

# Introduction

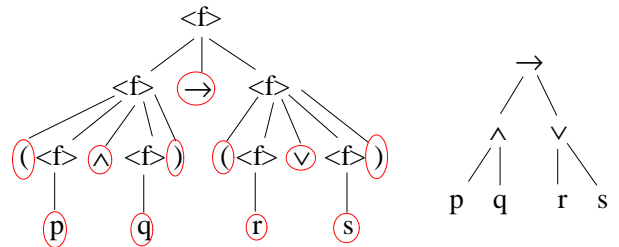
We do *propositional logic* for 6 weeks (*truth tables, semantic tableau, resolution*), and then do *predicate logic* (e.g.  $\exists x (p(x) \wedge q(x))$ ) and a little *Prolog*. **Four** applications of the course: I: Software correctness — formal *methods* and *logic*. (a) *Program verification*: program  $\rightarrow$  an equivalent logic form in which we can prove properties.

**Example:** if X then emit A else emit B goes to  $(X \wedge A) \vee (\neg X \wedge B)$ . (b) *Program specification*: we specify a program using logic. **Example:** BUTTON  $\rightarrow$  (LIGHT\_A  $\wedge$  LIGHT\_B) goes to if BUTTON then emit LIGHT\_A; emit LIGHT\_B;. II: Prolog and Deductive Databases (*facts, rules, ?questions*). III: Circuit Design (*specify using logic; reason; simplify*). IV: Automated Theorem Proving.

## Propositional Logic

(1)  $\langle \text{formula} \rangle ::= p$ , where  $p \in P$ , and  $p$  is an **atom** while  $P$  is the set of **propositional letters**. (2)  $\langle \text{formula} \rangle ::= \neg(\langle f \rangle)$ . (3)  $\langle f \rangle ::= (\langle f \rangle) \vee (\langle f \rangle)$ . (4)  $\langle f \rangle ::= (\langle f \rangle) \wedge (\langle f \rangle)$ . (5)  $\langle f \rangle ::= \langle f \rangle \rightarrow \langle f \rangle$ . (6)  $\langle f \rangle ::= \langle f \rangle \leftrightarrow \langle f \rangle$ . **Operators:**  $\neg$  (not),  $\vee$  (or/disjoint),  $\wedge$  (and/conjunct),  $\rightarrow$  (implies),  $\leftrightarrow$  (equivalence).  $F$  is the *set of formulae* that can be **derived** from this grammar. Examples:  $p, \neg p, p \wedge q, p \vee q, \dots$

Consider the *expression*  $(p \wedge q) \rightarrow (r \vee s)$ . Derivation: (1)  $\langle f \rangle$ . (2)  $\langle f \rangle \rightarrow \langle f \rangle$  by rule 5. (3)  $(\langle f \rangle \wedge \langle f \rangle) \rightarrow \langle f \rangle$  by rule 4. (4)  $(p \wedge q) \rightarrow \langle f \rangle$  by rule 1. (5)  $(p \wedge q) \rightarrow (\langle f \rangle \vee \langle f \rangle)$  by rule 3. (6)  $(p \wedge q) \rightarrow (r \vee s)$ . The *derivation tree* and the *formation tree* are as shown on the right. Note that in the formation tree, we replace  $\langle f \rangle$  by its child that is an **operator** or an **atom**. We then do *In Order* traversal of the tree: visit LEFT subtree, visit root, visit RIGHT subtree.



8th February 2002

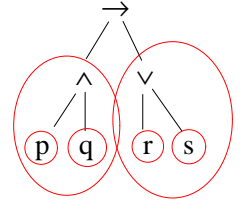
**Definition:** A *truth assignment* is a function  $v: P \rightarrow \{T, F\}$ , i.e.  $v$  assigns one of the truth values T or F to every *atom*, e.g.  $v(p) = T$ , or  $v(q) = F$ . **Definition:** The assignment  $v$  can be extended to a *function*  $v: F \rightarrow \{T, F\}$  mapping **formulae** to **truth values** by the inductive definition in the table shown on the right. Note that  $v$  is an *interpretation*.

<b>A</b>	<b>v(A<sub>1</sub>)</b>	<b>v(A<sub>2</sub>)</b>	<b>v(A)</b>
$\neg A_1$	T		F
$\neg A_1$	F		T
$A_1 \vee A_2$	F	F	F
$A_1 \vee A_2$	otherwise		T
$A_1 \wedge A_2$	T	T	T
$A_1 \wedge A_2$	otherwise		F
$A_1 \rightarrow A_2$	T	F	F
$A_1 \rightarrow A_2$	otherwise		T
$A_1 \leftrightarrow A_2$	$v(A_1) = v(A_2)$		T
$A_1 \leftrightarrow A_2$	$v(A_1) \neq v(A_2)$		F

**Definition:** Let  $A_1, A_2 \in F$ . If  $v(A_1) = v(A_2)$  for all interpretations  $v$ , then  $A_1$  is *logically equivalent* to  $A_2$ , written  $A_1 \equiv A_2$ . Q: Is  $p \wedge (q \vee r) \equiv (p \wedge q) \vee r$ ? A: If  $v(p) = F, v(q) = T$  and  $v(r) = T$ , then  $v(p \wedge (q \vee r)) = F \wedge (T \vee T) = F \wedge T = F$ , but  $v((p \wedge q) \vee r) = (F \wedge T) \vee T = F \vee T = T$ .

**Theorem 1.1:**  $A_1 \equiv A_2$  iff  $A_1 \leftrightarrow A_2$  in every interpretation. Note: ‘ $\equiv$ ’ is *English*, while ‘ $\leftrightarrow$ ’ is a *boolean operator*. **Proof** ( $\Rightarrow$ ): if  $A_1 \equiv A_2$  then  $A_1 \leftrightarrow A_2$ . Suppose that  $A_1 \equiv A_2$  and that  $v$  is an *arbitrary interpretation*. By the definition of **logical equivalence** ( $\equiv$ ),  $v(A_1) = v(A_2)$ . By the table on the previous page, then  $v(A_1 \leftrightarrow A_2) = T$ . Note that  $v$  was *arbitrary*, so that  $v(A_1 \leftrightarrow A_2) = T$  in **all** interpretations.

**Definition:**  $A$  is a *subformula* of  $B$  if the **formation tree** for  $A$  occurs as a subtree of the formation tree for  $B$ .  $A$  is a *proper subformula* of  $B$  if  $A$  is not **identical** to  $B$ . As an example, consider the formation tree of  $(p \wedge q) \rightarrow (r \vee s)$  as shown on the right. The **red** circles show *potential* (proper) subformulas.



**Definition:** If  $A$  is a *subformula* of  $B$ , and if  $A'$  is *any* formula, then  $B'$ , the *substitution* of  $A'$  for  $A$  in  $B$ , denoted by  $B\{A \leftarrow A'\}$ , is the formula obtained by replacing **all** occurrences of the subtree for  $A$  in  $B$  by the tree for  $A'$ . **Example:**  $B = (p \rightarrow q) \leftrightarrow (\neg p \rightarrow \neg q)$ ,  $A = p \rightarrow q$ ,  $A' = \neg p \vee q$ :  $B' = B\{A \leftarrow A'\} = (\neg p \vee \neg q) \leftrightarrow (\neg p \rightarrow \neg q)$ .

**Theorem 1.2:** Let  $A$  be a *subformula* of  $B$ , and let  $A'$  be a formula such that  $A \equiv A'$ . Then  $B \equiv B\{A \leftarrow A'\}$ . **Example:**  $B = (p \vee q) \wedge (s \vee t)$ ,  $A = p \vee q$ ,  $A' = q \vee p$ :  $B' = B\{A \leftarrow A'\} = (q \vee p) \wedge (s \vee t)$ , with  $B \equiv B'$ . Now consider the *logical equivalences* below, where ‘*false*’ is a **shorthand** for  $p \wedge \neg p$ , and ‘*true*’ is a **shorthand** for  $p \vee \neg p$ . Further,  $A \downarrow B$  stands for *nor*,  $\neg(A \vee B)$ ;  $A \uparrow B$  stands for *nand*,  $\neg(A \wedge B)$ ; and  $A \oplus B$  stands for *XOR*,  $(A \wedge \neg B) \vee (\neg A \wedge B)$ .

$A \vee \text{true} \equiv \text{true}$	$A \wedge \text{true} \equiv A$	$A \equiv \neg \neg A$		$A \vee B \equiv B \vee A$	$A \wedge B \equiv B \wedge A$
$A \vee \text{false} \equiv A$	$A \wedge \text{false} \equiv \text{false}$	$A \equiv A \wedge A$	$A \equiv A \vee A$	$A \leftrightarrow B \equiv B \leftrightarrow A$	$A \oplus B \equiv B \oplus A$
$A \rightarrow \text{true} \equiv \text{true}$	$\text{true} \rightarrow A \equiv A$	$A \vee \neg A \equiv \text{true}$	$A \wedge \neg A \equiv \text{false}$ (4)	$A \uparrow B \equiv B \uparrow A$	$A \downarrow B \equiv B \downarrow A$
$A \rightarrow \text{false} \equiv \neg A$	$\text{false} \rightarrow A \equiv \text{true}$	$A \rightarrow A \equiv \text{true}$		$A \rightarrow B \equiv \neg B \rightarrow \neg A$	
$A \leftrightarrow \text{true} \equiv A$	$A \oplus \text{true} \equiv \neg A$	$A \leftrightarrow A \equiv \text{true}$	$A \oplus A \equiv \text{false}$		
$A \leftrightarrow \text{false} \equiv \neg A$	$A \oplus \text{false} \equiv A$	$\neg A \equiv A \uparrow A$	$\neg A \equiv A \downarrow A$		
$A \vee (B \vee C) \equiv (A \vee B) \vee C$	$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$	$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$	$A \oplus B \equiv \neg(A \rightarrow B) \vee (B \rightarrow A)$		
$A \leftrightarrow (B \leftrightarrow C) \equiv (A \leftrightarrow B) \leftrightarrow C$	$A \oplus (B \oplus C) \equiv (A \oplus B) \oplus C$	$A \rightarrow B \equiv \neg A \vee B$ (3)	$A \rightarrow B \equiv \neg(A \wedge \neg B)$		
$A \uparrow (B \uparrow C) \equiv (A \uparrow B) \uparrow C$	$A \downarrow (B \downarrow C) \equiv (A \downarrow B) \downarrow C$	$A \vee B \equiv \neg(\neg A \wedge \neg B)$	$A \wedge B \equiv \neg(\neg A \vee \neg B)$		
		(de Morgan)	(de Morgan)		
$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ (2)	$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ (1)	$A \vee B \equiv \neg A \rightarrow B$	$A \wedge B \equiv \neg(A \rightarrow \neg B)$		
$A \wedge (A \vee B) \equiv A$	$A \vee (A \wedge B) \equiv A$	$A \rightarrow B \equiv A \leftrightarrow (A \wedge B)$	$A \rightarrow B \equiv B \leftrightarrow (A \vee B)$		
		$A \wedge B \equiv (A \leftrightarrow B) \leftrightarrow (A \vee B)$	$A \leftrightarrow B \equiv (A \vee B) \rightarrow (A \wedge B)$		

**Examples:** (1)  $p \wedge (\neg p \vee q) \equiv (p \wedge \neg p) \vee (p \wedge q)$  (by (1))  $\equiv (\text{false}) \vee (p \wedge q)$  (by (4))  $\equiv p \wedge q$ . (2)  $p \vee (\neg p \wedge (q \vee p)) \equiv p \vee ((\neg p \wedge q) \vee (\neg p \wedge p))$  (by (1))  $\equiv p \vee (\neg p \wedge q) \equiv (p \vee \neg p) \wedge (p \vee q) \equiv p \vee q$ . (3) Consider the *diagram* shown on the right. We could have the **code** ‘if  $X$  then  $\text{Light\_on}$ ; if  $Y$  then  $\text{Light\_off}$ ;’, or the **logic**  $(X \rightarrow \text{Light}) \wedge (Y \rightarrow \neg \text{Light})$ . Now  $(\neg X \vee \text{Light}) \wedge (\neg Y \vee \neg \text{Light}) \wedge X \wedge Y$  (by (3))  $\equiv ((\neg X \vee \text{Light}) \wedge \neg Y) \vee ((\neg X \vee \text{Light}) \wedge \neg \text{Light}) \wedge X \wedge Y$  (by (1))  $\equiv ((\neg X \wedge \neg Y) \vee (\text{Light} \wedge \neg Y)) \vee ((\neg X \wedge \neg \text{Light}) \vee (\text{Light} \wedge \neg \text{Light}) \wedge X \wedge Y) \equiv [(\neg X \wedge \neg Y) \vee (\text{Light} \wedge \neg Y)] \wedge X \wedge Y \vee [(\neg X \wedge \neg \text{Light}) \vee (\text{Light} \wedge \neg \text{Light})] \wedge X \wedge Y \equiv [(\neg X \wedge \neg Y) \wedge X \wedge Y] \vee [(\text{Light} \wedge \neg Y) \wedge X \wedge Y] \equiv \text{false}$ .



**Definition:** A propositional formula  $A$  is *satisfiable* iff  $v(A) = T$  for some interpretation  $v$ . A **satisfying** interpretation is called a *model* for  $A$ .  $A$  is *unsatisfiable/contradictory* iff it is not satisfiable, i.e.  $v(A) = F$  for all  $v$ . Q: Is  $p \wedge q \wedge (\neg p \vee \neg q)$  satisfiable? A: No — e.g. with  $v(p) = T$  and  $v(q) = T$ , we have  $T \wedge T \wedge (F \vee F) = F$ .

**Definition:**  $A$  is *valid*, written  $\models A$  (' $\models$ ' = 'double turnstile'), iff  $v(A) = T$  for all  $v$  ( $A$  is a *tautology*).  $A$  is not valid, or falsifiable,  $\not\models A$ , iff it is **not** valid, i.e.  $v(A) = F$  for some  $v$ . Q: Is  $p \wedge (q \vee r)$  valid? A: No — to see this, set  $v(p) = F$  and  $v(q) = v(r) = T$ .

**Theorem 1.3:** (a)  $A$  is *valid* iff  $\neg A$  is *unsatisfiable* (valid:  $v(A) = T$  for all  $v$ ; unsatisfiable:  $v(\neg A) = F$  for all  $v$ ). **Proof.** ( $\Rightarrow$ ) If  $A$  is *valid* then  $\neg A$  is *unsatisfiable*. Consider an **arbitrary** interpretation  $v$ .  $v(A) = T$  iff  $v(\neg A) = F$ .  $A$  is valid ( $A$  is true in *all* interpretations) iff  $\neg A$  is false in all interpretations — so  $\neg A$  is *unsatisfiable*.

( $\Leftarrow$ ) If  $\neg A$  is *unsatisfiable*, then  $A$  is *valid*. If  $\neg A$  is *unsatisfiable*, then  $v(\neg A) = F$  for an **arbitrary**  $v$ . Further,  $v(\neg A) = F$  iff  $v(A) = T$ . Since  $v$  is *arbitrary*, then  $\neg A$  is false in **all** interpretations iff  $A$  is true in **all** interpretations, i.e.  $A$  is valid. **End of Proof.** (b)  $A$  is *satisfiable* iff  $\neg A$  is *falsifiable*.

**Definition:** Let  $U$  be a *set* of formulae. An **algorithm** is a decision procedure (DP) for  $U$  if given an arbitrary formula  $A \in F$ , it *terminates* and returns the answer '**yes**' if  $A \in U$ , and returns '**no**' if  $A \notin U$ . Truth tables as DP's: *satisfiability* (a 'T' in the RHS), *validity* (**all** 'T' in the RHS), and *inefficiency*:  $n$  atoms  $\Rightarrow 2^n$  rows in the truth table.

Q: Is  $A$  valid? A: Apply DP for *satisfiability* to  $\neg A$ . **Refutation:** If DP reports  $\neg A$  is satisfiable, then  $A$  is *not* valid — and if DP reports  $\neg A$  is **not** satisfiable, then  $A$  is valid. **Definition:** A set of formulae  $U = \{A_1, \dots, A_n\}$  is (*simultaneously*) satisfiable iff there exists an interpretation  $v$  such that  $v(A_1) = \dots = v(A_n) = T$ . The *satisfying interpretation* is called a **model** of  $U$ .

$U$  is *unsatisfiable* iff for every interpretation  $v$  there exists an  $i$  such that  $v(A_i) = F$ . **Example:**  $U = \{p, \neg p \vee q, q \wedge r\}$ . Now  $v(p) = T$ ,  $v(q) = T$  and  $v(r) = T$  satisfies **every** formula — so  $U$  is satisfiable. **Example:**  $U = \{p, \neg p \vee q, q \wedge r, \neg p\}$ . **Not** satisfiable:  $v(p) = T$  means that  $v(\neg p) = F$ , and  $v(\neg p) = T$  means that  $v(p) = F$ .

**Theorem 1.4:** Let  $U = \{A_1, \dots, A_n\}$ . (a) If  $U$  is *satisfiable*, then so is  $U - \{A_i\}$  for *any*  $1 \leq i \leq n$ . (b) If  $U$  is *satisfiable* and if  $B$  is *valid*, then  $U \cup \{B\}$  is satisfiable. (c) If  $U$  is *unsatisfiable*, then for **any** formula  $B$ ,  $U \cup \{B\}$  is *also* unsatisfiable. (d) If  $U$  is *unsatisfiable*, and for some  $1 \leq i \leq n$   $A_i$  is valid, then  $U - \{A_i\}$  is *unsatisfiable*.

**Definition:** Let  $U$  be a set of *formulae*  $\{A_1, \dots, A_n\}$ , and let  $A$  be an arbitrary formula.  $A$  is a **logical consequence** of  $U$  if every truth assignment  $v$  making  $U$  true *also* makes  $A$  true. Notation:  $A_1, \dots, A_n \models A$ .

Consider  $U = \{p, \neg q\}$ , and let  $A = (p \vee r) \wedge (\neg q \vee \neg r)$ . Is  $U \models A$ ? Now  $v(p) = T$  and  $v(q) = F \Rightarrow A = (T \vee r) \wedge (T \vee \neg r) = T \wedge T = T$ , so that  $U \models A$ . Now consider  $U = \{p, \neg q\}$ , and let  $B = \neg p \wedge r$ . Is  $U \models B$ ? Now  $v(p) = T$  and  $v(q) = F \Rightarrow B = F \wedge r = F$ , so that  $U \not\models B$ . **Theorem 1.5:**  $U \models A$  iff  $\models A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow A$  ( $\models = \text{valid}$ ).

13th February 2002

## Tutorial 1

(1) Are the following formulae *true* or *false* under the assignment  $v(p) = T, v(q) = T, v(r) = T, v(s) = F, v(t) = F$  and  $v(w) = T$ ? (a)  $p \wedge q$ ; (b)  $\neg(p \vee q)$ ; (c)  $(p \wedge \neg q) \vee (r \wedge \neg s)$ ; (d)  $(p \wedge \neg q) \vee (r \wedge \neg s) \rightarrow (t \vee s \vee q)$ ; and (e)  $(p \wedge \neg q) \vee (r \wedge \neg s) \leftrightarrow (t \vee \neg w)$ . A: (a) T; (b) F; (c) T; (d)  $F \vee T \rightarrow T$  gives T; (e)  $T \leftrightarrow F$  gives F.

(2) For each of the following formulae, define *one assignment* that makes the formula **true**, and *one assignment* that makes the formula **false**: (a)  $p \vee q$ ; (b)  $(p \rightarrow (s \vee r))$ ; (c)  $(p \rightarrow (s \vee r)) \vee (q \leftrightarrow ((t \vee s) \wedge r))$ ; and (d)  $(p \leftrightarrow q) \vee \neg((p \rightarrow q) \wedge (q \rightarrow p))$ . A: (a) True:  $v(p) = T, v(q) = F$ ; False:  $v(p) = F, v(q) = F$ . (b) True:  $v(p) = v(s) = v(r) = T$ ; False:  $v(p) = T, v(s) = v(r) = F$ . (c) True:  $v(p) = v(s) = v(r) = T, v(q) = v(t) = F$ ; False:  $v(p) = T, v(s) = v(r) = F, v(q) = F, v(t) = F$ . (d) True:  $v(p) = v(q) = T$ ; False: *Not Possible* (the formula is a *tautology*).

(3) Are the following formulae *logically equivalent*? Either find an interpretation which proves that they are **not** logically equivalent, or show that they evaluate to the same truth value for **all** possible interpretations. (a)  $A \vee B \equiv \neg(\neg A \wedge B)$ ; (b)  $(A \vee (B \wedge C)) \equiv (A \vee B) \wedge (B \vee C)$ ; and (c)  $A \rightarrow B \equiv \neg A \vee B$ . A: (a) Show using a *truth table* that LHS = RHS in all cases. (b) With  $v(A) = T, v(B) = F$  and  $v(C) = F$ , we have **LHS** = T, but **RHS** =  $T \wedge F = F$  — so *not* logically equivalent. (c) As with (a), use a truth table to show that logical equivalence *holds*.

(4) Use *logical equivalences* and *substitution* to simplify the following formulae as much as possible: (a)  $(p \vee (q \wedge \neg p))$ ; (b)  $(p \wedge (q \vee \neg p))$ ; and (c)  $(p \rightarrow q) \wedge (\neg q \rightarrow r) \wedge \neg r$ . A: (a)  $(p \vee (q \wedge \neg p)) \equiv (p \vee q) \wedge (p \vee \neg p) \equiv (p \vee q) \wedge T \equiv p \vee q$ . (b)  $(p \wedge (q \vee \neg p)) \equiv (p \wedge q) \vee (p \wedge \neg p) \equiv p \wedge q$ . (c)  $(p \rightarrow q) \wedge (\neg q \rightarrow r) \wedge \neg r \equiv (\neg p \vee q) \wedge (q \vee r) \wedge \neg r \equiv (\neg p \vee q) \wedge ((q \wedge \neg r) \vee (r \wedge \neg r)) \equiv (\neg p \vee q) \wedge (q \wedge \neg r) \equiv (\neg p \wedge q \wedge \neg r) \vee (q \wedge q \wedge \neg r) \equiv (\neg p \wedge q \wedge \neg r) \vee (q \wedge \neg r)$ .

(5) Prove the **validity** of each formula D below by showing whether or not  $\neg D$  is *unsatisfiable*: (a)  $(A \rightarrow B) \vee \neg(\neg A \vee B)$ ; and (b)  $(A \rightarrow B) \vee (\neg A \vee B)$ . A: (a)  $\neg D = \neg((A \rightarrow B) \vee \neg(\neg A \vee B)) \equiv \neg((A \rightarrow B) \vee (A \wedge \neg B))$ . If we consider *all the different choices* for  $v(A)$  and for  $v(B)$ , then  $v(\neg D) = F$  in all cases, meaning that D is valid, e.g. for  $v(A) = v(B) = T$ , we have  $\neg D = \neg(T \vee ?) \equiv \neg T \equiv F$ . (b)  $\neg D = \neg((A \rightarrow B) \vee (\neg A \vee B))$ . Here, if we take  $v(A) = T$  and  $v(B) = F$ , then  $v(\neg D) = \neg(F \vee (F \vee F)) \equiv \neg F = T$ . Conclusion: D is not valid.

(6) With regards to the formulae of question 2, (a) which of these formulae are *satisfiable*, and (b) which of these formulae are *valid*? A: **All** are satisfiable; **only** (d) is valid. (7) Which of the following sets of formulae are *satisfiable*: (a)  $\{p \wedge q, p \wedge \neg r\}$ ; (b)  $\{p, p \vee q, r \vee \neg q, \neg r\}$ ; and (c)  $\{p, p \wedge q, \neg p \vee \neg q\}$ . A: (a)  $v(p) = v(q) = T, v(r) = F$  satisfies *all* the formulae. (b)  $v(p) = T, v(r) = v(q) = F$  satisfies *all* the formulae. (c) **Not** satisfiable.

(8) Which of the following are *logical consequences* of  $\{p \wedge q, p \wedge \neg r\}$ : (a)  $p \vee q$ ; (b)  $p \wedge r$ ; (c)  $p \vee s$ ; and (d)  $p \wedge s$ . A: **Let**  $U = \{p \wedge q, p \wedge \neg r\}$ .  $v(p) = T$ ,  $v(r) = F$  and  $v(q) = T$  is the only assignment that satisfies  $U$ . (a) If  $v(p) = T$ ,  $v(r) = F$  and  $v(q) = T$ , then  $p \vee q = T$ , OK. (b) If  $v(p) = T$ ,  $v(r) = F$  and  $v(q) = T$ , then  $p \wedge r = F$ , so **not** a logical consequence. (c) If  $v(p) = T$ ,  $v(r) = F$  and  $v(q) = T$ , then  $p \vee s = T$  for all  $s$ , so OK. (d) If  $v(p) = T$ ,  $v(r) = F$  and  $v(q) = T$ , then  $p \wedge s = F$  if  $v(s) = F$ , so **not** a logical consequence.

(9) The following are all true statements concerning a *murder mystery*. Convert to **symbolic** form and determine who murdered Lord H. (a) Lord H, the murdered man, was killed by a blow to the head with a *brass candlestick*. (b) Either *Lady H* or the *maid* was in the dining room at the time of the murder.

