temporal logic 时序逻辑

在传统的propositional逻辑中，一个formula的true值是constant一个常量，要么是对的要么是从错的

例如proposition命题，地球是圆的，什么时候都是true的

modal logic

modal logic 模态逻辑，modal logic模态逻辑是一组向形式逻辑formal logic提供扩展的相关系统

modal是一个用修饰来限定一个陈述句statement真实性的表达

例如

proposition”the earth is round”

可以用一组modal来限定

it is necessary that the earth is round

it is possible that the earth is round

Person a beleaves that the earth is round

计算机科学重点关注modal logic中temporal logic这一部分 时序逻辑

如同他名字，temprol logic 提供了一种概念，就是对formal logic进行了时间上的拓展

一个formula 可以是true sometimes，有时候又是false，但不能同时是false and true

我们把时间看作是一组moment（state）的集合。并且无限延伸至未来，直到我们可以忽视它

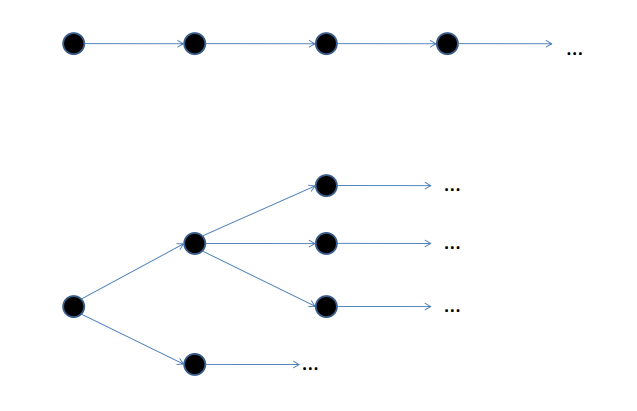
我们也可以明确给某一个moment指定下一Moment，这个下一moment是有具体时间的

这些moment串成的sequence叫做path

我们也可以给任意一个moment，指定多个next moment

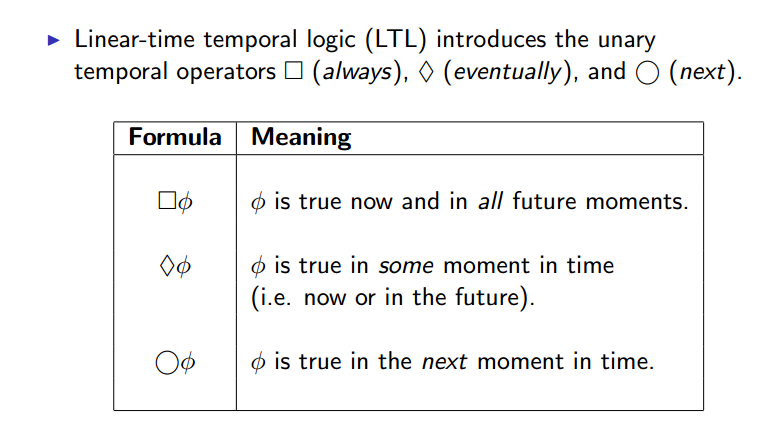
一个moment制定一个的叫做linear model

一个指定多个的叫做branching model



Linear-time temporal logic:

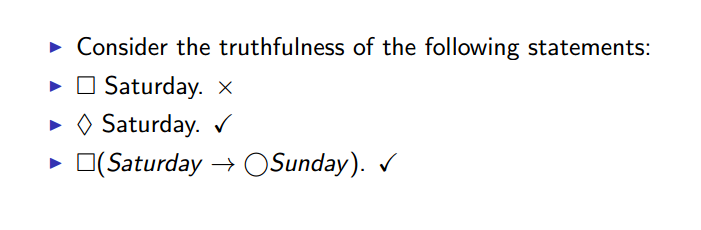
linear-time temporal logic(LTL) 用的都是一元符号



