1.propositional logic

概念翻译梳理

proposition:陈述句，命题

atom:不可分解的陈述句

compound可以分解的陈述句

判断命题唯一标准，有句号，陈述句，有正确错误而不是未知

negation否定，conjunction交，disjunction并，exclusinve异或，P与Q要不一样,implication推出，biconditional双箭头

前提primise，结论conclusion

formula，用字母括号逻辑符号组成的propositional logic，formula一定要语法正确，不是formula，单字母是atomic formula

优先级，否交并推互

tautology永真式，contradiction永否式

inverse否命题，converse逆命题，contrapositive逆否命题

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

argument，一组句子组成，一句是conclusion，其他是premise

valid，逻辑正确，

sound，逻辑正确的情况下并且前提也是正确的

consistent，一组句子有没有可能，是同时对的

propositional formula就是proposition

2.Predicate logic

Predicate logic 谓词逻辑，与propositional logic的区别在于 predicate logic可以给class中的Object描述特殊性质

比如说

untyped set comprehension ， typed set comprehension

quantifier，universal quantifierexistential quantifier 

bound variable，有一个free variable x ，我们给他赋值，或者加一个quantifier，他就成了bound variable

quantifier的优先级高于其他所有的逻辑符号，但是低于否定

意思就是对于任意X，P(X)成立，与Q(x)tf,的并集

则是p(x)与q(x)的并集，

否定的变换：否定符号往里移，quantifier对换

Categorical proposition 直言命题分为四类

A all A are B

E No A are B

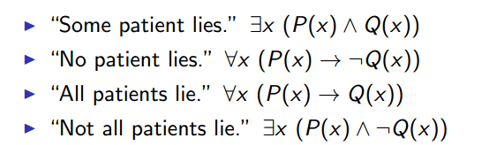
I some A are B

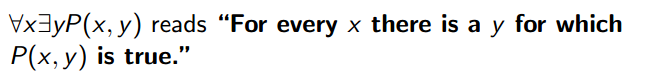
O some A are not B

affirmative肯定， negative否定 Universal 全部， Particular部分

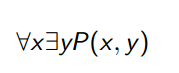
A :universal affirmative

particular用存在交集，universal用任意推理



nested quantifier

否认nested quantifier





否定写进去，任意存在互相转换

Contradictory categorical proposition互相矛盾的直言命题

比如，每个人都有一个房子，他的contradictory statement就是，不是每个人都有房子//假如前者对，后者错，假如前者错，后者对

AO EI

Contrary categorical proposition 相反直言命题，不能同时对，但能同时错。

所有人都有钱，没人有钱。A对B错，A错B对。但也可能有人有钱有人没钱，所以A错B错

一对universal或一对particular是contrary

subcontrary categorical proposition :不能同时错 但能同时对，

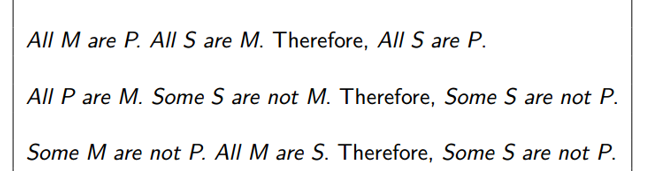
有的人撒谎了，有一个诚实的人。 A对B对可以，A错所有人都诚实，B错一个诚实的人都没有，就不行

