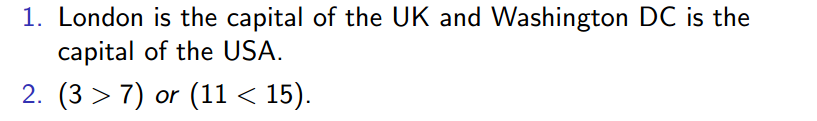
LOGIC:是亚里士多德创立的一门学科，处理命题的规则。

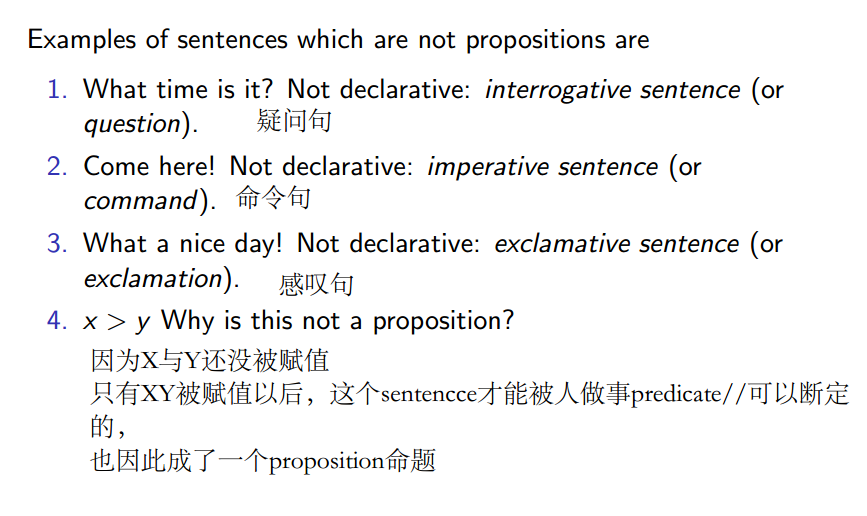
◮proposition：命题，是一个对或错(但不是同时兼具两者)的语句statement(或陈述句declaretive statement)

◮命题称为原子atomic如果他们不能被分解变成其他sentence，否则就叫compound。

compound proposition的例子



不是命题的例子

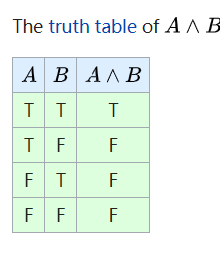


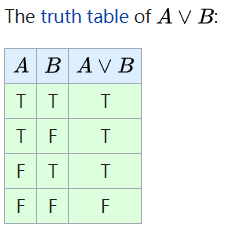
利用已有proposition来创造新proposition

我们可以使用logical operator逻辑符号来创造新命题proposition

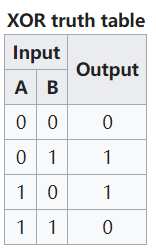
假设p与q是proposition

negation：否定

conjunction:  两者都对才能对

disjunction:  两者都错才是错

exclusive异或

 必须一对一错，不能相同

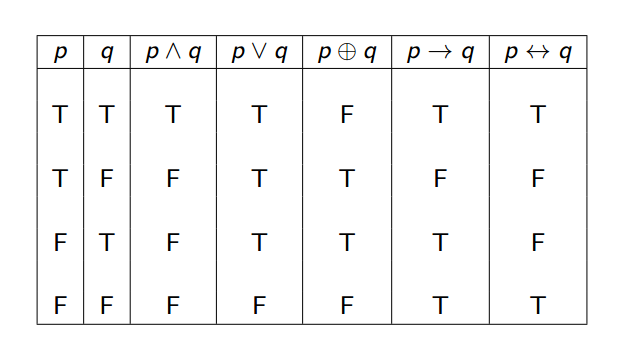
implication:推出

，如果p是对的q是错的，那么是false，不然都是对的（错推对也算对）

p叫做premise假设，q叫做conclusion结论

implication意味着，如果p是对的，那么q一定是对的，但并不意味着Q是对的，那么P就一定是对的

Biconditional:当pq同为对或错，true，



Arity:用来描述一个operator所使用的operand/argument的数量

unary operator 只用一个参数，比如否定negation

binary则使用两个参数，上面的除了negation都是binary operator

Formula

命题逻辑propositional logic也称为符号逻辑symbolic logic，因为它使用称为命题变量propositional的符号(也称为命题字母、句子变量或句子字母)以命题的形式表示知识

propositional variable与逻辑连接符symbols for logical connectives和圆括号构成了alphabet，让我们可以从这个alphabet中生成propositional formula命题公式