(c) If the *cook* was in the kitchen at the time of the murder, then the *butler* killed Lord H with the poison. (d) If Lady H was in the dining room at the time of the murder, then the *chauffeur* killed Lord H. (e) If the cook was *not* in the kitchen at the time of the murder, then the maid was *not* in the dining room when the murder was committed. (f) If the *maid* was in the dining room at the time the murder was committed, then the *wine steward* killed Lord H.

A: Information gathered from the text above (by *converting* each statement into a propositional logic formula): (a)  $p$  ( $p$  = killed with *candlestick*); (b)  $q \vee r$  ( $q$  = *Lady H* in dining room;  $r$  = *maid* in dining room); (c)  $s \rightarrow t$  ( $s$  = cook in kitchen;  $t$  = *butler* kills with *poison*); (d)  $q \rightarrow u$  ( $u$  = *chauffeur* kills); (e)  $\neg s \rightarrow \neg r$ ; (f)  $r \rightarrow v$  ( $v$  = *wine steward* kills). More information: there was **one** method of murder so that  $\neg(p \wedge t)$ , and there was only **one** murderer so that  $\neg(t \wedge u) \wedge \neg(t \wedge v) \wedge \neg(u \wedge v)$ .

If  $U = \{p, q \vee r, s \rightarrow t, q \rightarrow u, \neg s \rightarrow \neg r, r \rightarrow v, \neg(p \wedge t), \neg(t \wedge u) \wedge \neg(t \wedge v) \wedge \neg(u \wedge v)\}$ , then we ask whether  $U \models t$ ,  $U \models u$ , or  $U \models v$ . Recall that  $U \models B$  iff  $A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow B$  is valid. Firstly, we ask whether  $U \models t$ ? Now  $A_1 \wedge \dots \wedge A_n \rightarrow B$  is only *false* if we have  $T \rightarrow F$ . So we set  $v(t) = F$  and **try** to satisfy  $A_1 \wedge \dots \wedge A_n$ . Looking at the *equations*, we see that  $v(p) = T$ ,  $v(s) = F$ ,  $v(r) = F$ ,  $v(q) = T$ ,  $v(u) = T$  and  $v(v) = F$  satisfies *all* equations, so the **Butler** did not commit the murder.

Similarly, if we ask whether  $U \models v$ , then we find that this *cannot* happen if we set  $v(v) = F$  and let  $v(p) = T$ ,  $v(r) = F$ ,  $v(t) = F$ ,  $v(s) = F$ ,  $v(r) = F$ ,  $v(q) = T$  and  $v(u) = T$  to satisfy  $A_1 \wedge \dots \wedge A_n$ . Finally, to find out whether we have  $U \models u$ , we set  $v(u) = F$  and try to satisfy  $A_1 \wedge \dots \wedge A_n$ . But we *cannot* do this (there is a **contradiction**), so indeed we have shown that  $U \models u$  — and that the **chauffeur** killed Lord H.

15th February 2002

*Alternative solution: manipulate*  $A_1 \wedge \dots \wedge A_n$ :  $p \wedge (s \rightarrow t) \wedge (\neg s \rightarrow \neg r) \wedge (q \vee r) \wedge q \rightarrow u \wedge r \rightarrow v \wedge$  (extra) (where (extra) denotes the *extra rules* we deduce (e.g. only *one* murderer))  $\equiv p \wedge (\neg s \vee t) \wedge (s \vee \neg r) \wedge (q \vee r) \wedge (\neg q \vee u) \wedge (\neg r \vee v) \wedge$  (extra)  $\equiv ((p \wedge \neg s) \vee (p \wedge t)) \wedge (s \vee \neg r) \wedge (q \vee r) \wedge (\neg q \vee u) \wedge (\neg r \vee v) \wedge$  (extra)  $\equiv \dots \equiv p \wedge \neg s \wedge \neg r \wedge q \wedge u \wedge (\neg r \vee v) \wedge$  (extra)  $\rightarrow u$ .

# Binary Decision Diagrams

A BDD for a formula  $A$  is a *rooted directed binary acyclic graph*. Each **nonterminal** is labelled with an *atom*, and each **leaf** is labelled with a *truth value*. No atom appears more than once in a path from the root to a leaf. One *outgoing edge* is the **false** edge and is denoted by a **dotted** line; the *other* outgoing edge is the **true** edge and is denoted by a **solid** line.

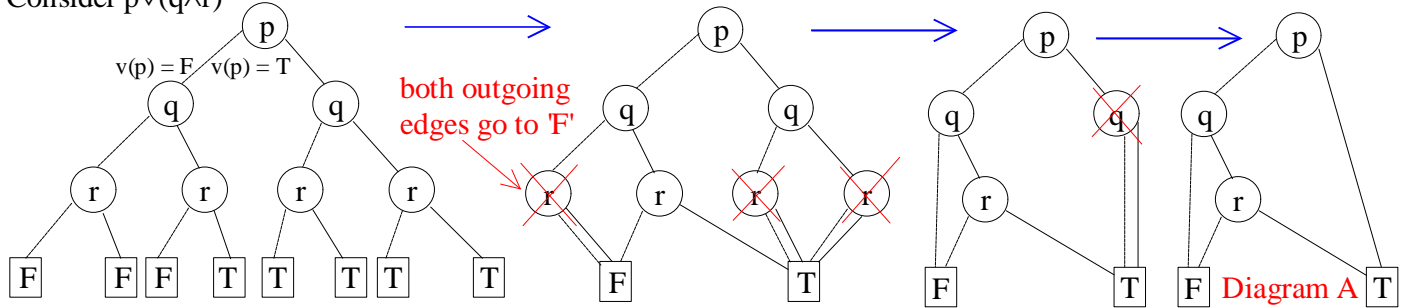
With each path from the root to a leaf is associated an *assignment* to the atoms of  $A$ . Assign F to the atom if the *false* edge is taken from the node labelled by the atom, and assign T if the *true* edge is taken. The leaf gives the value of  $A$  under this assignment. The path need not include **all** the atoms in  $A$ , but it must include assignments to **enough** atoms to enable the value of  $A$  to be computed.

**The Reduction Algorithm.** **Input:** An ordered binary decision diagram  $bdd$ . **Output:** An ordered binary decision diagram in *reduced form*. **Algorithm:** If  $bdd$  has more than **two** distinct leaves (one labelled T and one labelled F), then remove *duplicate* leaves and direct all edges that point to leaves to the single remaining leaf for each truth value. Then perform the following steps for *as long as possible*:

(1) If both the **false** and the **true** edges of a node labelled  $v_i$  point to the *same* node labelled  $v_j$ , **delete** this node  $v_i$  and direct  $v_i$ 's incoming edge(s) to  $v_j$ . (2) If **two** distinct nodes are the roots of identical sub-BDDs, delete *one* sub-BDD and direct its incoming edges to the *other* node. Examples on the *application* of (1) and (2) are shown below. Note that there is also an Apply Algorithm for combining two OBDDs, but this is *not examinable*.

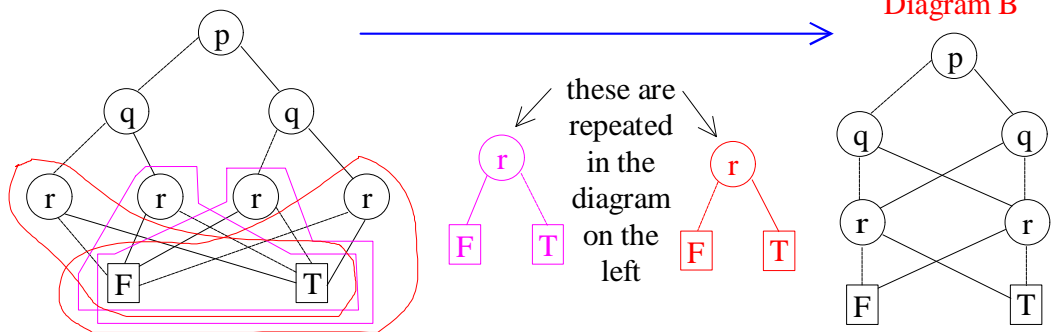
### Application of (1):

Consider  $p \vee (q \wedge r)$



**Definition:** The ordered sequence of atoms labelling the nodes on a path is called an *ordering* of the atoms. A set of orderings  $\{o_1, \dots, o_n\}$  is *compatible* iff there are no atoms  $p$  and  $p'$  s.t.  $p$  comes **before**  $p'$  in  $o_i$  and  $p'$  comes **before**  $p$  in  $o_j$ , with  $i \neq j$ . In Diagram A, we have orderings

### Application of (2):



( $p, q$ ), ( $p, q, r$ ), ( $p, q, r$ ) and ( $p$ ). **Definition:** An *ordered* BDD (OBDD) is a BDD s.t. the set of orderings of the atoms of *all paths* is **compatible**. Diagram B is an OBDD.

**Theorem 1.2 (Bryant):** The algorithm *reduce* is correct — it returns a reduced OBDD that is *equivalent* to the OBDD in the input, in the sense that they give the **same** value to each assignment. For a given *ordering* of atoms, the reduced OBDD's for logically equivalent formulae are *structurally identical*.

**Properties:** (1) A formula is *satisfiable* iff a boxed 'T' appears in the reduced OBDD. (2) A formula is *valid* iff its OBDD is simply a boxed 'T'. (3) A formula is *unsatisfiable* iff its OBDD is simply a boxed 'F'. (4)  $A \equiv B$  if their OBDD's are *structurally identical*. **Theorem 2.2 (Bryant):** The OBDD for the formula  $(p_1 \wedge p_2) \vee \dots \vee (p_{2n-1} \wedge p_{2n})$  has  $2n+2$  nodes under the *ordering*  $p_1, \dots, p_{2n}$ , and has  $2^{n+1}$  nodes under the *ordering*  $p_1, p_{n+1}, p_2, p_{n+2}, \dots, p_n, p_{2n}$ .

19th February 2002

## Semantic Tableau

**Definitions:** (i) A literal is an *atom* or the *negation* of an atom. (ii) An atom is a *positive literal* and the negation of an atom is a *negative literal*. (iii) For any atom  $p$ ,  $\{p, \neg p\}$  is a **complementary pair** of literals. (iv) For any formula  $A$ ,  $\{A, \neg A\}$  is a complementary pair of formulae. (v)  $A$  is the complement of  $\neg A$ , and  $\neg A$  is the complement of  $A$ . **Note:** *Literal* =  $p$  or  $\neg p$ , *Atom* =  $p$ .

The ' $\alpha$ ' and ' $\beta$ ' tables (conjunct and disjunct tables) associated to the forthcoming algorithm are as shown on the right. **Examples:**  $\alpha$ :  $\neg(A_1 \vee A_2) \equiv (\neg A_1 \wedge \neg A_2)$ ;  $\neg(A_1 \rightarrow A_2) \equiv \neg(\neg A_1 \vee A_2) \equiv A_1 \wedge \neg A_2$ .  $\beta$ :  $\neg(\beta_1 \wedge \beta_2) \equiv \neg\beta_1 \vee \neg\beta_2$ .

**Algorithm: The Construction of a Semantic Tableau.**

*Input:* A formula  $A$  of the propositional calculus. *Output:* A semantic tableau  $T$  for  $A$ , all of whose leaves are marked.

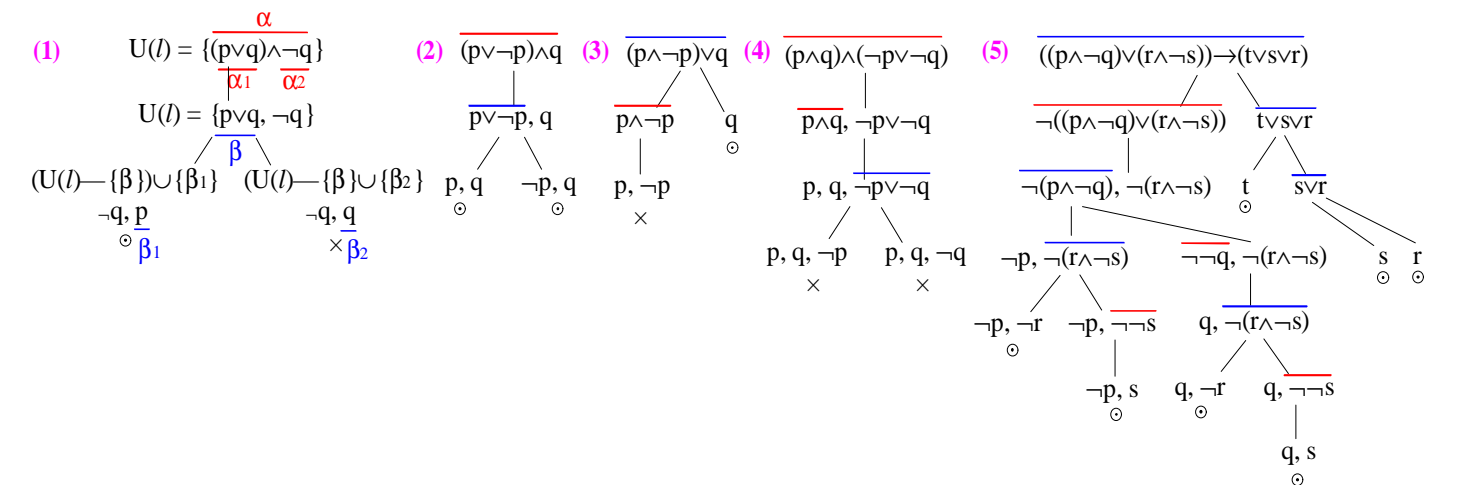
$\alpha$	$\alpha_1$	$\alpha_2$
$\neg\neg A_1$	$A_1$	
$A_1 \wedge A_2$	$A_1$	$A_2$
$\neg(A_1 \vee A_2)$	$\neg A_1$	$\neg A_2$
$\neg(A_1 \rightarrow A_2)$	$A_1$	$\neg A_2$
$A_1 \leftrightarrow A_2$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$
$\beta$	$\beta_1$	$\beta_2$
$\neg(B_1 \wedge B_2)$	$\neg B_1$	$\neg B_2$
$B_1 \vee B_2$	$B_1$	$B_2$
$B_1 \rightarrow B_2$	$\neg B_1$	$B_2$
$\neg(B_1 \leftrightarrow B_2)$	$\neg(B_1 \rightarrow B_2)$	$\neg(B_2 \rightarrow B_1)$

A semantic tableau  $T$  for  $A$  is a tree, each node of which will be *labelled* with a set of formulae. Initially,  $T$  consists of a single node, the **root**, labelled with the singleton set  $\{A\}$ . The tableau is built *inductively* by choosing an unmarked leaf  $l$  labelled with a set of formulae  $U(l)$ , and then applying one of the following two *rules*. The construction terminates when **all** leaves are marked with either a ' $\times$ ' or a ' $\odot$ '.

(1) If  $U(l)$  is a set of literals, check to see if there is a *complementary pair of literals* in  $U(l)$ . If so, mark the leaf as *closed*, ' $\times$ ' — and if not, mark the leaf as *open*, ' $\odot$ '. (2) If  $U(l)$  is **not** a set of literals, choose a formula in  $U(l)$  which is **not** a literal. (a) If the formula is an  $\alpha$ -formula, create a new node  $l'$  as a child of  $l$ , and label  $l'$  with  $U(l') = (U(l) - \{\alpha\}) \cup \{\alpha_1, \alpha_2\}$ . Note that in the case that  $\alpha$  is  $\neg\neg A_1$ , then there is no  $\alpha_2$ . (b) If the formula is a  $\beta$ -formula, create **two** new nodes  $l'$  and  $l''$  as children of  $l$ . Label  $l'$  with  $U(l') = (U(l) - \{\beta\}) \cup \{\beta_1\}$ , and label  $l''$  with  $U(l'') = (U(l) - \{\beta\}) \cup \{\beta_2\}$ .

Let us first consider two examples in which we treat things *informally*. **Example 1:** Let  $A \equiv p \wedge (\neg q \vee \neg p)$ . Now  $v(A) = T$  iff  $v(p) = v(\neg q \vee \neg p) = T$ , iff  $v(p) = T$  and  $v(\neg q) = T$ , or  $v(p) = T$  and  $v(\neg p) = T$ . *First case:*  $\{p, \neg q\}$ , OK. *Second case:*  $\{p, \neg p\}$ , not OK. **Example 2:** Let  $B \equiv (p \vee q) \wedge (\neg p \wedge \neg q)$ . Now  $v(B) = T$  iff  $v(p \vee q) = v(\neg p \wedge \neg q) = T$ , iff  $v(\neg p) = T$ ,  $v(\neg q) = T$  and  $v(p) = T$ , or  $v(\neg p) = T$ ,  $v(\neg q) = T$  and  $v(q) = T$ . In both cases ( $\{\neg p, \neg q, p\}$  and  $\{\neg p, \neg q, q\}$ ), we *cannot* satisfy the list of formulae. Using the algorithm, we could build up *diagrams* as shown on the left.

Let us now consider some *more examples*. Note that in the **first** example, we shall draw the diagram more *formally* than is usual. (1)  $(p \vee q) \wedge \neg q$ , (2)  $(p \vee \neg p) \wedge q$ , (3)  $(p \wedge \neg p) \vee q$ , (4)  $(p \wedge q) \wedge (\neg p \vee \neg q)$ , and (5) (*more complicated*)  $((p \wedge \neg q) \vee (r \wedge \neg s)) \rightarrow (t \vee s \vee r)$ .



**Definition:** A tableau whose construction has *terminated* is called a *completed tableau*. A completed tableau is termed **closed** if **all** leaves are marked closed. Otherwise (i.e. a leaf is marked *open*), it is termed open.

### Assignment 1: Set 19/2; In 5/3; Back 15/3

(1) (a) Prove whether the formulae below are satisfiable or otherwise. Based upon your results, determine whether the *negation* of each formula is valid. (i)  $\neg[(A \leftrightarrow (A \vee B)) \vee \neg(A \rightarrow B)]$ , (ii)  $\neg[(A \vee (B \wedge C)) \vee (\neg(A \vee B) \wedge A \wedge C)]$ . (b) Prove that  $\neg(\neg p \wedge \neg(q \vee r)) \equiv p \vee q \vee r$ . (c) Prove that  $(p \leftrightarrow (q \leftrightarrow q)) \equiv p$ . (d) Prove that  $(p \rightarrow \neg q) \wedge (q \vee (p \wedge r)) \wedge (\neg r \leftrightarrow p) \equiv \neg p \wedge q \wedge r$ . (e) Prove that  $A$  is *satisfiable* iff  $\neg A$  is *falsifiable*.

A: (a) (i) Let  $X$  be the *formula* in question. If  $v(A) = F$ , and if  $v(B) = T$ , then  $v(X) = \neg[(F \leftrightarrow (F \vee T)) \vee \neg(F \rightarrow T)] = \neg[(F \leftrightarrow T) \vee \neg(T)] = \neg[F \vee F] = T$ . This proves that  $X$  is satisfiable. Using part (a) of *Theorem 1.3* ( $A$  is valid iff  $\neg A$  is unsatisfiable), then  $\neg X$  cannot be valid as we have found a *truth assignment* that satisfies  $\neg\neg X = X$ . (ii) Let  $Y$  be the *formula* in question. If  $v(A) = F$ ,  $v(B) = F$  and  $v(C) = T$ , then  $v(Y) = \neg[(F \vee (F \wedge T)) \vee (\neg(F \vee F) \wedge F \wedge T)] = \neg[(F \vee F) \vee (\neg(F) \wedge F \wedge T)] = \neg[F \vee (T \wedge F \wedge T)] = \neg[F \vee F] = T$ . This proves that  $Y$  is *satisfiable*. As in the above, we can therefore say that  $\neg Y$  is **not** valid.

(b) LHS  $\equiv \neg(\neg p \wedge \neg(q \vee r)) \equiv \neg(\neg p \wedge \neg q \wedge \neg r) \equiv \neg\neg p \vee \neg(\neg q \wedge \neg r) \equiv p \vee \neg\neg q \vee \neg\neg r \equiv p \vee q \vee r \equiv$  RHS. (c) LHS  $\equiv (p \leftrightarrow (q \leftrightarrow q)) \equiv (p \leftrightarrow T) \equiv p \equiv$  RHS. (d) LHS  $\equiv (p \rightarrow \neg q) \wedge (q \vee (p \wedge r)) \wedge (\neg r \leftrightarrow p) \equiv (\neg p \vee \neg q) \wedge (q \vee (p \wedge r)) \wedge ((\neg r \rightarrow p) \wedge (p \rightarrow \neg r)) \equiv (\neg p \vee \neg q) \wedge ((q \vee p) \wedge (q \vee r)) \wedge ((r \vee p) \wedge (\neg p \vee \neg r)) \equiv (\neg p \vee \neg q) \wedge (q \vee p) \wedge (q \vee r) \wedge (r \vee p) \wedge (\neg p \vee \neg r) \equiv [((\neg p \vee \neg q) \wedge q) \vee ((\neg p \vee \neg q) \wedge p)] \wedge (q \vee r) \wedge (r \vee p) \wedge (\neg p \vee \neg r) \equiv [(\neg p \wedge q) \vee (\neg q \wedge p)] \wedge (q \vee r) \wedge (r \vee p) \wedge (\neg p \vee \neg r) \equiv ([\{(\neg p \wedge q) \vee (\neg q \wedge p)\} \wedge q] \vee [\{(\neg p \wedge q) \vee (\neg q \wedge p)\} \wedge r]) \wedge (r \vee p) \wedge (\neg p \vee \neg r) \equiv ([\{(\neg p \wedge q)\}] \vee [\{(\neg p \wedge q \wedge r) \vee (\neg q \wedge p \wedge r)\}]) \wedge (r \vee p) \wedge (\neg p \vee \neg r) \equiv [(\neg p \wedge q) \vee (\neg p \wedge q \wedge r) \vee (\neg q \wedge p \wedge r)] \wedge (r \vee p) \wedge (\neg p \vee \neg r) \equiv [(\neg p \wedge q \wedge r) \vee (\neg p \wedge q \wedge r) \vee (\neg q \wedge p \wedge r)] \vee [\neg q \wedge p \wedge r] \wedge (\neg p \vee \neg r) \equiv ((\neg p \wedge q \wedge r) \vee (\neg q \wedge p \wedge r)) \wedge (\neg p \vee \neg r) \equiv (\neg p \wedge q \wedge r) \equiv$  RHS.

(e) **If.** If A is satisfiable, then there is an *assignment* of truth values such that  $v(A) = T$ . But if  $v(A) = T$ , then  $v(\neg A) = F$ , and so the **same** truth assignment falsifies  $\neg A$  — and so  $\neg A$  is falsifiable. **Only If.** If  $\neg A$  is *falsifiable*, then there is an *assignment* of truth values such that  $v(\neg A) = F$ . But if  $v(\neg A) = F$ , then  $v(\neg\neg A) = v(A) = T$ , and so the **same** truth assignment satisfies A — and so A is *satisfiable*.

(2) (a) Define an *interpretation* which makes the set of formulae  $\{p \wedge q, \neg q \vee r, r \wedge (\neg p \vee s)\}$  *simultaneously satisfiable*. (b) Prove the following:  $\{((q \rightarrow (p \vee (q \rightarrow p))) \vee \neg(p \rightarrow q))\} \models (p \vee \neg q)$ . (c) Prove the *following*:  $\{p \rightarrow q, q \rightarrow r, \neg r \leftrightarrow s, s\} \models \neg p$ . (d) Prove that if a *set of formulae* U is satisfiable, and if a formula B is **valid**, then  $U \cup \{B\}$  is *satisfiable*. (e) Let  $U = \{A_1, \dots, A_n\}$ . Prove that  $U \models B$  iff  $\models A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow B$ .

A: (a) The interpretation  $v(p) = v(q) = v(r) = v(s) = T$  suffices to satisfy *all* the formulae in the set. (b) Method 1: By considering all the *possible* truth assignments to p and q in turn (i.e. T, T; T, F; F, T and F, F), we check that whenever the formula in the *set* is T, then the formula  $(p \vee \neg q)$  is true as well. **For example**, when  $v(p) = v(q) = T$ , then  $v((q \rightarrow (p \vee (q \rightarrow p))) \vee \neg(p \rightarrow q)) = ((T \rightarrow (T \vee (T \rightarrow T))) \vee \neg(T \rightarrow T)) = (T \rightarrow (T \vee T)) \vee \neg T = (T \rightarrow T) \vee F = T \vee F = T$ . Then as  $v(p \vee \neg q) = T \vee \neg T = T \vee F = T$ , then  $\{((q \rightarrow (p \vee (q \rightarrow p))) \vee \neg(p \rightarrow q))\} \models (p \vee \neg q)$  holds in *this* case.

