Massachusetts Institute of Technology

16.410-13 Principles of Autonomy and Decision Making

Due: Friday, 12/03/04

Paper solutions are due no later than 5pm on Friday, 12/03/04. Please give solutions to course secretary Brian O' Conaill, at his desk outside of 33-330.

Objectives

The first objective of this problem set is to develop your facility with propositional logic and the DPLL algorithm. For propositional logic, the objective is to develop your skill at encoding English statements in propositional logic, determining the truth of a sentence with respect to an interpretation, and reducing a sentence to clausal form. For satisfiability, the objective is to exercise your understanding of unit propagation and DPLL (backtrack search plus unit propagation).

The second objective is to develop your understanding of model-based diagnosis. This includes applying the concepts of constraint suspension, conflict extraction, candidate generation and candidate testing.

The third objective is to develop your understanding of decision tree learning. This includes the generation of a decision tree from a data set, and the application of the decision tree to new instances.

16.410 Assignment

For 16.410 students, this problem set is comprised of **both** an **online portion** and a **written portion**, given below. 16.410 students should log into the online tutor, and complete the online problems under problem set #9 by the due date, listed above. 16.410 students must complete the threee problems listed below, turning in hardcopy solutions as indicated above.

16.413 Assignment

16.413 students **do not** have any **online problems** assigned, **only** a **written assignment**. Please complete **the three problem**s listed below and turn in hardcopy solutions to the course secretary, as indicated above.

Problem 1 Propositional Logic and Satisfiability

Consider the following quote, taken from (Barwise and Etchemendy, 1993):

> If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a [mortal] mammal, then it is horned. The unicorn is magical if it is horned.

Represent this quote using the following propositional symbols:

MY	=	"Mythical Unicorn"
IU	=	"Immortal Unicorn"
MM	=	"Mortal Mammal"
HU	=	"Horned Unicorn"
MG	=	"Magical Unicorn"

Part a. Using the above propositional symbols, provide a propositional logic sentence S that is a **direct translation** of the above quote.

Solution:

(MY implies IU) and
((not MY) implies MM) and
((IU or MM) implies HU) and
(HU implies MG);

Part b. Determine the truth of your sentence S, from part a, for the following interpretation I:

{MY = False, IU = False, MM = True, HU = True, MG = True}

- Show how you derive the truth of S, using I and the semantics of the logical connectives. Your derivation should be similar to the derivation given in class.

Solution: S is True in I

Derivation:

Substituting I into sentence S produces:

(False implies False) and
((not False) implies True) and
((False or True) implies True) and
(True implies True);

Applying the definition of implies:

(A implies B) iff (not A or B)

results in:

((not False) or False) and
((not (not False)) or True) and
((not (False or True)) or True) and
((not True) or True);

Applying the Boolean operator not, simplifies S to:

(True or False) and
(False or True) and
((not (False or True)) or True) and
(False or True);

A disjunction is true if **any one** disjunct is true, simplifying S to:

True and
True and
True and
True;

A conjunction is true if **all** conjuncts are true, simplifying S to:

True

Part c. Reduce your sentence S, from part a, to a set of propositional clauses.

- Walk through your reduction step by step, starting from sentence S and ending with your set of clauses. Your reduction should be similar in style to the reduction given in the appendix of the class notes.

- List your resulting clauses, labeling them C1 … Cn.

Solution: First, start by substituting for the definition of implies:

(A implies B) iff (not A or B)

into s, producing:

((not MY) or IU) and
((not (not MY)) or MM) and
((not (IU or MM)) or HU) and
((not HU) or MG);

Second, drive the negations inward, by applying de Morgan's theorem:

(not (A or B)) → ((not A) and (not B))

and double negation elimination:

(not (not P)) → P

producing:

((not MY) or IU) and
(MY or MM) and
(((not IU) and (not MM)) or HU) and
((not HU) or MG);

Third, drive conjuncts (and) outward by using distribution:

(or (and A B) C) → (and (A or C)(B or C))

producing:

((not MY) or IU) and
(MY or MM) and
(((not IU) or HU) and ((not MM) or HU)) and
((not HU) or MG);

Nested conjuncts can be collapsed, by associativity and commutativity:

((C and B) and A) → (A and B and C)

Applying this to S results in five clauses:

C1: (not MY) or IU;
C2: MY or MM;
C3: (not IU) or HU;
C4: (not MM) or HU;
C5: (not HU) or MG;

Part d. Add the following clause, C0, to the set of clauses C1 … Cn, from part c:

C0: ¬ MG

Apply unit propagation to clauses C0, C1 … Cn.