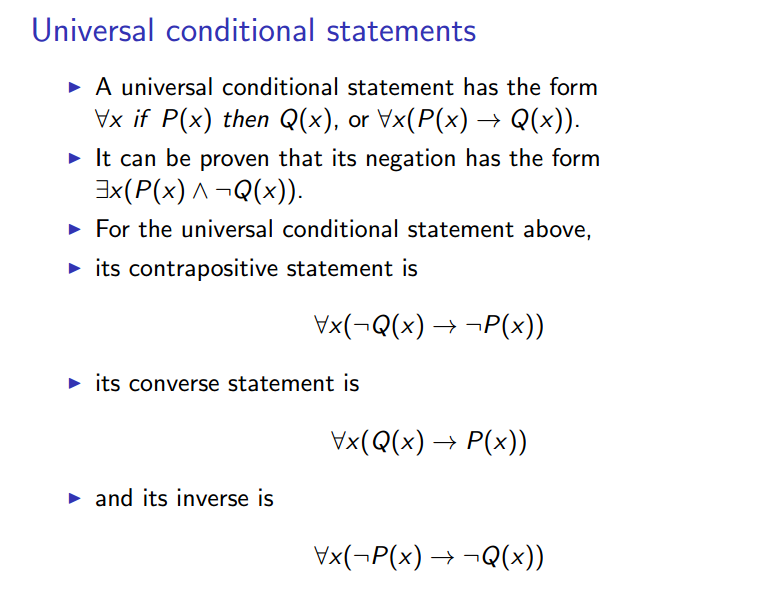
subaltern and superaltern categorical proposition，直言命题上下级关系

all是superaltern，如果all对，some对

categorical syllogisms直言三段论

也是由三部分组成，major premise, minor premise, conclusion。这三部分全部都是categorical propositions。





条件语句， A对应O， all A then B, some A not B, 逆否否逆这种不用改quantifier

set,集合

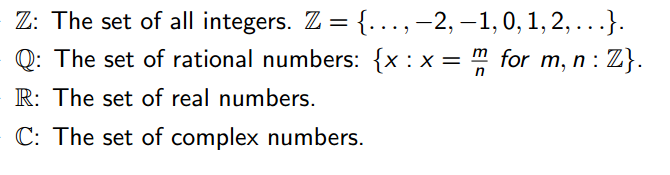
set是一组object集合，Object叫做elements

empty set 

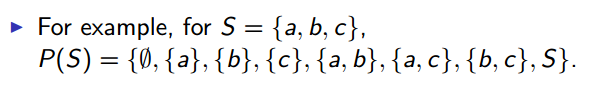
set有两个属性：element不能重复，element顺序不重要

enumeration：枚举，用来描述set的一种方法

comprehension：理解，



Z整数，Q有理数，R实数，C复数

Power set:记做P（S）所有子集

union并集

intersection交集

difference 差集，属于A不属于B ，也记做

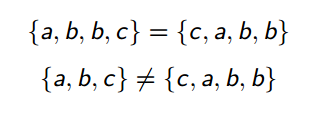
symmetric difference。 异或

disjoint 

cardinality,集合的大小





BAGS 顺序不重要，但是可以重复

tuple，坐标，但是数字部分用variable表示

圆括号，顺序重要，允许重复

对于两个集合来说，他们的ordered pair写成（a,b）形式a属于A，b属于B

笛卡尔积

List

list是可变的，tuple是不可变的，有顺序，可重复

我们可以进入list中的两个元素 ,head与剩下的tail

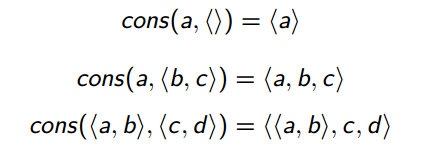


head是

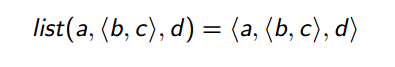
tail是

空List记做

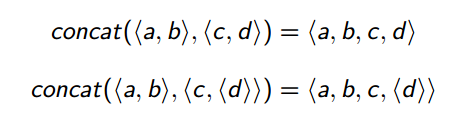
cons ，有两个参数，第一个作为head，第二个元素必须是 list形式，把第一个加到第二个里面



list，有多个参数，把所有参数组成一个新list，不回去除括号



concat: 把所有现有list连在一起，去除前后括号



String，一组有限零个或多个element组成的sequence，这些element都是从叫alphabet的有限集合中拿出来的

可以生成的string就有

没有element的string叫做empty string，记做

string的长度

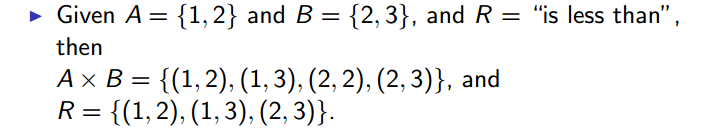
concat两个string

language，A生成的string的集合，

任何LANGUAGE都是的子集

binary relation: 两个元素的关系，

binary relation: 