Method 2 (use in an *exam!*): LHS  $= ((q \rightarrow (p \vee (q \rightarrow p))) \vee \neg(p \rightarrow q)) \equiv ((\neg q \vee (p \vee (\neg q \vee p))) \vee \neg(\neg p \vee q))$  (using  $A \rightarrow B \equiv \neg A \vee B$ )  $\equiv (\neg q \vee p \vee \neg q \vee p) \vee (p \wedge \neg q)$  (using a *derivative* of de Morgan's law)  $\equiv \neg q \vee p \vee (p \wedge \neg q) \equiv (p \vee \neg q) \wedge (p \vee \neg q) \rightarrow (p \vee \neg q)$ , so that we can use **Theorem 1.5** to say that  $\{((q \rightarrow (p \vee (q \rightarrow p))) \vee \neg(p \rightarrow q))\} \models (p \vee \neg q)$ . QED.

(c) Method 1: Looking at the *set of formulae*, we see that in order to **satisfy** the set of formulae, the *last* formula implies that we must have  $v(s) = T$ , the *second last* formula then implies that we must have  $v(r) = F$ , the *third last* formula then implies that we must have  $v(q) = F$ , and then the *first* formula implies that we must then have  $v(p) = F$ . So because the **only** interpretation that satisfies the set of formulae is  $v(s) = T$  and  $v(r) = v(q) = v(p) = F$  — and because this interpretation satisfies  $\neg p$ , then we have *proved* that  $\{p \rightarrow q, q \rightarrow r, \neg r \leftrightarrow s, s\} \models \neg p$ . QED.

Method 2 (used in the *solutions*): By Theorem 1.5,  $\{A_1, A_2, \dots, A_n\} \models A$  iff  $A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow A$ . So we manipulate  $A_1 \wedge A_2 \wedge \dots \wedge A_n$ , in this case  $(p \rightarrow q) \wedge (q \rightarrow r) \wedge (\neg r \leftrightarrow s) \wedge s$ , to get  $\neg p \wedge \neg q \wedge \neg r \wedge s$ , which implies  $\neg p$  (i.e.  $\neg p \wedge \neg q \wedge \neg r \wedge s \rightarrow \neg p$ ), and so because we have shown that  $(p \rightarrow q) \wedge (q \rightarrow r) \wedge (\neg r \leftrightarrow s) \wedge s \equiv \neg p \wedge \neg q \wedge \neg r \wedge s \rightarrow \neg p$ , then the *result* we want holds. QED.

(d) If  $U$  is satisfiable, then there is an *interpretation* that satisfies all the formulae in  $U$ . Because  $B$  is valid, then the interpretation that satisfies  $U$  will **also** satisfy  $B$ . It follows that because there is an interpretation that satisfies  $U$  **and**  $B$ , then  $U \cup \{B\}$  is satisfiable. QED.

C. J. Rudall's answer: Let  $U = \{A_1, \dots, A_n\}$ . If  $U$  is *satisfiable*, then there exists a  $v'$  such that  $v'(A_1) = v'(A_2) = \dots = v'(A_n) = T$ . If  $B$  is *valid*, then  $v(B) = T$  for **every** possible  $v$ . Let  $v$  be *arbitrary* such that  $v(B) = T$ . (i) Let  $v'$  and  $v$  assign truth values to *disjoint sets of atoms*. It follows that  $v'$  can be **extended** to include  $v$ , and now  $v'(A_1) = v'(A_2) = \dots = v'(A_n) = v'(B) = T$ , and so by the definition of *simultaneous satisfiability*,  $U \cup \{B\}$  is satisfiable.

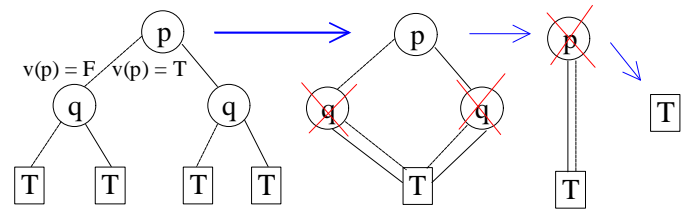
(ii) Consider the situation in which every atom given a *truth assignment in  $v$*  is **also** given one in  $v'$ : because  $v(B) = T$  for *every* possible  $v$ , then it follows that  $v'(B) = T$ , and so  $v'(A_1) = v'(A_2) = \dots = v'(A_n) = v'(B) = T$ , so that (*by definition*)  $U \cup \{B\}$  is satisfiable. (iii) Consider the situation in which only *some* atoms given a truth assignment in  $v$  are given a truth assignment in  $v'$ . Extend  $v'$  to *include* those assignments of  $v$  not already given in  $v'$  — then as  $v(B) = T$  for *every possible*  $v$ , then  $v'(B) = T$  — and so, with  $v'(A_1) = v'(A_2) = \dots = v'(A_n) = T$ , we note that  $U \cup \{B\}$  is satisfiable *by definition*. (iv) Consider the situation in which every atom given a truth assignment in  $v'$  is *also* given one in  $v$  — we treat this situation like we treated situation (ii).

(e) **If.** If  $U \models B$ , then every time that  $U$  is *simultaneously satisfiable*, we must have  $v(B) = T$ . But every time that  $U$  is *simultaneously satisfiable*, it follows that  $v(A_1 \wedge A_2 \wedge \dots \wedge A_n) = T$ . So because *every* time that  $v(A_1 \wedge A_2 \wedge \dots \wedge A_n) = T$  we have  $v(B) = T$ , it follows that  $A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow B$  is always true when  $v(A_1 \wedge A_2 \wedge \dots \wedge A_n) = T$ . But because in any **other** situation  $A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow B$  is always true (i.e. because  $F \rightarrow ? = T$ , where the  $F$  comes from the situations where the interpretation  $v$  makes *at least one* of the  $A_i$ 's false, so that  $v(A_1 \wedge A_2 \wedge \dots \wedge A_n) = F$ ), then it *follows* that  $A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow B$  is valid.

**Only If.** If  $A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow B$  is valid, then  $v(A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow B)$  is *never false*. But because  $v(A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow B)$  is only *ever false* when  $v(A_1 \wedge A_2 \wedge \dots \wedge A_n)$  is *true* and when  $v(B)$  is false, then *whenever*  $v(A_1 \wedge A_2 \wedge \dots \wedge A_n)$  is true, then we must have  $v(B) = T$  *as well*. But if  $v(A_1 \wedge A_2 \wedge \dots \wedge A_n)$  is true, then **all** the  $A_i$  ( $1 \leq i \leq n$ ) must be true so that  $U$  is *simultaneously satisfiable*. We have therefore found that **whenever**  $U$  is simultaneously satisfiable, it follows that we **must** have  $v(B) = T$ . It *follows* that  $U \models B$ . QED.

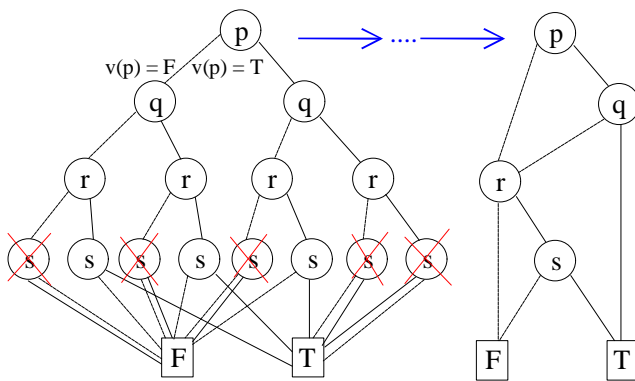
(3) (a) Using *Ordered Binary Decision Diagrams* (OBDDs), prove that the following formula is valid (**all stages** of the reduction process should be shown):  $(p \leftrightarrow q) \vee \neg((p \rightarrow q) \wedge (q \rightarrow p))$ . (b) Using OBDDs, prove that the formula  $\neg(\neg p \wedge \neg(q \vee r))$  is *equivalent* to the formula  $p \vee q \vee r$ . **All stages** of the reduction process should be shown. (c) Bryant states that the OBDD for the formula  $(p_1 \wedge p_2) \vee \dots \vee (p_{2n-1} \wedge p_{2n})$  has  $2n+2$  nodes under the ordering  $p_1, \dots, p_{2n}$ , and  $2^{n+1}$  nodes under the ordering  $p_1, p_{n+1}, p_2, p_{n+2}, \dots, p_n, p_{2n}$ . Show that the *first part* of this theorem is true for the formula  $(p \wedge q) \vee (r \wedge s)$ . **All stages** of the reduction process should be shown.

A: (a) The diagram is as shown on the right. **Notes:** we first draw the *initial* OBDD for the formula in question, writing down the calculations for the truth values below the diagram (**not** shown here). We then apply the *reduction* algorithm, the first step being to draw a diagram with a **single** boxed 'T' and a **single** boxed 'F'. We then go on to *simplify* as shown. Because the OBDD for the formula consists simply of a **single** boxed 'T', then it follows that the formula is **valid**. QED. **Tip:** Always show **all** stages of the reduction process, i.e. don't be tempted to *combine two or more stages* onto one diagram.



(b) In this question, we draw a diagram for *each* formula, simplify or **reduce** the diagrams, and because the reduced diagrams are **identical**, we conclude that the formulae are equivalent.

(c) Consider the formula  $(p \wedge q) \vee (r \wedge s)$  under the ordering  $p, q, r, s$ . We have an *initial* diagram as shown on the left, and after **reduction** (including removing *identical* sub-BDD's), we are left with the *second* diagram on the left. As you can see, there are 6 nodes in our reduced diagram (including the boxed 'T' and the boxed 'F'). This *agrees* with Bryant's theorem in that we were dealing with the formula  $(p \wedge q) \vee (r \wedge s)$  so that  $2n = 4$ , and so  $2n+2 = 6$ , which agrees with our *diagram*. QED.



22nd February 2002

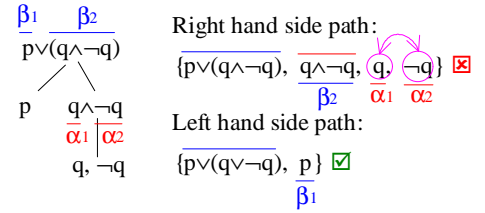
## Soundness and Completeness of Semantic Tableaux

Let  $T$  be a *completed tableau* for a formula  $A$ . **Theorem 3.1 (Soundness):** If  $T$  is *closed* then  $A$  is *unsatisfiable*. **Theorem 3.2 (Completeness):** If  $A$  is *unsatisfiable* then  $T$  is *closed*. **Proof** of Theorem 3.1. We will prove a more *general* theorem. Consider a subtree  $T$  rooted at  $n$ . We'll prove: if the subtree **closes**, then  $U(n)$  is **unsatisfiable**. Soundness is the *special* case where  $n$  is the *root*.

Proof by induction on the *height*  $h$  of the node  $n$  in  $T$ .  $h = 0$ , base case:  $n$  is a *leaf*. We know that  $T$  closes so that  $U(n)$  must contain a *complementary pair of literals*, and therefore  $U(n)$  is by definition *unsatisfiable*. Case  $h > 0$ : (a) An  $\alpha$  rule was used, so that  $n$  has a child  $n'$  of height  $h-1$ ,  $U(n) = \{A_1 \wedge A_2\} \cup U_0$ , where  $U_0$  is some *set of formulae*, and  $U(n') = \{A_1, A_2\} \cup U_0$ .

**Inductive step:** Assume that the theorem is *true* for  $n-1$ , i.e. that  $U(n')$  is *unsatisfiable* since the subtree rooted at  $n'$  closes. Let  $v$  be an *arbitrary interpretation*. We know that  $v(A') = F$  for some  $A' \in U(n')$ . (i)  $A'$  is  $A_1$ , therefore (by *definition*) if  $v(A_1) = F$  then  $v(A_1 \wedge A_2) = F$  and so  $A_1 \wedge A_2 \in U(n)$ . Since  $v$  was *arbitrary*, then  $U(n)$  is *unsatisfiable*. (ii)  $A'$  is  $A_2$  — same as (i). (iii)  $A' \in U_0$ . Now  $v(A') = F$ , and  $A' \in U(n)$ . Since  $v$  was *arbitrary*, then  $U(n)$  is *unsatisfiable*. (b) a  $\beta$  rule was used (**not** proved).

**Definition:** Let  $U$  be a set of formulae.  $U$  is a **Hintikka set** iff (1) For all atoms  $p$  appearing in a formula of  $U$ , either  $p \in U$  or  $\neg p \in U$ ; (2) If  $\alpha \in U$  is an  $\alpha$ -formula, then  $\alpha_1 \in U$  and  $\alpha_2 \in U$ ; (3) If  $\beta \in U$  is a  $\beta$ -formula, then  $\beta_1 \in U$  or  $\beta_2 \in U$ . An example is shown on the right (where the first set is not a Hintikka set and the second set is a Hintikka set). Further examples of valid Hintikka sets:  $\{p \vee (q \wedge r), q \wedge r, q, r\}$ ,  $\{p \vee (q \wedge r), p\}$  and  $\{(p \vee q) \wedge (\neg p \vee \neg q), p \vee q, \neg p \vee \neg q, p, \neg q\}$ .



**Theorem 3.3:** Let  $L$  be an open leaf in a completed tableau  $T$ . Let  $U = \cup_i U(i)$ , where  $i$  runs over the set of nodes on the branch from the root to  $L$ . Then  $U$  is a **Hintikka set**.

26th February 2002

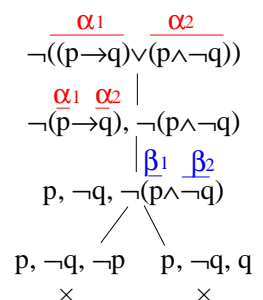
**Theorem 3.4 (Hintikka's Lemma):** Let  $U$  be a Hintikka set. Then  $U$  is satisfiable. **Proof:** Let  $P = \{p_1, \dots, p_m\}$  be the set of atoms appearing in all formulae in  $U$ . Define the following interpretation for  $U$  as follows:  $v(p) = T$  if  $p \in U$ ,  $v(p) = F$  if  $\neg p \in U$ , and  $v(p) = T$  if  $p \notin U$  and  $\neg p \notin U$ . By condition 1 of Hintikka sets, we know that each atom is given only one truth value.

*Proof by structural induction:* For any formula  $A$  in  $U$ , (a) If  $A$  is an atom  $p$ , then  $v(A) = v(p) = T$  as we know that  $p \in U$ . (b) If  $A$  is the negated atom  $\neg p$ , then  $v(A) = v(\neg p) = T$  as  $\neg p \in U$ . (c) If  $A$  is  $\alpha$ , then by condition 2 of Hintikka sets,  $\alpha_1, \alpha_2 \in U$ . *Inductive step:* Assume that  $v(\alpha_1) = v(\alpha_2) = T$ , then  $v(\alpha_1 \wedge \alpha_2) = T$ . Therefore,  $v(A) = T$ . (d) If  $A$  is  $\beta$ , then by condition 3 of Hintikka sets, either  $\beta_1 \in U$  or  $\beta_2 \in U$ . *Inductive step:* Assume that  $v(\beta_1) = T$  or  $v(\beta_2) = T$ , then  $v(\beta_1 \vee \beta_2) = T$ . Therefore,  $v(A) = T$ .

Now recall Theorem 3.2: Let  $T$  be a completed tableau for a formula  $A$ . If  $A$  is unsatisfiable then  $T$  is closed. **Proof:** We will prove the contrapositive ( $C \rightarrow D; \neg D \rightarrow \neg C$ ). Let  $T$  be a completed open tableau. Then by Theorem 3.3,  $U$ , the union of the labels of the nodes on an open branch is a Hintikka set.

By Theorem 3.4,  $U$  is satisfiable and a model can be found.  $A$  is the labelling of the root, so that  $A \in U$ . Therefore, the model for  $U$  is also a model for  $A$ . So if  $T$  is a completed open tableau for  $A$ , then  $A$  is satisfiable. Therefore, if  $A$  is not satisfiable, then  $T$  must be a completed closed tableau.

**Theorem 3.5 (Soundness and Completeness):** If  $T$  is a completed tableau, then  $A$  is unsatisfiable iff  $T$  is closed. **Corollary 3.1:**  $A$  is satisfiable iff  $T$  is open. **Proof:**  $A$  is satisfiable iff  $A$  is not unsatisfiable (by definition); iff  $T$  is not closed (by Theorem 3.5) iff  $T$  is open. **Corollary 3.2:**  $A$  is valid iff the tableau for  $\neg A$  closes.



Q: Is the formula  $(p \rightarrow q) \vee (p \wedge \neg q)$  valid? A: By showing that the semantic tableau for the negation closes (as shown on the right), we prove that  $(p \rightarrow q) \vee (p \wedge \neg q)$  is valid. Note: when drawing a diagram, you can stop expanding a branch once you find a pair of complementary literals.

## Tutorial

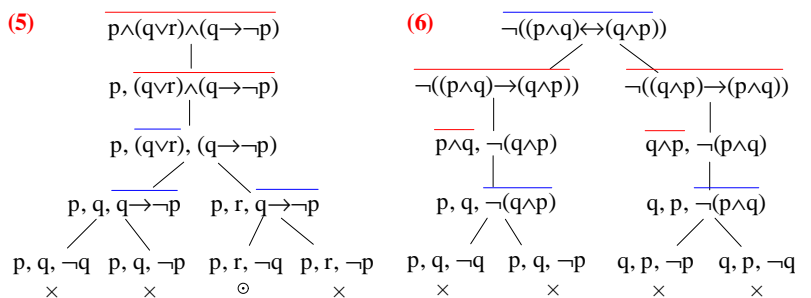
(1) Which of the following are *Hintikka sets*: (a)  $\{p \wedge (q \vee r), r\}$ ; (b)  $\{p \wedge (q \vee r), r, (q \vee r), p, \neg r\}$ ; (c)  $\{p \wedge (q \vee r), r, (q \vee r), p, q\}$ ; and (d)  $\{p \rightarrow (q \vee r), p, \neg(p \leftrightarrow q), p \rightarrow q\}$ . A: (a) *Not* a Hintikka set —  $p$  and  $q \vee r$  are **not** elements of the set. (b) *Not* a Hintikka set — the set contains a *complementary pair of literals*  $r$  and  $\neg r$ . (c) *Is* a Hintikka set. (d) As  $p \rightarrow (q \vee r)$  is in the set, then we must have  $\neg p$  or  $(q \vee r)$  in the set. But this is **not** the case, so we conclude that the set is **not** a Hintikka set.

(2) For the sets in question 1 that were **not** Hintikka sets, add/remove/alter formulae to *make them* into Hintikka sets. A: (a) **Add**  $(q \vee r)$  and  $p$  to the set. (b) **Remove**  $\neg r$  from the set. (c) **Add**  $\neg p$ ,  $(q \vee r)$  and  $r$ ; **remove**  $p$ . We now have  $Q = \{p \rightarrow (q \vee r), \neg p, (q \vee r), r, \neg(p \rightarrow q), p \rightarrow q\}$ . As  $\neg(p \leftrightarrow q) \in Q$  then we *must have*  $\neg(p \rightarrow q)$  or  $\neg(q \rightarrow p) \in Q$ . So **add**  $\neg(q \rightarrow p)$  to  $Q$ , say. Now  $\neg(q \rightarrow p) \in Q$  so we *must have*  $q$  and  $\neg p$  in  $Q$  — so **add** them in to give  $Q = \{p \rightarrow (q \vee r), \neg p, q, r, (q \vee r), \neg(p \leftrightarrow q), \neg(q \rightarrow p), p \rightarrow q\}$ . Now  $p \rightarrow q \in Q$  so we *must have*  $\neg p$  or  $q$  in  $Q$  — which is OK. Conclusion:  $Q$  is now a *Hintikka set*.

(3) Draw the **completed** semantic tableau for the following formula and construct the *Hintikka sets* for each **open** branch:  $p \vee ((q \rightarrow r) \wedge s)$ . A: The tableau is as shown on the left. **First** branch:  $\{p \vee ((q \rightarrow r) \wedge s), p\}$ . **Second** branch:  $\{p \vee ((q \rightarrow r) \wedge s), (q \rightarrow r) \wedge s, q \rightarrow r, s, \neg q\}$ . **Third** branch:  $\{p \vee ((q \rightarrow r) \wedge s), (q \rightarrow r) \wedge s, q \rightarrow r, s, r\}$ . (4) What interpretation satisfies the following *Hintikka sets*: (a)  $\{p \wedge (q \vee r), (q \vee r), p, q\}$ ; (b)  $\{\neg(p \wedge q \wedge r), \neg(q \wedge r), \neg q, p\}$ . A: (a)  $v(p) = T, v(q) = T, v(r) = ?$ , say  $T$ ; (b)  $v(p) = T, v(q) = F, v(r) = ?$ , say  $T$  again.

(5) Prove that the formula  $p \wedge (q \vee r) \wedge (q \rightarrow \neg p)$  is satisfiable by showing that a *completed* tableau for the formula is open. Using the tableau, give all the interpretations which make the formula **true**. A: The tableau is as shown below. The interpretation which makes the formula **true** is  $v(p) = T, v(r) = T$  and  $v(q) = F$ .

(6) Prove that the following formula is *valid* by showing that the completed tableau for the **negation** is closed:  $(p \wedge q) \leftrightarrow (q \wedge p)$ . A: The tableau is as shown on the right. From the tableau, we **conclude** that the formula  $(p \wedge q) \leftrightarrow (q \wedge p)$  is valid.



## Resolution

**Conjunctive Normal Form. Definition:** A formula is in CNF iff it is a *conjunction* of *disjunctions* of literals. *Examples:*  $(\neg p \vee q \vee r) \wedge (\neg q \vee r) \wedge \neg r$  is in CNF,  $(\neg p \vee q \vee r) \wedge ((p \wedge \neg q) \vee r) \wedge \neg r$  is **not** in CNF,  $(\neg p \vee q \vee r) \wedge \neg(\neg q \vee r) \wedge \neg r$  is **not** in CNF, and  $(\neg p \vee q \vee r)$  is in CNF.

**Theorem 4.1:** Every formula in the propositional calculus can be transformed into an equivalent formula in CNF. Examples:  $p \rightarrow q \equiv \neg p \vee q$ , and  $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$ . **Proof:** Perform the following on  $A$  to turn it into CNF: (a) Use *logical equivalences* to eliminate all operators except for negation, conjunction and disjunction.

(b) Push all *negations* inwards using de Morgan's laws,  $\neg(A \wedge B) \equiv \neg A \vee \neg B$ , and  $\neg(A \vee B) \equiv \neg A \wedge \neg B$ . (c) Eliminate *double negations* using the equivalence  $\neg\neg A \equiv A$ . (d) The formula now consists of *conjunctions* and *disjunctions of literals*. Use the following *distributive* laws to eliminate conjunctions within disjunctions:  $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ , and  $(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$ .

**Example:**  $(\neg p \rightarrow \neg q) \rightarrow (p \rightarrow q) \equiv \neg(\neg p \rightarrow \neg q) \vee (p \rightarrow q) \equiv \neg(\neg\neg p \vee \neg q) \vee (\neg p \vee q)$  (step (a) now *completed*);  $\equiv (\neg\neg\neg p \wedge \neg\neg q) \vee (\neg p \vee q)$  (step (b) now *completed*);  $\equiv (\neg p \wedge q) \vee (\neg p \vee q)$  (step (c) now *completed*);  $\equiv (\neg p \vee (\neg p \vee q)) \wedge (q \vee (\neg p \vee q)) \equiv (\neg p \vee \neg p \vee q) \wedge (q \vee \neg p \vee q)$  (step (d) now *completed*).

**Definition.** A *clause* is a set of literals which is considered to be an implicit *disjunction*, e.g.  $\neg p \vee q \vee r$  is written as  $\{\neg p, q, r\}$ . A *unit clause* is a clause consisting of exactly **one** literal, e.g.  $\neg p$  is written as  $\{\neg p\}$ , a *unit clause*. A formula in *clausal form* is a set of clauses which is considered to be an implicit **conjunction**. **Example:**  $(p \vee q \vee r) \wedge (p \vee \neg q \vee r)$  is written as  $\{\{p, q, r\}, \{p, \neg q, r\}\}$ . Notice that the *commas* represent ' $\vee$ ' or ' $\wedge$ ' depending on their **position**.

**Example:** For the formula  $(\neg q \vee \neg p \vee q) \wedge (p \vee r \vee \neg s)$  in CNF, the *clause* representing  $(\neg q \vee \neg p \vee q)$  is  $\{\neg q, \neg p, q\}$ , the *clause* representing  $(p \vee r \vee \neg s)$  is  $\{p, r, \neg s\}$ , and the *clausal form* of the **entire** formula is  $\{\{\neg q, \neg p, q\}, \{p, r, \neg s\}\}$ . **Corollary 4.2:** Every formula in the propositional calculus can be transformed into an *equivalent* formula in clausal form.

**Notation:** (a) Write the clauses with the *delimiters* '{' and '}' removed, and (b) write the clauses with **negation** as a bar over the propositional letter, e.g.  $\bar{p}$  represents  $\neg p$ . **Example:**  $\{\{\neg q, \neg p, q\}, \{p, r, \neg s\}\}$  is written as  $\{\bar{q}\bar{p}q, pr\bar{s}\}$ .

