CDM Quiz

Name:_____ Student ID:_____

1 SAT or UNSAT, that is the question. (15')

1. Use DPLL algorithm to solve the following problem. You need to write detailed process. When using decide rule, please decide 1 is true at the first time. (10')

$$\emptyset \parallel 1 \vee 4,\ \overline{1} \vee 2,\ \overline{2} \vee 3,\ \overline{3} \vee \overline{4},\ \overline{1} \vee \overline{2} \vee 4$$

Solution.

$$\begin{aligned}
\emptyset &\parallel 1 \vee 4,\ \overline{1} \vee 2,\ \overline{2} \vee 3,\ \overline{3} \vee \overline{4},\ \ \overline{1} \vee\ \overline{2} \vee 4 \Rightarrow (Decide)\\
1^d &\parallel 1 \vee 4,\ \overline{1} \vee 2,\ \overline{2} \vee 3,\ \overline{3} \vee \overline{4},\ \ \overline{1} \vee\ \overline{2} \vee 4 \Rightarrow (UnitProp)\\
1^d 2 &\parallel 1 \vee 4,\ \overline{1} \vee 2,\ \overline{2} \vee 3,\ \overline{3} \vee \overline{4},\ \ \overline{1} \vee\ \overline{2} \vee 4 \Rightarrow (UnitProp)\\
1^d 23 &\parallel 1 \vee 4,\ \overline{1} \vee 2,\ \overline{2} \vee 3,\ \overline{3} \vee \overline{4},\ \ \overline{1} \vee\ \overline{2} \vee 4 \Rightarrow (UnitProp)\\
1^d 234 &\parallel 1 \vee 4,\ \overline{1} \vee 2,\ \overline{2} \vee 3,\ \overline{3} \vee \overline{4},\ \ \overline{1} \vee\ \overline{2} \vee 4 \Rightarrow (BackTrack)\\
\overline{1} &\parallel 1 \vee 4,\ \overline{1} \vee 2,\ \overline{2} \vee 3,\ \overline{3} \vee \overline{4},\ \ \overline{1} \vee\ \overline{2} \vee 4 \Rightarrow (UnitProp)\\
\overline{1}4 &\parallel 1 \vee 4,\ \overline{1} \vee 2,\ \overline{2} \vee 3,\ \overline{3} \vee \overline{4},\ \ \overline{1} \vee\ \overline{2} \vee 4 \Rightarrow (UnitProp)\\
\overline{1}4\overline{3} &\parallel 1 \vee 4,\ \overline{1} \vee 2,\ \overline{2} \vee 3,\ \overline{3} \vee \overline{4},\ \ \overline{1} \vee\ \overline{2} \vee 4 \Rightarrow (UnitProp)\\
\overline{1}4\overline{32} &\parallel 1 \vee 4,\ \overline{1} \vee 2,\ \overline{2} \vee 3,\ \overline{3} \vee \overline{4},\ \ \overline{1} \vee\ \overline{2} \vee 4
\end{aligned}$$

2. Convert the following program to a propositional logic formula, which is unsatisfiable iff. the assertion never fails. (5')

