You can review the latex source for this assignment-file to learn and use latex to prepare your homework submission. You will see the use of macros (to write uniformly formatted text), different text-styles (emphasized, bold-font), different environments (figures, enumerations).

It is not required that you use exactly this latex source to prepare your submission.

Homework 4 (CTL & BDD): ComS/CprE/SE 412, ComS 512

Due-date: Apr 2 at 11:59PM.

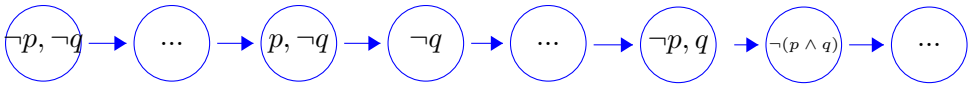
Submit online on Canvas two files: the source file in latex format and the pdf file generated from latex. Name your files: ⟨your-net-id⟩-hw4.⟨tex/pdf⟩.

Homework must be individual's original work. Collaborations and discussions of any form with any students or other faculty members or soliciting solutions on online forums are not allowed. Please review the academic dishonesty policy on our syllabus. If you have any questions/doubts/concerns, post your questions/doubts/concerns on Piazza and ask TA/Instructor.

1. Prove or disprove: the following properties can be expressed in CTL

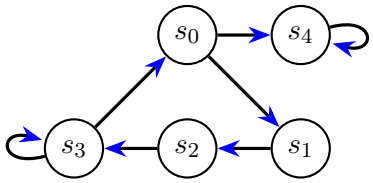
   (a) Whenever p holds in a state (say s), q holds within 4 steps from s.
   $\mathtt{AG}(p \Rightarrow (q \vee \mathtt{AX}(q \vee \mathtt{AX}(q \vee \mathtt{AX}(q \vee \mathtt{AX}(q))))))$

   (b) There exists a path where p is true eventually and q is true eventually but they are never true in the same state.
   A possible path where p occurs before q looks like: ("..." refers to same labels as previous)

   Similarly, also consider the case where q occurs before p
   $\mathtt{E}((\neg p \wedge \neg q)\ \mathtt{U}\ ((p \wedge \neg q) \wedge \mathtt{E}(\neg q\ \mathtt{U}\ ((\neg p \wedge q) \wedge \mathtt{EG}(\neg(p \wedge q)))))) \vee$
   $\mathtt{E}((\neg p \wedge \neg q)\ \mathtt{U}\ ((\neg p \wedge q) \wedge \mathtt{E}(\neg p\ \mathtt{U}\ ((p \wedge \neg q) \wedge \mathtt{EG}(\neg(p \wedge q))))))$

2. For the following Kripke structure with $b \in L(s_3)$, $\{a, b\} \subseteq L(s_2)$, $a \in L(s_1)$ and $b \in L(s_4)$,

   We are given a function defined as follows:

   $$A_\varphi(Z) = [\![\varphi]\!]_M \ \cap\ R_\forall(R_\forall(Z))$$

   where φ is some CTL formula, $[\![\varphi]\!]_M$ is the semantics of φ (in the context of a Kripke structure $M = (S, T, L)$) defined as a set of states of a Kripke structure, and $R_\forall$ is

   $$R_\forall(Z) = \{s \mid \forall s'.(s, s') \in T \ \Rightarrow\ s' \in Z\}$$

(a) Compute the greatest fixed point A_b for the given Kripke structure.

$$
\begin{aligned}
A_b(S) &= [\![b]\!]_M \cap R_\forall(R_\forall(S)) \\
&= \{s_3, s_2, s_4\} \cap S \\
&= \{s_3, s_2, s_4\} \\
A_b^2(S) &= A_b(\{s_3, s_2, s_4\}) \\
&= \{s_3, s_2, s_4\} \cap R_\forall(R_\forall(\{s_3, s_2, s_4\})) \\
&= \{s_3, s_2, s_4\} \cap R_\forall(\{s_2, s_1, s_4\}) \\
&= \{s_3, s_2, s_4\} \cap \{s_1, s_0, s_4\} \\
&= \{s_4\} \\
A_b^3(S) &= A_b(\{s_4\}) \\
&= \{s_3, s_2, s_4\} \cap R_\forall(R_\forall(\{s_4\}) \\
&= \{s_4\}
\end{aligned}
$$

gfp of $A_b = \{s_4\}$

(b) **512 only** It is claimed that the semantics of the property expressed by the greatest fixed point of A_b cannot be expressed in CTL. Justify the validity of the claim.
Let S' be the gfp of A_b. So, $A_b(S') = S'$

$$
\begin{aligned}
S' &= [\![b]\!]_M \cap R_\forall(R_\forall(S')) \\
&= \{s | s \in [\![b]\!]\} \cap \{s | \forall \pi \in Paths(s). \forall \pi[1] \in R_\forall(S')\} \\
&= \{s | s \in [\![b]\!]\} \cap \{s | \forall \pi \in Paths(s). \forall \pi' \in Paths(\pi[1]). \pi'[1] \in S'\} \\
&= \{s | s \in [\![b]\!]\} \cap \{s | \forall \pi \in Paths(s). \pi[2] \in S'\} \\
&= \{s | s \in [\![b]\!] \wedge \forall \pi \in Paths(s). \pi[2] \in S'\} \\
&= \{s | \forall \pi \in Paths(s). \forall i. i \; is \; even. \pi[i] \in [\![b]\!]\}
\end{aligned}
$$

Cannot express using CTL because the status of the property b on odd states is not known and it is not possible to specify the property on only even states. If the property on odd-states were known, that could be used to identify the even states via AX.

