CSC 380/480 – Foundations of Artificial Intelligence

Winter 2007

Assignment 3 Solutions to Selected Problems

1. Suppose we have a knowledge base KB containing the following 4 sentences:

KB1: $(\mathbf{A} \vee \mathbf{B}) \wedge \mathbf{C} \Rightarrow \mathbf{E}$
KB2 $\mathbf{D} \wedge \mathbf{F} \Rightarrow \mathbf{A}$
KB3: $\neg\mathbf{E} \vee \mathbf{D}$
KB4: $\mathbf{B} \wedge \mathbf{C}$
KB5: $\mathbf{F}$

Want to derive sentence **A** (i.e., show that **A** is entailed by the knowledge base KB).

Solution:

Using forward chaining and rules of inference:

1.	$\mathbf{C}$	KB4 and AND-Elimination
2.	$\mathbf{B}$	(again) KB4 and AND-Elimination
3.	$\mathbf{A} \vee \mathbf{B}$	Step 2, using OR-Introduction (introducing **A**)
4.	$(\mathbf{A} \vee \mathbf{B}) \wedge \mathbf{C}$	Step 3, Step 1, and AND-Introduction
5.	$\mathbf{E}$	Step 4, KB1, and Modus Ponens
6.	$\mathbf{D}$	Step 5, KB3, and Resolution
7.	$\mathbf{D} \wedge \mathbf{F}$	Step 6, KB5, and AND-Introduction
8.	$\mathbf{A}$	Step 7, KB2, and Modus Ponens

Note that this is not the only possible derivation. There are rules that can be used at various steps eventually leading to A.

Using resolution-refutation procedure: First we convert the KB into clause form (note that KB1 and KB4 each become two separate clauses):

KB1.1: $\neg\mathbf{A} \vee \neg\mathbf{C} \vee \mathbf{E}$
KB1.2: $\neg\mathbf{B} \vee \neg\mathbf{C} \vee \mathbf{E}$
KB2: $\neg\mathbf{D} \vee \neg\mathbf{F} \vee \mathbf{A}$
KB3: $\neg\mathbf{E} \vee \mathbf{D}$
KB4.1: $\mathbf{B}$
KB4.2: $\mathbf{C}$
KB5: $\mathbf{F}$

Details of converting KB1 to clause form:

$$(\mathbf{A} \vee \mathbf{B}) \wedge \mathbf{C} \Rightarrow \mathbf{E} \Leftrightarrow \neg [\, (\mathbf{A} \vee \mathbf{B}) \wedge \mathbf{C}\,] \vee \mathbf{E} \Leftrightarrow \neg(\mathbf{A} \vee \mathbf{B}) \vee \neg\mathbf{C} \vee \mathbf{E} \Leftrightarrow (\neg\mathbf{A} \wedge \neg\mathbf{B}) \vee (\neg\mathbf{C} \vee \mathbf{E})$$

Now: using distributivity, we get a conjunction of two clauses: $(\neg\mathbf{A} \vee \neg\mathbf{C} \vee \mathbf{E}) \wedge (\neg\mathbf{B} \vee \neg\mathbf{C} \vee \mathbf{E})$. These can then just be divided into two independent clauses KB1.1 and KB1.2, above.

To perform resolution-refutation we start with $\neg\mathbf{A}$ and applying resolution at each step, try to arrive at the empty clause:

$\neg\mathbf{A}$	resolve with	KB2: $\neg\mathbf{D} \vee \neg\mathbf{F} \vee \mathbf{A}$
$\neg\mathbf{D} \vee \neg\,\mathbf{F}$		KB3: $\neg\mathbf{E} \vee \mathbf{D}$
$\neg\mathbf{F} \vee \neg\mathbf{E}$		KB5: $\mathbf{F}$
$\neg\mathbf{E}$		KB1.2: $\neg\mathbf{B} \vee \neg\mathbf{C} \vee \mathbf{E}$
$\neg\mathbf{B} \vee \neg\mathbf{C}$		KB4.1: $\mathbf{B}$
$\neg\mathbf{C}$		KB4.2: $\mathbf{C}$
☒ (empty clause)		

Since empty clause (contradiction) was reached by starting with $\neg\mathbf{A}$, then we can conclude that **A** entails from the knowledgebase KB.

Again, there are other possible derivations here. Also, some choices of clauses selected in intermediate steps, may not lead to success. I such cases, other possible clauses must be tried until no other options are available.