现在我们归纳性的把formula这样描述：

atomic formula：每一个propositional variable本身，比如p,q，不需要任何符号//最基本的formula

如果∅是formula，那么¬∅也是formula//formula加上否定符号仍是formula，这里的∅只是代表一个formula，而不是空集

如果都是formula,而是任意binary连接符，那么也是formula

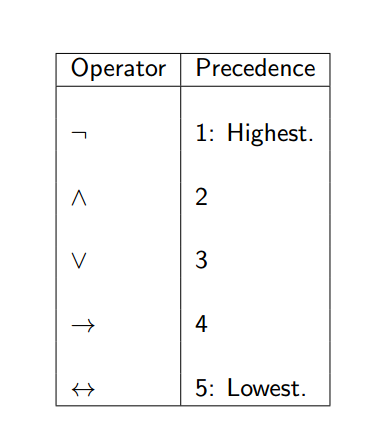
上面这组描述被称作grammar，可以帮助我们决定formulas语法上的的正确性

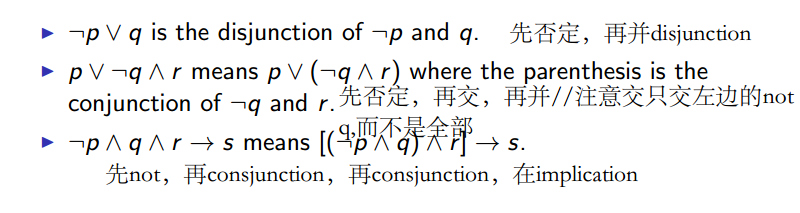
比如就是语法正确的，而就是语法错误的，所以他就不是formula

compound proposition复合命题：把Individual proposition连起来，我们可以吧proposition理解为陈述句。

除非用了圆括号parentheses,否定negation的优先级是compound proposition中最高的，

conjunction第二交集，disjunction并集第三，implication第四推出，biconditional第五互推





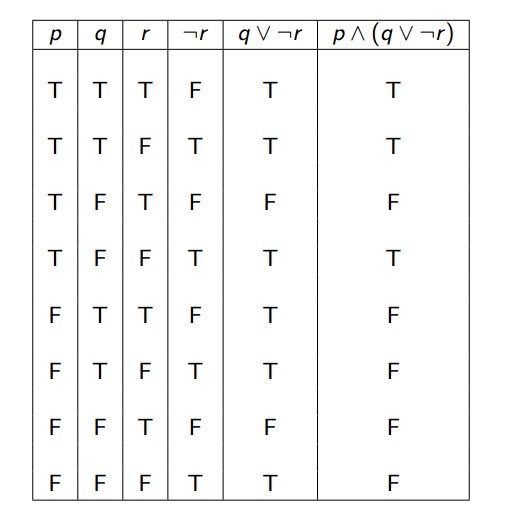
让compound proposition可视化：truth table

为了创造一个compound proposition的真值表，我们必须创造足够的行数row来包括所有的可能性（interpretation）

已知每个individual proposition只有两个值，错or对，我们一共可以创造2^n行

描述individual proposition值的列叫做operand columns//PQ值

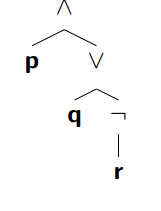
描述compound proposition叫做result column



左三列是operand column,右三列是result column

另一个让compound proposition可视化的方法是expression tree，System specifications



，外部节点leaf是individual proposition， Non-leaf node是逻辑符号

tautology 是一个永远true的compound proposition无论变量如何改变

contradiction永远false的compound proposition无论变量如何改变

contingency，tautology或contradiction，当compound proposition的值与value无关的时候

System specification系统规范，

用自然语言表达的句子通常出现在系统规范中。

这类规范的好处是可以阅读的。

坏处是有些啰嗦和含混

我们可以把这类句子转换成逻辑表达来取出ambiguity含混性，并且决定这组句子是否是一致的或可以推导出结论的.

