02246 Mandatory Assignment
L07 - SMT, introduction to Bounded Model Checking

To be submitted on DTU Learn - see deadline on DTU Learn

You are encouraged to work in groups, but you must clearly identify the contributions of each group member, and you will be jointly responsible for the finished report. Register your group on DTU Learn before submitting as group submission.

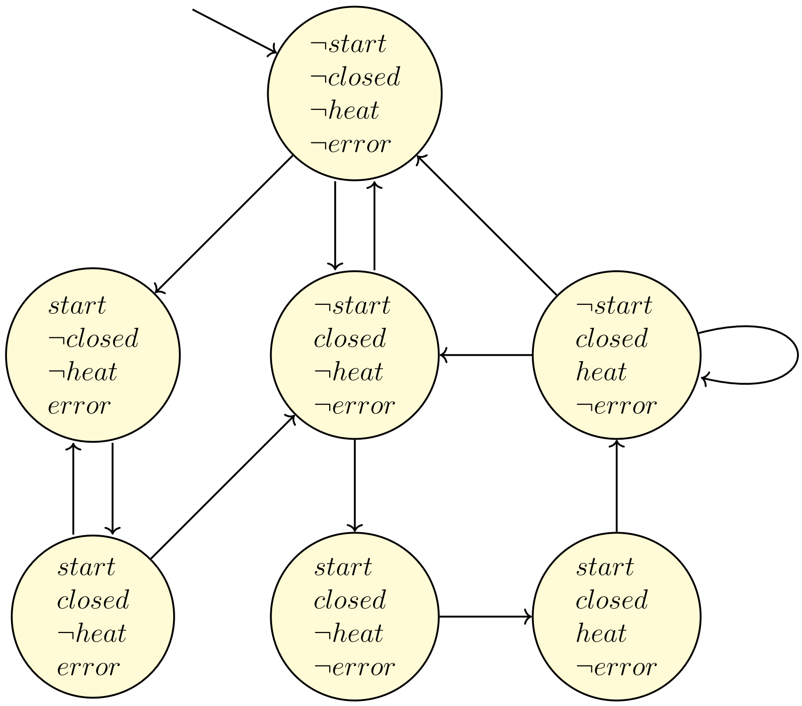
Answers to all parts should be typed up using LaTeX and submitted electronically as a PDF report using the provided template. Drawings and formulae may be handwritten and scanned. More detailed instructions as to the style of answer we expect for each part are included below.

Group contribution: We have worked together on every aspect of this assignment. Problems have been solved and discussed together. We have both had a part in the formulation and answer of every problem, and it is therefore difficult to put names on any specific section.

L07 - SMT, introduction to Bounded Model Checking

L07P: Practical Problems

L07P.1



Given the above transition system T representing the microwave oven controller, check whether $T \models_k \mathbf{E}(\neg heat\, \mathbf{U}\, close)$ for $k = 2$.[1] To do so, perform the following steps:

a) Encode the above problem as propositional formula $[\![T, \neg heat\, \mathbf{U}\, close]\!]_2$

We let each state be represented by a bitvector of length 4 such that each digit represents a property, (*start, closed, heat, error*) respectively; a 0 represents the property being false, and 1 represents the property being true. Note that *start* is the most significant bit and *error* is the least significant bit in this setup.

Now we define the transition relation between two states

$$\begin{aligned} s \longrightarrow s' := {} & (s = 0000 \wedge s' = 1001) \vee (s = 0000 \wedge s' = 0100) \vee (s = 0100 \wedge s' = 0000) \vee \\ & (s = 0110 \wedge s' = 0000) \vee (s = 0110 \wedge s' = 0110) \vee (s = 0110 \wedge s' = 0100) \vee \\ & (s = 1001 \wedge s' = 1101) \vee (s = 1101 \wedge s' = 1001) \vee (s = 1101 \wedge s' = 0100) \vee \\ & (s = 0100 \wedge s' = 1100) \vee (s = 1100 \wedge s' = 1110) \vee (s = 1110 \wedge s' = 0110), \end{aligned}$$

and the intial state condition as follows

$$I(s) := s = 0000.$$

[1]The property $(\neg heat\, \mathbf{U}\, closed)$ reads as "the microwave does not heat up *until* the door is closed".