Solution: In the following, a literal is crossed out if false, and underlined if assigned by unit propagation:

C0: not MG → MG = False
C1: ~~(not MY)~~ or ~~IU~~; Violated
C2: MY or ~~MM~~; → MY = True
C3: (not IU) or ~~HU~~; → IU = False
C4: (not MM) or ~~HU~~; → MM = False
C5: (not HU) or ~~MG~~; → HU = False

- List the truth assignment (one of "True," "False" or "Unassigned") that unit propagation determines for each proposition. For each proposition that is assigned "True" or "False," give its support, that is, the clause and assignments used to determine the proposition's truth.

Solution: MG = False with support C0,
HU = False with support C5,
MM = False with support C4,
IU = False with support C3,
MY = True with support C2.

- List all clauses that are violated by this truth assignment.

Solution: C1

- List all clauses that are satisfied by this truth assignment.

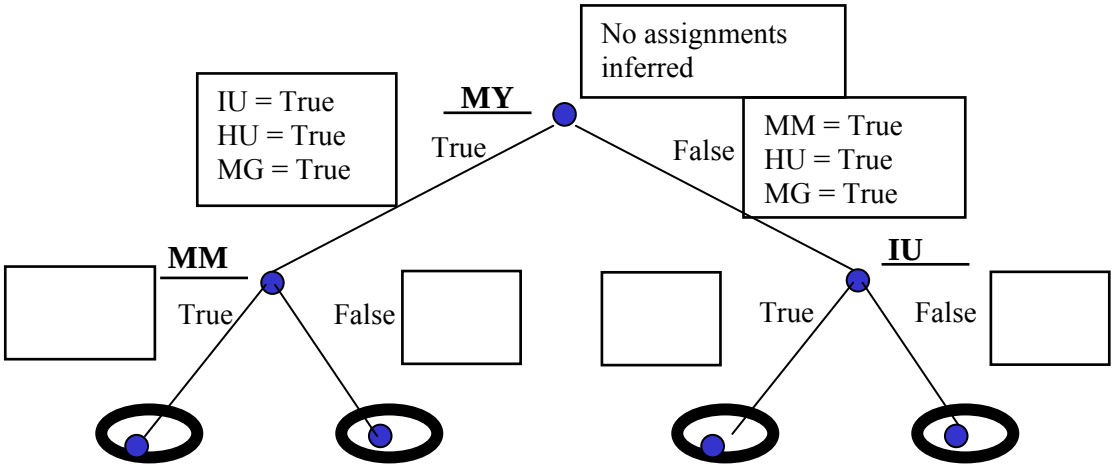
Solution: C0, C2-C5

Part e. Use the DPLL algorithm (Backtrack Search + Unit Propagation) to enumerate **all models** of clauses C1 … Cn, from part c (**do not** include C0). Recall that a model = assigns True or False to all propositions, such that all clauses are satisfied.

You must assign the propositions in the order: MY, IU, MM, HU and, finally, MG. For each proposition, first assign "True," followed by "False."

- Give the **full** backtrack search tree resulting from applying DPLL, that is, run the search until **all** subtrees have been explored or pruned. Label each branch with the assignment made for that branch. In addition, next to each branch, list all assignments that are derived by unit propagation for that branch. Cross off any node that is pruned as inconsistent. Circle any (leaf) node that denotes a model.

Solution:



- List all models that you find by applying DPLL.

