Maggie Johnson
CS103A

Handout #6

Connectives

Key Topics

* Introduction to Connectives
* And, Or, Not
* Truth Tables
* The "Game"
* Ambiguity
* Equivalences

- Introduction to Connectives

  We build complex claims from atomic ones using the Boolean connectives and, or, not.

  Truth-functional: the truth value of a complex sentence built using connectives depends *only* on the truth values of the simpler sentences from which it is built.

- And, Or, Not

  A Truth Table shows the truth value of a complex sentence given all possible truth values of the simple sentences of which it is made.

Truth table for AND (^):	p	q	p^q	
	T	T	T	
	T	F	F	(conjunction)
	F	T	F	
	F	F	F	

Truth table for OR (v):	p	q	pvq	
	T	T	T	
	T	F	T	(disjunction)
	F	T	T	
	F	F	F	

Truth table for NOT (¬)	p	¬p	
	T	F	
	F	T	(negation)

Literal: A sentence that is either atomic or the negation of an atomic sentence.

- Truth Tables

Algorithm for constructing truth tables to determine truth value of a complex sentence:

Either I will wash my car, or it will not rain.

Translation: p = I will wash my car.
q = It will rain.

This sentence has the form p v ~ q.

Step 1: Specify the different combinations of true and false for the variables:

p	q
T	T
T	F
F	T
F	F

no other possible combination, right?

Step 2: Write the entire complex sentence to the right of this part of the table with a separate column under each variable or connective. Then we copy the true/false values assigned in step 1 to the variables to the right:

p	q	p	v	~	q
T	T	T			T
T	F	T			F
F	T	F			T
F	F	F			F

Step 3: Evaluate the results of the logical operations using the following precedence rules:

1) Go to the farthest inside set of parentheses that has not been done yet;
2) Within the innermost set of parentheses, do the computations in this order:
negation is done first, then ^ and v from left to right.

p	q	p	v	~	q	
T	T	T	T	F	T	(negation done first, then v)
T	F	T	T	T	F	
F	T	F	F	F	T	
F	F	F	T	T	F	
			*			

Now we know all possible combinations for the complex proposition.

A more involved example: [(p ^ q) v r] ^ [~ (p ^ r)]

p	q	r	p	^	q	v	r	^	~	p	^	r
T	T	T	T	T	T	T	T	F	F	T	T	T
T	T	F	T	T	T	T	F	T	T	T	F	F
T	F	T	T	F	F	T	T	F	F	T	T	T
T	F	F	T	F	F	F	F	F	T	T	F	F
F	T	T	F	F	T	T	T	T	T	F	F	T
F	T	F	F	F	T	F	F	F	T	F	F	F
F	F	T	F	F	F	T	T	T	T	F	F	T
F	F	F	F	F	F	F	F	F	T	F	F	F
								*				

evaluated in this order:

1. [**(p ^ q)** v r] ^ [~ (p ^ r)]
2. [(p ^ q) v r] ^ [~ **(p ^ r)**]
3. **[(p ^ q) v r]** ^ [~ (p ^ r)]
4. [(p ^ q) v r] ^ **[~ (p ^ r)]**
5. [(p ^ q) v r] **^** [~ (p ^ r)]

- S is a *tautology* if and only if every row of the truth table assigns T to S.
- If S is a tautology, then S is a *logical truth*, i.e., it is *logically necessary*.
- Some logical truths are not tautologies, e.g., a = a.
- S is *TT-possible* if and only if at least one row of the truth table assigns T to S.

- The "Game"

  Henkin-Hintikka Game Rules:

  ¬P: If you commit yourself to the truth of ¬P, it's the same as committing yourself to the falsity of P. If you commit to the falsity of ¬P, you are committing yourself to the truth of P.

  P ^ Q: If you commit to the truth of P ^ Q, then you have committed yourself to the truth each of P and of Q. If you commit to the falsity of P ^ Q, then at least one of P and Q must be false, maybe both.

  P v Q: If you commit to the truth of P v Q, then you have committed yourself to the truth of either P or of Q. If you commit to the falsity of P ^ Q, then both P and Q must be false.

- Ambiguity

  Elliot is playing Nintendo or Robert is playing Nintendo and Brian is playing gizmos.

- Equivalences

Two propositions are *logically equivalent* if they have exactly the same truth values under all circumstances, i.e., they have the same *truth conditions*. For example, if we were to construct a truth table for ~ (p ^ r) v [(p ^ q) ^ ~ r], we would see that the last evaluated column is exactly the same as the last evaluated column for [(p ^ q) v r] ^ [~ (p ^ r)]. If two statements are logically equivalent, you can substitute one for the other.

Identity Laws	p ^ T <=> p p v F <=> p
Domination Laws	p v T <=> T p ^ F <=> F
Idempotent Laws	p v p <=> p p ^ p <=> p
Double Negation	~(~p) <=> p
Commutative Laws	p v q <=> q v p p ^ q <=> q ^ p
Associative Laws	(p v q) v r <=> p v (q v r) (p ^ q) ^ r <=> p ^ (q ^ r)
Distributive Laws	p v (q ^ r) <=> (p v q) ^ (p v r) p ^ (q v r) <=> (p ^ q) v (p ^ r)
DeMorgan's Laws	~(p ^ q) <=> ~p v ~q ~(p v q) <=> ~p ^ ~q
Other Useful Equivalences	p v ~p <=> T p ^ ~p <=> F