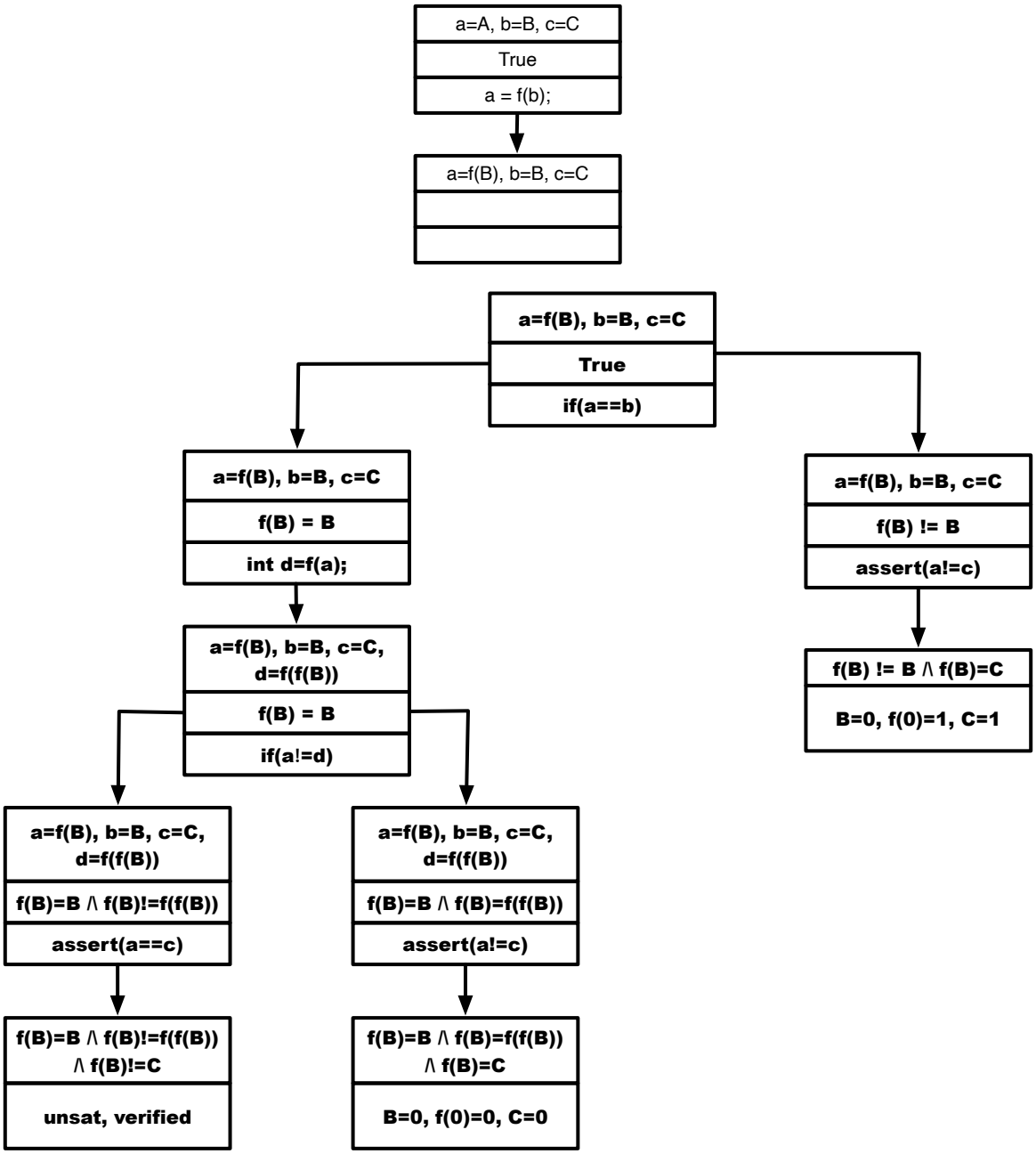
```
1 void func(bool a, bool b, bool c) {
2   if(a)
3     b = b & c;
4   assert(b == c);
5 }
```

Solution. $(A1 \leftrightarrow A0) \wedge (C1 \leftrightarrow C0) \wedge (B1 \leftrightarrow ((A0 \wedge (B0 \wedge C0)) \vee (\neg A0 \wedge B0))) \wedge \neg(B1 \leftrightarrow C1)$

2 Working with SMT solver is like magic. (35')

Consider the following program. We treat f to be an uninterpreted function.

```
1  void func(int a, int b, int c) {
2    a = f(b);
3    if(a == b) {
4      int d = f(a);
5      if(a != d) {
6        assert(a == c);
7        return;
8      }
9    }
10   assert(a != c);
11 }
```



1. The picture above is part of the symbolic execution tree. Please give the whole tree. You should write all formulas produced by assertions. If some formulas are satisfiable, please give possible counterexamples. (18')

   Solution. The tree is above.

2. Now we want to use Lazy SMT techniques to solve the formula produced by *assert(a==c)*. Please give every step of DPLL(T), including the input and output of SAT solver and theory solver. You do not need to give the internal process of EUF solver and DPLL. (8')

   Solution. First, transform the formula to propositional logic formula $p1 \wedge \neg p2 \wedge \neg p3$.

   Second, SAT solver solves the formula and returns sat (p1=T, p2=F, p3=F).