Then we encode $[\![T]\!]_2$ as follows

$$\begin{aligned}[\![T]\!]_2 &:= I(s_0) \wedge \bigwedge_{i=0}^{1} (s_i \longrightarrow s_{i+1}) \\ &= I(s_0) \wedge (s_0 \longrightarrow s_1) \wedge (s_1 \longrightarrow s_2).\end{aligned}$$

Now the loop condition is encoded as follows

$$\begin{aligned}L_2 := \bigvee_{l=0}^{2} {}_lL_2 &= {}_0L_2 \vee {}_1L_2 \vee {}_2L_2 \\ &= (s_2 \longrightarrow s_0) \vee (s_2 \longrightarrow s_1) \vee (s_2 \longrightarrow s_2).\end{aligned}$$

It now remains only to determine

$$[\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^0 \quad \text{and} \quad {}_l[\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^0.$$

We find

$$\begin{aligned}[\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^0 &= [\![\mathit{closed}]\!]_k^0 \vee ([\![\neg \mathit{heat}]\!]_2^0 \wedge [\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^1) \\ &= \mathit{closed}(s_0) \vee (\neg \mathit{heat}(s_0) \wedge ([\![\mathit{closed}]\!]_2^1 \vee ([\![\neg \mathit{heat}]\!]_2^1 \wedge [\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^2))) \\ &= \mathit{closed}(s_0) \vee (\neg \mathit{heat}(s_0) \wedge (\mathit{closed}(s_1) \vee (\neg \mathit{heat}(s_1) \wedge ([\![\mathit{closed}]\!]_2^2 \\ &\quad \vee ([\![\neg \mathit{heat}]\!]_2^2 \wedge [\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^3)))) \\ &= \mathit{closed}(s_0) \vee (\neg \mathit{heat}(s_0) \wedge (\mathit{closed}(s_1) \vee (\neg \mathit{heat}(s_1) \wedge (\mathit{closed}(s_2)))),\end{aligned}$$

and

$$\begin{aligned}{}_l[\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^0 &= {}_l[\![\mathit{closed}]\!]_2^0 \vee ({}_l[\![\neg \mathit{heat}]\!]_2^0 \wedge {}_l[\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^{\mathrm{succ}(0)}) \\ &= \mathit{closed}(s_0) \vee (\neg \mathit{heat}(s_0) \wedge {}_l[\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^1) \\ {}_l[\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^1 &= {}_l[\![\mathit{closed}]\!]_2^1 \vee ({}_l[\![\neg \mathit{heat}]\!]_2^1 \wedge {}_l[\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^{\mathrm{succ}(1)}) \\ &= \mathit{closed}(s_1) \vee (\neg \mathit{heat}(s_1) \wedge {}_l[\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^2) \\ {}_l[\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^2 &= {}_l[\![\mathit{closed}]\!]_2^2 \vee ({}_l[\![\neg \mathit{heat}]\!]_2^2 \wedge {}_l[\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^{\mathrm{succ}(2)}) \\ &= \mathit{closed}(s_2) \vee (\neg \mathit{heat}(s_2) \wedge {}_l[\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^l).\end{aligned}$$

Thus when inserting, we find

$$\begin{aligned}{}_l[\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^0 = \mathit{closed}(s_0) \vee \Big(&\neg \mathit{heat}(s_0) \wedge \\ &\big(\mathit{closed}(s_1) \vee (\neg \mathit{heat}(s_1) \wedge \\ &\big(\mathit{closed}(s_2) \vee (\neg \mathit{heat}(s_2) \wedge {}_l[\![\neg \mathit{heat}\ \mathbf{U}\ \mathit{closed}]\!]_2^l))))\Big)\end{aligned}$$

We achieve the final encoding by inserting the above-found expressions into the formula

$$[\![T, f]\!]_k = [\![T]\!]_k \wedge \left((\neg L_k \wedge [\![f]\!]_k^0) \vee \bigvee_{l=0}^{k} \big({}_lL_k \wedge {}_l[\![f]\!]_k^0\big) \right).$$

Note however that we can omit the loop condition, as a finite path satisfying the existence property $\mathbf{E}(\neg heat\,\mathbf{U}\,close)$ is sufficient. We therefore get

$$\begin{aligned}
[\![T, \neg heat\ \mathbf{U}\ closed]\!]_2 =& [\![T]\!]_2 \wedge [\![\neg heat\ \mathbf{U}\ closed]\!]_2^0 \\
=& I(s_0) \wedge (s_0 \longrightarrow s_1) \wedge (s_1 \longrightarrow s_2) \wedge \\
& closed(s_0) \vee (\neg heat(s_0) \wedge (closed(s_1) \vee (\neg heat(s_1) \wedge (closed(s_2))))).
\end{aligned}$$

b) Implement the encoding in Z3 and check its satisfiability.