Now if  $L$  is a *literal*, then  $L^c$  is its **complement**, i.e. if  $L = p$  then  $L^c = \bar{p}$ , and if  $L = \bar{p}$  then  $L^c = p$ . **Lemma 4.3:** Suppose that a literal  $L$  appears in *some* clause of  $S$  (where  $S$  is a **set** of clauses), but  $L^c$  does not appear in **any** clause of  $S$ . Let  $S'$  be obtained from  $S$  by deleting *every clause* containing  $L$ . Then  $S$  is **satisfiable** iff  $S'$  is **satisfiable** (denoted by  $S \approx S'$ ).

**Example:**  $S = \{pq\bar{r}, p\bar{q}, q\bar{p}\} (= (p \vee q \vee \neg r) \wedge (p \vee \neg q) \wedge (\neg q \vee p))$ . Take the *letter* ' $r$ ':  $\bar{r}$  appears but  $r$  does **not**, so  $S' = \{p\bar{q}, q\bar{p}\}$ . Case (a):  $S$  is *satisfiable*. As  $S' \subset S$ , then  $S'$  is *automatically* satisfiable. Case (b):  $S'$  is satisfiable. This means that there is an **interpretation** which makes  $v(p\bar{q}) = v(q\bar{p}) = T$ . So either  $v(p) = T$  and  $v(q) = T$ , which makes  $v(pq\bar{r}) = T$ ; or  $v(p) = F$  and  $v(q) = F$ , in which case  $v(pq\bar{r})$  depends on the value of  $v(r)$ . **Trick:** in this situation, simply *extend*  $S'$ 's interpretation by adding  $v(L) = T$  — i.e. we set  $v(r) = F$  so that  $v(pq\bar{r}) = T$ .

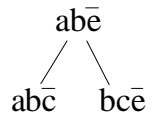
**Lemma 4.4:** Let  $C = \{L\} \in S$  be a *unit clause*. Let  $S'$  be obtained from  $S$  by deleting every clause *containing*  $L$  and by deleting  $L^c$  from every *remaining clause*. Then  $S \approx S'$ . Example: When  $S = \{r, pq\bar{r}, p\bar{q}, \bar{q}p\}$ ,  $S' = \{pq, p\bar{q}, qp\}$ .

**Lemma 4.5:** Suppose that  $L \in C$  and that  $L^c \in C$  for some  $C \in S$ . Let  $S' = S - \{C\}$ . Then  $S \approx S'$ . Example: When  $S = \{p\bar{p}q, qrs\}$ ,  $S' = \{qrs\}$ . **Lemma 4.6:** Let  $C_1, C_2 \in S$  with  $C_1 \subseteq C_2$ . Let  $S' = S - \{C_2\}$ . Then  $S \approx S'$ . Example: When  $S = \{pq, pqr, \bar{p}\bar{q}\bar{r}\}$ ,  $S' = \{pq, \bar{p}\bar{q}\bar{r}\}$ . **Definition:** An *empty clause* is denoted by the symbol  $\square$ , and an *empty set of clauses* is denoted by  $\phi$ . **Lemma 4.7:**  $\phi$  is **valid** and  $\square$  is **unsatisfiable**.

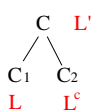
**Exercises:** Simply the *following*: (a)  $S = \{p\bar{q}\bar{r}, \bar{p}qr, s\bar{r}\}$ , (b)  $S = \{p\bar{q}\bar{r}, s, qs, \bar{s}\bar{q}p\}$ , (c)  $S = \{p\bar{p}rs, q\bar{r}s, \bar{r}s\}$ , (d)  $S = \{s, \bar{q}\bar{s}, qr\}$ . A: (a)  $S' = \{p\bar{q}\bar{r}, \bar{p}qr\}$  by Lemma 4.3 (s). (b)  $S' = \{p\bar{q}\bar{r}, \bar{q}p\}$  by Lemma 4.4;  $S'' = \phi$  by Lemma 4.3 (p). (c)  $S' = \{q\bar{r}s, \bar{r}s\}$  by Lemma 4.5 (p,  $\bar{p}$ );  $S'' = \phi$  by Lemma 4.3 (s). (d)  $S' = \{\bar{q}, qr\}$  by Lemma 4.4 (s);  $S'' = \{r\}$  by Lemma 4.4 ( $\bar{q}$ );  $S''' = \phi$  by Lemma 4.4 (r).

**Resolution Rule.** Let  $C_1$  and  $C_2$  be clauses such that  $L \in C_1$  and  $L^c \in C_2$ . The clauses  $C_1$  and  $C_2$  are then said to be *clashing clauses* and are said to clash on  $L$  and  $L^c$ .  $C$ , the **resolvent** of  $C_1$  and  $C_2$ , is the clause  $C = \text{Res}(C_1, C_2) = (C_1 - \{L\}) \cup (C_2 - \{L^c\})$ . Note:  $C_1$  and  $C_2$  are the *parent clauses* of  $C$ .

Example:  $C_1 = ab\bar{c}$ ,  $C_2 = bc\bar{c}$ : as  $C_1$  and  $C_2$  clash on  $c$  and  $\bar{c}$ , then it follows that  $C = \text{Res}(C_1, C_2) = (ab\bar{c} - \{\bar{c}\}) \cup (bc\bar{c} - \{c\}) = ab \cup b\bar{c} = ab\bar{c}$ . Example: If  $C_1 = p\bar{q}r$  and if  $C_2 = sqt$ , then  $\text{Res}(C_1, C_2) = prst$ .



**Theorem 4.8:** The resolvent  $C$  is *satisfiable* iff the parent clauses  $C_1$  and  $C_2$  are *mutually satisfiable*. **Proof** ( $\Rightarrow$  direction): If the resolvent  $C$  is satisfiable, then  $C_1$  and  $C_2$  are. Let  $v$  be an *interpretation* which satisfies  $C$ , let  $L$  be in  $C_1$ , and let  $L^c$  be in  $C_2$ . Then  $v(L) = T$  for *at least one literal* in  $C$ . By the resolution rule,  $L' \in C_1$  and/or  $L' \in C_2$ . If  $L' \in C_1$  (similarly for  $C_2$ ), then  $v(C_1) = T$ , and neither  $L \in C$  or  $L^c \in C$  (the resolution rule **removed**  $L$  and  $L^c$ ). It follows that  $v$  is *not defined* on  $L$ , and so  $v$  can be *extended* to an interpretation  $v'$  by defining  $v'(L^c) = T$ .  $\therefore$ ,  $v'(C_2) = T$ , and so  $v'(C_1) = v(C_1) = T$  because  $v'$  is an **extension** of  $v$ .



**Resolution Procedure** (for *propositional logic*). Input: A set of clauses  $S$ . Output:  $S$  is *satisfiable* or *unsatisfiable*. Algorithm: Let  $S$  be a set of clauses, and define  $S_0 = S$ . Assume that  $S_i$  has been **constructed**. Choose a pair of *clashing clauses*  $C_1, C_2 \in S_i$  that has **not** been chosen before. Let  $C$  be the clause  $\text{Res}(C_1, C_2)$  defined by the *resolution rule*, and let  $S_{i+1} = S_i \cup \{C\}$ . If  $C = \square$ , then terminate the procedure —  $S$  is **unsatisfiable**. If  $S_{i+1} = S_i$  for *all possible choices of clashing clauses*, then terminate the procedure —  $S$  is **satisfiable**.

Example: Is  $S = \{p\bar{q}, \bar{p}, q\} = \{C_1, C_2, C_3\}$  satisfiable? Now  $\text{Res}(C_1, C_2) = \{\bar{q}\}$  so that  $S' = S \cup \{\bar{q}\}$ . Is  $S'$  *satisfiable*? If there is an interpretation that makes  $S$  true, then that **same** interpretation will make  $S \cup \text{Res}(C_1, C_2)$  true.

Now let  $S = \{p, \bar{p}q, \bar{r}, \bar{p}\bar{q}\bar{r}\} = \text{clauses } \{1, 2, 3, 4\}$ . Choose a pair of *clashing* clauses: 3 and 4, say. Then  $\text{Res}(\bar{r}, \bar{p}\bar{q}\bar{r}) = \bar{p}\bar{q}$ , clause 5, and so  $S_1 = S \cup \bar{p}\bar{q}$ . Now choose a *different* pair of clashing clauses: 5 and 2, say. Then  $\text{Res}(\bar{p}\bar{q}, \bar{p}q) = \bar{p}$ , clause 6, and so  $S_2 = S_1 \cup \bar{p}$ . Choose **another** pair of clashing clauses: 6 and 1, say. Then  $\text{Res}(\bar{p}, p) = \square$ , the *empty* clause. It follows that  $S$  is **unsatisfiable**. **Definition:** A derivation of  $\square$  from  $S$  is called a *refutation* of  $S$ .

In the following examples, *do not* simplify  $S$  (i.e. don't apply **lemmas** to find  $S'$ , where  $S \approx S'$ ). (1)  $S = \{pq, \bar{p}qr, \bar{r}, \bar{q}\} = \{1, 2, 3, 4\}$ ; (2)  $S = \{pq, p\bar{q}, r\bar{p}, \bar{r}\bar{p}\} = \{1, 2, 3, 4\}$ ; (3)  $S = \{p\bar{q}\bar{r}, pqr, \bar{p}q\} = \{1, 2, 3\}$ .

A: (1) *Clashing clauses:* 2 and 3:  $\text{Res}(\bar{p}qr, \bar{r}) = \bar{p}q$ , clause 5. 1 and 5:  $\text{Res}(pq, \bar{p}q) = q$ , clause 6. 4 and 6:  $\text{Res}(\bar{q}, q) = \square$ . It follows that  $S$  is **unsatisfiable**. (2) *Clashing clauses:* 1 and 2:  $\text{Res}(pq, p\bar{q}) = p$ , clause 5. 3 and 4:  $\text{Res}(r\bar{p}, \bar{r}\bar{p}) = \bar{p}$ , clause 6. 5 and 6:  $\text{Res}(p, \bar{p}) = \square$ . It follows that  $S$  is **unsatisfiable**. (3) *Clashing clauses:* 1 and 2:  $\text{Res}(p\bar{q}\bar{r}, pqr) = pq$ , clause 4. 1 and 3:  $\text{Res}(p\bar{q}\bar{r}, \bar{p}q) = q\bar{r}$ , clause 5. 2 and 3:  $\text{Res}(pqr, \bar{p}q) = qr$ , clause 6. 1 and 6:  $\text{Res}(p\bar{q}\bar{r}, qr) = pq$ . 2 and 5:  $\text{Res}(pqr, q\bar{r}) = pq$ . 3 and 4:  $\text{Res}(\bar{p}q, pq) = q$ , clause 7. As there are *no more clashing clauses* to consider that will give **more** elements (when we have  $S_i = \{p\bar{q}\bar{r}, pqr, \bar{p}q, pq, q\bar{r}, qr, q\}$ ), then it follows that  $S$  is **satisfiable**.

Consider a *set of clauses*  $S$  and define  $\text{Res}(S) = S \cup \{R \mid R \text{ is a } \mathbf{resolvent} \text{ of 2 clauses in } S\}$ . Also define  $\text{Res}^0(S) = S$ ,  $\text{Res}^{n+1}(S) = \text{Res}(\text{Res}^n(S))$ , and  $\text{Res}^*(S) = \bigcup_{n=0}^{\infty} \text{Res}^n(S)$ . It follows that  $\text{Res}^*(S) = \text{Res}^0(S) \cup \text{Res}^1(S) \cup \text{Res}^2(S) \cup \dots = S \cup \text{Res}(S) \cup \text{Res}(\text{Res}(S)) \cup \dots$  **Theorem 4.9** (*Soundness and completeness*):  $S$  is *unsatisfiable* iff  $\square \in \text{Res}^*(S)$ .

**Soundness** ( $\Leftarrow$ ): if  $\square$  is derived, then  $S$  is *unsatisfiable*. (Informal) **Proof:** Theorem 4.8 says that  $\text{Res}(C_1, C_2)$  is satisfiable iff  $C_1$  and  $C_2$  are *mutually satisfiable*. Let  $S = \{C_1, C_2, \dots\}$ , and consider  $S \cup \text{Res}(C_1, C_2)$ . The interpretation that makes  $C_1$  and  $C_2$  *mutually satisfiable* also makes  $\text{Res}(C_1, C_2)$  *satisfiable*. So we can **add** the resolvent to  $S$  and not *affect* the satisfiability of  $S$ . Further, if  $S = \{L, L^c, \dots\}$ , then  $\text{Res}(L, L^c) = \square$ , and it is *obvious* then that  $L \wedge L^c \wedge \dots \equiv \text{false}$ .

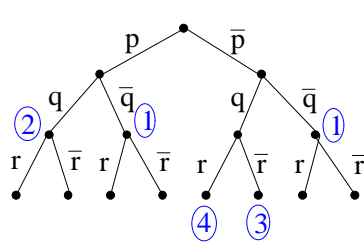
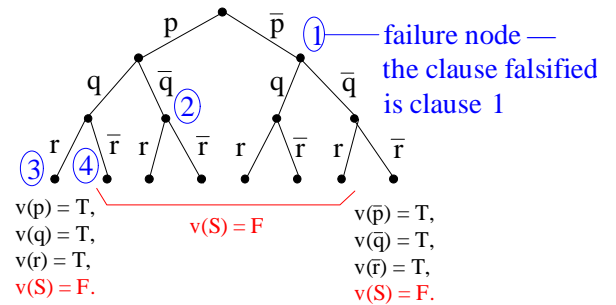
12th March 2002

**Completeness** ( $\Rightarrow$ ): If  $S$  is *unsatisfiable*, then the empty clause  $\square$  will be derived by the resolution procedure. *Two questions:* does the resolution procedure **terminate**, and will we get  $\square$ ? In the Resolution Completeness Proof, we will use the following *algorithm* for the construction of a Semantic Tree:

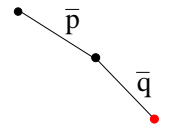
Construction of a Semantic Tree. **Input:** A set of *clauses*  $S$ . **Output:** A *semantic tree*  $T$  for  $S$  which is either *open* or *closed*. **Algorithm:** Let  $\{p_1, \dots, p_n\}$  be the propositional letters appearing in  $S$ . Form the *complete binary tree*  $T$  of depth  $n$ , and label the left-branching edges from a node of depth  $i-1$  by  $p_i$ , and the right-branching edges by  $\bar{p}_i$ .

Each branch  $b$  in  $T$  defines an *interpretation*  $v_b$  by  $v_b(p_i) = T$  if  $p_i$  labels the  $i^{\text{th}}$  edge in  $b$ , otherwise  $\bar{p}_i$  labels the  $i^{\text{th}}$  edge in  $b$ , and  $v_b(p_i) = F$ . A branch  $b$  is termed *closed* if  $v_b(S) = F$ , otherwise  $b$  is *open*.  $T$  is *closed* if all branches are closed, otherwise  $T$  is *open*.

**Example:** Consider  $S = \{p, \bar{p}q, \bar{r}, \bar{p}\bar{q}r\} =$  clauses  $\{1, 2, 3, 4\}$ . Propositional letters:  $\{p, q, r\}$ . We construct the tree as shown on the right. **Definition:** Let  $T$  be a closed semantic tree for a set of clauses  $S$ , and let  $b$  be a branch in  $T$ . The node in  $b$  *closest* to the root which falsifies  $S$  is a **failure node**. A clause falsified by a failure node is known as a closure *associated* with the failure node.



*Another Example:*  $S = \{q, \bar{q}\bar{p}, \bar{q}p\bar{r}, \bar{q}p\bar{r}\} =$  clauses  $\{1, 2, 3, 4\}$  as shown on the left (Propositional letters =  $\{p, q, r\}$ ). **Lemma 4.10:** A clause  $C$  associated with a failure node  $n$  is a *subset* of the complement of the literals appearing on the branch  $b$  to  $n$ . **Example:** In the example on the left, take *failure node 2*: *Subset* of literals on branch =  $\{p, q\}$ , *complement* of subset =  $\{\bar{p}, \bar{q}\}$ , *clause falsified* =  $\{\bar{q}\bar{p}\}$ . For the first failure node 1, *subset* =  $\{p, \bar{q}\}$ , *complement* =  $\{\bar{p}, q\}$ , and *clause falsified* =  $\{q\}$ . An *informal proof* is shown on the right, where the red node is a failure node, so that  $C$  is falsified, e.g.  $C = p\vee q$ . It follows that  $v(C) = F$ ,  $v(p\vee q) = F$ ,  $v(p) = F$  and  $v(q) = F$ .



13th March 2002

## Tutorial

**Q:** Transform the following formulae into *conjunctive normal form* (CNF) and then write them in the equivalent clausal form. **Note:** in any exam, any logical equivalences that you need will be given. Also note that you can use any sequence of (*logically equivalent*) steps you like to get to CNF — you don't need to use the exact sequence that we used in the proof of Theorem 4.1. (a)  $(p \leftrightarrow q) \wedge r \wedge s$ , (b)  $(p \rightarrow q) \vee (p \wedge q)$ .

**A:** (a)  $(p \leftrightarrow q) \wedge r \wedge s \equiv [(p \rightarrow q) \wedge (q \rightarrow p)] \wedge r \wedge s \equiv [(\neg p \vee q) \wedge (\neg q \vee p)] \wedge r \wedge s \equiv (\neg p \vee q) \wedge (\neg q \vee p) \wedge r \wedge s$ , CNF. *Clausal form:*  $\{\bar{p}q, \bar{q}p, r, s\}$ . (b)  $(p \rightarrow q) \vee (p \wedge q) \equiv (\neg p \vee q) \vee (p \wedge q) \equiv ((\neg p \vee q) \vee p) \wedge ((\neg p \vee q) \vee q) \equiv (\neg p \vee p \vee q) \wedge (\neg p \vee q \vee q) \equiv (\text{true} \vee q) \wedge (\neg p \vee q) \equiv \text{true} \wedge (\neg p \vee q) \equiv \neg p \vee q$ , CNF. *Clausal form:*  $\{\bar{p}q\}$ .

**Q:** For each set of clauses  $S$  below, derive the *simplest possible set*  $S'$  s.t.  $S \approx S'$ : (a)  $S = \{p, rs, prs, \bar{p}qrs\}$ , (b)  $S = \{qr, qrs, r\}$ , and (c)  $\{p\bar{p}qr, s, rs, qr\bar{s}, \bar{s}r\bar{q}\}$ . **A:** (a)  $r$  appears and  $\bar{r}$  does not, so  $S' = \{p\}$ . Now  $p$  appears and  $\bar{p}$  does not, so  $S'' = \emptyset$ , and so  $S$  is *satisfiable*.

(b)  $r$  is a *unit clause* so  $S' = \emptyset$ , and so  $S$  is *satisfiable*. (c)  $s$  is a unit clause so  $S' = \{p\bar{p}qr, qr, \bar{r}q\}$ . Now  $p$  and  $\bar{p}$  appear in the *first* clause, so  $S'' = \{qr, \bar{r}q\}$ . There is *no more* reduction.

**Q:** Work out the following resolvents and find a *mutually satisfying interpretation* for  $C_1$  and  $C_2$ . Show that the interpretation also satisfies their resolvent, noting that this is always the case — this is why in the resolution procedure we can add the resolvent to the original set of clauses and **not affect** the satisfiability of that set. (a)  $\text{Res}(qr, \bar{r})$ , (b)  $\text{Res}(pqr, \bar{q}r)$ , (c)  $\text{Res}(p, \bar{p})$ .

A: (a)  $\text{Res}(qr, \bar{r}) = q$ . Now  $v(q) = T$  and  $v(r) = F$  satisfies  $C_1$  and  $C_2$ , and this satisfies the resolvent *as well*. (b)  $\text{Res}(pqr, \bar{q}r) = pr$ .  $v(p) = T$ ,  $v(q) = T$  and  $v(r) = T$  satisfies  $C_1$  and  $C_2$ , and this satisfies the resolvent *as well*. (c)  $\text{Res}(p, \bar{p}) = \square$ . There is **no** interpretation which satisfies both  $C_1$  and  $C_2$ .

Q: Determine the *satisfiability* of the following sets of clauses by applying the resolution procedure. No **simplification** of the sets should be carried out before applying the procedure (i.e. don't work out a *simpler*  $S'$  (s.t.  $S \approx S'$ ) and apply the resolution procedure to that set). (a)  $S = \{\bar{p}r, p, q, \bar{p}\bar{q}r\}$ , (b)  $S = \{qrs, \bar{q}r\bar{s}, \bar{r}\bar{q}, \bar{q}\}$ , (c)  $S = \{qrs, \bar{s}pq, \bar{r}qp, w\bar{q}, \bar{p}w, \bar{w}\}$ , and (d)  $S = \{qrs, qr\bar{s}, \bar{p}\bar{r}, \bar{p}q, r\bar{q}, \bar{r}\bar{q}\}$ .

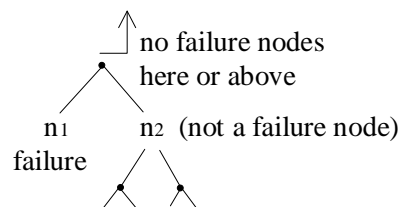
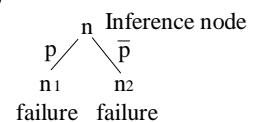
A: (a)  $S =$  clauses  $\{1, 2, 3, 4\}$ . Clashing clauses: 1 and 2:  $\text{Res}(\bar{p}r, p) = r$  (5); 3 and 4:  $\text{Res}(q, \bar{p}\bar{q}r) = \bar{p}r$  (6); 2 and 4:  $\text{Res}(p, \bar{p}\bar{q}r) = \bar{q}r$  (7). There is *nothing more to consider*, so  $S$  is satisfiable. (b)  $S =$  clauses  $\{1, 2, 3, 4\}$ . Clashing clauses: 1 and 2:  $\text{Res}(qrs, \bar{q}r\bar{s}) = rs\bar{s}$  (5) or  $q\bar{q}r$  (9). In this question, (as in (a)) we carry on finding clashing clauses until we cannot find anything new. Note that in order to verify that  $S$  is indeed *satisfiable*, reduce it ( $q$  is a unit clause) to give  $S' = \{rs\}$ , which is *obviously* satisfiable.

(c)  $S =$  clauses  $\{1, 2, 3, 4, 5, 6\}$ . Clashing clauses: 4 and 6:  $\text{Res}(w\bar{q}, \bar{w}) = \bar{q}$  (7); 5 and 6:  $\text{Res}(\bar{p}w, \bar{w}) = \bar{p}$  (8); 3 and 7:  $\text{Res}(\bar{r}qp, \bar{q}) = \bar{r}p$  (9); 8 and 9:  $\text{Res}(\bar{p}, \bar{r}p) = \bar{r}$  (10); 2 and 8:  $\text{Res}(\bar{s}pq, \bar{p}) = \bar{s}q$  (11); 7 and 11:  $\text{Res}(\bar{q}, \bar{s}q) = \bar{s}$  (12); 1 and 12:  $\text{Res}(qrs, \bar{s}) = qr$  (13); 10 and 13:  $\text{Res}(\bar{r}, qr) = q$  (14); 7 and 14:  $\text{Res}(\bar{q}, q) = \square$ . It follows that  $S$  is *unsatisfiable*.

(d)  $S =$  clauses  $\{1, 2, 3, 4, 5, 6\}$ . Clashing clauses: 1 and 2:  $\text{Res}(qrs, qr\bar{s}) = qr$  (7); 3 and 4:  $\text{Res}(\bar{p}\bar{r}, \bar{p}q) = \bar{r}q$  (8); 6 and 8:  $\text{Res}(\bar{r}\bar{q}, \bar{r}q) = \bar{r}$  (9); 5 and 9:  $\text{Res}(r\bar{q}, \bar{r}) = \bar{q}$  (10); 7 and 9:  $\text{Res}(qr, \bar{r}) = q$  (11); 10 and 11:  $\text{Res}(\bar{q}, q) = \square$ . It follows that  $S$  is *unsatisfiable*.

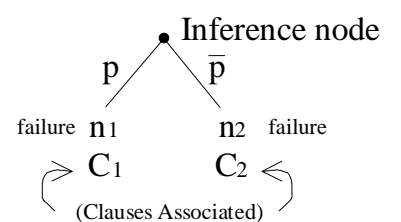
15th March 2002

**Definition:** Let  $n_1$  and  $n_2$  be failure nodes which are children of a node  $n$  —  $n$  is then called an inference node. **Lemma 4.11:** In a closed semantic tree  $T$  for a set of clauses  $S$ , there is *at least one inference node*. **Informal Proof:** Let's assume **no** inference nodes. Looking at the diagram on the left,  $n_1$  is a

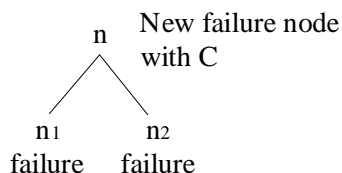


failure node while  $n_2$  is *not*. No *ancestor* of  $n_2$  is a failure node because  $n_1$  is a failure node — *by definition*  $n_1$  is the **closest** failure node to the root. But  $T$  is closed so there is *at least one* failure node in  $n_2$ 's subtree. If there are **no** inference nodes, then we get a strictly decreasing sequence of nodes which goes on forever **or** until you get an inference node.