Solution:

- MY = True, IU = True, HU = True, MG = True, MM = True
- MY = True, IU = True, HU = True, MG = True, MM = False
- MY = False, IU = True, HU = True, MG = True, MM = True
- MY = False, IU = False, HU = True, MG = True, MM = True

Derivation of Solutions: Apply DPLL to S = C1-C5

C1: (not MY) or IU;
C2: MY or MM;
C3: (not IU) or HU;
C4: (not MM) or HU;
C5: (not HU) or MG;

while branching in the order MY, IU, MM, HU and, finally, MG.

Starting with the root node, no propagations are possible, and we branch on MY.

Node MY = True

C1: ~~(not MY)~~ or IU; → IU = True
C2: MY or MM; Satisfied
C3: ~~(not IU)~~ or HU; → HU = True
C4: (not MM) or HU; Satisfied
C5: ~~(not HU)~~ or MG; → MG = True

Note that, as a result of unit propagation, C1-C5 are all satisfied, even though MM has not been assigned. This means that either assignment to MM is guaranteed to be satisfy the clauses.

Note that the partial assignment MY = True, IU = True, HU = True, MG = True is called an *implicant* of C1-C5, since the partial assignment guarantees the truth of C1-C5.

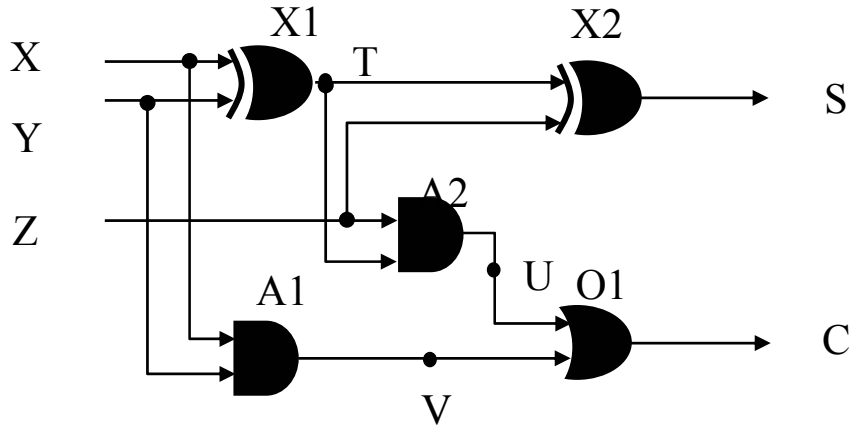
Node MY = False

C1: (not MY) or IU; Satisfied
C2: ~~MY~~ or MM; → MM = True
C3: (not IU) or HU; Satisfied
C4: ~~(not MM)~~ or HU; → HU = True
C5: ~~(not HU)~~ or MG; → MG = True

Again, C1-C5 are all satisfied, this time with MY = False, HU = True, MG = True, and MM = True, and IU unassigned. The tree at this point is exhausted.

Problem 2 – Model-based Diagnosis

Consider the application of Single_Fault_w_Conflicts (constraint suspension) to the full adder circuit, shown in the schematic below.



The full adder is a digital circuit that has three binary inputs (X, Y and Z), and computes a two bit sum (00, 01, 10, 11), where the higher order bit of the sum is the output C and the lower order bit of the sum is S. For example, if X = 1, Y = 0 and Z = 1, then the result is C = 1 and S = 0.

The circuit consists of two eXclusive OR gates (X1 and X2), two AND gates (A1 and A2) and one OR gate (O1). In constraint suspension, we model each component C as having two possible modes: the good mode, denoted G(C), and the unknown mode, denoted U(C). If a component is working correctly, then it is in the good mode; otherwise, it is in the unknown mode.

Each component has no behavior specified when in the unknown mode, that is, its constraints are "suspended." This means that any behavior is possible (consistent) in the unknown mode.

Each component has its standard behavior when in the good mode; that is:

- the output of an AND gate is 1 if and only if both its inputs are 1,
- the output of an OR gate is 0 if and only if both its inputs are 0, and
- the output of an exclusive OR gate is 1 if and only if one input is 1 and the other input is 0.