logical equivalance逻辑相等

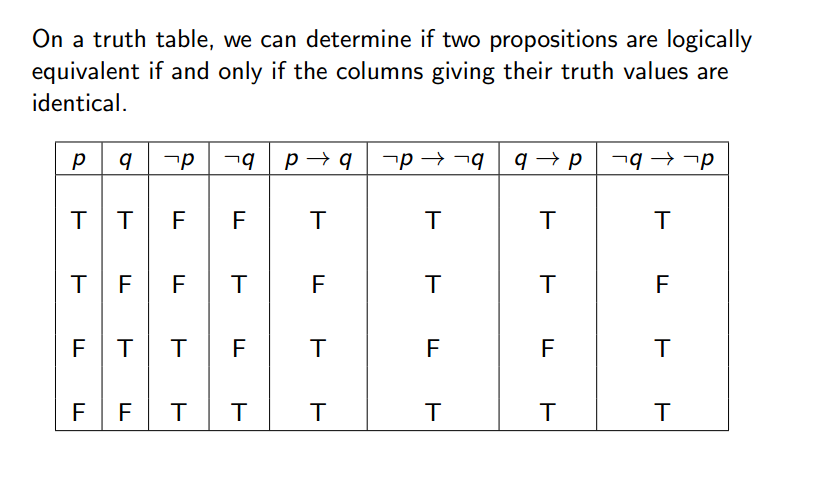
一组sentence是consistent一致的如果存在可能性，让每一个sentence都是true

两个proposition p与q被认作是逻辑相等，记做如果任意可能性他们都有着相同的truth value换句话说，是永真式tautology。

Inverse 否命题

converse 逆命题

contrapositive逆否命题

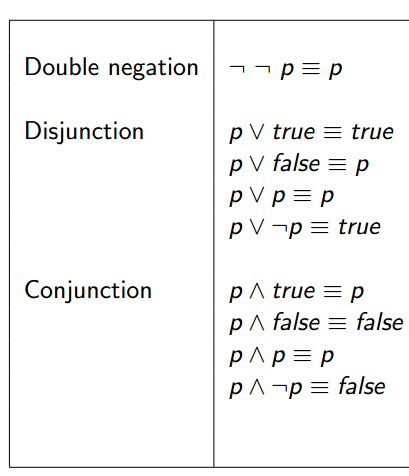
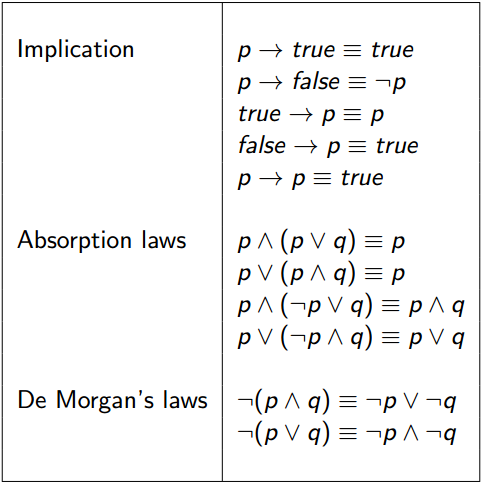