   Third, EUF solver tells $(f(B) = B) \wedge (f(B) \neq f(f(B))) \wedge (f(B) \neq C)$ is not satisfiable.

   Fourth, SAT solver solves $p1 \wedge \neg p2 \wedge \neg p3 \wedge \neg(p1 \wedge \neg p2 \wedge \neg p3)$ and returns unsat.

   So lazy SMT returns unsat finally.

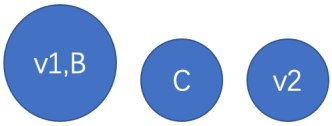
3. DPLL(T) invokes EUF solver in 2. Please draw circles to show the detailed process of EUF solver. You need to explain the reasons of merging circles and its final answers. (5')

Solution. EUF solver solves $(f(B) = B) \wedge (f(B) \neq f(f(B))) \wedge (f(B) \neq C)$.

Step1. Convert the formula to $(v1 = B) \wedge (v1 \neq v2) \wedge (v1 \neq C)$, v1=f(B), v2=f(v1).

Step2. Draw circles.

Step3. Because v1=B, we merge circles.



Step4. Because $(v1 = B) \rightarrow (f(B) = f(v1))$, we merge circles.

Step5. Because $v1 \neq v2$, but v1 and v2 are in the same circle. So the formula is unsatisfiable.

4. If the variables are bool type and we use an eager SMT solver, please convert the formula produced by *assert(a==c)* into a propositional logic formula. (4')

Solution.

$$
\begin{aligned}
&(f_B \leftrightarrow B) \wedge \\
&\neg(f_B \leftrightarrow f_{fB}) \wedge \\
&\neg(f_B \leftrightarrow C) \wedge \\
&((B \leftrightarrow f_B) \rightarrow (f_B \leftrightarrow f_{fB}))
\end{aligned}
$$

3 Hoare Logic: assign logic to program. (20')

1. Write the pre/post-condition of the following Hoare triples to make them valid. (2')

$$
\begin{aligned}
&\{\quad\}x := y + z; \{x > 4\} \\
&\{x = 4\} while(x > 0) x := x - 1; \{\quad\}
\end{aligned}
$$

Solution. $\{y + z = 8\}x := y + z; \{x > 4\}$, $\{x = 4\} while(x > 0) x := x - 1; \{x = 0\}$

2. Consider the following program, please determine if $a \geq 0$ is the loop invariant. If not, please describe the reasons. (5')

```
1 int func() {
2   int a := 9;
3   while(a > 0) a := a - 2;
4   return 0;
5 }
```

Solution. It is not the loop invariant. Because $\{(a \geq 0) \wedge (a > 0)\}a := a - 2\{a \geq 0\}$ does not hold.

3. Prove $\{b \geq 20\}a := b; while(a > 2)a := a - 1; \{a > 1\}$ via Hoare inference rules and axioms. The loop invariant is $a \geq 2$. Variables are integers. Please write the proof steps with names of rules like that in slides. (13')

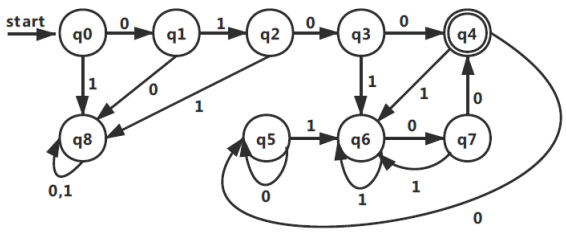
Solution.

$(1)\{a - 1 \geq 2\}a := a - 1; \{a \geq 2\}$	AS
$(2)(a \geq 2 \wedge a > 2) \rightarrow (a - 1 \geq 2)$	$predicate \quad logic$
$(3)\{a \geq 2 \wedge a > 2\}a := a - 1; \{a \geq 2\}$	$SP(1)(2)$
$(4)\{a \geq 2\}while(a > 2)a := a - 1; \{a \geq 2 \wedge a \leq 2\}$	$WHP(3)$
$(5)(a \geq 2 \wedge a \leq 2) \rightarrow (a > 1)$	$predicate \quad logic$
$(6)\{a \geq 2\}while(a > 2)a := a - 1; \{a > 1\}$	$WC(4)(5)$
$(7)\{b \geq 2\}a := b\{a \geq 2\}$	AS
$(8)b \geq 20 \rightarrow b \geq 2$	$predicate \quad logic$
$(9)\{b \geq 20\}a := b\{a \geq 2\}$	$SP(7)(8)$
$(10)\{b \geq 20\}a := b; while(a > 2)a := a - 1; \{a > 1\}$	$SC(6)(9)$