□一直是true

◇有可能在这个或以后的Moment正确（path里有一个是正确的

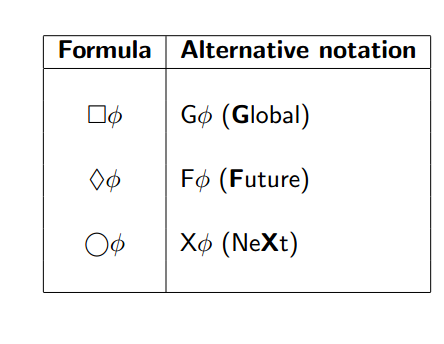
圈，下一个Moment正确

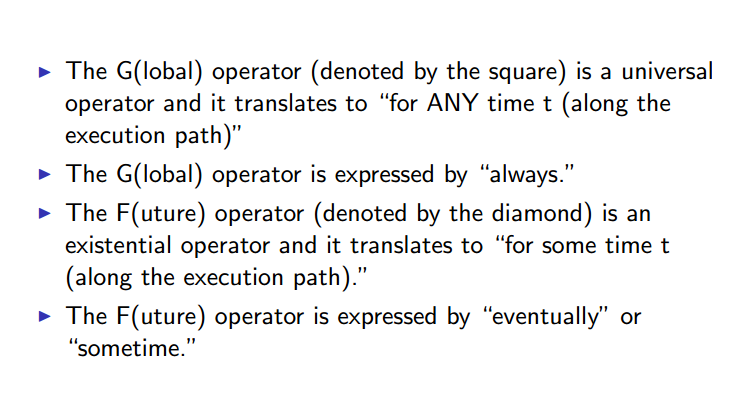


一直是saturday ,x

some moment是saturday

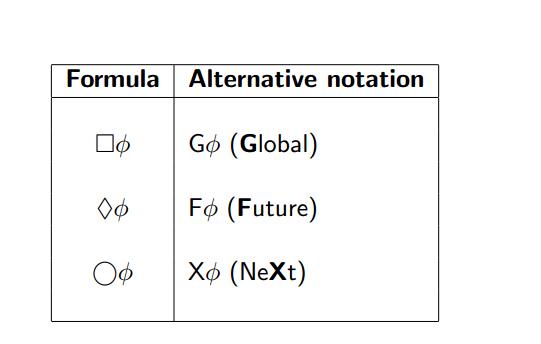
saturday后面的圈sunday一直是正确的，

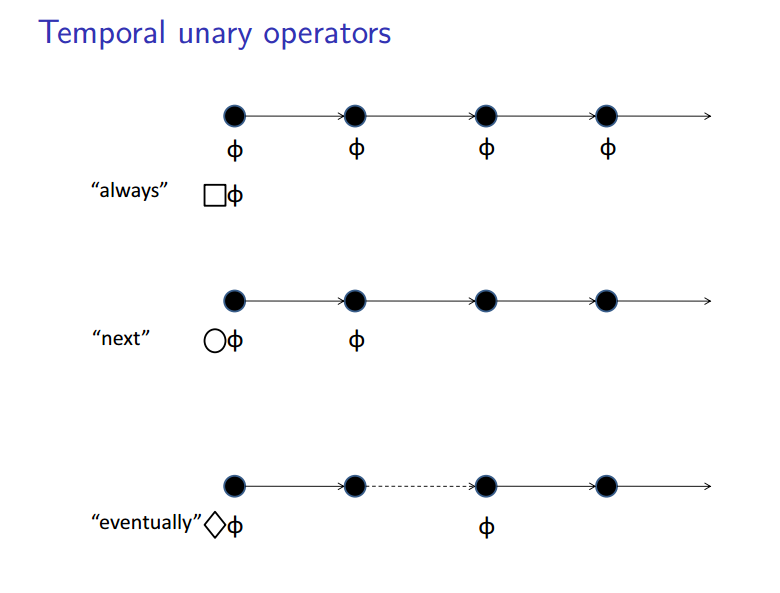




G operator是global的意思，对于所有moment

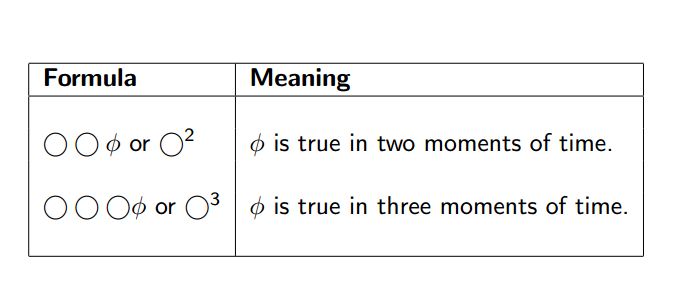
F意思是future，对于正在进行的path的有些时候