2. Translation to First-Order Predicate Logic:

Solution:

We will use the following predicates for this problem. Others used are clear from context:

- takes(x, c, t) → student x takes course c during term s
- passes(x, c, t) → student x passes course c during term s
- buys(x, y, z) → x buys y from z
- fools(x, y) → x cooks for y

(a) **Some students took history in Fall 04**: $\exists x$ [student(x) ∧ takes(x, `hist`, `fall04`)]

(b) **Not everyone who took sociology, passed it:**
$\neg\forall x\forall t$ [(student(x) ∧ takes(x, `soc`, t)) => passes(x, `soc`, t)]

Note: this one can be written as existential if negation is moved inside.

(c) **Every student who passes both history and sociology, takes anthropology:**
$\forall x\forall t$ [(student(x) ∧ takes(x, `hist`, t) ∧ takes(x, `soc`, t)) => takes(x, `anthro`, t)]

(d) **Only one student failed (did not pass) both history and sociology**:
$\exists x\exists t$ [student(x) ∧ ¬ passes(x, `soc`, t) ∧ ¬ passes(x, `hist`, t)) ∧
$\forall y\forall t$ ((student(y) ∧ ¬passes(x, `soc`, t) ∧ ¬passes(x, `hist`, t)) => (y = x)]

Note: this one can be written in several other ways.

(e) **There is a person who cooks for all those who do not cook for themselves**:
$\exists x$ [person(x) ∧ $\forall y$ [person(y) ∧ ¬cooks(y, y) => cooks(x, y)]]

(g) **There is a broker who sells stocks to people who do not own any stocks**:
$\exists x$ [broker(x) ∧ $\forall y\forall s$ [person(y) ∧ stock(t) ∧ ¬own(y, s) => ($\exists t$ stock(t) ∧ sells(x, y, t))]]

Note that here we need to distinguish between some stock that is sold (t) and all stocks (s) not owned by y.

(h) **There is a barber who shaves all men in town who do not shave themselves**:
$\forall x$ [investor(x) ∧ $\exists y$ (broker(y) ∧ buys(x, `goog`, y)) => smart(x)]

3. Resolution-Refutation Problem:

Solution:

(a) The knowledge base in first order logic:

$\forall x$ food(x) => likes(`john`, x)
food(`apple`)
food(`chicken`)
$\forall x\ \forall y$ [eats(x, y) $\wedge$ $\neg$killed-by(x, y) => food(y)]
eats(`bill`, `peanuts`) $\wedge$ alive(`bill`)
$\forall x\ \forall y$ [killed-by(x, y) => ~alive(x)]
$\forall x$ eats(`bill`, x) => eats(`sue`, x)

Note: This can also be written as:

$\forall x$ [($\exists y$ killed-by(x, y)) => ~alive(x)]

The translation into clause form will still be the same as below.

(b) Translation to clausal form

1. $\neg$food(x) $\vee$ likes(`john`, x)
2. food(`apple`)
3. food(`chicken`)
4. $\neg$eats(x, y) $\vee$ killed-by(x, y) $\vee$ food(y)

5a. eats(`bill`, `peanuts`)
5b. alive(`bill`)

6. $\neg$killed-by(x,y) $\vee$ $\neg$alive(x)
7. $\neg$eats(`bill`, x) $\vee$ eats(`sue`, x)

(c) Prove that `john` likes `peanuts`:

resolve with	**subgoal (resolvant)**
	$\neg$likes(`john`, `peanuts`)
(1)	$\neg$food(`peanuts`)
(4)	$\neg$eats(x, `peanuts`) $\vee$ killed-by(x, `peanuts`)
(5a)	killed-by(`bill`, `peanuts`)
(6)	$\neg$alive(`bill`)
(5b)	⊠ (empty clause)

Since we arrived at the empty clause, the original goal likes(`john`, `peanuts`) is proved.

(d) What food does sue eat? (i.e., $\exists x$ food(x) $\wedge$ eats(sue, x)?). Negating this goal results in: $\neg$ ($\exists x$ food(x) $\wedge$ eats(sue, x)). Conversion to clause form results in: $\neg$food(x) $\vee$ $\neg$eats(sue, x) . This will be the starting goal for resolution-refutation.

resolve with	**subgoal (resolvant)**	
	$\neg$food(x) $\vee$ $\neg$eats(sue, x)	
(7)	$\neg$eats(bill, x) $\vee$ $\neg$food(x)	
(5a)	$\neg$food(peanuts)	(with substitution { x = peanuts }

After this point the rest of the derivation becomes exactly the same as part (c) above (after the 2nd resolution step):

(e) The new axioms, replacing eats(bill, peanuts) $\wedge$ alive(bill), are as follows:

$\forall x$ ($\neg\exists y$ eats(x, y)) => dead(x) (if people don't eat anything then they'll die)
$\forall x$ dead(x) => $\neg$alive(x)
alive(bill)

Translating the first statement above to clause form will be done as follows:

$\forall x$ ($\neg\exists y$ eats(x, y)) => dead(x) $\Leftrightarrow$ $\forall x$ $\neg$ ($\neg\exists y$ eats(x, y)) $\vee$ dead(x)
$\Leftrightarrow$ $\forall x$ $\exists y$ eats(x, y) $\vee$ dead(x)
$\Leftrightarrow$ $\forall x$ eats(x, sk1(x)) $\vee$ dead(x)
$\Leftrightarrow$ eats(x, sk(x)) $\vee$ dead(x)

Note that the elimination of the existential quantifier (within the scope of $\forall x$) has resulted in the creation of a Skolem function (of x), sk(x), representing a specific but unknown object.

