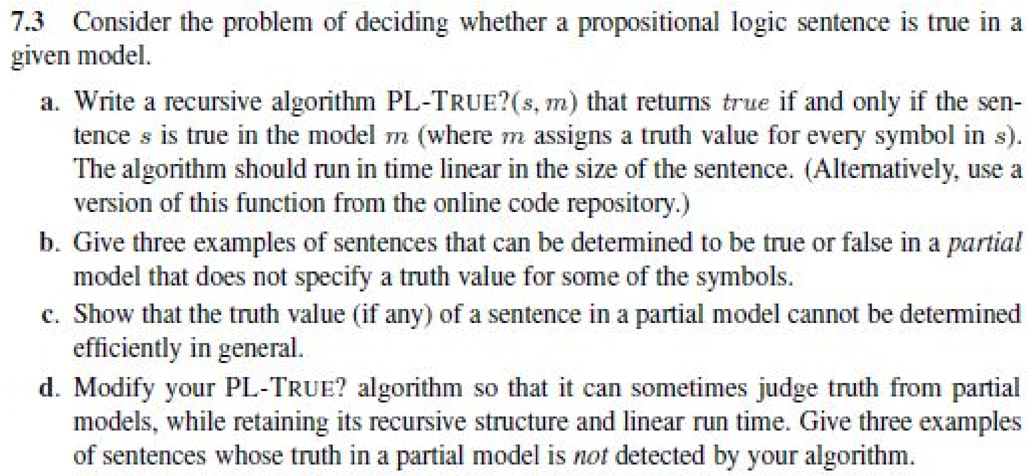
**Artificial Intelligence (CS 534)** - Assignment 3
By: Yousef Fadila yfadila@wpi.edu

---



**7.3** Consider the problem of deciding whether a propositional logic sentence is true in a given model.

**a.** Write a recursive algorithm PL-TRUE?$(s, m)$ that returns *true* if and only if the sentence $s$ is true in the model $m$ (where $m$ assigns a truth value for every symbol in $s$). The algorithm should run in time linear in the size of the sentence. (Alternatively, use a version of this function from the online code repository.)

**b.** Give three examples of sentences that can be determined to be true or false in a *partial* model that does not specify a truth value for some of the symbols.

**c.** Show that the truth value (if any) of a sentence in a partial model cannot be determined efficiently in general.

**d.** Modify your PL-TRUE? algorithm so that it can sometimes judge truth from partial models, while retaining its recursive structure and linear run time. Give three examples of sentences whose truth in a partial model is *not* detected by your algorithm.

a)
The code in the python library doesn’t assume that the model would assign value for every symbol. (the None return value). By assuming that we could make it further simple.
Here is a simple version to solve the problem in time linear to the size of the expression.
The function **pl_true_revised** is based on the function `pl_true` from the online code repository.

```
def pl_true_revised(exp, model={}):
   if exp in (True, False): return exp
   op, args = exp.op, exp.args
   if is_prop_symbol(op): return model.get(exp)
   elif op == '~': return not pl_true(args[0], model)

   p, q = args
   if op == '|': return pl_true(p, model) or pl_true(q, model)
   elif op == '&': return pl_true(p, model) and pl_true(q, model)
   if op == '==>': return pl_true(~p | q, model)
   elif op == '<==': return pl_true(p | ~q, model)
   pt = pl_true(p, model)
   qt = pl_true(q, model)
   if op == '<=>': return pt == qt
```

b) we can choose the examples such as they are valid or unsatisfiable regardless the value of the some symbols
i) $\alpha \lor \neg\alpha \lor \beta$ - can be determine to be True even if partial model doesn’t assign value to all
ii) $\alpha \land \neg\beta \land \beta$ can be determine to be False even if partial model doesn’t assign value to all
iii) $(\alpha \land \beta) \Rightarrow \alpha$ can be determine to be True even if partial model doesn’t assign value to all

c) as we show in (b) partial model in some cases can find the truth value of a sentence by determining if it's valid or unsatisfiable regardless the value of unassigned symbols. As we learn in class, deciding validity or unsatisfiability is NP-complete problem so there is no general algorithm that work in polynomial time for such problem.