换句话说，R的目的就是把那些满足relation的ordered pair(a,b)归纳连接起来

（A,B）的relation可以写成aRb,或R（a,b）,例如a<b.<(a.b)

relation的性质

1.reflexive:自反性，对于任意一个A中的元素a，aRa, 这个元素自己和自己满足关系，例如divide//整除，is equal to//等于

2.irreflexive/antireflexive

A中所有元素a都与他自己无关，例如大于,不等于

注意not reflexive 与irreflexive是不一样的

比如relation likes，他不是reflexive，不是每个人都爱自己，但也不是irreflexive，因为不是每个人都不爱自己

3.Symmetric,对称性，任意两个元素，如果a b之间有关系，那么b与a之间也有关系，例如，是血亲，互相结婚

4.assymetric,不对称性，任意两个元素，如果a b之间有关系，那么b与a之间一定没有关系，例如，a是b的父亲，a小于b

一个关系不可能又是symmetric又是assymetric

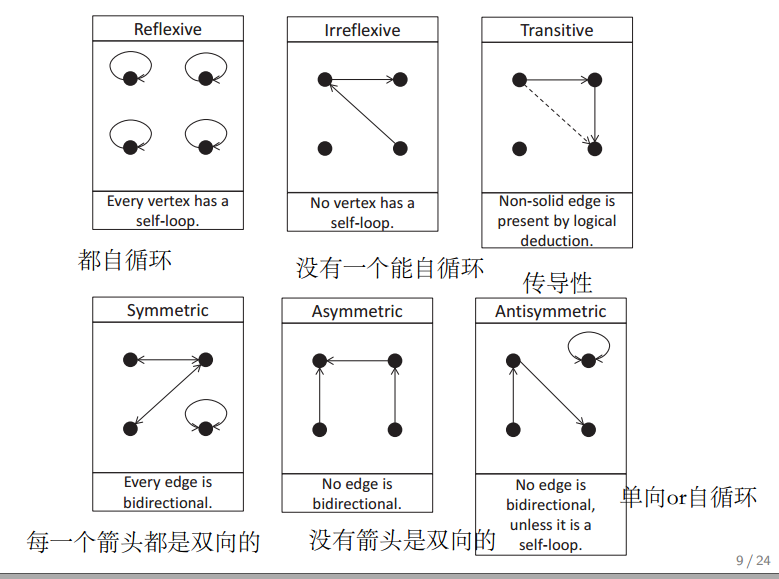
注意asymmetric不代表着not symmetric，

可以既不是symmetric又不是assymetric，例如likes

5.antisymmetric，反对称的

对于任何元素，他与自己是有双向关系的，如果他和别人有关系，那一定是单向的

6.transitive 传导性，任意abc，如果a与b有关系，b与c有关系，那么a与c有关系，例如，ancestor



relation的域

我们知道relation实际上有两层含义

1.两个元素之间的关系

2.ordered pair组成的set

domain 第一个元素的集合 range第二个元素的集合

binary relation的inverse，任意binary relation R都有一个inverse relation，记做

domain range互换

equivalence relation，如果有任何关系既是reflexive, symmetric与transitive的，那么就是equivalance relation，例如，is equal to