Translating to everything into clausal form, the full revised knowledge base becomes:

1. $\neg$food(x) $\vee$ likes(john, x)
2. food(apple)
3. food(chicken)
4. $\neg$eats(x, y) $\vee$ killed-by(x, y) $\vee$ food(y)
5. $\neg$killed-by(x, y) $\vee$ $\neg$alive(x)

6a. eats(x, sk(x)) $\vee$ dead(x) (where sk is a Skolem function)
6b. alive(bill)
6c. $\neg$dead(x) $\vee$ $\neg$alive(x)

7. $\neg$killed-by(x,y) $\vee$ $\neg$alive(x)
8. $\neg$eats(bill, x) $\vee$ eats(sue, x)

Now, we use resolution to again (as in part d) ask what `sue` eats (this time we don't know that `bill` eats peanuts):

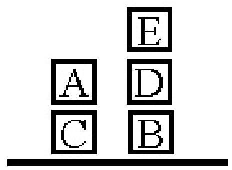
What food does `sue` eat? (i.e., $\exists z$ food(z) $\wedge$ eats(`sue`, z)?). Negating this goal results in:
$\neg$ ($\exists z$ food(z) $\wedge$ eats(sue, z)). Conversion to clause form results in: $\neg$food(z) $\vee$ $\neg$eats(`sue`, z) . [**Note:** this time variable name z was used instead of x, just to distinguish different occurrence of variables in the proof below].

resolve with	**subgoal (resolvant)**
	$\neg$eats(`sue`, z) $\vee$ $\neg$food(z)
(8)	$\neg$eats(`bill`,z) $\vee$ $\neg$food(z)
(6a)	$\neg$food(`sk`(`bill`)) $\vee$ dead(`bill`) (with z = `sk`(`bill`))
(6c)	$\neg$food(`sk`(`bill`)) $\vee$ $\neg$alive(`bill`)
(6b)	$\neg$food(`sk`(`bill`))
(4)	$\neg$eats(x, `sk`(`bill`)) $\vee$ killed-by(x, `sk`(`bill`))
(5)	$\neg$eats(x, `sk`(`bill`)) $\vee$ $\neg$alive(z)
(6b)	$\neg$eats(`bill`, `sk`(`bill`))
(6a)	dead(`bill`)
(6c)	$\neg$alive(`bill`)
(6b)	☒ (empty clause)

3. Backward Chaining and AND/OR Graphs:

Solution:

The knowledge base and the query to be answered are depicted below:



Query: $\exists w$ above(w,B)?

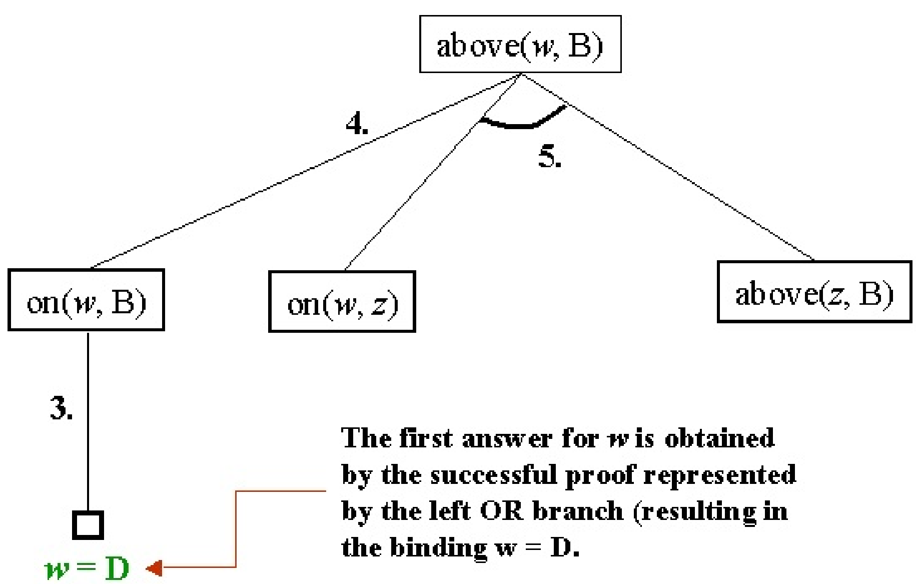
KB for Blocks World Problem

```
1. on(A,C)
2. on(D,B)
3. on(E,D)
4. on(x,y) => above(x,y)
5. on(x,z) ∧ above(z,y) => above(x,y)
```