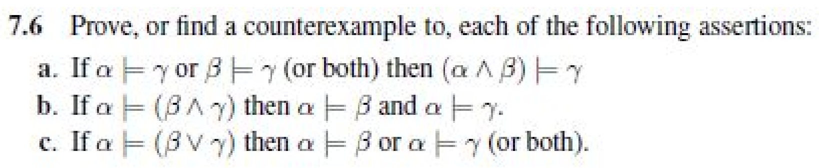
d) we could modify the algorithim to use Short-circuit evaluation
https://en.wikipedia.org/wiki/Short-circuit_evaluation
In such case instead of evaluate all args before deciding the outcome, the second argument is executed or evaluated only if the first argument does not suffice to determine the value of the expression.

The current solution in (a) already support this feature as a result of python supporting of short circuit evalution (
https://stackoverflow.com/questions/2580136/does-python-support-short-circuiting) .
For example
```
if op == '|': return pl_true(p, model) or pl_true(q, model)
```
Will return True even if `pl_true(p, model)` resolved as True, even if the model is unable to evaluate pl_true(q, model) because of missing assignments for symbols in q.

Three examples which truth is partial model is not detected by the suggested algorithm:

1) $\alpha \lor \neg\beta \lor \beta$ - if the partial model was unable to evaluate $\beta$ it won't return True for this sentence .
2) $\alpha \land False$ - if the partial model was unable to evaluate $\alpha$ it won't return False for this sentence .( for instance it would return False for $False \land \alpha$ as it won't try to evaluate the 2nd argument $\alpha$ .
3) $\alpha \lor$ True for the same reason as above.

---



**7.6** Prove, or find a counterexample to, each of the following assertions:

**a.** If $\alpha \models \gamma$ or $\beta \models \gamma$ (or both) then $(\alpha \land \beta) \models \gamma$

**b.** If $\alpha \models (\beta \land \gamma)$ then $\alpha \models \beta$ and $\alpha \models \gamma$.

**c.** If $\alpha \models (\beta \lor \gamma)$ then $\alpha \models \beta$ or $\alpha \models \gamma$ (or both).

a) True:

$if\ a\,|=\gamma\ iff\ M(a)\ \subseteq\ M(\delta)$

$if\ \beta\,|=\gamma\ iff\ M(\beta)\ \subseteq\ M(\delta)$

Using Monotonicity feature we know that $M(\alpha\ \land\ \beta)\ \subseteq\ M(a)\ and\ also\ M(\alpha\ \land\ \beta)\ \subseteq\ M(\beta)$

So if $M(a)\ \subseteq\ M(\gamma)$ then $M(\alpha\ \land\ \beta)\ \subseteq\ M(a)\ \subseteq\ M(\gamma)$

If $M(\beta)\ \subseteq\ M(\gamma)$ then $M(\alpha\ \land\ \beta)\ \subseteq\ M(\beta)\ \subseteq\ M(\gamma)$

$\Rightarrow\ if\ M(a)\ \subseteq\ M(\gamma)\ or\ M(\beta)\ \subseteq\ M(\gamma)\ then\ M(\alpha\ \land\ \beta)\ \subseteq\ M(\gamma)$

b) True: By intuitive, if $\alpha$ entails both $\beta$ *and* $\gamma$ *togather* then it must entails $\beta$ *and* $\gamma$ *individually* we can proof that by set theory.

*from set theory we know that* $M(\beta \wedge \gamma) \subseteq M(\beta)$ *and* $M(\beta \wedge \gamma) \subseteq M(\gamma)$

*if* $M(a) \subseteq M(\beta \wedge \gamma)$ and using the thumb rule above: $M(a) \subseteq M(\beta \wedge \gamma) \subseteq M(\gamma)$ and $M(a) \subseteq M(\beta \wedge \gamma) \subseteq M(\beta)$

c) False.