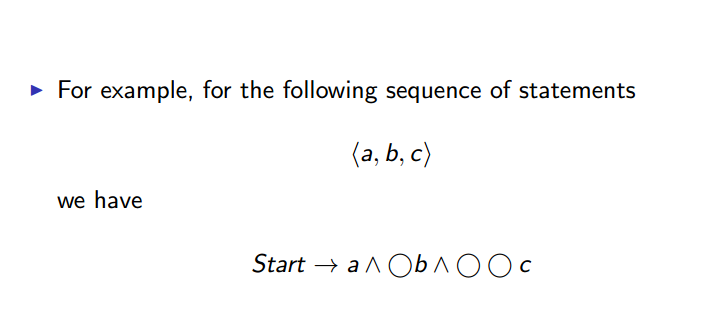
替换选项



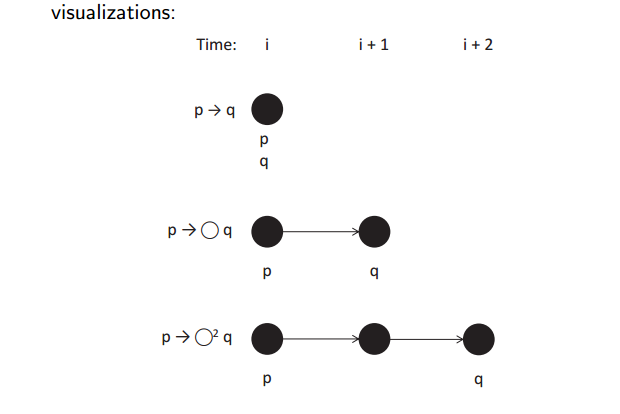
next是对将要发生的Moment加的符号。

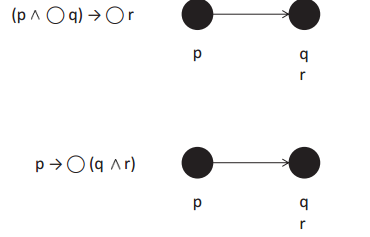
k阶future moment





对于这个sequence，注意是将要发生的moment前面加圈





注意区别，箭头代表的是推出，而交集才是陈述

虽然这两组moment是一样的

但是第一个代表的意思是，如果P的下一刻是Q，那么下一刻也会R

第二个意思是，如果P，那么下一刻就会Q且R

代表这个东西最终是true(未来某个moment)，但是在这个moment之后不保证仍然是true//只能指定true一次

如果我们想让这个东西sometime开始无限true

//可能第一次true后还有false，但是还是有true//指定多次true，但不是一直true