partition

A被分割为几个无交集的子集，那么这个子集就叫做一个partition

partial order

任何一个关系如果是reflexive antisymmetric transitive，那么就叫做partial order

例如less than or equal to,（不是两个关系，是小于等于这一个符号）

一个描述partial order的set叫做poset

total order，如果set A中的每一对元素都满足partial order，那么这个setA就是total order的

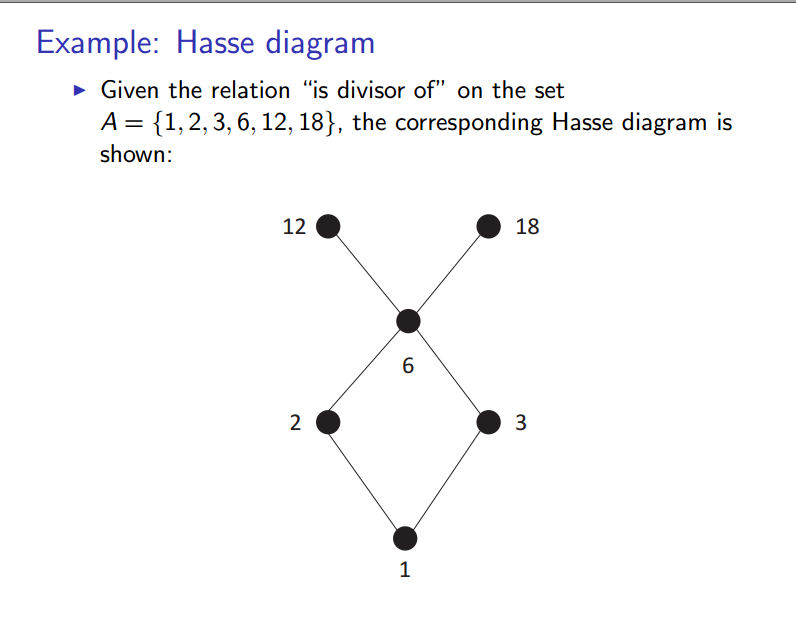
posets中的predecessors前辈和successors

如果R是建立在setA上的partial order， xRy，那么x就是y的predecessor，y是x的successor

一个element也许possess许多Predecessor

如果x是y的processor并且没有z，让zRy,yRz,那么x就是y的immediate predeecessor，记做x<y

Hasse diagram,是一个用来表达Poset的graph，如果x<y,那么x就在y的下方

..

任意Poset中，必然有minimal element//没有predecessor，与maximal element//没有successor

composition，给你两个binary relation，我们可以构建一个新的binary relation，这个过程叫做composition

例如，is brother of, is parent of，我们可以构建is uncle of，记做

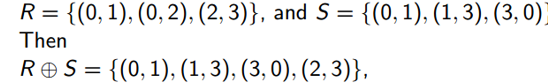
override, 把R所有的relation(第一个元素不在S的domain//第一元素集)加入S

R override S

1.原样copy S

2.找到S的domain， ｛0,1,3｝

3.R中的2,3不在domain中，加入



function:

这样的形式

A是function的domain

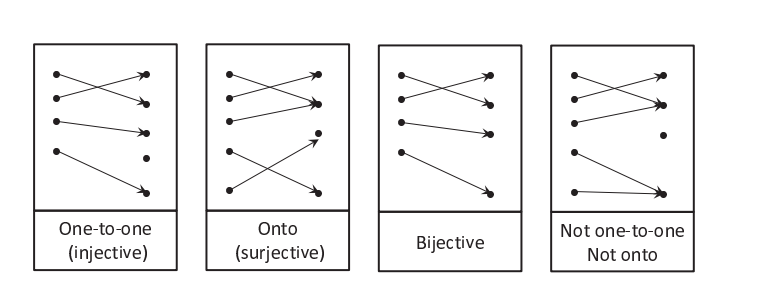
B是function的codomain

B中所有被对应到的元素的集合是range

propositional function也是function,比如P(x):x is an island P(Montreal)=true 就是function，这里true 就是B