Suppose you observe the following inputs and outputs for the full adder:

OBS = { X = 0, Y = 0, Z = 1, S = 0 and C = 1}

Given OBS and the model described above, diagnose the circuit using the algorithm Single_Fault_w_Conflicts, as presented in lecture.

Part a. Candidate Generation

Generate the initial candidate set. First assume that all components are in the "good" mode G. Use propagation to derive a symptom, and extract a conflict.

- List the conflict found.
- List all candidates determined from this conflict.

Solution: Starting with:

X= 0	Given	Y=0	Given	Z=1	Given
S=0	Given	C=1	Given		

You deduce the following by propagation:

T=0 by X1=G, X=0, Y=0
S=1 by X2=G, T=0, Z=1

Which is inconsistent with the observable for S=0.

The **conflict** is:

Not (X1=G and X2=G)

This conflict results in **two single fault candidates**:

A1=G and A2=G and O1=G and X1=U and X2=G
A1=G and A2=G and O1=G and X1=G and X2=U

Alternatively, you could have first discovered a conflict at C:

T=0 by X1=G, X=0, Y=0
U=0 by A2=G, T=0
V=0 by A1=G, X=0
C=0 by O1=G, U=0, V=0

This is inconsistent with the observable for C=1. The resulting conflict is:

Not (A1=G and A2=G and O1=G and X1=G)

This conflict results in **four single fault candidates**:

A1=U and A2=G and O1=G and X1=G and X2=G
A1=G and A2=U and O1=G and X1=G and X2=G
A1=G and A2=G and O1=U and X1=G and X2=G
A1=G and A2=G and O1=G and X1=U and X2=G

The first, smaller conflict is preferable, since it leads to a smaller candidate set.

Part b. Derive Diagnoses by Testing Candidates

Identify the candidates from part a that are diagnoses, by applying the algorithm Single_Fault_Test_Candidates.

- For each candidate, in the order tested, list:
  - the candidate tested,
  - each assignment to a variable that is derived by propagation,
  - whether or not the candidate is consistent with the model and observations, and
  - a conflict extracted when the candidate is inconsistent,

- List your final diagnoses.

Solution: We test the candidates in lexicographic order:
Again we start with:

X= 0	Given	Y=0	Given	Z=1	Given
S=0	Given	C=1	Given		

Candidate: A1=G and A2=G and O1=G and **X1=U** and X2=G

Consistent?	Yes
Derived assignments:	C=1, T=1, U=1, V=0
New conflict:	None
Pruned by conflict:	No

You deduce the following by propagation:

V=0 by A1=G, X=0
T=1 by X2=G, Z=1, S=0
U=1 by A2=G, T=1, Z=1
C=1 by O1=G, U=1

Candidate: A1=G and A2=G and O1=G and X1=G and **X2=U**

Consistent?	No
Derived assignments:	C=0, T=0, U=0, V=0
New conflict:	Not (A1=G and A2=G and O1=G and X1=G)
Pruned by conflict:	No

You deduce the following by propagation.

T=0 by X1=G, X=0, Y=0
U=0 by A2=G, T=0
V=0 by A1=G, X=0
C=0 by O1=G, U=0, V=0

Diagnoses: X1 is broken, that is (A1=G and A2=G and O1=G and **X1=U** and X2=G)

Alternate Solution:

If you alternatively discovered the conflict:
Not (A1=G and A2=G and O1=G and X1=G)

Then your candidates are:
A1=U and A2=G and O1=G and X1=G and X2=G
A1=G and A2=U and O1=G and X1=G and X2=G
A1=G and A2=G and O1=U and X1=G and X2=G
A1=G and A2=G and O1=G and X1=U and X2=G

Testing these candidates in lexicographic order results in the following:

Candidate: A1=U and A2=G and O1=G and X1=G and X2=G
Consistent? No
Derived assignments: C=1, T=1, U=1, V=0
New conflict: not (X1=G and X2=G)
Pruned by conflict: No

You deduce the following by propagation:
T=0 by X1=G, X=0, Y=0
S=1 by X2=G, T=0, Z=1

Where S=1 is inconsistent with observation S=0.