We implement the above encoding in Z3 as shown below. Running this repeatedly yields one of the following 3 outputs

```
[s1 = 4, s2 = 0 , s0 = 0]
[s1 = 9, s2 = 13, s0 = 0]
[s1 = 4, s2 = 12, s0 = 0]
```

which in binary represents the 3 paths

$$\{0000, 1001, 1101\}, \{0000, 0100, 1100\}, \{0000, 0100, 0000\}$$

also shown here

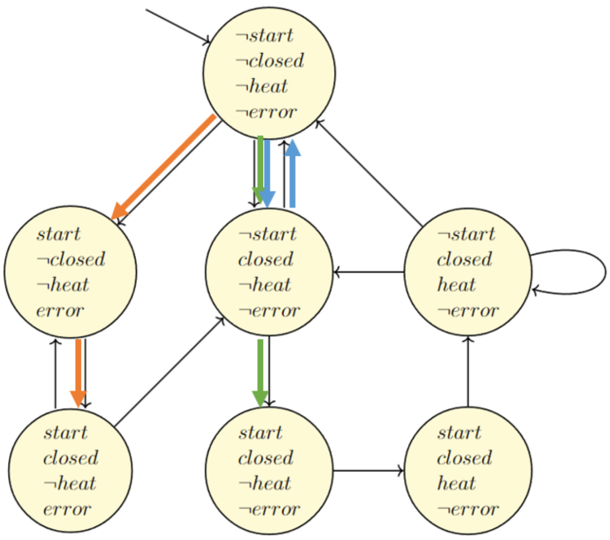


Figure 1: The three paths satisfying the property highlighted in different colors.

```
from z3 import *

# transition system
Trans = {(0b1110, 0b0110), (0b1101, 0b0100), (0b0100, 0b1100), (0b1100, 0b1110), (0
    b0000, 0b0100), (0b0100, 0b0000), (0b0110, 0b0110), (0b0110, 0b0100), (0b1001, 0
    b01101), (0b0110, 0b0000), (0b1101, 0b1001), (0b0000, 0b1001)}

# is closed true
def closed(state):
  return state & (1 << 2) == 0b0100

# is heat true
def heat(state):
  return state & (1 << 1) == 0b0010

# initial state
```

```python
def I(state):
    return state == 0

# function check if s --> s' in the transition system
def T(state1, state2):
  concat = []
  for (s1, s2) in Trans:
    cond = And(state1 == s1, state2 == s2)
    concat.append(cond)
  return Or(concat)

# setup z3 bitvectors
states = []
for i in range(3):
    states.append(BitVec("s" + str(i), 4))

# conditions
T2 = And(I(states[0]), And(T(states[0], states[1]), T(states[1], states[2])))
no_loop_cond = Or(
  closed(states[0]),
  And(
    Not(heat(states[0])),
    Or(
      closed(states[1]),
      And(
        Not(heat(states[1])),
        closed(states[2])
      )
    )
  )
)
Tf2 = And(T2, no_loop_cond)

s = Solver()
s.add(Tf2)

# get result
satisfied = s.check()
if satisfied == sat:
  print(s.model())
```

For step a), you must provide a detailed explanation of all the steps related to the construction of $[\![T, \neg \mathit{heat}\,\mathbf{U}\,\mathit{close}]\!]_2$. For step b), attach the corresponding Python script to your submission.

L07T: Theoretical Problems

DPLL-T inference sequences in all the problems below should use states $M||F$, where F is obtained from the previous state by applying the partial assignment from M.

L07T.1 Check the satisfiability of the formulas below and provide full DPLL-T inference sequences demonstrating the result. Obtained satisfiability results can be tested in Z3. The theory used in both of the examples is EUF. Thus, remember all the relevant "rules" such as the one of congruence closure.