**Lemma 4.12:** In a *closed semantic tree*  $T$ , let  $b$  be the branch from the root terminating at inference node  $n$ . The children  $n_1$  and  $n_2$  of  $n$  are failure nodes, so let  $C_1$  and  $C_2$  be any clauses associated with them respectively. Then  $C_1$  and  $C_2$  clash, and  $v_b$ , the partial interpretation associated with  $n$ , *falsifies*  $C$ , their **resolvent clause**.

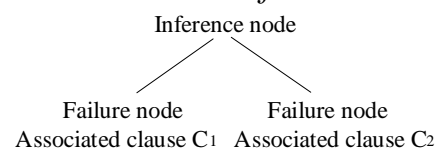


**Informal Proof:** (1)  $C_1$  and  $C_2$  clash. Assuming that  $b_1$  is the branch finishing at  $n_1$ , and that  $b_2$  is the branch finishing at  $n_2$ , then  $b_1$  and  $b_2$  are *identical* except for the edges  $n$  to  $n_1$  and  $n$  to  $n_2$ . By *Lemma 4.10*,  $C_1$  is a **subset** of the **complement** of the literals on  $b_1$ . Similarly for  $C_2$  and  $b_2$ . So  $C_1$  and  $C_2$  **clash** on (say)  $p$  and  $\bar{p}$ . (2)  $\forall_b(\text{Res}(C_1, C_2)) = F$ .  $n_1$  is a *failure node* so  $C_1$  is **not** satisfied — but  $C_1$  is a *disjunction*. Similarly for  $n_2$  and  $C_2$ . But  $\text{Res}(C_1, C_2) = (C_1 - \{p\}) \cup (C_2 - \{\bar{p}\})$ , e.g.  $\text{Res}(qrp, r\bar{p}) = qrs$  — therefore  $C$  is *false*.

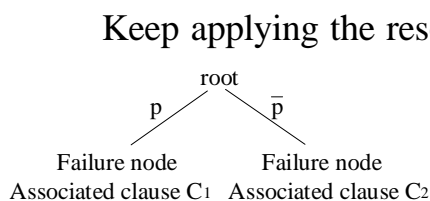


**Lemma 4.13:** Let  $n$  be an *inference node*, let  $C_1, C_2 \in S$  be clauses associated with the failure nodes that are the *children* of  $n$ , and let  $C$  be their resolvent. Then  $S \cup \{C\}$  has a failure node that is either  $n$  or is an *ancestor* of  $n$ , and  $C$  is a clause *associated* with that failure node.

**Completeness** (of the resolution procedure): If a set of clauses  $S$  is *unsatisfiable*, then  $\square$  (the empty clause) will be derived by the resolution procedure. **Proof:**  $S$  is an *unsatisfiable* set of clauses, so it follows that there is a closed semantic tree  $T$  for  $S$ . Clauses of  $S$  can be associated with failure nodes in  $T$ , and, by *Lemma 4.11*, there is *at least one* inference node in  $T$ .



An application of the *resolution rule* at this inference node will do the following **three** things: (1) Add the resolvent to the set  $S$ , i.e. add  $(\text{Res}(C_1, C_2))$ ; (2) Create a *new failure node* (by *Lemma 4.13*); and (3) Delete **two** failure nodes (remember that the definition of a failure node is the node **closest** to the root which falsifies  $S$ . So if you add a failure node *further up the branch*, the original failure node will *no longer* be a failure node).



Keep applying the resolution rule at *inference nodes* and eventually the number of failure nodes decreases to **two**, so you have the situation shown on the left. Applying the resolution rule at this inference node (i.e. the root) adds  $\text{Res}(C_1, C_2) = \square$  to the set, so it follows that the **empty clause** will be derived.

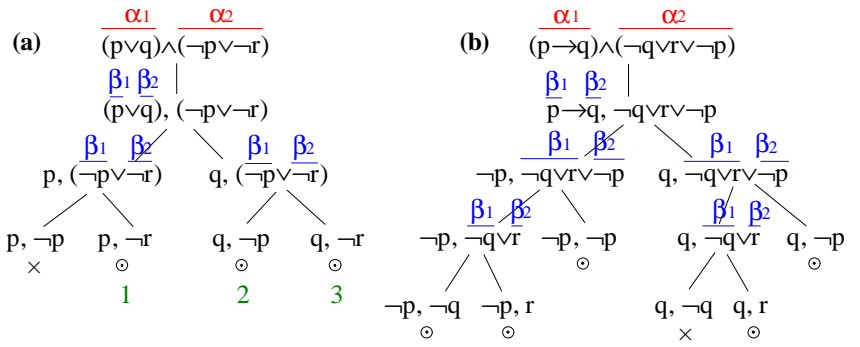
## Assignment 2: Set 15/3; In 12/4; Back 19/4

(1) (a) Construct the *semantic tableau* for the following formula and for each branch, where appropriate, construct the corresponding **Hintikka set**:  $(p \vee q) \wedge (\neg p \vee \neg r)$ . (b) Use a *semantic tableau* to discover a **model** for the following formula and *define* that model:  $(p \rightarrow q) \wedge (\neg q \vee r \vee \neg p)$ .

(c) Prove *Corollary 3.2*, that  $A$  is valid iff the tableau for  $\neg A$  closes. Hint: the proof should only take a *couple of lines*. (d) In the **inductive proof** of Soundness for semantic tableaux, we proved the case where an  $\alpha$  rule is used and the height of the subtree is *greater than zero*, i.e.  $h > 0$ . Prove the case where a  $\beta$  rule is used and  $h > 0$ .

A: (a) Looking at the tableau on the right, here are the Hintikka sets for the *three open leaves* in the tableau:

*Hintikka set 1*:  $\{(p \vee q) \wedge (\neg p \vee \neg r), (p \vee q), (\neg p \vee \neg r), p, \neg r\}$ ;  
*Hintikka set 2*:  $\{(p \vee q) \wedge (\neg p \vee \neg r), (p \vee q), (\neg p \vee \neg r), q, \neg p\}$ ;  
*Hintikka set 3*:  $\{(p \vee q) \wedge (\neg p \vee \neg r), (p \vee q), (\neg p \vee \neg r), q, \neg r\}$ .



(b) A *model* for the formula  $A = (p \rightarrow q) \wedge (\neg q \vee r \vee \neg p)$  is an interpretation  $v$  that satisfies  $A$ . As the shown tableau has an **open** leaf, then we know that there *is* an interpretation  $v$  that satisfies  $A$ . Looking at the *left-most* open leaf, we can define a model for  $A$  by setting  $v(p) = F$ ,  $v(q) = F$  and  $v(r) = T$ , say, where the choice for the truth assignment of  $r$  was *arbitrary*.

(c) Theorem 3.5 says that if  $T$  is a *completed semantic tableau* for a formula  $A$ , then  $A$  is *unsatisfiable* iff  $T$  is *closed*. Let  $T'$  be a completed semantic tableau for the formula  $\neg A$ . It follows from the above that  $\neg A$  is *unsatisfiable* iff  $T'$  is *closed*. Now Theorem 1.3 says that  $A$  is *valid* iff  $\neg A$  is *unsatisfiable*, and, using Theorem 3.5, we can therefore say that  $A$  is *valid* iff  $\neg A$  is *unsatisfiable* iff  $T'$  is *closed*, i.e. a formula  $A$  is *valid* iff the tableau for  $\neg A$  *closes*, as required. QED.

(d) **Claim:** If  $T$  is a *completed semantic tableau* for a formula  $A$ , then if  $T$  is *closed* then  $A$  is *unsatisfiable*. **Proof:** [Consider a subtree  $T$  rooted at  $n$ . We'll prove: if the subtree **closes**, then  $U(n)$  is **unsatisfiable**. Proof by induction on the *height*  $h$  of the node  $n$  in  $T$ .  $h = 0$ , base case:  $n$  is a *leaf*. We know that  $T$  closes so that  $U(n)$  contains a *complementary pair of literals*, and therefore  $U(n)$  is by definition *unsatisfiable*. Case  $h > 0$ : (a) An  $\alpha$  rule was used: this proof has been done *previously*].

(b) Case  $h > 0$  and a  $\beta$  rule was used: It follows that  $n$  has **two** children, say  $n'$  and  $n''$ , of height  $h-1$ ;  $U(n) = \{(B_1 \vee B_2)\} \cup U_0$ , where  $U_0$  is some *set of formulae*;  $U(n') = B_1 \cup U_0$ ; and  $U(n'') = B_2 \cup U_0$ . *Inductive step:* Assume that the theorem is **true** for  $h-1$ , i.e. that  $U(n')$  and  $U(n'')$  are *unsatisfiable* since the subtrees rooted at  $n'$  and  $n''$  *close*.

Let  $v$  be an *arbitrary* interpretation. We know that if the subtrees rooted at  $n'$  and  $n''$  **close**, then  $v(B') = F$  for some  $B' \in U(n')$ , and  $v(B'') = F$  for some  $B'' \in U(n'')$ . There are now **four** cases to consider: (i)  $B'$  is  $B_1$  and  $B''$  is  $B_2$ . In this situation, if  $v(B_1) = v(B_2) = F$ , then  $v(B_1 \vee B_2) = F$ , and so as  $B_1 \vee B_2 \in U(n)$ , and as  $v$  was chosen to be *arbitrary*, then it follows that  $U(n)$  is *unsatisfiable*.

(ii)  $B'$  is  $B_1$  and  $B'' \in U_0$ . If  $B'' \in U_0$ , with  $v(B'') = F$ , then as  $B'' \in U(n)$  (*because*  $B'' \in U_0$  and  $U_0 \in U(n)$ ), and because  $v$  was chosen to be *arbitrary*, then it follows that  $U(n)$  is **unsatisfiable**. (iii)  $B' \in U_0$  and  $B''$  is  $B_2$ . If  $B' \in U_0$ , with  $v(B') = F$ , then as  $B' \in U(n)$  (*because*  $B' \in U_0$  and  $U_0 \in U(n)$ ), and because  $v$  was chosen to be *arbitrary*, then it follows that  $U(n)$  is **unsatisfiable**. (iv)  $B' \in U_0$  and  $B'' \in U_0$ . Exactly the *same argument* as in parts (ii) or (iii). As in all possible cases we have shown that  $U(n)$  is **unsatisfiable**, then the result follows.

(2) The following *equivalences* may be used in the following sub questions:  $\neg(A \vee B) \equiv \neg A \wedge \neg B$ ,  $\neg(A \wedge B) \equiv \neg A \vee \neg B$ ,  $A \rightarrow B \equiv \neg A \vee B$ ,  $(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$ ,  $(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$ , and  $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$ . (a) Transform the following formula into *conjunctive normal form* and then write the formula as an equivalent set  $S$  of clauses. Derive the **simplest** set  $S'$  such that  $S$  is satisfiable if and only if  $S'$  is satisfiable, i.e.  $S \approx S'$ :  $(p \leftrightarrow q) \wedge ((q \rightarrow r) \vee (\neg p \wedge q)) \wedge \neg q$ .

(b) Determine the *satisfiability* of the following sets of clauses by applying the Resolution procedure. No *simplification* of the sets should be carried out **before** applying the procedure: (i)  $\{pqr, \bar{p}q, \bar{r}, \bar{q}\}$ , (ii)  $\{pqrs, \bar{p}rs, \bar{r}, \bar{s}, \bar{q}\}$ . (c) Let clause  $C$  be the *resolvent* of clauses  $C_1$  and  $C_2$ . Prove that if  $C_1$  and  $C_2$  are *mutually satisfiable*, then  $C$  is *also satisfiable*.

A: (a)  $(p \leftrightarrow q) \wedge ((q \rightarrow r) \vee (\neg p \wedge q)) \wedge \neg q \equiv (p \rightarrow q) \wedge (q \rightarrow p) \wedge ((q \rightarrow r) \vee (\neg p \wedge q)) \wedge \neg q \equiv (\neg p \vee q) \wedge (\neg q \vee p) \wedge ((\neg q \vee r) \vee (\neg p \wedge q)) \wedge \neg q \equiv (\neg p \vee q) \wedge (\neg q \vee p) \wedge [(\neg q \vee r) \vee \neg p] \wedge [(\neg q \vee r) \vee q] \wedge \neg q \equiv (\neg p \vee q) \wedge (\neg q \vee p) \wedge (\neg q \vee r \vee \neg p) \wedge (\neg q \vee r \vee q) \wedge \neg q \equiv$  an expression in CNF. The *equivalent clausal form* is given by  $S = \{\bar{p}q, \bar{q}p, \bar{q}r\bar{p}, \bar{q}rq, \bar{q}\}$ . Now  $\bar{q}$  is a *unit clause* so we can delete every clause containing  $\bar{q}$  and delete  $q$  from *every other clause* to give the set  $S' = \{\bar{p}\}$ . Now  $\bar{p}$  is a unit clause so we can *delete it* to give  $S'' = \emptyset$ , which shows that (because  $\emptyset$  is valid) the set  $S$  is **satisfiable** (e.g.  $v(p) = v(q) = v(r) = F$  satisfies  $S$ ).

(b) (i) Let  $S = \{pqr, \bar{p}q, \bar{r}, \bar{q}\} =$  clauses  $\{1, 2, 3, 4\}$ . Apply the *resolution procedure*: Clashing Clauses: 1 and 2:  $\text{Res}(pqr, \bar{p}q) = qr$  (5); 4 and 5:  $\text{Res}(\bar{q}, qr) = r$  (6); 3 and 6:  $\text{Res}(\bar{r}, r) = \square$  (7). Because  $\square$  has been *derived* using the resolution procedure, then we must conclude that  $S$  is **unsatisfiable**.

(ii) Let  $S = \{pqrs, \bar{p}rs, \bar{r}, \bar{s}, \bar{q}\} =$  clauses  $\{1, 2, 3, 4, 5\}$ . Apply the *resolution procedure*: Clashing clauses: 1 and 2:  $\text{Res}(pqrs, \bar{p}rs) = qrs$  (6); 3 and 6:  $\text{Res}(\bar{r}, qrs) = qs$  (7); 4 and 7:  $\text{Res}(\bar{s}, qs) = q$  (8); 5 and 8:  $\text{Res}(\bar{q}, q) = \square$  (9). Because  $\square$  has been *derived* using the resolution procedure, then we must conclude that  $S$  is **unsatisfiable**.

(c) **Claim**: If the clauses  $C_1$  and  $C_2$  are mutually satisfiable, then  $C = \text{Res}(C_1, C_2)$  is also satisfiable. **Proof**: Assume (w.l.o.g.) that  $C_1$  and  $C_2$  clash on the literal 'p', with  $p \in C_1$  and  $\bar{p} \in C_2$ . Knowing that  $C_1$  and  $C_2$  are *mutually satisfiable*, then there must be an interpretation  $v$  which makes **both**  $C_1$  and  $C_2$  satisfiable. There are now two cases to consider, dependent on the truth value that the *interpretation*  $v$  assigns to  $p$ .

Case 1:  $v(p) = T$ . If  $v(p) = T$ , then we *automatically* know that  $v(C_1) = T$  (*because*  $p \in C_1$ ), and so  $C_1$  is satisfiable. *Further*, because  $v(\bar{p}) = F$ , then there must be **another** literal  $q \in C_2$  with  $v(q) = T$  in order for  $C_2$  to be satisfiable. But this literal  $q$  *must* appear in  $C = \text{Res}(C_1, C_2)$  by the *definition* of  $\text{Res}(C_1, C_2)$ : we therefore conclude that as  $q \in C_2 - \{\bar{p}\}$ , then  $q \in \text{Res}(C_1, C_2) = C$ , so that the interpretation  $v$  that *mutually satisfies*  $C_1$  and  $C_2$  **also** satisfies  $C$  (as  $v(q) = T$ ). QED.

**Case 2:**  $v(p) = F$ . If  $v(p) = F$ , then  $v(\bar{p}) = T$ , and so we *automatically* know that  $v(C_2) = T$  (because  $\bar{p} \in C_2$ ), and so  $C_2$  is satisfiable. *Further*, because  $v(p) = F$ , then there must be **another** literal  $r \in C_1$  with  $v(r) = T$  in order for  $C_1$  to be satisfiable. But this literal  $r$  *must* appear in  $C = \text{Res}(C_1, C_2)$  by the *definition* of  $\text{Res}(C_1, C_2)$ : we therefore conclude that as  $r \in C_1 - \{p\}$ , then  $r \in \text{Res}(C_1, C_2) = C$ , so that the interpretation  $v$  that *mutually satisfies*  $C_1$  and  $C_2$  **also** satisfies  $C$  (as  $v(r) = T$ ). QED. End of Proof.

9th April 2002

## Summary of the First Half of the Course

	<b>Satisfiability</b> <i>Formula true for at least one interpretation</i>	<b>Unsatisfiability</b> <i>Formula cannot be true for any interpretation</i>	<b>Validity</b> <i>Formula A true for every interpretation</i>	<b>Equivalence</b> <i>Is A equivalent to B?</i>
<b>Truth Tables</b>	At least one T in the RHS column	All F in the RHS column	All T in the RHS column	The truth tables for A and B are identical
<b>BDDs</b>	At least one leaf with T in it	The OBDD can be reduced to just a boxed F	The OBDD can be reduced to just a boxed T	A and B are equivalent if their OBDDs are structurally identical
<b>Semantic Tableaux</b>	At least one open leaf	All leaves are closed	The tableau for $\neg A$ is closed	See if $A \leftrightarrow B$ is valid, i.e. whether the tableau for $\neg(A \leftrightarrow B)$ is closed
<b>Resolution</b>	Cannot derive the empty clause	Can derive the empty clause	Write $\neg A$ in clausal form and derive the empty clause	Write $\neg(A \leftrightarrow B)$ in clausal form and derive the empty clause

**Note:** If A is *valid*, then not every leaf of the semantic tableau for A is open, e.g. try  $(p \vee \neg p) \wedge (p \vee \neg p)$ .

	<b>Advantages</b>	<b>Drawback</b>	<b>Exponential or polynomial time?</b>
<b>Truth Tables</b>		Very inefficient — $2^n$ rows, where n is the number of atoms in the formula. Have to evaluate formula for each row	For some formulae the method runs in exponential time
<b>BDDs</b>	Very efficient for large formulae		For some formulae the method runs in exponential time
<b>Semantic Tableaux</b>	Easy and efficient for propositional calculus	Becomes arbitrary and inefficient for predicate calculus	Usually more efficient than truth tables but for some formulae the method runs in exponential time
<b>Resolution</b>	Efficient for both propositional and predicate calculus		Usually more efficient than truth tables but for some formulae the method runs in exponential time

**Note:** Due to algorithms that we have *not covered*, in practice BDDs can be more efficient than truth tables. **Cook** proved in 1971 that satisfiability in propositional logic is an **NP-Complete** problem, i.e. if an efficient algorithm to solve satisfiability can be found, then an efficient algorithm can be found for *every* NP problem. It is highly unlikely that there is a polynomial algorithm for satisfiability.

## Predicate Calculus

Consider the statement “*Every student is younger than some lecturer*”. (1) We have *predicates*  $S$ ,  $L$ ,  $Y$  and *constants* Kevin, Jane:  $S(\text{Kevin})$  denotes Kevin is a **student**,  $L(\text{Jane})$  denotes Jane is a **lecturer**, and  $Y(\text{Kevin}, \text{Jane})$  denotes Kevin is **younger** than Jane. (2) We have *variables*, e.g.  $x$ :  $S(x)$  denotes  $x$  is a **student**, ...,  $Y(x, y)$  denotes  $x$  is **younger** than  $y$ . (3) We have *quantifiers*:  $\forall$  means “for all”:  $\forall x A$  means “for **every**  $x$ ,  $A$  is *true*”;  $\exists$  means “exists”:  $\exists x A$  means “for **some**  $x$ ,  $A$  is *true*”.

**Examples:** (1)  $\exists x S(x)$  means there exists an  $x$  s.t.  $x$  is a student. (2)  $\exists x (S(x) \wedge L(x))$  means there exists an  $x$  s.t.  $x$  is a **student** and a **lecturer**. (3)  $\neg \exists x (S(x) \wedge L(x))$  means there does **not** exist an  $x$  who is a **student** and a **lecturer**. (4)  $\exists x, y (S(x) \wedge L(y) \wedge Y(x, y))$  means there is an  $x$  and a  $y$  s.t.  $x$  is a **student**,  $y$  is a **lecturer** and  $x$  is **younger** than  $y$ . (5)  $\forall x (L(x) \rightarrow S(x))$  means for every  $x$ , if  $x$  is a **lecturer** then  $x$  is a **student**, or every lecturer is a student.

(6)  $\neg \forall x (L(x) \rightarrow S(x))$  means **not every** lecturer is a student, or some lecturer is **not** a student. Our statement “*Every student is younger than some lecturer*” may now be written as  $\forall x (S(x) \rightarrow \exists y (L(y) \wedge Y(x, y)))$ : for every  $x$ , if  $x$  is a **student** then there exists a  $y$  s.t.  $y$  is a **lecturer** and  $x$  is **younger** than  $y$ .

Consider another example:  $B(x)$  denotes  $x$  is a **bird**,  $F(x)$  denotes  $x$  can **fly**, and  $Q(x)$  denotes  $x$  **quacks**. Interpretations:  $\exists x (B(x) \wedge F(x))$  means there exists an  $x$  which is a **bird** that **flies**;  $\neg \exists x (B(x) \wedge F(x))$  means there is **no** bird that can **fly**;  $\forall x (B(x) \rightarrow F(x))$  means for every  $x$ , if  $x$  is a **bird** then  $x$  can **fly**;  $\neg \forall x (B(x) \rightarrow F(x))$  means **not every** bird can fly;  $\exists x (B(x) \wedge Q(x))$  means there exists a bird that **quacks**; and  $\forall x (B(x) \rightarrow Q(x))$  means **all** birds quack.

12th April 2002

**Aside:** Consider 2 heat sensors.  $\text{Sensor1}(x)$  means temperature  $x$  is *detected* by sensor 1,  $\text{Sensor2}(x)$  means temperature  $x$  is *detected* by sensor 2,  $G(x, y)$  denotes  $x > y$ , and  $\text{alarm}(x)$  denotes that the Alarm is *sounded* and that the value of  $x$  is *displayed*. (1) If sensor 1 detects a temperature **greater** than that of sensor 2, then sound the alarm and display sensor 1’s temperature:  $\forall x \forall y [\text{Sensor1}(x) \wedge \text{Sensor2}(y) \wedge G(x, y) \rightarrow \text{alarm}(x)]$ . (2)  $\forall x \forall y [(\text{Sensor1}(x) \wedge \text{Sensor2}(y)) \rightarrow \neg G(x, y)]$  means Sensor 1 **never** detects a temperature *greater* than that of sensor 2 (*unrealistic*). (3)  $\forall x \forall y [(\text{Sensor1}(x) \wedge \text{Sensor2}(y) \wedge \neg G(x, y)) \rightarrow \neg \exists z \text{alarm}(z)]$ .

## Predicate Formulae

We have a set of *predicate letters*  $P$ , a set of *variables*  $V$ , and a set of *constants*  $A$ . A *term* is a **variable** or a **constant**. **Formula:** (1) If  $p$  is a predicate taking  $n$  arguments ( $n \geq 1$ ), and if  $t_1, \dots, t_n$  are terms, then  $p(t_1, \dots, t_n)$  is a formula. (2) If  $\phi$  is a formula, then so is  $\neg \phi$ ; and if  $\phi$  and  $\psi$  are formulae, then so are  $\phi \wedge \psi$ ,  $\phi \vee \psi$  and  $\phi \rightarrow \psi$ . (3) If  $\phi$  is a formula and if  $x$  is a variable, then  $\forall x \phi$  and  $\exists x \phi$  are formulae; and if  $p, q \in P$ ,  $a \in A$  and  $x, y \in V$ , then  $p(x)$ ,  $p(a)$ ,  $q(a, x)$  and  $q(x, y)$  are formulae.

**Scope.** In a *quantified formula*  $\exists x A$  or  $\forall x A$ ,  $x$  is called the *quantified variable* and  $A$  is the *scope* of the quantified variable. *Examples* (scope shown in red):  $\exists x p(x)$ ,  $\exists x (p(x) \wedge q(x))$ , and  $\exists x (p(x) \wedge \forall x (q(x) \wedge w(x)))$ . In the *final* formula, we may get rid of the *confusion* by replacing the formula with the formula  $\exists y (p(y) \wedge \forall x (q(x) \wedge w(x)))$ .

**Definition:** Let  $A$  be a *formula*. An occurrence of the variable  $x$  in  $A$  is a **free** variable of  $A$  iff  $x$  is **not** within the scope of the quantified variable  $x$ . A variable which is **not free** is **bound**, and a formula with **no** free variables is **closed**. Examples: in  $p(x, y)$ ,  $x$  and  $y$  are *free*; in  $\exists x p(x, y)$ ,  $x$  is *bound* while  $y$  is *free*; in  $\forall y \exists x p(x, y)$ ,  $x$  and  $y$  are *bound* (we have a *closed* formula); in  $p(x) \wedge \forall x q(x)$ , the first  $x$  is *free* while the second  $x$  is *bound*; and in  $\exists y (p(x, y) \wedge \forall x q(x, z))$ , *free variables* are shown in red while blue variables are *bound* ( $q, p \in P$ ;  $x, y, z \in V$ ).