If ( $\beta \vee \gamma$ ) is valid, for example $\beta = \neg\gamma$ then $M(a) \subseteq M(\beta \vee \gamma)$ for all $\alpha$ as M( $\beta \vee \gamma$ ) is always true. So in such case it is enough to show $\alpha$ that does not entail $\beta$ *or* $\neg\beta$

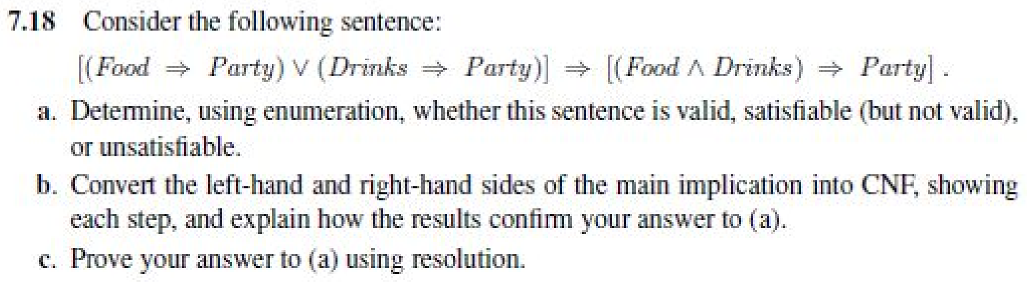
$\alpha = x \vee y$

$\beta = \neg y \wedge x$

$\gamma = \neg\beta = \neg(\neg y \wedge x) = y \vee \neg x$

In such case a entails $\beta \vee \gamma$ but it doesn't entail either $\beta$ (for example when x is false but y is true) nor $\gamma$ (for example when x is true but y is false).

---



**7.18** Consider the following sentence:

$$[(Food \Rightarrow Party) \vee (Drinks \Rightarrow Party)] \Rightarrow [(Food \wedge Drinks) \Rightarrow Party] .$$

**a.** Determine, using enumeration, whether this sentence is valid, satisfiable (but not valid), or unsatisfiable.

**b.** Convert the left-hand and right-hand sides of the main implication into CNF, showing each step, and explain how the results confirm your answer to (a).

**c.** Prove your answer to (a) using resolution.

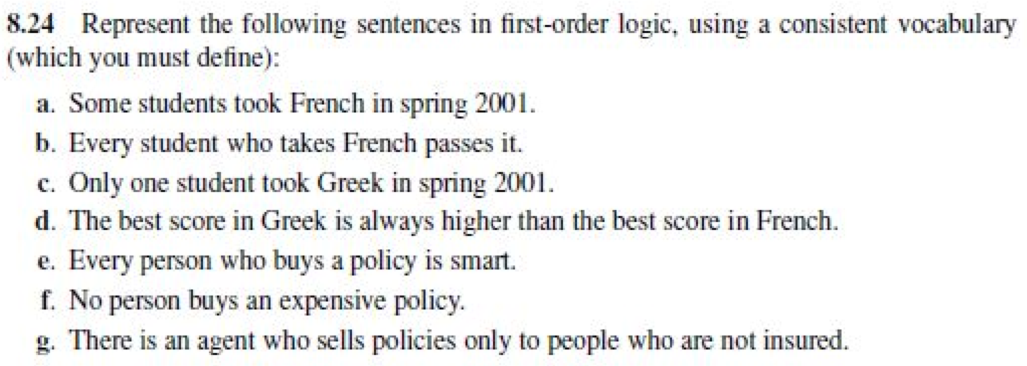
| Food = F | Party=P | Drinks=D | i= F ⇒ P | ii= D ⇒ P | $i \vee ii$ | iii = $F \wedge D$ | iii ⇒ P | $i \vee ii$ => ($iii$ ==> P) |
|---|---|---|---|---|---|---|---|---|
| F | F | F | T | T | T | F | T | T |
| F | F | T | T | F | T | F | T | T |
| F | T | F | T | T | T | F | T | T |
| F | T | T | T | T | T | F | T | T |
| T | F | F | F | T | T | F | T | T |
| T | T | F | T | T | T | F | T | T |
| T | F | T | F | F | F | T | F | T |
| T | T | T | T | T | T | T | T | T |