1. $(b = c \vee b = d) \wedge b = a \wedge \neg(f(b) = f(d)) \wedge (\neg(f(a) = f(c)) \vee \neg(f(a) = f(b)))$

   Given the above formula, we convert the above problem into a propositional problem as follows:

$$(\underbrace{b = c}_{x_1} \vee \underbrace{b = d}_{x_2}) \wedge \underbrace{b = a}_{x_3} \wedge \neg \underbrace{(f(b) = f(d))}_{x_4} \wedge (\neg \underbrace{(f(a) = f(c))}_{x_5} \vee \neg \underbrace{(f(a) = f(b))}_{x_6}),$$

   which yields

$$(x_1 \vee x_2) \wedge x_3 \wedge \neg x_4 \wedge (\neg x_5 \vee \neg x_6).$$

   Now using DPLL-T and using notation similar to the one of the slides, we find

$$\begin{aligned}
\emptyset \,||\, (x_1 \vee x_2), x_3, \neg x_4, (\neg x_5 \vee \neg x_6) &\xRightarrow[x_3]{\text{UnitProp.}} \\
x_3 \,||\, (x_1 \vee x_2), \neg x_4, (\neg x_5 \vee \neg x_6) &\xRightarrow[\neg x_4]{\text{UnitProp.}} \\
x_3, \neg x_4 \,||\, (x_1 \vee x_2), (\neg x_5 \vee \neg x_6) &\xRightarrow[x_1]{\text{Decide}} \\
x_3, \neg x_4, x_1^d \,||\, (\neg x_5 \vee \neg x_6) &\xRightarrow[\neg x_5]{\text{Decide}} \\
x_3, \neg x_4, x_1^d, \neg x_5^d \,||\, \{\} &\xRightarrow[(\neg x_3 \vee x_4 \vee \neg x_1 \vee x_5)]{\text{T-Learn}} \\
x_3, \neg x_4, x_1^d, \neg x_5^d \,||\, (\neg x_3 \vee x_4 \vee \neg x_1 \vee x_5) &\xRightarrow{\text{T-Backjump}} \\
x_3, \neg x_4, x_1^d, x_5 \,||\, \neg x_6 &\xRightarrow[\neg x_6]{\text{UnitProp.}} \\
x_3, \neg x_4, x_1^d, x_5, \neg x_6 \,||\, \{\} &\xRightarrow[(\neg x_3 \vee x_4 \vee \neg x_1 \vee \neg x_5 \vee x_6)]{\text{T-Learn}} \\
x_3, \neg x_4, x_1^d, x_5, \neg x_6 \,||\, (\neg x_3 \vee x_4 \vee \neg x_1 \vee \neg x_5 \vee x_6) &\xRightarrow{\text{T-Backjump}} \\
x_3, \neg x_4, \neg x_1 \,||\, x_2, (\neg x_5 \vee \neg x_6) &\xRightarrow[x_2]{\text{UnitProp.}} \\
x_3, \neg x_4, \neg x_1, x_2 \,||\, (\neg x_5 \vee \neg x_6) &\quad \text{FAIL}
\end{aligned}$$

   Note that we FAIL at the last step, even though we have possible Decision rules to apply. This is done as the model $x_3, \neg x_4, \neg x_1, x_2$ is T-inconsistent with no decision labels to backjump to – thus a FAIL.

   Verifying with Z3 using the following code gives the output `no solution`.

```
from z3 import *
S = DeclareSort('S')
a, b, c, d = Consts('a b c d', S)
f = Function('f', S, S)
solve([Or(b == c, b == d), b == a, Not(f(b) == f(d)), Or(Not(f(a) == f(b)), Not(f(a)
     == f(b)))])
```

2. $f(f(a)) = g(b,b) \wedge f(b) = g(f(a),b) \wedge g(a,b) = b \wedge (f(a) = a \vee f(a) = b) \wedge (\neg(f(b) = b) \vee \neg(f(g(a,b)) = g(b,b)))$