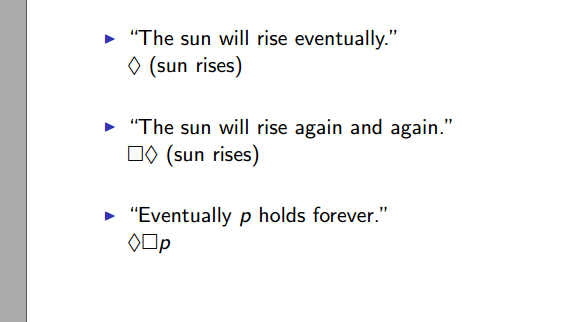
□与◇是队里的

，不永远true=存在一刻false

反过来也是成立的

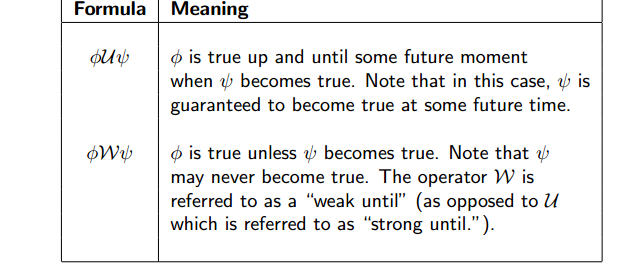


逆否命题， 如果某一时刻是true，那么就不会是一直不是true的状态



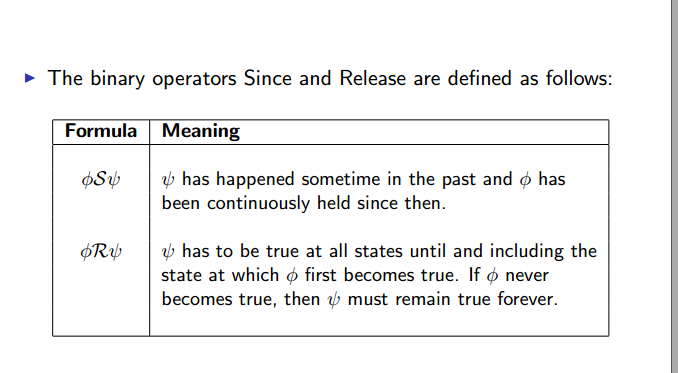
注意◇□与□◇区别，前者是从某一刻永远，后者是不间断的总是有true

binary符号 在Linear-time temporal logic



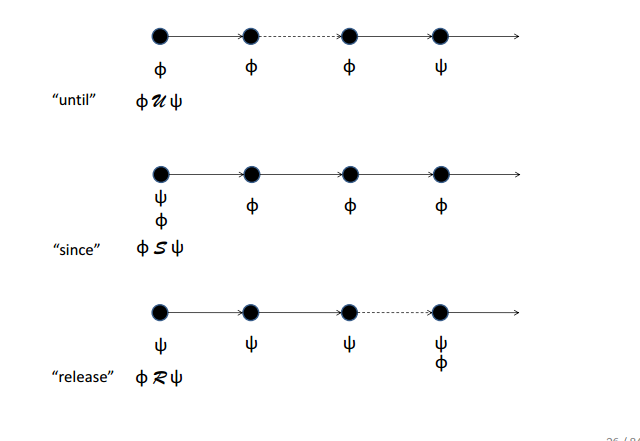
U，语句一直是true直到未来某一Moment另外语句是true，在这个语句中，语句B是在未来某一时刻一定会true的（相当于◇），也叫strong until