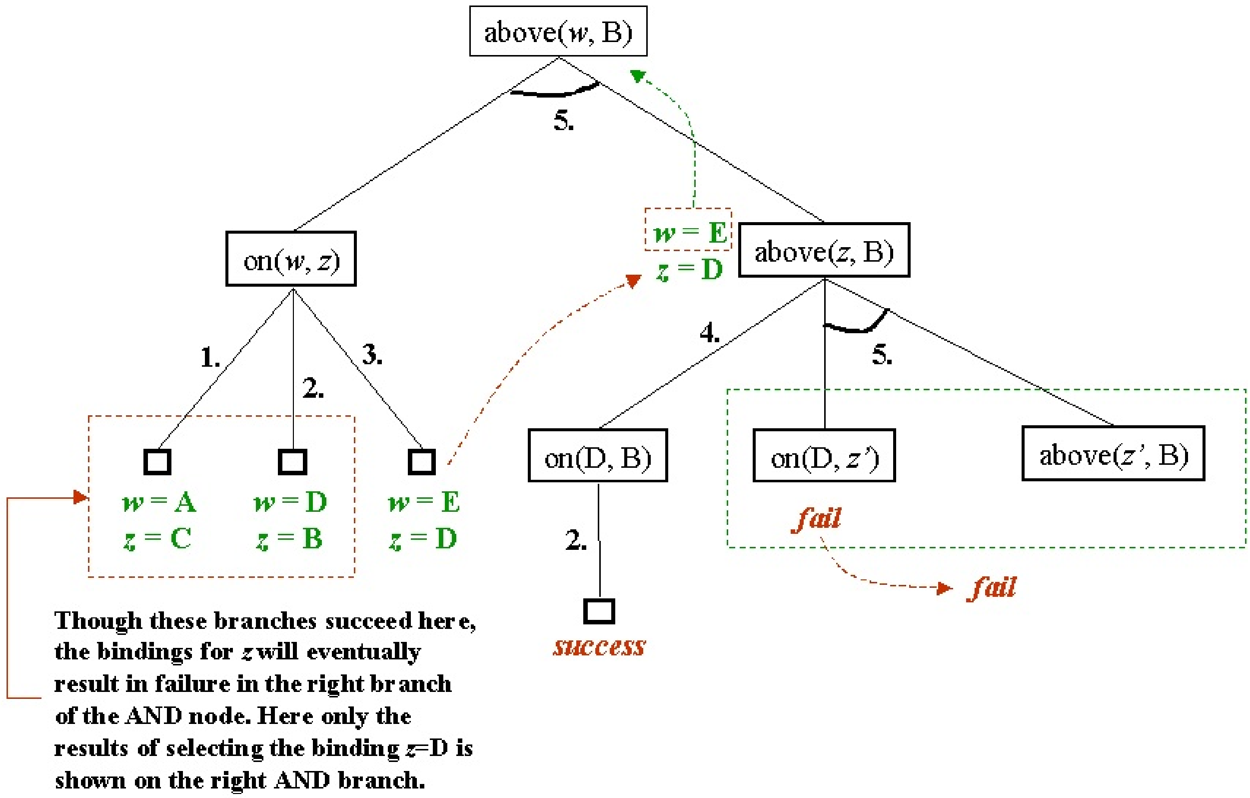
In effect, the query is asking the system to provide answers (binding for the variable *w*) to the question: "which blocks are above the block B?"

We show the solution in stages. The query above(*w*,B) can unify with the head of two rules in the KB (4, and 5). These represent alternative (OR) branches in the proof tree. Success along either of these alternatives will result in an answer for the query (a constructive proof). In the case of rule 5, we get an AND branch. This means that success along this branch will require success along both subtrees corresponding to the two conjuncts on the left-hand-side of rule 5.

The successful proof along the left OR branch is shown below:



The success along the right branch of the OR node (which itself is and AND branch) is shown below:



Note that the left subtree of the AND branch (with root on(*w*,*z*)) can unify with KB facts 1, 2, and 3, each resulting in a successful proof of this node with its own bindings for variables *w* and *z*. Selecting any of these successful branches will result in the corresponding bindings to propagate to the right subtree of the AND node. In this case, selecting the branches corresponding to the substitutions {*w*=A,*z*=C} and {*w*=D,*z*=B} will eventually result in failure in the right subtree of the AND node (not shown here). In the above figure, we have only shown the right subtree given the selection of the substitution {*w*=E,*z*=D} in the left subtree.