clauses have at most 3 literals.

Iteratively reduce the size $m$ of each clause $C_i$ to 3. Start with $k = m$; as long as $k > 3$:

1. We have $C_i = (\ell_{i,1} \vee (\ell_{i,2} \vee (\ldots \vee (\ell_{i,k-1} \vee \ell_{i,k}))))$
2. Introduce new literal $o_{i,k}$
3. Convert $o_{i,k} \Leftrightarrow (\ell_{i,k-1}, \vee \ell_{i,k})$ into CNF and add it to the formula
4. Set $C_i = (\ell_{i,1} \vee (\ell_{i,2} \vee (\ldots \vee o_{i,k}))$

Note that all the clauses generated during the conversion in step 3 contain at most 3 literals (see slides), and by construction, the formula is equi-satisfiable.

**Task 5 (5 points):**

1. Consider the following formulas in propositional logic; are they satisfiable? If yes, provide a satisfying assignment over booleans, if not, give the reasoning that leads to this conclusion.

$$(\neg a \vee \neg b) \ \wedge \ c \ \wedge \ (b \vee d) \ \wedge \ \neg d \ \wedge \ e \ \wedge \ (\neg e \vee \neg c \vee a) \tag{1}$$

   Unsatisfiable (by unit propagation):

   - $c \wedge e \wedge (\neg e \vee \neg c \vee a) \Rightarrow a$
   - $a \wedge (\neg a \vee \neg b) \Rightarrow \neg b$
   - $\neg b \wedge (b \vee d) \Rightarrow d$, contradiction.

$$f \ \wedge \ (\neg g \vee f) \ \wedge \ (h \vee \neg f) \ \wedge \ (g \vee h) \ \wedge \ (\neg g \vee \neg h) \tag{2}$$

   Satisfiable. ($g = \mathsf{false}, h = \mathsf{true}, f = \mathsf{true}$)

2. Consider the following formulas in Equality Logic; are they satisfiable? If yes, provide a satisfying assignment over integers, if not, give the reasoning based on equivalence classes that leads to this conclusion.

$$i = j \ \wedge \ k = \ell \ \wedge \ k \neq m \ \wedge \ \ell \neq i \ \wedge \ m = i \ \wedge \ f = j \tag{3}$$

   Satisfiable. $f = i = j = m = 0, k = \ell = 1$

$$n \neq o \wedge p = q \wedge r = s \wedge r = t \wedge t = n \wedge s = q \wedge q = g \wedge g = o \wedge r = g \wedge t = q \tag{4}$$

   Unsatisfiable. The cycle $n = t = q = g = o$ contradicts $n \neq o$.

3. Check the satisfiability of the following SMT formulas. Assume that $u, v, w, x, y, z \in \mathbb{Z}$ are integer constants, and $f : \mathbb{Z} \to \mathbb{Z}$ and $g : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ are unary and binary uninterpreted functions over integers respectively. Whenever a formula is satisfiable, give a satisfying assignment for it, i.e., integer values for all constants and function interpretations over integers that make the formula true under the assignment. Whenever a formula is not satisfiable, give a reason why it is unsatisfiable.

$$g(3, y) = 5 \ \wedge \ g(y, 3) = 5 \ \wedge \ g(y, x) \neq g(x, y) \tag{5}$$

Satisfiable. Note that we need to provide an interpretation (a model, respectively) for the constants as well as for the functions. For the functions, a (partial) map from parameter values (domain) to return values (range) is sufficient.

$$y = 3, x = 0, g(3, 3) = 5, g(3, 0) = 3, g(0, 3) = 0$$

$$\begin{aligned} g(1, x) = 2 \ \wedge \ g(1, x) = g(x, 1) \ \wedge \ f(f(x)) = g(1, x) \\ \wedge \ f(f(f(1))) = 1 \ \wedge \ f(1) \neq g(x, 1) \ \wedge \ 1 = f(x) \end{aligned} \tag{6}$$

Unsatisfiable.

- $g(x, 1) = g(1, x) = 2$, therefore $f(f(x)) = 2$
- $f(x) = 1$, therefore $f(f(x)) = f(1) = 2$, this contradicts $f(1) \neq g(x, 1)$.

$$\begin{aligned} g(z, z) = z \ \wedge \ g(u, u) = u \\ \wedge \ (g(z, z) = 0 \vee g(z, z) = 1) \ \wedge \ (g(u, u) = 0 \vee g(u, u) = 1) \\ \wedge \ g(z, u) = g(u, z) \ \wedge \ g(3, 2) = w \ \wedge \ g(2, 3) = v \ \wedge \ v \neq w \end{aligned} \tag{7}$$

Satisfiable:

$$z = u = 0, v = 5, w = 4, g(0, 0) = 0, g(3, 2) = 4, g(2, 3) = 5$$

For those interested in playing with Z3, the SMTLib encoding for Formula 7 is provided on the next page. (You're allowed to use Z3 during the exam.)

Formula 7:

$$g(1,x) = 2 \ \wedge \ g(1,x) = g(x,1) \ \wedge \ f(f(x)) = g(1,x) \\ \wedge \ f(f(f(1))) = 1 \ \wedge \ f(1) \neq g(x,1) \ \wedge \ 1 = f(x)$$

SMTLib encoding:

```
(declare-fun g (Int Int) Int)
(declare-fun z () Int)
(declare-fun u () Int)
(declare-fun w () Int)
(declare-fun v () Int)
(assert (= z (g z z)))
(assert (= u (g u u)))
(assert (or (= 0 (g z z)) (= 1 (g z z))))
(assert (or (= 0 (g u u)) (= 1 (g u u))))
(assert (or (= 0 (g z u)) (= 1 (g u z))))
(assert (= w (g 3 2)))
(assert (= v (g 2 3)))
(assert (not (= w v)))
(check-sat)
(get-model)
```