W，语句A一直是true除了语句B是true的，但是语句B并不一定在未来某一时刻会变成true(一直是FALSE)，也叫weak until



S:语句B将在某一刻发生，从那以后语句A就一直是true

R:语句B必须一直是true直到语句A变成true，在A实现的那一MOMENT，B仍然是true的，如果A一直没发生·，B就永远是true的



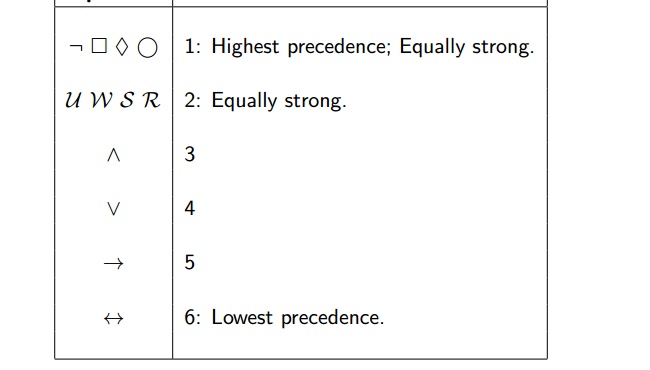
U是瞬间异或，S是遇B瞬间持续改A，release是直到遇到A才释放B

理解，A一直对Until B， A一直对 SINCE B(完成时)。 A RELEASE了B，

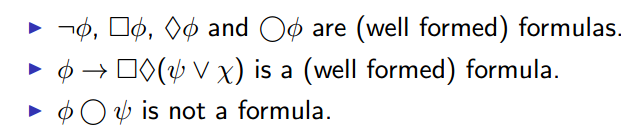
一元符号的优先级永远高于二元符号

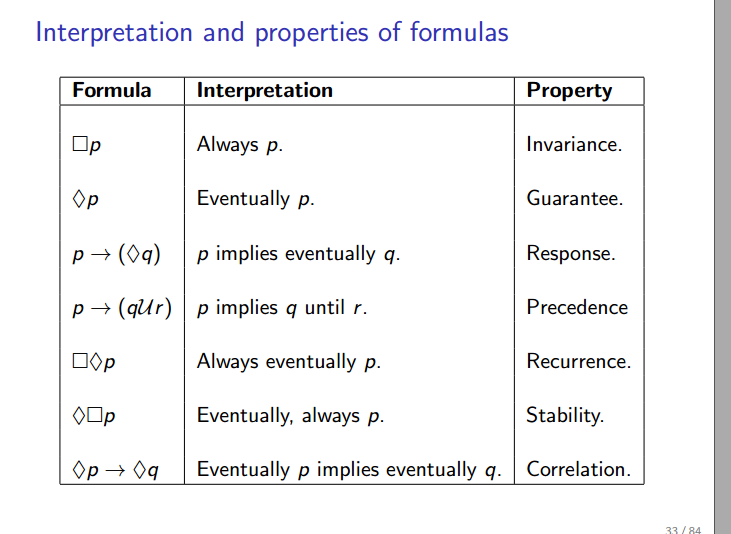
、

时序符号优先级大于∩∪推



well formed formula，没有结构问题的表达式



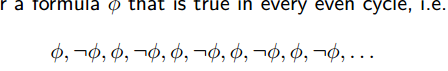


Expressiveness of temporal logic

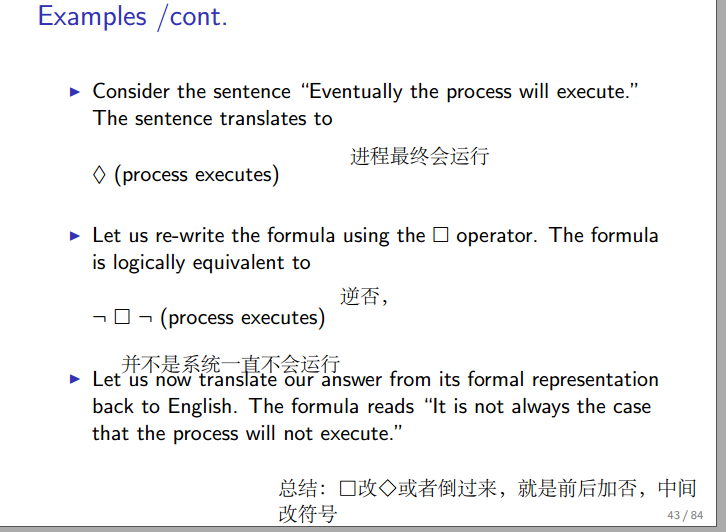
temporal logic的表现力

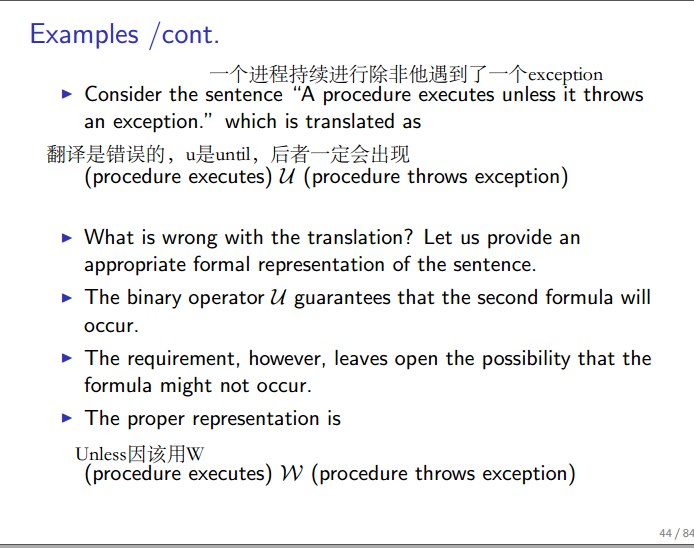
linear temporal logic 是无法表达所有的时序逻辑情况

假设一个formula，单数true，偶数false



LTL formula是无法表达这样关系的





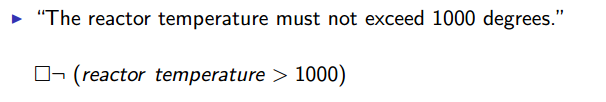
concurrent and reactive system的性质：safety 并行反应系统

safety意味着有些bad thing不会happen。