3. Draw the ROBDD for the combination of the propositional variables describing a safe set of states. The following conditions hold in the safe set of states:

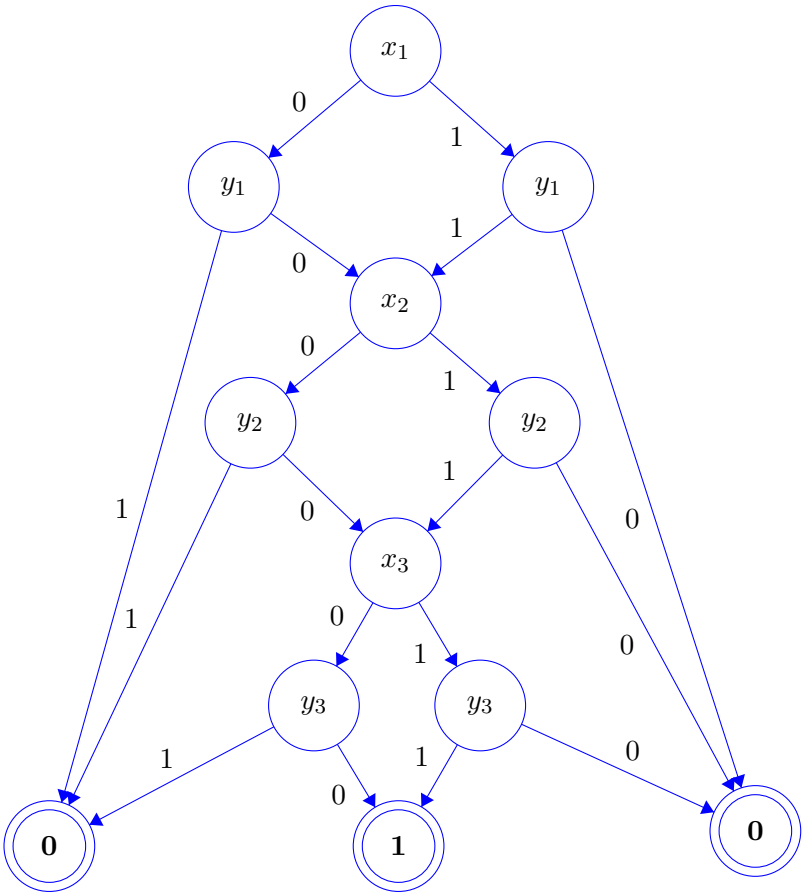
   (a) x_1 and y_1 are either both true or both false

   (b) x_2 and y_2 are either both true or both false

   (c) x_3 and y_3 are either both true or both false

   Your grade will depend on size of the ROBDD.

   The minimal size ROBDD has 9 nodes + 2 terminal nodes. This is obtained by using a variable order that is any permutation of variables within parentheses $((x_1, y_1), (x_2, y_2), (x_3, y_3))$. Variables that together define the final status should stay together in the ROBDD. Note that in the figure two **0** terminal nodes are drawn only for clarity purpose.



4. You company requires that in any collaborative project, code commits must be reviewed by collaborator(s) of the person who commits the code. One fine morning, you wake up to see your collaborator has committed some late night code in language pseudo-lang with minimal comments. Your job is to review the commit and provide some constructive comments.

   The only code-comments are as follows:

   - The code utilizes ROBDD implementation where each node of the ROBDD is described using a structure *node* containing three properties: *name* of the decision variable for that node; a pointer to the root of the left-subtree (*left*); and a pointer to the root of the right-subtree (*right*).

     The terminal or leaf nodes are associated with the *name* true or false (and have null pointers for *left* and *right*).
   - The code corresponds to some function with two formal parameters each of type node, and returns a boolean type.

```
function wit(struct node n1, struct node n2) {
   if ((n1.name == true) && (n2.name == false)) return true;
   if ((n2.name == true) && (n1.name == false)) return true;
   if ((n1.left == null) || (n2.left == null)) return false;
   if (n1.name == n2.name) {
```

```
      if (wit(n1.left, n2.left))
          return wit(n1.right, n2.right)
      }
   return false;
}
```

(a) What is the functionality of the function *wit*?

Returns true if $n1$ and $n2$ are complement of each other.

(b) What is the runtime of the function *wit*?

We know that number of nodes, $n = 2^k$ where k is number of variables.But when considering the time complexity, you need to fix what is the parameter.

1. If parameter is the number of nodes (n): Worst-case scenario: If the ROBDDs that are under consideration for apply() have shared nodes due to reduction (i.e nodes with multiple parents), apply operations involving those will be recomputed, leading to exponential blow up. Hence, $O(2^n)$.

2. If the parameter is the number of variables (k): Worst-case scenario: If the ROBDDs that are under consideration for apply() may or may not have shared nodes and are full diagrams, apply operations will be applied as many times as there are nodes(2^k). Hence, $O(2^k)$. Of course one may claim that this refers to $O(n)$ but then if the ROBDDs had shared nodes, $O(n)$ would be incorrect.