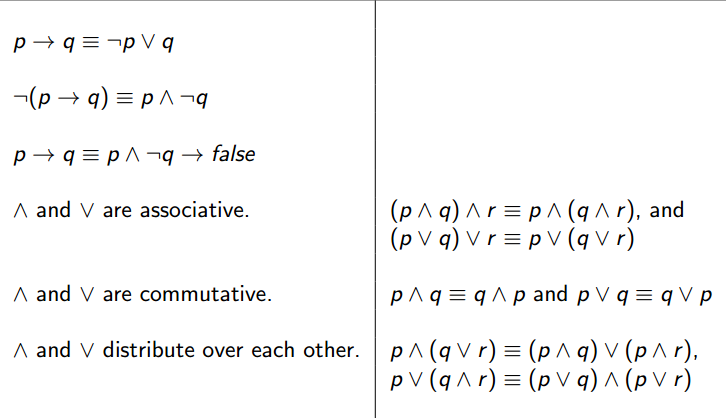
在真值表上，当且仅当给出真值的列相同时，我们可以确定两个命题在逻辑上是否等价。



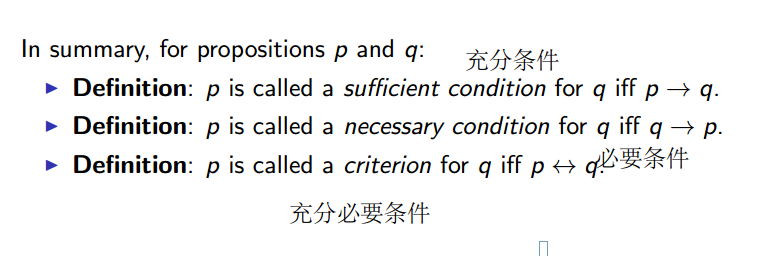


基本equivalance规则



对于命题PQ，



sufficient condition充分条件，necessary condition必要条件，criterion充分必要条件

argument：argument是一组句子组成，其中一个句子作为结论conclusion，其他的句子作为premise前提

Validity：有效性

一个argument被称作valid当且仅当

所有前提premise是对的而结论conclusion是错的这种情况是不可能的。

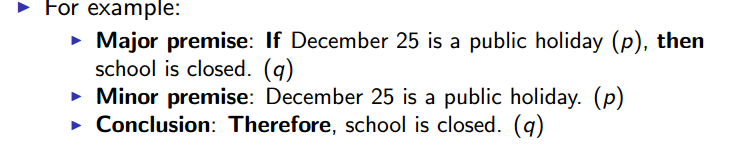
invalidity，存在情况所有premise对而结论错

Syllogisms三段论

一个argument由两句前提premise与一个结尾conclusion组成称为syllogism三段式，第一个premise称作major premise主前提，第二个称作minor premise次前提。

modus ponens演绎推理，三段论的一种：

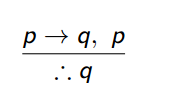
这个符号读作yield 产生



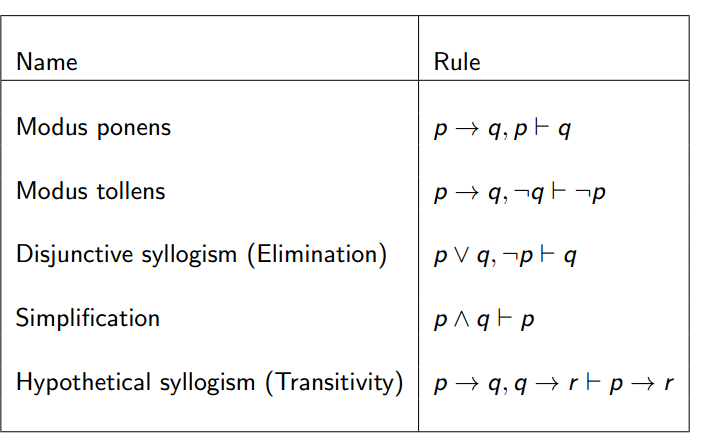
前提是P->Q.已知P，所以我们能得到Q

inference推理：通过一系列前提premise得出一个结论的过程

rule of inference，valid argument form，有很多种，modus ponens就是其一

也可以写成以下形式

inference rules 也叫做common validating rules



truthfulness validity的区别

truthfulness是Individual proposition的一个性质，取决于P/Q是对还是错

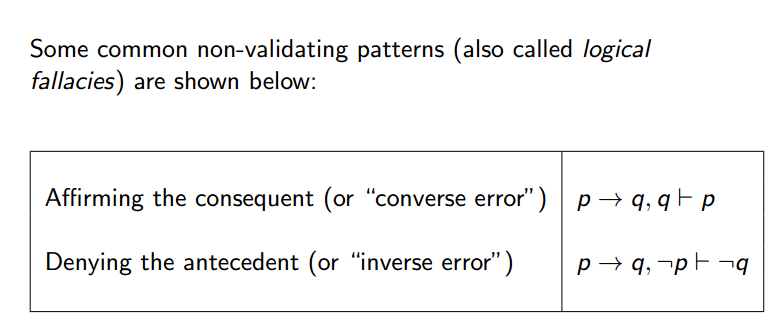
validity是argument forms的性质，取决于是否有一种结构是premise对而结论错

换句话说有效推论就是，如果前提对，那么结论必对

编程语言中的条件语句

绝大部分编程语言支持implication//推出，比如if then else结构,nested if结构，condition?expression1;expression2结构，unless then else结构

有些常见的无效推论也叫做逻辑谬误



soundness：一个argument被称作sound如果他是valid的，并且所有前提都是对的。

注意可能储存在deductive argument，逻辑是对的，但是前提就出错了，那他就不是sound的

所有argument要么是valid的，要么是invalid的。而valid argument中，要么是sound的，要么是unsound的。

在日常生活中，人们往往通过conclusion正确，认为论证是有效的

然而一个argument是否valid只取决于他逻辑本身，是不是sound只取决于前提是对的

最后总结：truthfulness, PQ的正确性，VALID，argument推理是否有效，sound，推理前置条件是否正确。

有可能一个argument是valid但是结论是错的

也有可能一个argument是invalid而结论是对的

一个argument推论只有一种情况是可以被接受的，那就是sound

