1)

Assumptions:

(i) The sentence s is made unambiguous by the use of parentheses.

(ii) For brevity, we only include the symbols ∧, ∨ and ¬

(iii) We use m[a] to denote the True/False value of a in model m.

Function: PL-True? (s, m)

↳ returns True/False

a) Let LHS = '', RHS = '', logic operator = ''.

b) Scan s from left to right, looking for a ¬, left parenthesis or symbol.

c) If symbol found, LHS = symbol, go to II

ii) If ¬ found, logic operator = ¬.
Goto 2.

e) If left (found :

Let n=1

while n >0:

LHS += next character

if next char =) : n=n-1

If next char == (: n=n+1

f) If logic operator == "", logic operator = next character.

g) Repeat lines 3-11, replacing LHS and RHS

h) If LHS or RHS is not a symbol,
LHS = PL_TRUE?(LHS, m),
RHS = PL_TRUE?(RHS, m)

(i) If logic - operator == ¬:
return (True) if LHS is False,
False otherwise.

(j) Else if logic - operator == ∧:
Return (True) if LHS and RHS are both true,
False otherwise.

(K) Else if logic - operator == ∨ :
Return (True) if either LHS/RHS are equal to (True),
False otherwise

---

2)

(i) True

(ii) False

(iii) For All ∨ True

c) Let k be number of symbols in s that do not appear in the partial model m.

Some sentences are always True or always False, no matter the Truth-Table assignment of the symbols in the sentence.

The only way to be sure is to evaluate the 2k rows of the truth table 1 by 1, waiting to see one true and one False evaluation, to determine if the sentence has Truth value.

However, due to the existence of Always True/False sentences, the worst case is O(2k).