4 DFA (12')

We define $L_s = \{s | s \ starts \ with \ '010' \ and \ ends \ with \ '100'\}$ over the input alphabet $\{0, 1\}$. For example, $010100 \in L_s$, but $0100010 \notin L_s$.

1. Design a DFA A_s such that $L(A_s) = L_s$. Specify the transition diagram, and the intuitive purpose of each state. (8')



Solution. Purpose of each state:

(a) q0: Start state

(b) q1: Identify the prefix "0"

(c) q2: Identify the prefix "01"

(d) q3: Identify the prefix "010"

(e) q4: Accept state

(f) q5: Prefix satisfied and end with more than 3 '0's

(g) q6: Prefix satisfied and end with "1"

(h) q7: Prefix satisfied and end with "10"

(i) q8: Prefix unsatisfied and reject

2. Show how A_s accepts 0101100 by computing transition function like $\hat{\delta}(q_0, 0101100)$ (4')

Solution.

$$\begin{aligned}
\hat{\delta}(q_0, 0101100) &= \hat{\delta}(q_1, 101100) && (1)\\
&= \hat{\delta}(q_2, 01100) && (2)\\
&= \hat{\delta}(q_3, 1100) && (3)\\
&= \hat{\delta}(q_6, 100) && (4)\\
&= \hat{\delta}(q_6, 00) && (5)\\
&= \hat{\delta}(q_7, 0) && (6)\\
&= q_4 && (7)\\
& && (8)
\end{aligned}$$

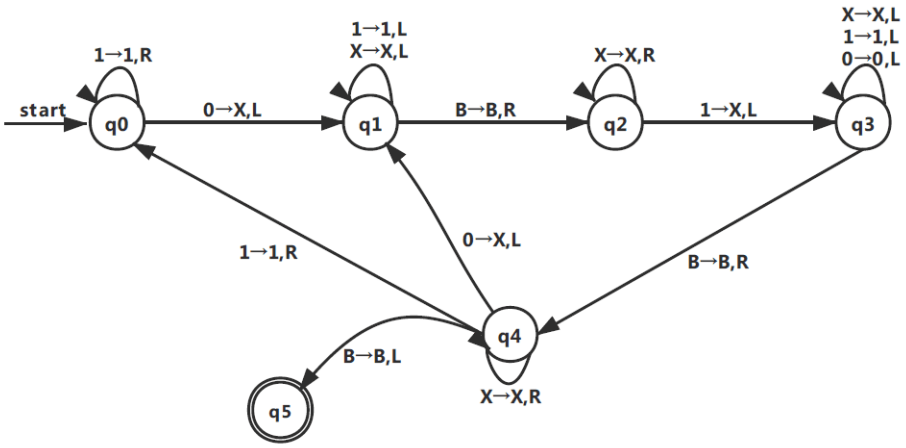
5 Turing Machine (18')

We define $L_{count} = \{s | s \text{ has same count of 0s and 1s}\}$ over the input alphabet $\{0, 1\}$. For example, $00101101 \in L_{count}$, but $01010 \notin L_{count}$.

1. Briefly explain why no DFA can recognize L_{count}. (4')

Solution. L_{count} cannot be defined by a regular expression.
To accept L_{count}, we need at least n state, but n is unbounded.
(Give points to whatever makes sense)

2. Design a Turing Machine M_{count} such that $L(M_{count}) = L_{count}$. You can only use the naive TM without using extra states, multitrack or multitape, etc. Specify the states, transitions, and the intuitive purpose of each state. (8')



Solution.

3. Is L_{count} a recursive language (R) or just recursive enumerate but not recursive language (RE but not R)? Explain why. (6')

Solution. Recursive language means Turing machine never loops. (2')
L_{count} is recursive language. (2')
The count of 1s and 0s is decreasing, and the TM will finally halt. (2')