Candidate: A1=G and **A2=U** and O1=G and X1=G and X2=G
Consistent? No
Derived assignments: none
New conflict: none
Pruned by conflict: **Yes**

Candidate: A1=G and A2=G and **O1=U** and X1=G and X2=G
Consistent? No
Derived assignments: none
New conflict: none
Pruned by conflict: **Yes**

Candidate: A1=G and A2=G and O1=G and **X1=U** and X2=G

Consistent?	Yes
Derived assignments:	C=1, T=1, U=1, V=0,
New conflict:	none
Pruned by conflict:	no

You deduce the following by propagation:

V=0 by A1=G, X=0
T=1 by X2=G, Z=1, S=0
U=1 by A2=G, T=1, Z=1
C=1 by O1=G, U=1

Completing all candidates. Again the solution is:

Diagnoses: X1 is broken, that is (A1=G and A2=G and O1=G and **X1=U** and X2=G)

Problem 3 Decision Tree Learning

You are using a rover on Mars to determine whether or not rocks you encounter are Martian in origin or are extra-Martian meteorites. You have collected eight samples, and used a human expert to determine whether or not the eight sample rocks are meteorites. You now want the rover to learn to classify meteorites by itself using the labeled examples.

Example	IsHeavy	IsShiny	IsSpotted	IsSmooth	IsMeteorite
A	0	0	0	0	0
B	0	0	1	0	0
C	1	1	0	1	0
D	1	0	0	1	1
E	0	1	1	0	1
F	0	0	1	1	1
G	0	0	0	1	1
H	1	1	0	0	1
U	1	1	1	1	?
V	0	1	0	1	?
W	1	1	0	0	?

You know whether or not rocks A through H are meteorites, but you do not know about U through W.

Part a. Build an ID-3 decision tree to classify rocks.

Part b. Classify rocks U, V and W as meteorites or not, using this decision tree.

Problem 3 Decision Tree Learning

You are using a rover on Mars to determine whether or not rocks you encounter are Martian in origin or are extramartian meteorites. You have collected eight samples, and used a human expert to determine whether or not the eight sample rocks are meteorites. You now want the rover to learn to classify meteorites by itself using the labelled examples.

Example	IsHeavy	IsShiny	IsSpotted	IsSmooth	IsMeteorite
A	0	0	0	0	0
B	0	0	1	0	0
C	1	1	0	1	0
D	1	0	0	1	1
E	0	1	1	0	1
F	0	0	1	1	1
G	0	0	0	1	1
H	1	1	0	0	1
U	1	1	1	1	?
V	0	1	0	1	?
W	1	1	0	0	?

You know whether or not rocks A through H are meteorites, but you do not know about U through W.

Part a. Build an ID-3 decision tree to classify rocks.
Initial entropy: 0.95443
Gain of IsHeavy: .003229
Gain of IsShiny: .003229
Gain of IsSpotted: .048795
Gain of IsSmooth: .048795

Tie between IsSmooth and IsSpotted: Choose IsSmooth

Entropy of ¬ IsSmooth: 1.0
Gain of IsHeavy: .311278
Gain of IsShiny: 1.0
Gain of IsSpotted: 0.0

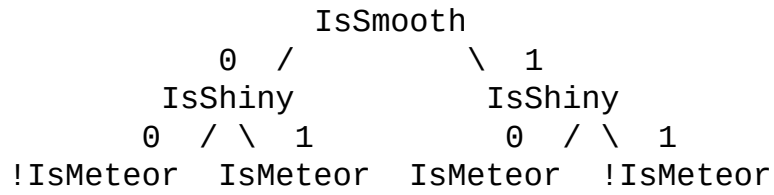
Choose IsShiny

Entropy of IsSmooth: 0.811278

Gain of IsHeavy: .311278
Gain of IsShiny: .811278
Gain of IsSpotted: 0.122556

Choose IsShiny

Done.



Part b. Classify rocks U, V and W as meteorites or not using this decision tree.

U and V: IsSmooth: 1 IsShiny: 1
Therefore not a meteorite.

W: IsSmooth: 0 IsShiny: 1
Therefore is a meteorite.