一个buffer必须不能同时接受超过一个wrriter‘这种 safety性质叫做self exclusion自我排斥

不会有两个进程同时被block，A进行，B等到A进行完以后继续，这种叫做deadlock freedom

关键的·temporal 概念就是引入never，这件坏事never发生，确保了safety



concurrent and reactive system的性质：liveness活跃度,

意思是something good 最终会happen

一个thread最终会进入CS来完成他的任务

关键词就是◇，at least once

concurrent and reactive system的性质3：fairness公平

fairness意味着对某一服务的请求最终都会得到确认并进行对应操作

fairness通常都会被用来证明liveness

starvation freedom：假设一个writer不停申请shared buffer（attempt something continuously），那么他在未来能否成功申请到，把fairness分为了三类

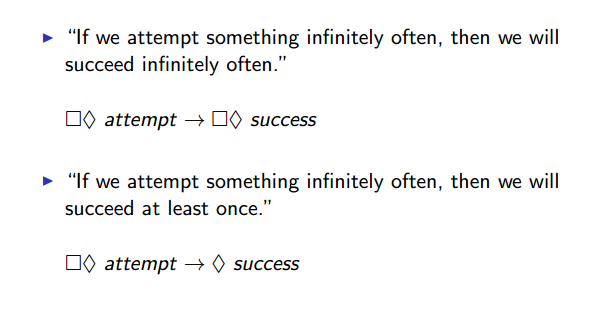
Fairness又分为三类

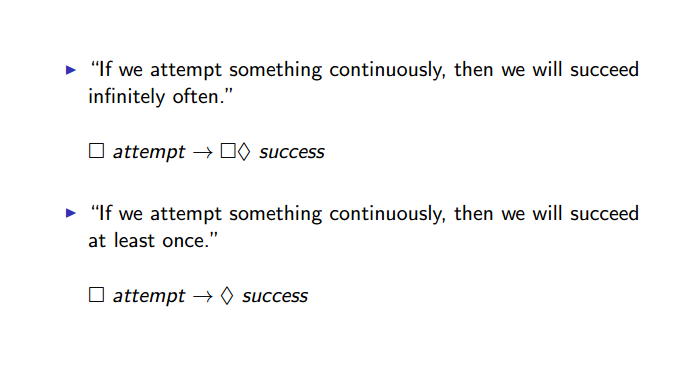
weak fairness：如果一个程序持续申请/尝试，那么它终将被无限次成功进行（l两次成功之间可能有间隔）

strong fairness：如果一个程序被无限次申请尝试（有间隔），那么他就会无限次成功（有间隔）

unconditional fairness，这个程序将无限次被运行（有间隔）

有间隔的叫做infinitely often，符号是□◇





at least once是◇，continuously是正方形