## Interpretations

**Propositional:** For  $p \wedge q$ , set  $v(p) = T$  and  $v(q) = T$  so that  $v(p \wedge q) = T$ . **Predicate:** Consider  $\forall x p(a, x)$  ( $p \in P$ ,  $a \in A$ ,  $x \in V$ ). *Meaning:* For every  $x$ ,  $p(a, x)$  must be true. Interpretation 1 ( $I = (\mathbf{N}, \{<\}, \{22\})$ ):  $p$  is the *less than* relation,  $<$ , i.e.  $p(x, y)$  is true iff  $x < y$ ;  $a$  is 22; and  $x$  is taken from the set  $\mathbf{N}$  of *natural* numbers (+ve integers). Now  $\forall x p(a, x)$  means  $p(22, x)$  is true for *every*  $x \in \mathbf{N}$ , i.e.  $22 < x$  is true for *every*  $x \in \mathbf{N}$ , which is definitely **not** true (!), so that  $\forall x p(a, x)$  is false.

16th April 2002

Interpretation 2 ( $I = (\mathbf{N}, \{\leq\}, \{0\})$ ):  $p$  is  $\leq$ ,  $a$  is 0,  $x \in \mathbf{N}$ . Here,  $\forall x p(a, x)$  is true iff for every  $x$ ,  $a \leq x$ ; iff for every  $x$ ,  $0 \leq x$  — which is true. Interpretation 3 ( $I = (\mathbf{M}, \{\text{is the father of}\}, \{\text{Kevin}\})$ ):  $p$  denotes '*is the father of*' (so that  $p(x, y)$  is  $T$  iff  $x$  is the father of  $y$ );  $a = \text{Kevin}$ ;  $x \in \mathbf{M}$ , where  $\mathbf{M}$  is '*all Mankind*'. Here,  $\forall x p(a, x)$  is true iff for every  $x$ ,  $p(a, x)$  is true; iff for every  $x$ ,  $a$  is the *father* of  $x$ ; iff for every  $x$ , *Kevin* is the father of  $x$ , which is false.

Now consider  $\exists x p(a, x)$ : for **some**  $x$ ,  $p(a, x)$  is true. Interpretation 1:  $p$  is  $=$ ,  $a$  is 44,  $x$  is any integer ( $x \in \mathbf{Z}$ ). Here, for some  $x$ , we must have  $p(a, x) = T$ , i.e. we must have  $44 = x$ , where  $x \in \mathbf{Z}$  — which is *true* if we set  $x = 44$ . Interpretation 2:  $p$  denotes '*is the brother of*',  $a$  is *Mary*, and  $x \in \mathbf{M}$  (all Mankind again). Here, for some  $x$ , *Mary* must be the brother of  $x$ . *False?*

**Definition:** For a *predicate formula* with *predicates*  $p_1, \dots, p_m$ , *constants*  $a_1, \dots, a_k$ , and *variables*  $x_1, \dots, x_n$ , an *interpretation*  $I$  is a triple  $(D, \{R_1, \dots, R_m\}, \{d_1, \dots, d_k\})$ , where  $x_i$  is a *member* of  $D$ ,  $x_i \in D$ ;  $a_i$  is *assigned* a value  $d_i$  from  $D$ ; and  $p_i$  is *assigned* a relation  $R_i$  on  $D$ .

**Example:**  $\exists x (p(a, x) \wedge q(x) \wedge w(x, b))$ . Interpretation 1:  $I = (\mathbf{N}, \{>, \text{is even}, <\}, \{10, 13\})$ : There is an  $x$  s.t.  $a > x$ ,  $x$  is even and  $x < b$ , i.e. *there is an*  $x$  s.t.  $10 > x$ ,  $x$  is even and  $x < 13$  —  $x = 6$  is OK. Interpretation 2:  $I = (\mathbf{N}, \{=, \text{is even}, \text{is divisible by}\}, \{14, 7\})$ : There is an  $x$  s.t.  $14 = x$ ,  $x$  is even and  $w(x, 7)$  is true —  $x = 14$  is OK.

## Tutorial

Q:  $A(x)$  means  $x$  is an *animal*,  $F(x)$  means  $x$  has *four legs*, and  $T(x)$  means  $x$  has a *tail*. Translate the following into **predicate logic**: *Some animal has 4 legs*, *No animal has 4 legs*, *Every animal has four legs*, *If an animal has a tail then it has four legs*, and *No animal with four legs has a tail*. A:  $\exists x(A(x) \wedge F(x))$ ,  $\neg \exists x(A(x) \wedge F(x))$ ,  $\forall x(A(x) \rightarrow F(x))$ ,  $\forall x([A(x) \wedge T(x)] \rightarrow F(x))$ , and  $\neg \exists x(F(x) \wedge A(x) \wedge T(x))$ .

Q:  $B(x, y)$  means  $x$  *beats*  $y$ ,  $R(x)$  means  $x$  is a *rugby team*,  $P(y, x)$  means  $y$  is a *prop forward* for  $x$ ,  $L(x, y)$  means  $x$  *loses* to  $y$ , and Kevin, Llanelli and Merthyr are *constants*. Translate the following into **predicate logic**: *Every rugby team has a prop forward*, *If Llanelli beats Merthyr then Merthyr lost to Llanelli*, *Llanelli does not always lose*, and *if Kevin is a prop forward for team  $x$ , then  $x$  will always lose*. A:  $\forall x(R(x) \rightarrow \exists y(P(y, x)))$ ,  $B(\text{Llanelli}, \text{Merthyr}) \rightarrow L(\text{Merthyr}, \text{Llanelli})$ ,  $\exists y(\neg L(\text{Llanelli}, y))$ , and  $\forall x((P(\text{Kevin}, x) \wedge R(x)) \rightarrow \forall y(L(x, y)))$ .

Q: For the following predicate formulae, classify each variable occurrence as *free* or *bound*, and then determine if the formula is **closed** or not: (a)  $\forall x p(x, y)$ , (b)  $\forall x p(x, y) \vee \exists x P(y, x)$  ( $\forall, \exists, \neg$  bind most *tightly*; then  $\vee, \wedge$ ; then  $\rightarrow, \leftrightarrow$ ), (c)  $\exists y [\forall x p(x, y) \vee \exists x p(y, x)]$ .

A: (a) In  $p(x, y)$ , the  $x$  is *bound* and the  $y$  is *free* — so the formula is **not** closed. (b) The formula rewrites as  $[\forall x p(x, y)] \vee [\exists x P(y, x)]$ . In  $p(x, y)$ , the  $x$  is *bound* while the  $y$  is *free*; and in  $P(y, x)$ , the  $y$  is *free* while the  $x$  is *bound*. Conclusion: the formula is **not** closed. (c) All the variables are *bound* (the  $y$ 's this time to the first ' $\exists$ '), so the formula **is** closed.

Q: Consider the formula  $\exists x(p(a, x) \vee q(x))$ , where  $x$  is a *variable*,  $a$  is a constant, and  $p$  and  $q$  are predicates. Determine whether the formula is **true** for the following *interpretations* and explain your reasons: (a)  $I_1 = (\mathbf{Z}, \{<, \text{is odd}\}, \{0\})$ , where  $\mathbf{Z}$  is the set of *positive and negative integers*, and '*isodd*' denotes 'is an odd number'; (b)  $I_2 = (\mathbf{Z}^+, \{>, \text{isnegative}\}, \{0\})$ , where  $\mathbf{Z}^+$  is the set of positive integers, and '*isnegative*' denotes 'is a negative number'.

A:  $I_1$ : We need to find an  $x$  s.t.  $0 < x$  or s.t.  $x$  is an *odd* number — this can be achieved if  $x = 1$ , for instance, so that  $I_1 \models \exists x(p(a, x) \vee q(x))$ .  $I_2$ : We need to find an  $x$  s.t.  $0 > x$  or s.t.  $x$  is a -ve number: we cannot do this because there is **no** -ve  $x$  (because  $x \in \mathbf{Z}^+$ ), and there is **no**  $x \in \mathbf{Z}^+$  which is  $< 0$  — therefore the interpretation is *false*, i.e.  $I_2 \not\models \exists x(p(a, x) \vee q(x))$ .

Q: For each of the following formulae, write **two** interpretations, one that makes the formula *true*, and one which makes it *false*: (a)  $\forall x \exists y p(x, y)$ , (b)  $\forall x \exists y [p(x, y) \wedge q(y, a)]$ . A: (a)  $I_T = (\mathbf{Z}, \{<\}, \{ \})$ : for all  $x$ , there exists an integer  $y$  which is *greater* than  $x$  ( $x \in \mathbf{Z}$ ) — this is true, so that  $I_T \models \forall x \exists y p(x, y)$ .  $I_F = (\{\mathbf{Z}^+\}, \{>\}, \{ \})$ : for all  $x \in \mathbf{Z}^+$ , there exists a  $y \in \mathbf{Z}^+$  such that  $x > y$  — this is **not** true if  $x = 0$ , and so  $I_F \not\models \forall x \exists y p(x, y)$ . (b)  $I_T = (\mathbf{Z}^+, \{=, >\}, \{-1\})$ : for all +ve integers  $x$ , there exists a  $y \in \mathbf{Z}^+$  s.t.  $x = y$  and  $y$  is *greater* than  $-1$  — this is true, so that  $I_T \models \forall x \exists y [p(x, y) \wedge q(y, a)]$ .  $I_F = \{\mathbf{Z}^+, \{=, >\}, \{5\}\}$ : for all  $x \in \mathbf{Z}^+$ , there exists a  $y \in \mathbf{Z}^+$  s.t.  $x = y$  and  $y$  is *greater* than  $5$  — this is false for e.g.  $x = 4$ , so that  $I_F \not\models \forall x \exists y [p(x, y) \wedge q(y, a)]$ .

## The Truth Value of a Predicate Formula

[Handout] Let  $A$  be a *formula*,  $I$  an *interpretation*, and  $\sigma_I$  an *assignment*.  $v_{\sigma_I}(A)$ , the value of  $A$  under  $\sigma_I$ , is defined by *induction* on the structure of  $A$ : (a) Let  $A = p_k(c_1, \dots, c_n)$  be an *atomic formula*, where each  $c_i$  is either a **variable**  $x_i$  or a **constant**  $a_i$ .  $v_{\sigma_I}(A) = T$  iff  $\{d_1, \dots, d_n\} \in R_k$ , where  $R_k$  is the *relation* assigned by  $I$  to  $p_k$ , and  $d_i$  is the *domain element* assigned to  $c_i$ , either by  $I$  if  $c_i$  is a *constant*, or by  $\sigma_I$  if  $c_i$  is a *variable*.

(b)  $v_{\sigma_I}(\neg A) = T$  iff  $v_{\sigma_I}(A) = F$ . (c)  $v_{\sigma_I}(A_1 \vee A_2) = T$  iff  $v_{\sigma_I}(A_1) = T$  or  $v_{\sigma_I}(A_2) = T$ , and *similarly* for the other Boolean operators  $\wedge$ ,  $\rightarrow$  and  $\leftrightarrow$ . (d)  $v_{\sigma_I}(\forall x A_1) = T$  iff  $v_{\sigma_I[x \leftarrow d]}(A_1) = T$  for **all**  $d \in D$ . (e)  $v_{\sigma_I}(\exists x A_1) = T$  iff  $v_{\sigma_I[x \leftarrow d]}(A_1) = T$  for **some**  $d \in D$ .

[Notes] Let  $I$  be an *interpretation*. An assignment  $\sigma_I: V \rightarrow D$  is a function which maps every *variable* of  $I$  to an *element* of the domain of  $I$ . As an example, consider  $\exists x, y p(x, y)$  with  $I = (\mathbf{N}, \{<\}, \{ \})$ . Here,  $\sigma_I$  is defined as  $\sigma_I: \{x, y\} \rightarrow \mathbf{N}$ , e.g.  $x \leftarrow 4$  and  $y \leftarrow 5$ . Further,  $\sigma_I[x_i \leftarrow d]$  is an assignment that is the *same* as  $\sigma_I$  except that  $x_i$  is mapped to  $d$ . With  $\sigma_I[x \leftarrow 66]$ , for example, we would have  $y \leftarrow 5$  and  $x \leftarrow 66$ .

Let us now consider  $I = (\mathbf{N}, \{iseven\}, \{22\})$ . (1) When is  $p(a)$  (for  $a \in A$ ) true? Now  $v_{\sigma_I}(p(a)) = T$  iff {value assigned to  $a$  by  $I$ }  $\in iseven$  ( $iseven = \{x \mid x \text{ is even, } x \in \mathbf{N}\}$ ). As  $\{22\} \in iseven$ , then the value of  $p(a)$  under the assignment  $\sigma_I$  is **true**. (2) When is  $p(x)$  (for  $x \in V$ ) true? Now  $v_{\sigma_I}(p(x)) = T$  iff {value assigned to  $x$  by  $\sigma_I$ }  $\in iseven$  — but we cannot conclude from this *whether or not*  $p(x)$  is true. (3) When is  $\exists x p(x)$  (for  $x \in V$ ) true? Now  $v_{\sigma_I}(\exists x p(x)) = T$  iff  $v_{\sigma_I[x \leftarrow d]}(p(x)) = T$  for some  $d \in \mathbf{N}$ , iff {value assigned to  $x$  by  $\sigma_I[x \leftarrow d]$ }  $\in iseven$  for some  $d \in \mathbf{N}$ , iff  $\{d\} \in iseven$  for some  $d \in \mathbf{N}$  — which is *true*.

Definition: A **closed** formula  $A$  is true in  $I$  if  $v_{\sigma_I}(A) = T$ , written  $I \models A$ . In this situation,  $I$  is a **model** for  $A$ . *Example*: Consider  $A: \exists y p(a, y)$  with  $I_1 = (\mathbf{N}, \{<\}, \{22\})$ . Now  $v_{\sigma_I}(A) = T$  iff  $v_{\sigma_I[y \leftarrow d]}(p(a, y)) = T$  for some  $d \in \mathbf{N}$ , iff  $(22, d) \in <$  for some  $d \in \mathbf{N}$  (where  $< = \{(x, w) \mid x < w; x, w \in \mathbf{N}\}$ ), which is *true*, so that  $v_{\sigma_I}(A) = T$ , i.e.  $I_1 \models A$  (so that  $A$  is *satisfiable*).

Definition: A formula  $A$  is *satisfiable* if for some interpretation  $I$  we have  $I \models A$ . *Example*: Consider  $A: \exists y p(a, y)$  with  $I_2 = (\mathbf{Z}^+, \{>\}, \{0\})$ . Now  $v_{\sigma_I}(A) = T$  iff ... iff  $(0, d) \in >$  for some  $d \in \mathbf{Z}^+$ . This *cannot* be true, so that  $v_{\sigma_I}(A) = F$ , and so  $I_2 \not\models A$  ( $A$  is *falsifiable*). *Example*: Consider  $A: \forall x p(x) \rightarrow \exists x p(x)$ . **Meaning**: if  $p(x)$  is true for *every*  $x$ , then  $p(x)$  is true for *some*  $x$ . This is true for **all interpretations** so that  $A$  is valid, written  $\models A$ .

Definition: A formula  $A$  is *valid*, written  $\models A$ , if for **all** interpretations  $I$  we have  $I \models A$ . Definition: A formula  $A$  is *falsifiable* if  $A$  is not valid. *Example*: Consider  $A: \neg \exists x p(x) \wedge p(a)$  for  $a \in A$  and  $x \in V$ . **Meaning**:  $p(a)$  is true, and it is **not** true that there is an  $x$  such that  $p(x)$  is true ( $a, x \in D$ ). As we cannot have **both**  $p(x)$  true for no  $x$  and  $p(a)$  true for  $a$ , then it follows that  $A$  is *unsatisfiable*. Definition: A formula is *unsatisfiable* if it is **not** satisfiable.

**Theorem 5.1:** If  $A$  is *not* a closed formula (i.e. if  $A$  has free variables  $x_1, \dots, x_n$ ), and if  $I$  is an *interpretation*, then (a)  $v_{\sigma_I}(A) = T$  for *some* assignment  $\sigma_I$  iff  $v_{\sigma_I}(\exists x_1, \dots, \exists x_n A) = T$ ; (b)  $v_{\sigma_I}(A) = T$  for *all* assignments  $\sigma_I$  iff  $v_{\sigma_I}(\forall x_1, \dots, \forall x_n A) = T$ . Example:  $v_{\sigma_I}(p(x)) = T$  for *some*  $\sigma_I$  iff  $v_{\sigma_I}(\exists x p(x)) = T$ , and  $v_{\sigma_I}(p(x)) = T$  for *all*  $\sigma_I$  iff  $v_{\sigma_I}(\forall x p(x)) = T$ .

## Logical Equivalence

Definition: Given *two* closed formulae  $A_1$  and  $A_2$ , if  $v_{\sigma_I}(A_1) = v_{\sigma_I}(A_2)$  for *all* interpretations  $I$ , then  $A_1$  is *logically equivalent* to  $A_2$ , written  $A_1 \equiv A_2$ . **Theorem 5.2:**  $A \equiv B$  iff  $\models A \leftrightarrow B$ , i.e.  $A \leftrightarrow B$  is **valid**. Memorise These:  $\forall x A \equiv \neg \exists x \neg A$ , and  $\exists x A \equiv \neg \forall x \neg A$ . *Tautologies* in propositional logic also apply to predicate logic, e.g. knowing that  $A \vee \neg A$  is a tautology in *propositional logic* implies that  $(\forall x A) \vee \neg(\forall x A)$  is a tautology in *predicate logic*.

**Q:** Prove  $\exists x A(x) \equiv \neg \forall x \neg A(x)$ , i.e. prove that  $\exists x A(x) \leftrightarrow \neg \forall x \neg A(x)$  is valid. **A:** ( $\rightarrow$ ) Suppose that  $\exists x A(x)$  is true in **some** interpretation. Then there exists an  $x_0 \in D$  s.t.  $A(x_0)$  is true. Now assume that  $\neg \forall x \neg A(x)$  is *false* in this interpretation, then  $\forall x \neg A(x)$  is true, i.e. for all  $x_1 \in D$ ,  $\neg A(x_1)$  is true, i.e. for all  $x_1 \in D$ ,  $A(x_1)$  is **false**. But we have now found a *contradiction*, so it follows that we have proved that  $\exists x A(x) \rightarrow \neg \forall x \neg A(x)$  is valid.

( $\leftarrow$ ) Suppose that  $\neg \forall x \neg A(x)$  is true in **some** interpretation, then  $\forall x \neg A(x)$  is *false* in that interpretation, i.e. for all  $x_1 \in D$ ,  $\neg A(x_1)$  is false, i.e. for *all*  $x_1 \in D$ ,  $A(x_1)$  is true. Let us now *assume* that  $\exists x A(x)$  is **false** in this interpretation, then there exists an  $x_0 \in D$  s.t.  $A(x_0)$  is false. But this is *another* contradiction, so we have proved that  $\exists x A(x) \leftarrow \neg \forall x \neg A(x)$  is valid. **Conclusion:** Whenever the LHS is true then the RHS must be true and *vice-versa* — so that  $LHS \equiv RHS$ .

Now consider  $\forall x \exists y p(x, y) \rightarrow \exists x \forall y p(x, y)$ , with  $I = (\mathbf{N}, \{<\}, \{ \})$ . LHS:  $\forall x \exists y p(x, y)$  means  $\forall x \exists y x < y$ , i.e. for all  $x \in \mathbf{N}$ , there exists a  $y \in \mathbf{N}$  s.t.  $x < y$ , which is *true*. RHS:  $\exists x \forall y p(x, y)$  means  $\exists x \forall y x < y$ , i.e. there exists an  $x \in \mathbf{N}$  s.t. for all  $y \in \mathbf{N}$ , we have  $x < y$ . Now if  $y = 0$ , then there is *no such*  $x$  — so the RHS is not true.

Now consider  $\models \forall x A(x) \rightarrow \exists x A(x)$ . For this not to be *valid*, then  $\forall x A(x)$  must be **true** and also  $\exists x A(x)$  must be **false**. But if  $\forall x A(x)$  is true, then for all  $x_0 \in D$ , we must have  $A(x_0)$  true. *Further*, if  $\exists x A(x)$  is *false*, then  $\neg \forall x \neg A(x)$  is *false*, i.e.  $\forall x \neg A(x)$  is *true*, i.e. for all  $x_1 \in D$ ,  $\neg A(x_1)$  is *true*, i.e. for all  $x_1 \in D$ ,  $A(x_1)$  is *false*. But this is a **contradiction**, so when  $\forall x A(x)$  is true, then  $\exists x A(x)$  must also be true — so it's a *valid* formula.

Now consider (a)  $[\forall x A(x) \vee \forall x B(x)] \rightarrow \forall x (A(x) \vee B(x))$ , and (b)  $\forall x (A(x) \vee B(x)) \rightarrow [\forall x A(x) \vee \forall x B(x)]$ , with  $I = (\mathbf{N}, \{\text{isodd, iseven}\}, \{ \})$ . (a) Meaning: If all  $x$  are *odd* or if all  $x$  are *even*, **then** all  $x$  are *odd* or *even*. The LHS is *false*, so using  $F \rightarrow ? \equiv T$ , we can therefore "factor the  $\forall x$  out". (b) Meaning: If all  $x$  are *odd* or *even* **then** either all  $x$  are *odd* or all  $x$  are *even*. In this situation, we have  $T \rightarrow F \equiv F$ , so that the formula is **not** valid, i.e. we *can't* "multiply the  $\forall x$  inwards".

## Assignment 3: Set 23/4; In 7/5; Back 10/5

Q:  $S(x, y)$  means  $x$  is the *sister* of  $y$ ,  $H(x, y)$  means  $x$  is the *husband* of wife  $y$ , and  $P(x)$  means  $x$  *plays the piano*. Translate the following into **predicate logic**: There *exists* a sister who plays the piano, *All* Fred's sisters play the piano, *None* of Fred's sisters play the piano, and *All* Fred's *sisters-in-law* play the piano. A:  $\exists x, y (S(x, y) \wedge P(x))$ ,  $\forall x (S(x, \text{Fred}) \rightarrow P(x))$ ,  $\neg \exists x (S(x, \text{Fred}) \wedge P(x))$ , and  $\forall x, y ((S(x, y) \wedge H(\text{Fred}, y)) \rightarrow P(x))$ .

Q: Determine if the formula  $\exists x \forall y (p(a, x) \wedge (q(y, x) \vee w(y, x)))$  is **true** for the following interpretations (*explain* your answers):  $I = (\mathbf{Z}, \{>, \geq, <\}, \{13\})$ , where  $\mathbf{Z}$  is the set of all *integers*; and  $I = (\mathbf{N}, \{=, \neq, <\}, \{99\})$ , where  $\mathbf{N}$  is the set of *natural numbers*.

A: Meaning of Interpretation 1: There exists an  $x$  such that *for all*  $y$ , we have  $13 > x$  and ( $y \geq x$  or  $y < x$ ), i.e. **there exists** an  $x < 13$  such that for all  $y$ , we either have  $y \geq x$  or  $y < x$ . But *it doesn't matter* which  $x$  less than 13 we choose from the set of integers — in all cases, an arbitrary integer  $y$  will always be **either** greater than or equal to  $x$  **or** less than  $x$  — so we conclude that the interpretation is true for any  $x < 13$ , e.g.  $x = 12$ .

Meaning of Interpretation 2: There exists an  $x$  such that *for all*  $y$ , we have  $99 = x$  and ( $y = x$  or  $y < x$ ). The **only** choice for  $x$  that satisfies  $99 = x$  is obviously  $x = 99$ , but with this choice of  $x$ , we *cannot* have either  $y = x$  or  $y < x$  for *all*  $y \in \mathbf{N}$  — take for example  $y = 100$ , in which case neither  $x = 99$  or  $y < 99$  is true — we therefore conclude that the interpretation is *false*.

Q: Prove that  $\forall x p(x) \equiv \neg \exists x \neg p(x)$  (remember to prove **both** directions of  $\leftrightarrow$ ). A: In order to prove that  $\forall x p(x) \equiv \neg \exists x \neg p(x)$ , we need to prove that  $\forall x p(x) \leftrightarrow \neg \exists x \neg p(x)$  is valid. ( $\rightarrow$ ). The *only way* in which  $\forall x p(x) \rightarrow \neg \exists x \neg p(x)$  is **false** is when  $\forall x p(x)$  is **true** and  $\neg \exists x \neg p(x)$  is **false**. Let us now prove by *contradiction* that this cannot happen. Suppose that  $\forall x p(x)$  is **true** in some interpretation. Then for all  $x \in D$  we know that  $p(x)$  is true.

Now *assume* that  $\neg \exists x \neg p(x)$  is false in this interpretation, which *means* that  $\exists x \neg p(x)$  is true in this interpretation. But if this **is** the case, then there exists an  $x \in D$  which makes  $\neg p(x)$  true, i.e. there exists an  $x \in D$  which makes  $p(x)$  false. But this *contradicts* our earlier statement about  $p(x)$  being true for **all**  $x \in D$ , which means that there is **no** interpretation which makes  $\forall x p(x)$  true and  $\neg \exists x \neg p(x)$  false, so that  $\forall x p(x) \rightarrow \neg \exists x \neg p(x)$  must be valid.