   Firstly, we introduce the variables x_i, $i = 1, ..., 7$, representing the terms of the formula:

$$\underbrace{f(f(a)) = g(b,b)}_{x_1} \wedge \underbrace{f(b) = g(f(a),b)}_{x_2} \wedge \underbrace{g(a,b) = b}_{x_3} \wedge (\underbrace{f(a) = a}_{x_4} \vee \underbrace{f(a) = b}_{x_5}) \wedge (\neg(\underbrace{f(b) = b}_{x_6}) \vee \neg(\underbrace{f(g(a,b)) = g(b,b)}_{x_7}))$$

   Then we use DPLL(T) to check the satisfiability:

$$\begin{aligned}
\emptyset \; || x_1,\; x_2,\; x_3,\; (x_4 \vee x_5),\; (\neg x_6 \vee \neg x_7) &\xRightarrow[x_1]{\text{UnitProp.}} \\
x_1 \; || x_2,\; x_3,\; (x_4 \vee x_5),\; (\neg x_6 \vee \neg x_7) &\xRightarrow[x_2]{\text{UnitProp.}} \\
x_1,\; x_2 \; || x_3,\; (x_4 \vee x_5),\; (\neg x_6 \vee \neg x_7) &\xRightarrow[x_3]{\text{UnitProp.}} \\
x_1,\; x_2,\; x_3 \; || (x_4 \vee x_5),\; (\neg x_6 \vee \neg x_7) &\xRightarrow[x_4]{\text{Decide}} \\
x_1,\; x_2,\; x_3,\; x_4^d \; || (\neg x_6 \vee \neg x_7) &\xRightarrow[\neg x_6]{\text{Decide}} \\
x_1,\; x_2,\; x_3,\; x_4^d,\; \neg x_6^d \; || \{\} &\xRightarrow[(\neg x_1 \vee \neg x_2 \vee \neg x_3 \vee \neg x_4 \vee x_6)]{\text{T-learn}} \\
x_1,\; x_2,\; x_3,\; x_4^d,\; \neg x_6^d \; || (\neg x_1 \vee \neg x_2 \vee \neg x_3 \vee \neg x_4 \vee x_6) &\xRightarrow{\text{T-backjump}} \\
x_1,\; x_2,\; x_3,\; x_4^d,\; x_6 \; || \neg x_7 &\xRightarrow[\neg x_7]{\text{UnitProp}} \\
x_1,\; x_2,\; x_3,\; x_4^d,\; x_6,\; \neg x_7 \; || \{\} &
\end{aligned}$$

   Note that after the Decide on x_4, we get that

$$a \underset{\text{by } x_4}{=} f(a) \underset{\text{by } x_4}{=} f(f(a)) \underset{\text{by } x_1}{=} g(b,b) \text{ as well as } f(b) \underset{\text{by } x_2}{=} g(f(a),b) \underset{\text{by } x_4}{=} g(a,b) \underset{\text{by } x_3}{=} b.$$

   Therefore when we decide on $\neg x_6$ and get that $f(b) \neq b$, we reach a T-inconsistency. However, instead having $\neg x_7$ to be true gives us

$$b \underset{\text{by } x_6}{=} f(b) \underset{\text{by } x_3}{=} f(g(a,b)) \underset{\text{by } x_7}{\neq} g(b,b) \underset{\text{from above}}{=} a,$$

   which is perfectly consistent with our model.
   We verify the above result with Z3 using the following code

```
S = DeclareSort('S')
a, b = Consts('a b', S)
f = Function('f', S, S)
g = Function('g', S, S, S)
solve([f(f(a)) == g(b, b), f(b) == g(f(a), b), g(a, b) == b, Or(f(a) == a, f(a) == b
    ), Or(Not(f(b) == b), Not(f(g(a, b)) == g(b, b)))])
```

which gives a model which satisfies the formula

$$
\begin{aligned}
&[\mathtt{b} = \mathtt{S!val!1}, \\
&\ \mathtt{a} = \mathtt{S!val!0}, \\
&\ \mathtt{f} = [\mathtt{S!val!1} -> \mathtt{S!val!1}, \mathtt{else} -> \mathtt{S!val!0}], \\
&\ \mathtt{g} = [(\mathtt{S!val!0}, \mathtt{S!val!1}) -> \mathtt{S!val!1}, \mathtt{else} -> \mathtt{S!val!0}]]
\end{aligned}
$$