$i \vee ii$ => ($iii$ ==> P) is always True thus the sentence is valid

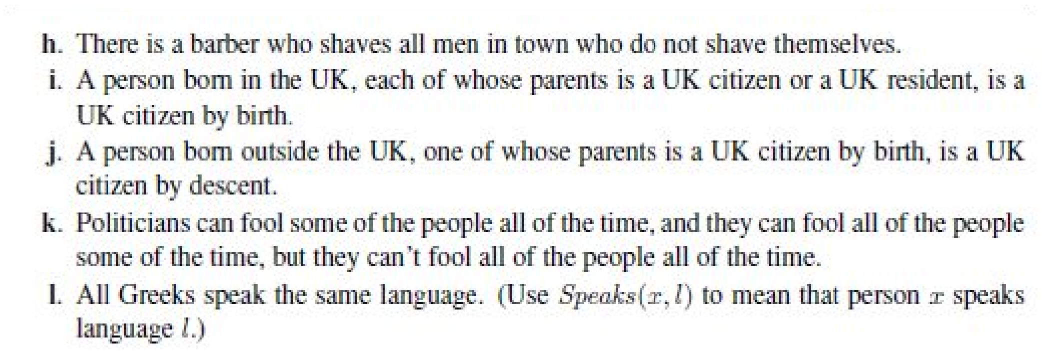
b)
Left side:
$(F \Rightarrow P) \vee (D \Rightarrow P)$
$(F \Rightarrow P) \vee (D \Rightarrow P)$
$(\neg F \vee P) \vee (\neg D \vee P)$
$(\neg F \vee P) \vee (\neg D \vee P)$
$\neg F \vee P \vee \neg D \vee P$
$\neg F \vee \neg D \vee P$
Right side:
$(F \wedge D) \Rightarrow P$
$\neg (F \wedge D) \vee P$
$(\neg F \vee \neg D) \vee P$
$\neg F \vee \neg D \vee P$
After converting both sides to CNF we see that they are the same.
So the sentence is actually of the form $\alpha \Rightarrow \alpha$ which is equal to $\neg\alpha \vee \alpha$ which is **valid.**

c) to prove it using resolution, we need to prove that negation is unsatisfiable
$\neg([(F \Rightarrow P) \vee (D \Rightarrow P)] \Rightarrow [(F \wedge D) \Rightarrow P])$
$\neg(\neg[(F \Rightarrow P) \vee (D \Rightarrow P)] \vee [(F \wedge D) \Rightarrow P])$
$[(F \Rightarrow P) \vee (D \Rightarrow P)] \wedge \neg [(F \wedge D) \Rightarrow P])$
From B we know that $(F \Rightarrow P) \vee (D \Rightarrow P)$ = $\neg F \vee \neg D \vee P$ and $(F \wedge D) \Rightarrow P$ = $\neg F \vee \neg D \vee P$
So the sentence become:
$(\neg F \vee \neg D \vee P) \wedge \neg(\neg F \vee \neg D \vee P)$
$(\neg F \vee \neg D \vee P) \wedge F \wedge D \wedge \neg P$
$\oslash$
As we failed to satisfy the negation, that's prove using resolution that the original is valid.

---


**8.24** Represent the following sentences in first-order logic, using a consistent vocabulary (which you must define):

**a.** Some students took French in spring 2001.
**b.** Every student who takes French passes it.
**c.** Only one student took Greek in spring 2001.
**d.** The best score in Greek is always higher than the best score in French.
**e.** Every person who buys a policy is smart.
**f.** No person buys an expensive policy.
**g.** There is an agent who sells policies only to people who are not insured.



h. There is a barber who shaves all men in town who do not shave themselves.

i. A person born in the UK, each of whose parents is a UK citizen or a UK resident, is a UK citizen by birth.

j. A person born outside the UK, one of whose parents is a UK citizen by birth, is a UK citizen by descent.

k. Politicians can fool some of the people all of the time, and they can fool all of the people some of the time, but they can't fool all of the people all of the time.

l. All Greeks speak the same language. (Use $Speaks(x, l)$ to mean that person $x$ speaks language $l$.)