( $\leftarrow$ ) The only way in which  $\neg \exists x \neg p(x) \rightarrow \forall x p(x)$  is **false** is when  $\neg \exists x \neg p(x)$  is **true** and  $\forall x p(x)$  is **false**. Let us now prove by *contradiction* that this cannot happen. Suppose that  $\neg \exists x \neg p(x)$  is **true** in some interpretation. This says that there *does not exist* an  $x \in D$  which makes  $\neg p(x)$  true, i.e. there does **not** exist an  $x \in D$  which makes  $p(x)$  false.

Now *assume* that  $\forall x p(x)$  is *false* in this interpretation. This means that in this interpretation, it is **not true** to say that for all  $x \in D$   $p(x)$  is true, i.e. there must be *at least one*  $x \in D$  which makes  $p(x)$  false. But this *contradicts* our earlier statement about there **not** being an  $x \in D$  which makes  $p(x)$  false, which means that there is **no** interpretation which makes  $\neg \exists x \neg p(x)$  true and  $\forall x p(x)$  false, so that  $\neg \exists x \neg p(x) \rightarrow \forall x p(x)$  must be **valid**.

Now because we have shown that **both**  $\forall x p(x) \rightarrow \neg \exists x \neg p(x)$  and  $\neg \exists x \neg p(x) \rightarrow \forall x p(x)$  are **valid**, then it follows that  $\forall x p(x) \leftrightarrow \neg \exists x \neg p(x)$  is valid, and so we have *proved* the statement  $\forall x p(x) \equiv \neg \exists x \neg p(x)$ . QED.

Q: Prove that the formula  $\exists x (p(x) \wedge q(x)) \rightarrow (\exists x p(x) \wedge \exists x q(x))$  is *valid*. A: The **only** way in which  $\exists x (p(x) \wedge q(x)) \rightarrow (\exists x p(x) \wedge \exists x q(x))$  is **false** is when  $\exists x (p(x) \wedge q(x))$  is **true** and  $\exists x p(x) \wedge \exists x q(x)$  is **false**. Let us now prove by *contradiction* that this cannot happen. Suppose that  $\exists x (p(x) \wedge q(x))$  is true in *some* interpretation. This means that there exists an  $x \in D$  such that **both**  $p(x)$  and  $q(x)$  are true at the *same time*.

Now assume that  $\exists x p(x) \wedge \exists x q(x)$  is **false** in this interpretation. Using  $\exists x A \equiv \neg \forall x \neg A$ , this means that  $\neg \forall x \neg p(x) \wedge \neg \forall x \neg q(x)$  is false in this interpretation. Therefore, *either* (i)  $\neg \forall x \neg p(x)$  is *false* and  $\neg \forall x \neg q(x)$  is *true*, (ii)  $\neg \forall x \neg p(x)$  is *true* and  $\neg \forall x \neg q(x)$  is *false*, or (iii) *both*  $\neg \forall x \neg p(x)$  and  $\neg \forall x \neg q(x)$  are *false*. (i) If  $\neg \forall x \neg p(x)$  is *false*, then  $\forall x \neg p(x)$  is true, which implies that there **does not** exist an  $x \in D$  making  $p(x)$  true, which *contradicts* our previous statement regarding there being an  $x \in D$  making **both**  $p(x)$  and  $q(x)$  true at the *same time*. (ii) As in (i) but with  $q$ 's replacing  $p$ 's. (iii) As in *either* (i) or (ii).

As we have shown that there is a **contradiction** in *each* of the situations where  $\neg \forall x \neg p(x) \wedge \neg \forall x \neg q(x)$  could be *false*, then we conclude that  $\exists x p(x) \wedge \exists x q(x)$  has to be **true** when  $\exists x (p(x) \wedge q(x))$  is true — and so  $\exists x (p(x) \wedge q(x)) \rightarrow (\exists x p(x) \wedge \exists x q(x))$  must be *valid*. QED.

26th April 2002

Prove that the following formula is *valid*, i.e. true for **every** interpretation:  $[(\forall x A(x) \vee \forall x B(x)) \rightarrow \forall x (A(x) \vee B(x))]$ . **Proof.** Say  $[\forall x A(x) \vee \forall x B(x)]$  is true — then *either*  $A(x_1)$  is true for all  $x_1 \in D$  (call this 1), or  $B(x_2)$  is true for all  $x_2 \in D$  (call this 2). Now let  $\forall x (A(x) \vee B(x))$  be *false* — then  $\neg \exists x \neg (A(x) \vee B(x))$  is false (using the *equivalence*  $\neg \exists x \neg C(x) \equiv \forall x C(x)$ ).

But then  $\exists x \neg (A(x) \vee B(x))$  is true, and so for some  $x_3 \in D$ ,  $\neg (A(x_3) \vee B(x_3))$  is true, i.e.  $\neg A(x_3) \wedge \neg B(x_3)$  is true (using the *equivalence*  $\neg (C \vee D) \equiv \neg C \wedge \neg D$ ), i.e.  $\neg A(x_3)$  and  $\neg B(x_3)$  are true, i.e. for some  $x_3 \in D$ ,  $A(x_3)$  and  $B(x_3)$  are false. But this **contradicts** 1 and 2, so if the LHS of  $\rightarrow$  is true then the RHS of  $\rightarrow$  **must** be true — and if the LHS of  $\rightarrow$  is false then we *don't care* if the RHS is true or false — so the formula is true for **every** interpretation and is therefore *valid*.

Now consider that  $M(x, y)$  means that  $x$  is  $y$ 's *mother*, that  $Eq(x, y)$  means that  $x$  and  $y$  are the *same*, and that Andy and Paul are *constants*. **Express** the following: "*Andy and Paul have the same maternal grandmother*", i.e. Andy's mother's mother = Paul's mother's mother. Using the techniques that we have looked at **so far**, we would write  $\forall x \forall y \forall u \forall v [(M(x, y) \wedge M(y, Andy) \wedge M(u, v) \wedge M(v, Paul)) \rightarrow Eq(x, u)]$ .

Consider now that we have a *function*  $m$ , where  $m(y)$  returns the **mother** of  $y$ . Using the function, the expression above could be written as  $Eq(m(mAndy)), m(mPaul))$ . Let us now look more closely at the *use* of functions of this type.

## Predicates Extended with Functions

**Interpretations.** For a formula with *predicates*  $p_1, \dots, p_m$ , constants  $a_1, \dots, a_k$ , variables  $x_1, \dots, x_n$ , and functions  $f_1, \dots, f_j$ , an interpretation  $I$  is a 4-tuple  $I = (D, \{R_1, \dots, R_m\}, \{F_1, \dots, F_j\}, \{d_1, \dots, d_k\})$ , where  $x_i$  is a *member* of  $D$ ,  $x_i \in D$ ;  $a_i$  is assigned a *value*  $d_i$  from  $D$ ;  $p_i$  is assigned a *relation*  $R_i$  on  $D$ ; and  $f_i$  is assigned a *function*  $F_i$  on  $D$ .

**Predicate Formulae.** For the set of *predicate letters*  $P$ , the set of *variables*  $V$ , the set of *constants*  $A$ , and the set of *functions*  $F$ , a **term** is any variable in  $V$ , any constant in  $A$ , or any expression of the form  $f(t_1, \dots, t_n)$ , where  $t_1, \dots, t_n$  are terms and  $f \in F$  has arity  $n$ . *Similarly*, a **predicate formula** can be *specified* as follows: if  $p$  is a predicate taking  **$n$  arguments** ( $n \geq 1$ ), and if  $t_1, \dots, t_n$  are terms, then  $p(t_1, \dots, t_n)$  is a formula; if  $\phi$  is a formula then *so is*  $\neg\phi$ ; if  $\phi$  and  $\psi$  are formulae then *so are*  $\phi \wedge \psi$ ,  $\phi \vee \psi$ ,  $\phi \rightarrow \psi$  and  $\phi \leftrightarrow \psi$ ; and if  $\phi$  is a formula with  $x$  a variable, then  $\forall x \phi$  and  $\exists x \phi$  are formulae.

Now *consider*  $A = \forall x \forall y [p(x, y) \rightarrow p(f(x, a), f(y, a))]$  with  $I = (\mathbf{N}, \{>\}, \{+\}, \{22\})$ . Meaning: for all  $x, y \in \mathbf{N}$ , if  $x > y$  then  $f(x, 22) > f(y, 22)$ , i.e.  $x+22 > y+22$ . This is *true*, so that  $I \models A$ . Similarly, *consider*  $A = \forall x \forall y [p(x, y) \rightarrow p(f(x, a), f(y, a))]$  with  $I = (\mathbf{N}, \{>\}, \{*\}, \{0\})$ . Meaning: for all  $x, y \in \mathbf{N}$ , if  $x > y$  then  $f(x, 0) > f(y, 0)$ , i.e.  $x*0 > y*0$ . This is **not** true, so that  $I \not\models A$ .

New situation: Consider that  $C(x)$  means  $x$  is a *child*, that  $Y(x, y)$  means  $x$  is *younger* than  $y$ , that  $M(x, y)$  means  $x$  is  $y$ 's *mother*, and that  $Eq(x, y)$  means  $x$  and  $y$  are the *same*. Q1: (a) Express the following: "*Every child is younger than its mother*". (b) Use the function  $m$ , where  $m(y)$  returns  $y$ 's mother, to **express** the sentence in (a).

A1: (a)  $\forall x \forall y [(C(x) \wedge M(y, x)) \rightarrow Y(x, y)]$ , (b)  $\forall x (C(x) \rightarrow Y(x, m(x)))$ . Q2: (a) Express the following: "*Sally and Sue do not have the same mother*". (b) Use the function  $m$ , where  $m(y)$  return's  $y$ 's mother, to **express** the sentence in (a). A2: (a)  $\forall x \forall y ((M(y, Sally) \wedge M(x, Sue)) \rightarrow \neg Eq(x, y))$ , (b)  $\neg(Eq(m(Sally), m(Sue)))$ . **Note**: Only use functions where we want to denote a **single** object, for example  $m(x)$  returning  $x$ 's genetic *mother* is OK, but  $b(x)$  returning  $x$ 's *brothers* is not — it is ambiguous — consider  $Y(\text{Paul}, b(\text{Paul}))$ .

**The truth value of a predicate formula.** Let  $A$  be a *formula*,  $I$  an *interpretation*, and  $\sigma_I$  an *assignment*.  $v_{\sigma_I}(A)$ , the value of  $A$  under  $\sigma_I$ , is defined by *induction* on the structure of  $A$  as follows (where the *first three statements* are **new** statements):  $v_{\sigma_I}(a_i) = d_i$  for  $a_i$  a *constant*,  $a_i \in A$ ;  $v_{\sigma_I}(f_i(t_1, \dots, t_n)) = F_i(v_{\sigma_I}(t_1), \dots, v_{\sigma_I}(t_n))$ ;  $v_{\sigma_I}(p_i(t_1, \dots, t_n)) = T$  iff  $(v_{\sigma_I}(t_1), \dots, v_{\sigma_I}(t_n)) \in R_i$ ;  $v_{\sigma_I}(\neg A) = T$  iff  $v_{\sigma_I}(A) = F$ ;  $v_{\sigma_I}(A_1 \vee A_2) = T$  iff  $v_{\sigma_I}(A_1) = T$  or  $v_{\sigma_I}(A_2) = T$  — and similarly for the *other* Boolean operators  $\wedge$ ,  $\rightarrow$  and  $\leftrightarrow$ ;  $v_{\sigma_I}(\forall x A_1) = T$  iff  $v_{\sigma_I[x \leftarrow d]}(A_1) = T$  for **all**  $d \in D$ ; and  $v_{\sigma_I}(\exists x A_1) = T$  iff  $v_{\sigma_I[x \leftarrow d]}(A_1) = T$  for **some**  $d \in D$ .

## Resolution in Predicate Logic

**Definition:** A closed formula is in clausal form iff it is of the form  $Q_1x_1\dots Q_nx_nM$ , where the  $Q_i$  are universal quantifiers ( $\forall$ ) and  $M$  is a *quantifier free formula* in CNF. **Example:**  $\forall x\forall y (p(x)\wedge q(y))$  and  $\forall x\forall y\forall z [(p(x)\vee q(y))\wedge w(z)]$  are written in *clausal form* as  $\{\{p(x)\}, \{q(y)\}\}$  and as  $\{\{p(x), q(y)\}, \{w(z)\}\}$ . Notice that  $\exists x (p(x)\wedge q(x))$ ,  $\forall x p(x) \wedge \exists y q(y)$  and  $\forall x p(x) \wedge \forall y q(y)$  are *NOT* in clausal form, while  $\forall x\forall y (p(x)\wedge q(y))$  *IS* in clausal form.

**Skolemization.** If  $A$  is a *closed* formula, then there exists a formula  $A'$  in clausal form s.t.  $A$  is satisfiable iff  $A'$  is satisfiable, written as  $A \approx A'$ , where we note that  $A$  and  $A'$  do not *necessarily* have to be equivalent.

**Substitution.** A substitution of *terms* for *variables* is a set  $\theta = \{x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n\}$ , where each  $x_i$  is a *distinct variable* and each  $t_i$  is a term which is **not** identical to the corresponding variable  $x_i$ . **Note:** An expression is a *term (variable, constant or function)*, a *literal*, a *clause* or a *set of clauses*; and an instance  $E\theta$  of  $E$  is obtained by *simultaneously replacing* each occurrence of  $x_i$  in  $E$  by  $t_i$ . As an example, if  $E = p(x)\vee q(f(y))$ , and if  $\theta = \{x \leftarrow y, y \leftarrow f(a)\}$ , then  $E\theta = p(y)\vee q(f(f(a)))$ . Similarly, if  $E = p(x, y)\wedge q(z)$ , and if  $\theta = \{x \leftarrow f(a), y \leftarrow f(b), z \leftarrow f(a)\}$ , then  $E\theta = p(f(a), f(b))\wedge q(f(a))$ .

**Composition of Substitutions.** Let  $\theta = \{x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n\}$  and  $\sigma = \{y_1 \leftarrow s_1, \dots, y_k \leftarrow s_k\}$  be *two substitutions*, and let  $X$  and  $Y$  be the *sets of variables* substituted for in  $\theta$  and  $\sigma$  respectively.  $\theta\sigma$ , the **composition** of  $\theta$  and  $\sigma$ , is the *substitution*  $\theta\sigma = \{x_i \leftarrow t_i \mid x_i \in X, x_i \neq t_i\sigma\} \cup \{y_j \leftarrow s_j \mid y_j \in Y, y_j \notin X\}$ . In *words*, (step 1) apply the *substitution*  $\sigma$  to the *terms*  $t_i$  of  $\theta$  (as long as resulting substitutions **do not collapse** to the vacuous  $x_i \leftarrow x_i$ ); (step 2) *append* the substitutions from  $\sigma$  whose variables are *not already substituted for* in  $\theta$ , i.e. do not appear on the LHS of a  $\leftarrow$  in  $\theta$ .

**Example:** If  $\theta = \{x \leftarrow a, y \leftarrow w\}$ , and if  $\sigma = \{a \leftarrow p, c \leftarrow q\}$ , then  $\theta\sigma = \{x \leftarrow p, y \leftarrow w, a \leftarrow p, c \leftarrow q\}$ . Similarly, if  $\theta = \{x \leftarrow a, y \leftarrow w, z \leftarrow d\}$ , and if  $\sigma = \{a \leftarrow p, c \leftarrow w, d \leftarrow z\}$ , then  $\theta\sigma = \{x \leftarrow p, y \leftarrow w, z \leftarrow z, a \leftarrow p, c \leftarrow w, d \leftarrow z\} = \{x \leftarrow p, y \leftarrow w, a \leftarrow p, c \leftarrow w, d \leftarrow z\}$ .

Now in *propositional calculus*, we write  $(p\vee q)\wedge\neg p\wedge\neg q$  in *clausal form* as follows:  $(p\vee q)\wedge\neg p\wedge\neg q = \{\{pq\}, \{\bar{p}\}, \{\bar{q}\}\} = \text{clauses } \{1, 2, 3\}$  — and we can then do  $\text{Res}(1, 2) = q$  (4) and finally  $\text{Res}(4, 3) = \square$ . Now consider  $p(a, b)\wedge\neg p(a, b)$ , with  $a, b \in A$  and  $p \in P$ . In *clausal form*, we have  $p(a, b)\wedge\neg p(a, b) = \{\{p(a, b)\}, \{\neg p(a, b)\}\} = \text{clauses } \{1, 2\}$ , and then  $\text{Res}(1, 2) = \square$ . Similarly,  $\text{Res}(p(f(a), g(b)), \neg p(f(a), g(b))) = \square$ , but what about  $\forall x\forall y\forall z [p(f(x), g(y))\wedge\neg p(f(a), g(z))] = \{\{p(f(x), g(y))\}, \{\neg p(f(a), g(z))\}\}$ ? Well, under the *substitution*  $\{x \leftarrow a, y \leftarrow z\}$ , the clauses become  $\{\{p(f(a), g(z))\}, \{\neg p(f(a), g(z))\}\}$ , so that the resolvent of the two clauses is again  $\square$ , but this time *only under the substitution*.

Definition: Given a set of atoms, a *unifier* is a substitution that makes the atoms of the set *identical*. A **most general unifier** (mgu) is a unifier  $\mu$  s.t. any other unifier  $\theta$  can be obtained from  $\mu$  by a *further* substitution  $\lambda$  s.t.  $\theta = \mu\lambda$ . Example 1:  $\{p(x), p(y)\}$  has mgu  $\{x \leftarrow y\}$ . A non mgu for this example:  $\{x \leftarrow a, y \leftarrow a\}$ . Example 2:  $\{p(f(x)), p(y)\}$  has unifier  $\{y \leftarrow f(x)\}$ . Example 3:  $\{p(x), q(y)\}$  has **no** unifier. Note: Do not worry about unifiers for the *exam*.

## General Resolution Rule

Let  $L = \{l_1, \dots, l_n\}$  be a set of literals (e.g.  $p(a), \neg p(a), \dots$ ) — then  $L^c = \{l_1^c, \dots, l_n^c\}$ . Further, let  $C_1$  and  $C_2$  be two clauses with **no variables** in common, and let  $L_1 = \{l_{11}, \dots, l_{1n_1}\} \subseteq C_1$  and  $L_2 = \{l_{21}, \dots, l_{2n_2}\} \subseteq C_2$  be subsets of literals s.t.  $L_1$  and  $L_2^c$  can be *unified* by a mgu  $\sigma$ . In this situation,  $C_1$  and  $C_2$  are said to be *clashing clauses* and are further said to clash on literals  $L_1$  and  $L_2$ .  $C$ , the resolvent of  $C_1$  and  $C_2$ , is given by  $\text{Res}(C_1, C_2) = \{C_1\sigma - L_1\sigma\} \cup \{C_2\sigma - L_2\sigma\}$ .

Example: Consider  $\forall x \forall y [p(f(x)) \wedge \neg p(y)] =$  (in *clausal form*)  $= \{\{p(f(x))\}, \{\neg p(y)\}\}$ . Here,  $L_1 = \{p(f(x))\}$ ,  $L_2 = \{\neg p(y)\}$ ,  $L_2^c = \{p(y)\}$ , and the *mgu* for  $L_1$  and  $L_2^c$  is  $\{y \leftarrow f(x)\}$ . It therefore follows that  $\text{Res}(\{p(f(x))\}, \{\neg p(y)\}) = \{C_1\sigma - L_1\sigma\} \cup \{C_2\sigma - L_2\sigma\} = \{p(f(x)) - p(f(x))\} \cup \{\neg p(f(x)) - \neg p(f(x))\} = \phi \cup \phi = \square$ , the *empty clause*.

## Prolog, Predicate Calculus and Resolution

A Prolog program has *three sections*: a **database of facts**, a **set of rules** and **queries/goals**. (1) **Database of Facts**. Example: John is the *father* of Mike, Mike is the *father* of Terry, and David is the *father* of Sally. In *Predicate Calculus*, we have  $\text{Father}(\text{John}, \text{Mike})$ ,  $\text{Father}(\text{Mike}, \text{Terry})$ , and  $\text{Father}(\text{David}, \text{Sally})$ . In *Prolog*, we have the same as above.

(2) **Sets of Rules**. Example: if  $x$  is the father of  $y$ , and if  $y$  is the father of  $z$ , then  $x$  is the *grandfather* of  $z$ . In *Predicate Calculus*, we have  $\forall x \forall y \forall z [( \text{Father}(x, y) \wedge \text{Father}(y, z) ) \rightarrow \text{Gfather}(x, z)]$ , and this can be turned into *clausal form* (by using the equivalence  $A \rightarrow B \equiv \neg A \vee B$ ) to give  $\forall x \forall y \forall z [\neg \text{Father}(x, y) \vee \neg \text{Father}(y, z) \vee \text{Gfather}(x, z)]$ . In *Prolog*, we have  $\text{Gfather}(x, z) :- \text{Father}(x, y), \text{Father}(y, z)$ , where we can think of the symbol  $:-$  as  $\leftarrow$  and can think of the comma as the symbol  $\wedge$ .

(3) **Goals**. Goals represent a *query*, e.g. does Terry have any grandfathers and, if so, who are they? In *Predicate Calculus*, we have  $\exists x \text{Gfather}(x, \text{Terry})$  — and what are the possible values of  $x$ . In *Prolog*, we have  $\text{Gfather}(x, \text{Terry})$  — and the Prolog system when given this goal can *keep* returning possible values of  $x$ .

Prolog would resolve a query as follows: (1) Construct the set of clauses consisting of *all the rules, facts and the negation of the goal G*:  $\{\text{rule 1 in clausal form}, \dots, \text{rule n in clausal form}, \text{fact 1}, \dots, \text{fact m}, \neg G\}$ . (2) Apply **resolution** to this set, attempting to yield the empty clause  $\square$ . (3) If resolution succeeds and thus  $\square$  is derived, then this set of clauses is *unsatisfiable*, so that  $\neg G$  is **not** a logical consequence of the set of clauses, and therefore **G is** a logical consequence of the set of clauses. (4) Furthermore, the substitutions made during resolution provide a set of *answers* to  $G$ .

For our example, (1) The *set of clauses* would be  $\{\neg\text{Father}(x, y) \vee \neg\text{Father}(y, z) \vee \text{Gfather}(x, z), \text{Father}(\text{John}, \text{Mike}), \text{Father}(\text{Mike}, \text{Terry}), \text{Father}(\text{David}, \text{Sally}), \neg\text{Gfather}(x, \text{Terry})\}$  — which can be written as  $\{\{\neg\text{Father}(x, y), \neg\text{Father}(y, z), \text{Gfather}(x, z)\}, \{\text{Father}(\text{John}, \text{Mike})\}, \{\text{Father}(\text{Mike}, \text{Terry})\}, \{\text{Father}(\text{David}, \text{Sally})\}, \{\neg\text{Gfather}(x, \text{Terry})\}\}$ .

(2) For clauses 1 and 5,  $L_1 = \{\text{Gfather}(x, z)\}$ ,  $L_2 = \{\neg\text{Gfather}(x, \text{Terry})\}$ ,  $L_2^c = \{\text{Gfather}(x, \text{Terry})\}$ , the *mgu* for  $L_1$  and  $L_2^c$  is  $\{z \leftarrow \text{Terry}\}$ , and so  $\text{Res}(1, 5) = \neg\text{Father}(x, y) \vee \neg\text{Father}(y, \text{Terry})$  (call this clause 6). (3) For clauses 3 and 6,  $L_1 = \{\text{Father}(\text{Mike}, \text{Terry})\}$ ,  $L_2 = \{\neg\text{Father}(y, \text{Terry})\}$ ,  $L_2^c = \{\text{Father}(y, \text{Terry})\}$ , the *mgu* for  $L_1$  and  $L_2^c$  is  $\{y \leftarrow \text{Mike}\}$ , and so  $\text{Res}(3, 6) = \neg\text{Father}(x, \text{Mike})$  (call this clause 7).

(4) For clauses 2 and 7,  $L_1 = \{\text{Father}(\text{John}, \text{Mike})\}$ ,  $L_2 = \{\neg\text{Father}(x, \text{Mike})\}$ ,  $L_2^c = \{\text{Father}(x, \text{Mike})\}$ , the *mgu* for  $L_1$  and  $L_2^c$  is  $\{x \leftarrow \text{John}\}$ , and so  $\text{Res}(7, 2) = \square$ . (5) In this case, resolution has proved that the set of clauses is **not** satisfiable — therefore  $\forall x \neg\text{Gfather}(x, \text{Terry})$  is **not** a logical consequence of the rules and facts, and so  $\exists x \text{Gfather}(x, \text{Terry})$  **is** a logical consequence of the rules and facts.

(6) Furthermore, *during the refutation*, the substitution  $\{z \leftarrow \text{Terry}, y \leftarrow \text{Mike}, x \leftarrow \text{John}\}$  was made. Making this substitution in the goal clause  $\exists x \text{Gfather}(x, \text{Terry})$  provides *one possible value of x*, i.e. **John** is the grandfather of Terry. (7) Prolog can then *repeat* resolution using a different substitution and therefore provide *another* possible value of  $x$ , i.e. another grandfather of Terry.

7th May 2002

## General Overview of the Course

(1) *General Propositional Logic* (equivalences, consequences, validity, satisfiability). (2) BDD's. (3) *Semantic Tableaux*. (4) *Resolution* for propositions. (5) *Predicates*. The **last two** lectures of the course will not be examined (i.e. predicate calculus *clausal form*, *substitution* for predicate calculus, *unification* for predicate calculus and *resolution* for predicate calculus).

On the exam paper, the  $\alpha$  and  $\beta$  tables and the following *equivalences* will be given:  $\neg(A \vee B) \equiv \neg A \wedge \neg B$ ,  $\neg(A \wedge B) \equiv \neg A \vee \neg B$ ,  $A \rightarrow B \equiv \neg A \vee B$ ,  $(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$ , and  $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$ . You *are* expected to memorise *some* things, such as (i) **simple equivalencies** such as  $\neg\neg r \equiv r$ ,  $A \wedge B \equiv B \wedge A$  and  $A \wedge \neg A \equiv \text{false}$ ; (ii)  $\forall x A \equiv \neg \exists x \neg A$  and  $\exists x A \equiv \neg \forall x \neg A$ ; (iii) **truth tables** for e.g.  $\wedge$ ,  $\vee$ ,  $\rightarrow$ ,  $\leftrightarrow$  and  $\neg$ ; and (iv) **algorithms** e.g. for semantic tableaux and resolution. Further, you will be asked to prove *theorems* etc. in the exam, and if an *interpretation*  $I$  for a predicate formula is given as a 3-tuple, then there are **no functions** in the formula.

Common mistakes: (1) In propositional logic, an interpretation needs a truth assignment for *every* proposition. (2) Make sure that you follow the algorithms and not your *own versions* of them e.g. in semantic tableau do not merge together the  $\alpha$  and  $\beta$  steps, and in OBDDs do not merge together steps. (3)  $\forall$ ,  $\exists$ ,  $\neg$  bind most tightly, then  $\vee$ ,  $\wedge$  and then  $\rightarrow$ ,  $\leftrightarrow$ , so e.g.  $\neg p \wedge q$  means  $(\neg p) \wedge q$ .

## Sample Exam Paper

Q1: The following equivalences may be used in the following sub questions:  $\neg(A \vee B) \equiv \neg A \wedge \neg B$ ,  $\neg(A \wedge B) \equiv \neg A \vee \neg B$ ,  $A \rightarrow B \equiv \neg A \vee B$ ,  $(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$ ,  $(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$ , and  $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$ . (a) By finding an *interpretation*, prove whether the following formula is **satisfiable**, **valid** or **contradictory**:  $\neg[(A \leftrightarrow (A \vee B)) \vee \neg(A \rightarrow B)]$ . Based upon your results, determine whether the *negation* of the formula is valid. (b) Prove that  $\neg(\neg p \wedge \neg(q \vee r)) \equiv p \vee q \vee r$ . (c) Prove that  $A$  is *satisfiable* iff  $\neg A$  is *falsifiable*. (d) Prove the following:  $\{p \rightarrow q, q \rightarrow r, \neg r \leftrightarrow s, s\} \models \neg p$ .

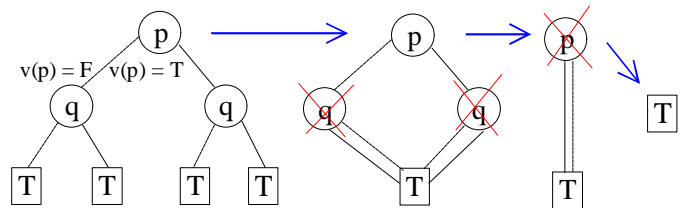
A: (a) Let  $X$  be the *formula* in question. If  $v(A) = F$ , and if  $v(B) = T$ , then  $v(X) = \neg[(F \leftrightarrow (F \vee T)) \vee \neg(F \rightarrow T)] = \neg[(F \leftrightarrow T) \vee \neg(T)] = \neg[F \vee F] = T$ , which proves that  $X$  is satisfiable. Part (a) of *Theorem 1.3* ( $A$  is valid iff  $\neg A$  is unsatisfiable) then enables us to say that  $\neg X$  cannot be valid as we have found a *truth assignment* that satisfies  $\neg\neg X = X$ . Further,  $X$  is *not valid* as the truth assignment  $v(A) = F$  and  $v(B) = T$  **falsifies**  $X$ . Note: Contradictory means *always false*.

(b) LHS  $\equiv \neg(\neg p \wedge \neg(q \vee r)) \equiv \neg(\neg p \wedge \neg q \wedge \neg r) \equiv \neg\neg p \vee \neg(\neg q \wedge \neg r) \equiv p \vee \neg\neg q \vee \neg\neg r \equiv p \vee q \vee r \equiv$  RHS. (c) **If**. If  $A$  is satisfiable, then there is an *assignment* of truth values such that  $v(A) = T$ . But if  $v(A) = T$ , then  $v(\neg A) = F$ , and so the **same** truth assignment falsifies  $\neg A$  — and so  $\neg A$  is falsifiable. **Only If**. If  $\neg A$  is *falsifiable*, then there is an *assignment* of truth values such that  $v(\neg A) = F$ . But if  $v(\neg A) = F$ , then  $v(\neg\neg A) = v(A) = T$ , and so the **same** truth assignment satisfies  $A$  — and so  $A$  is *satisfiable*. QED.

(d) By Theorem 1.5,  $\{A_1, A_2, \dots, A_n\} \models A$  iff  $A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow A$  is valid. So we manipulate  $A_1 \wedge A_2 \wedge \dots \wedge A_n$ , in this case  $(p \rightarrow q) \wedge (q \rightarrow r) \wedge (\neg r \leftrightarrow s) \wedge s$ , as follows:  $(p \rightarrow q) \wedge (q \rightarrow r) \wedge (\neg r \leftrightarrow s) \wedge s \equiv (\neg p \vee q) \wedge (\neg q \vee r) \wedge (\neg r \rightarrow s) \wedge (s \rightarrow \neg r) \wedge s \equiv (\neg p \vee q) \wedge (\neg q \vee r) \wedge (\neg\neg r \vee s) \wedge (\neg s \vee \neg r) \wedge s \equiv (\neg p \vee q) \wedge (\neg q \vee r) \wedge (r \vee s) \wedge ((\neg s \wedge s) \vee (\neg r \wedge s)) \equiv (\neg p \vee q) \wedge (\neg q \vee r) \wedge (r \vee s) \wedge \neg r \wedge s \equiv (\neg p \vee q) \wedge (\neg q \vee r) \wedge ((r \wedge \neg r \wedge s) \vee (s \wedge \neg r \wedge s)) \equiv (\neg p \vee q) \wedge (\neg q \vee r) \wedge (\text{false} \vee (s \wedge \neg r \wedge s)) \equiv (\neg p \vee q) \wedge (\neg q \vee r) \wedge (s \wedge \neg r \wedge s) \equiv (\neg p \vee q) \wedge \neg q \wedge \neg r \wedge s \rightarrow \neg p$ , so the result *holds*.

Q2: (a) Using **Ordered Binary Decision Diagrams** (OBDD's), prove that the following formula is valid (all stages of the reduction process should be shown):  $(p \leftrightarrow q) \vee \neg((p \rightarrow q) \wedge (q \rightarrow p))$ . (b) Construct the OBDD for the formula  $(p \wedge q) \vee w \vee s$ , and then reduce this OBDD as much as possible, showing *all stages of the reduction process*. (c) Let  $C = \{l\} \in S$  be a unit clause, and let  $S'$  be obtained from  $S$  by deleting *every clause containing  $l$*  and by deleting  $l^c$  from every remaining clause. Prove that  $S$  is *satisfiable* iff  $S'$  is *satisfiable*.

A: (a) The diagram is as shown on the *right*, and because the reduced OBDD for the formula consists simply of a **single** boxed 'T', then it follows that the formula is **valid**.



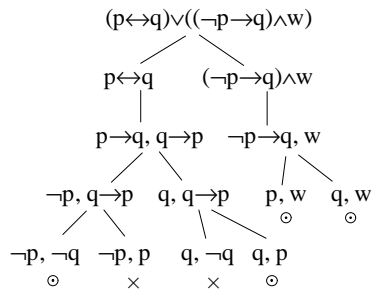
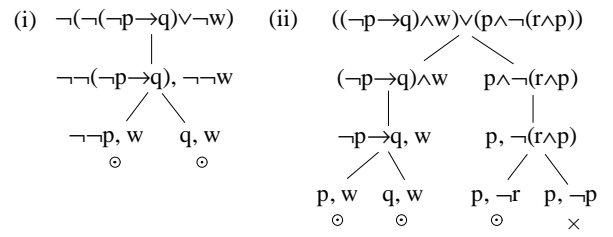
(b) Standard example — the *method* to use is as follows: (1) Fill in the *leaf truth values*, (2) Reduce the leaves to just **one boxed T** and **one boxed F**, (3) Both edges go to the same node — *delete*, (4) Identify *Sub-BDD's*, (5) **Repeat** (3) and (4) until no more reductions can be made, (6) *Sanity* check!

(c) **If:** Let  $v$  be a **model** for  $S$ . We'll prove that  $v$  is *also* a model for  $S'$  by showing that for every  $C_i$  s.t.  $C_i' = C_i - \{l^c\}$ , then  $v(C_i') = T$ . Now  $v(C) = v(l) = T$  since  $v$  is a model for  $S$ , so that  $v(l^c) = F$ . Further, since  $v$  is a model for  $S$ , then  $v(C_i) = T$ , and thus there must be some other literal  $l_i \in C_i$  s.t.  $v(l_i) = T$  — and so it follows that  $v(C_i') = T$ .

**Only If:** If  $S'$  is *satisfiable*, then there is a **model**  $v$  for  $S'$  s.t.  $v(C') = T$  for every  $C' \in S'$  (because a set of clauses is just an *implicit conjunction* so that all clauses must be true in order for  $v$  to be a model). Now extend  $v$  by defining  $v(l) = T$  — then for every  $C \in S - S'$ , *either*  $C$  has  $l$  in it, so that as  $v(l) = T$  then  $v(C) = T$  (a clause is an implicit disjunction, so that it is sufficient that we only have one literal true in order for the clause to evaluate to be true), *or*  $C$  has  $l^c$  in it, but knowing that  $C_i - \{l^c\} \in S'$  and therefore  $v(C_i - \{l^c\}) = T$ , then  $v(C_i) = T$  so that  $v(C) = T$  for every  $C \in S - S'$ . QED.

Q3: (The  $\alpha$  and  $\beta$  tables are given as an aid to this question...) (a) Construct the semantic tableaux for the following *formulae* and for each branch, where appropriate, construct the corresponding **Hintikka set**: (i)  $\neg(\neg(\neg p \rightarrow q) \vee \neg w)$ , (ii)  $((\neg p \rightarrow q) \wedge w) \vee (p \wedge \neg(r \wedge p))$ . (b) Use the method of semantic tableaux to discover a *model* for the formula  $(p \leftrightarrow q) \vee ((\neg p \rightarrow q) \wedge w)$  and define that model. (c) Prove *Hintikka's Lemma* which states that if  $U$  is a Hintikka set then  $U$  is **satisfiable**.

A: (a) (i) Hintikka sets: *first* branch =  $\{\neg(\neg(\neg p \rightarrow q) \vee \neg w), (\neg p \rightarrow q), p, w\}$ ; *second* branch =  $\{\neg(\neg(\neg p \rightarrow q) \vee \neg w), (\neg p \rightarrow q), q, w\}$ . (ii) Hintikka sets *first* branch =  $\{((\neg p \rightarrow q) \wedge w) \vee (p \wedge \neg(r \wedge p)), (\neg p \rightarrow q) \wedge w, \neg p \rightarrow q, p, w\}$ ; *second* branch =  $\{((\neg p \rightarrow q) \wedge w) \vee (p \wedge \neg(r \wedge p)), (\neg p \rightarrow q) \wedge w, \neg p \rightarrow q, q, w\}$ ; *third* branch =  $\{((\neg p \rightarrow q) \wedge w) \vee (p \wedge \neg(r \wedge p)), p \wedge \neg(r \wedge p), p, \neg(r \wedge p), \neg r\}$ . (b) The semantic tableau is as shown on the *left*. Looking at the first branch of the tableau, we see that a **model** for the formula in question is given by  $v(p) = F, v(q) = F$  and then  $v(w) = T$ , say. (c) See page 12.



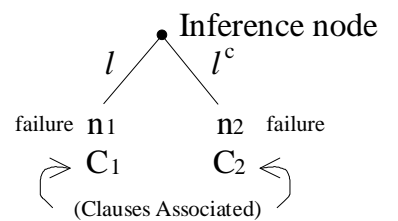
Q4: (a) For the following sets  $S$ , derive the *simplest set*  $S'$  such that  $S$  is satisfiable if and only if  $S'$  is satisfiable, i.e.  $S \approx S'$ : (i)  $S = \{p\bar{q}r, \bar{q}\bar{s}, s, \bar{s}q\}$ , (ii)  $S = \{rst, qq\bar{q}, \bar{s}\bar{t}, rq, t\}$ . Remember to explain your moves. (b) Determine the *satisfiability* of the following sets of clauses by applying the **resolution procedure** (no *simplification* of the sets should be carried out before applying the procedure): (i)  $\{pqr, st, \bar{r}, \bar{p}q, \bar{q}\}$ , (ii)  $\{rst, \bar{t}sw, \bar{w}r, \bar{s}r, \bar{r}\}$ .

(c) In a **closed** semantic tree, let  $b$  be the branch from the root terminating at an *inference node*  $n$ . The **children**  $n_1$  and  $n_2$  of  $n$  are **failure nodes**, so let  $C_1$  and  $C_2$  be any clauses associated with them respectively. Clauses  $C_1$  and  $C_2$  therefore clash, so prove that  $v_b$ , the partial interpretation associated with  $n$ , *falsifies*  $C$ , their resolvent clause.

A: (i)  $S = \{p\bar{q}r, \bar{q}r\bar{s}, s, s\bar{r}q\}$ . Now  $p$  appears but  $\bar{p}$  *does not* so that  $S' = \{\bar{q}r\bar{s}, s, s\bar{r}q\}$ ;  $s$  is a *unit clause* so that  $S'' = \{\bar{q}r\}$ ; and  $r$  appears but  $\bar{r}$  *does not* so that  $S''' = \emptyset$  — and so  $S$  is **satisfiable**. (ii)  $S = \{rst, qp\bar{q}, \bar{s}\bar{t}, rq, t\}$ . Now  $t$  is a *unit clause* so that  $S' = \{qp\bar{q}, \bar{s}, rq\}$ ;  $\bar{s}$  is a *unit clause* so that  $S'' = \{qp\bar{q}, rq\}$ ;  $p$  appears but  $\bar{p}$  *does not* so that  $S''' = \{rq\}$ ; and  $r$  appears but  $\bar{r}$  *does not* so that  $S'''' = \emptyset$  — and so  $S$  is **satisfiable**.

(b) (i) Let  $S = \{pqr, st, \bar{r}, \bar{p}q, \bar{q}\} =$  clauses  $\{1, 2, 3, 4, 5\}$ . *Clashing clauses*: 1 and 3:  $\text{Res}(pqr, \bar{r}) = pq$ , clause 6. 4 and 6:  $\text{Res}(\bar{p}q, pq) = q$ , clause 7. 5 and 7:  $\text{Res}(\bar{q}, q) = \square$  — and so  $S$  is **unsatisfiable**. (ii) Let  $S = \{rst, \bar{t}sw, \bar{w}r, \bar{s}r, \bar{r}\} =$  clauses  $\{1, 2, 3, 4, 5\}$ . *Clashing clauses*: 1 and 5:  $\text{Res}(rst, \bar{r}) = st$ , clause 6. 2 and 3:  $\text{Res}(\bar{t}sw, \bar{w}r) = \bar{t}sr$ , clause 7. 5 and 7:  $\text{Res}(\bar{r}, \bar{t}sr) = \bar{t}s$ , clause 8. 6 and 8:  $\text{Res}(st, \bar{t}s) = s$ , clause 9. 4 and 9:  $\text{Res}(\bar{s}r, s) = r$ , clause 10. 5 and 10:  $\text{Res}(\bar{r}, r) = \square$  — and so  $S$  is **unsatisfiable**.

(c) See page 19, and consider the following: Let  $b_1$  and  $b_2$  be the branches that terminate at failure nodes  $n_1$  and  $n_2$  respectively, so that  $b_1$  and  $b_2$  are identical except for the edges from  $n$  to  $n_1$  and from  $n$  to  $n_2$ . It follows that  $n_1$  and  $n_2$  are associated with clauses  $C_1$  and  $C_2$  which clash on the literals  $l$  and  $l^c$  of the atom  $p$ . Now the partial interpretation  $v_b$  is the same as  $v_{b_1}$  and  $v_{b_2}$  except that it *does not assign* to  $p$ . But  $v_{b_1}(C_1) = v_{b_2}(C_2) = F$  since  $n_1$  and  $n_2$  are failure nodes, so certainly  $v_{b_1}(C_1 - \{l\}) = F$  and  $v_{b_2}(C_2 - \{l^c\}) = F$  — so it follows that  $v_b(C) = v_b((C_1 - \{l\}) \cup (C_2 - \{l^c\})) = F$ . QED.



Q5: (a) Let  $G(x)$  mean ' $x$  is a gardener', let  $D(x, y)$  mean ' $x$  dances to the music of composer  $y$ ', and let  $P(x, y)$  mean ' $x$  performs in band  $y$ '. Using these predicate symbols, translate the **following sentences** into predicate logic formulae: "*Everybody who dances to Bach is a gardener*", "*Nobody who performs in the band Oasis dances to Bach*", and "*A gardener never dances*".

(b) Determine if the **formula**  $\forall x (w(x, a) \rightarrow (p(a) \wedge \exists y q(y, x)))$  is true for the *interpretation*  $(\mathbf{Z}, \{>, is\_positive, divisible\_by\}, \{0\})$ , where  $\mathbf{Z}$  is the set of all positive or negative integers,  $is\_positive$  is a predicate denoting that the argument is a positive number, and  $divisible\_by$  is a predicate denoting that the **first** argument is divisible by the **second** argument.

(c) Determine if the formula  $\exists y (q(y, a) \wedge \neg \forall z w(z, y))$  is true for the *interpretation*  $(\mathbf{Z}, \{<, \geq\}, \{0\})$ , where  $\mathbf{Z}$  is the set of all positive or negative integers. (d) Prove that the *formula*  $\exists x (p(x) \wedge q(x)) \rightarrow (\exists x p(x) \wedge \exists x q(x))$  is **valid**.

A: (a) The *required formulae* are as follows:  $\forall x (D(x, \text{Bach}) \rightarrow G(x))$ ,  $\neg \exists x (P(x, \text{Oasis}) \wedge D(x, \text{Bach}))$ , and  $\exists x (G(x) \wedge \neg \exists y D(x, y))$ . (b) Meaning: for all  $x$ ,  $x > 0$  implies that 0 is +ve and there exists a  $y$  such that  $y$  is *divisible* by  $x$ . This is true or false depending on whether 0 is considered to be a positive number or not (assume that it **does** in an exam). (c) Meaning:  $\exists y (y < 0 \wedge \neg \forall z z \geq y)$ . This is true with e.g.  $y = -1$ : not all  $z \in \mathbf{Z}$  will therefore be greater than or equal to  $y$  in this instance, e.g.  $-2$  is less than  $-1$  while  $1$  is greater than  $-1$ . (d) See Assignment 3.

## Exam Paper: May 2002

### Answer 3 questions out of 5 (Questions Done: 1, 3, 5)

(1) The following *equivalences* may be used in the following sub questions:

$$\neg(A \vee B) \equiv \neg A \wedge \neg B$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B$$

$$A \rightarrow B \equiv \neg A \vee B$$

$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$$

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$$

$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$$

(a) By defining appropriate *interpretations*, prove whether the following formulae are **satisfiable**, **valid** or **contradictory**:

(i)  $(A \rightarrow (B \vee C))$

(ii)  $(B \leftrightarrow (A \vee B)) \wedge \neg(A \rightarrow B)$

[4 marks]

(b) Prove the following *equivalences*, showing *all* steps.

(i)  $\neg(\neg r \wedge \neg(q \vee t)) \equiv r \vee q \vee t$

(ii)  $((p \rightarrow q) \wedge \neg(q \rightarrow r)) \wedge \neg r \wedge p \equiv (q \wedge \neg r \wedge p)$

[8 marks]

(c) Prove, showing *all* steps, that if  $U = \{p \leftrightarrow q, q \rightarrow r, p, \neg r \vee s\}$ , then  $U \models s$ .

[8 marks]

(2) (a) Use *Ordered Binary Decision Diagrams* to prove that the following formula is valid:  $((p \vee q) \wedge r) \vee \neg((p \vee q) \wedge r)$ . **All** stages of the reduction process should be shown.

[5 marks]

(b) Using *Ordered Binary Decision Diagrams*, prove that the following formula is satisfiable but not valid:  $(p \wedge q) \vee r$ . **All** stages of the reduction process should be shown.

[6 marks]

(c) Let  $U = \{A_1, \dots, A_n\}$ . Prove that if  $U \models B$  then  $\models A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow B$ . [9 marks]

(3) The following two tables represent the  $\alpha$  and  $\beta$  formulae used in the construction of semantic tableaux and are provided as an *aid* to the following sub questions:

$\alpha$	$\alpha_1$	$\alpha_2$
$\neg\neg A_1$	$A_1$	
$A_1 \wedge A_2$	$A_1$	$A_2$
$\neg(A_1 \vee A_2)$	$\neg A_1$	$\neg A_2$
$\neg(A_1 \rightarrow A_2)$	$A_1$	$\neg A_2$
$A_1 \leftrightarrow A_2$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$

$\beta$	$\beta_1$	$\beta_2$
$\neg(B_1 \wedge B_2)$	$\neg B_1$	$\neg B_2$
$B_1 \vee B_2$	$B_1$	$B_2$
$B_1 \rightarrow B_2$	$\neg B_1$	$B_2$
$\neg(B_1 \leftrightarrow B_2)$	$\neg(B_1 \rightarrow B_2)$	$\neg(B_2 \rightarrow B_1)$

(a) Construct the semantic tableaux for the following formulae and for each branch, where appropriate, construct the corresponding *Hintikka set*.

(i)  $p \wedge (\neg q \vee \neg p)$

(ii)  $(p \wedge q) \vee (\neg p \wedge \neg q)$

[6 marks]

(b) Use a semantic tableau to discover a *model* for the following formula and *define* that model:  $p \wedge (q \vee r) \wedge (q \rightarrow \neg p)$ .

[5 marks]

(c) Prove the following formula is *valid* by showing that the completed semantic tableau for the *negation* of the formula is closed:  $(p \vee (q \wedge r)) \leftrightarrow ((p \vee q) \wedge (p \vee r))$ .

[9 marks]

(4) (a) For each set of clauses  $S$  below, derive the *simplest* possible set  $S'$  such that  $S$  is satisfiable if and only if  $S'$  is satisfiable, i.e.  $S \approx S'$ . Show *all* stages of the derivation.

(i)  $S = \{qr, \bar{q}p\bar{s}, \bar{p}pqs\}$

(ii)  $S = \{pqr, \bar{q}, p\bar{r}s, qs, \bar{p}\bar{s}\}$

[7 marks]

(b) Determine the *satisfiability* of the following sets of clauses by applying the resolution procedure. No **simplification** of the sets should be carried out before applying the procedure.

(i)  $\{\bar{p}\bar{q}r, pr, qr, \bar{r}\}$

(ii)  $\{pqr, q\bar{r}s, sp, \bar{p}, \bar{q}\}$

[7 marks]

(c) Let clause  $C$  be the *resolvent* of clauses  $C_1$  and  $C_2$ . Prove that if  $C_1$  and  $C_2$  are *mutually satisfiable*, then  $C$  is also *satisfiable*.

[6 marks]

- (5) (a) Let  $F(x, y)$  mean ' $x$  is the father of  $y$ ', and let  $M(x, y)$  mean ' $x$  is the mother of  $y$ '. Using these predicate symbols, translate the following sentences into predicate logic formulae:
- (i) Everybody has a mother,
  - (ii) Everybody has a father and a mother,
  - (iii) Whoever has a mother has a father. **[8 marks]**
- (b) Consider the formula  $\forall x \exists y (p(x, a) \wedge p(y, x))$ , where  $a$  is a *constant symbol*,  $p$  is a *predicate symbol*, and  $x$  and  $y$  are *variables*. Determine if the formula is true in the following interpretations, explaining your reasons:
- (i)  $I = (\mathbf{N}, \{\geq\}, \{0\})$ , where  $\mathbf{N}$  is the set of *positive integers* including zero.
  - (ii)  $I = (\mathbf{Z}, \{=\}, \{5\})$ , where  $\mathbf{Z}$  is the set of *all integers*, positive or negative. **[6 marks]**
- (c) Prove the formula  $\forall x p(x) \rightarrow \exists x p(x)$  is *valid*. **[6 marks]**