Let's start building vocabulary that has enough power to represent sentences in first order logic:

**Takes(Student, Course, Semester)**
Infer that student "Student" has taken Course in Semester
**Succeed(Student, Course, Semester)**
Infer that student "Student" success Course in Semester
**Grade(Student, Course, Semester)**
Infer the final score of student "Student" in Course taken in Semester
**Student(id)**
Infer that "id" is student
**Objects:** "Spring 2001", "French","Greek"

a) $\exists$ id Student(id) $\wedge$ Takes(id, French, Spring 2001).
b) $\forall$ id, semester Student(id) $\wedge$ Takes(id, French, semester) $\Rightarrow$ Succeed(id, French, semester)
c) $\exists$ id Student(id) $\wedge$ Takes(id,Greek, Spring 2001) $\wedge$
$\forall$ id2 $id2 \neq id$ $\Rightarrow$ $\neg$Takes(id2,Greek, Spring 2001).
d) $\forall$ semester $\exists$ id1 $\forall$ id2 Grade(id1,Greek, semester) > Grade(id2, French, semester).

For the next sentences. we need first to extend our vocabulary to be able to represent the sentences in first order logic
**Person(id), Policy(id), ExpensivePolicy(id), Smart(id), Agent(id), InsuredPerson(id),**
Infer that "id" is person, policy, ExpensivePolicy smart,Agent or InsuredPerson respectively.
**Buys(id,item)**
Infer that id buys item
**Sells(id, item,id2)**
Infer that id sells item to id2

e) $\forall$ id Person(id) $\wedge$ ($\exists$ item Policy(item) $\wedge$ Buys(id, item)) $\Rightarrow$ Smart(id).
f) $\forall$ id, item Person(id) $\wedge$ ExpensivePolicy(item) $\Rightarrow$ $\neg$Buys(id, item).
g) $\exists$ id Agent(id) $\wedge$ $\forall$ item, id2 Policy(item) $\wedge$ Sells(id, item, id2) $\Rightarrow$$\neg$InsuredPerson(id2)

For the next sentences. we need first to extend our vocabulary to be able to represent the sentences in first order logic

**Barber(id), Man(id),**
Infer that “id” is barber, man respectively.
**Shaves(id,id1)**
Infer that id1 shaves id2

h) ∃ B Barber(B) ∧ ∀ M Man(M) ∧ ¬Shaves(M, M) ⇒ Shaves(B, M)

For the next sentences. we need first to extend our vocabulary to be able to represent the sentences in first order logic
**Borns(id, country)**
Infer that id borns in country
**Parent(id, id1)**
Infer that id is parent of id1
**Citizen(id, country)**
Infer that id is a citizen of country
**Resident(id, country)**
Infer that id is a resident of country
**CitizenByBirth(id, country)**
Infer that id is a citizen of country by birth
**CitizenByDescent(id, country)**
Infer that id is a citizen of country by descent
**Politician(id)**
Infer that id is a Politician
**Fools**(id, id1, t)
Infer that id fools id1 at time t;

i) ∀ id Person(id)∧Borns(id,UK)∧(∀ id1 Parent(id1,id) ⇒ ( Resident(id1,UK) ∨ Citizen(id1,UK) )) ⇒ CitizenByBrith(id,UK).

j) ∀ id Person(id) ∧ (∃ id1 Parent(id1, id) ∧ CitizenByBirth(id1,UK)) ∧ ¬Born(id,UK) ⇒ CitizenByDescent(id,UK).

k) ∀ id Politician(id) ⇒ (∃ t ∀ id1 Person(id1) ⇒ Fools(id, id1, t)) ∧ (∃ id2 ∀ t Person(id2) ∧ Fools(id, id2, t)) ∧ ¬(∀ t ∀ id3 Person(id3) ⇒ Fools(id, id3, t))

As suggested in the question: **Speaks(x,l)** means x speaks l

l) ∀ id, id1, l Citizen(id,Greece) ∧ Citizen(id2,Greece) ∧ Speaks(id, l) ⇒ Speaks(id2, l)