任意一个propositional operator也可以看作function，交并否推互

partial function：并不会A中每一个元素都对应到B，只能说A中子集A'会与B形成function



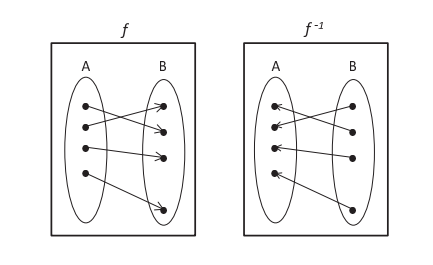
one to one codomain里面的元素最多被对应一次

Onto，codomain里任何元素都有对应元素

bijective,codomain里面任何元素都被对应且只对应一次

inverse

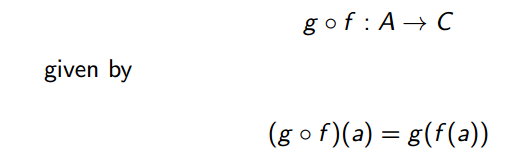
只有出现在bijection情况下才能有inverse



Composition



那么AC之间就存在composite relation



我们可以把function写成集合的形式

定义域： .

那么就是 

而ordered pair又是relation

本质上一个ordered pair就是一个对应关系

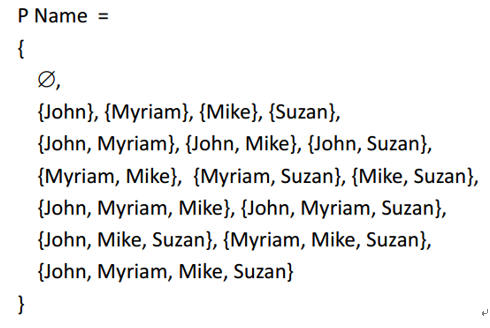
所以一个function，就是一个relation

Z-specification prelude

声明一个变量

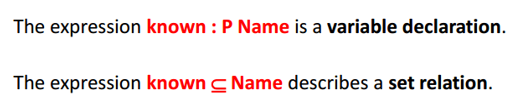


前者是variable，后者是type，有时候type需要我们自己指定



P NAME就是我们的新type，包括空集，也包括全集

新的变量,known的值可以使P NAME的任意一个元素

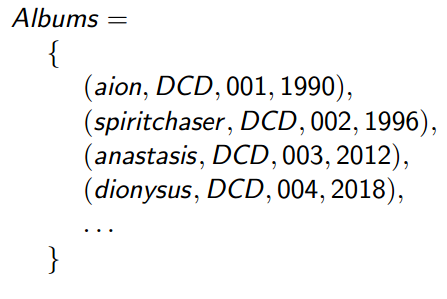


Z specification

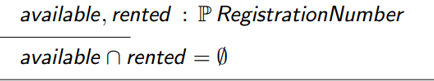
基本的构建过程

1,Declaration:声明，介绍这个变量

2.expressions:表示，描述了variable可以等于的值



3.predicate:描述，给这些变量可以等于的值加上一个限制constraint



Z中变量类型分为两种，simple type与composite type

simple type一共有两种，

1.primitivve type：Z里已经有的type

2.Basic type:用户定义的type  ，中括号括起来

composite type利用type constructor 在simple type上构建

1.Power set type:

2.Cartesian product types:笛卡尔积type



这样的一个type的变量是一个tuple

1.构建的方式就是X==Y，Y被缩写成X



2.，缩写的时候定义了一种新type，room number可以的值是右边type的子集之一//注意左边可以并不完全等于右边，可以等于右边的子集

3.我们可以明确指定这个newtype是哪部分子集，利用点



free type，：：==，用枚举的方式构建新的type/用递归的方式构建新的type

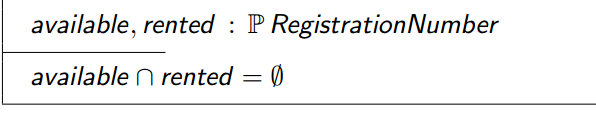


第二个就是递归类型，要么是nil，要么是不停的在原list的基础上加上N

Variable

一个variable就是object的名字，在declaration部分中写出来

axiomatic definition，用来描述一个global variable,并且可以加限制（也可以不加）



第一行，declaration,定义来了两类object,available,rented，他们都是 P registrationNumber的一部分

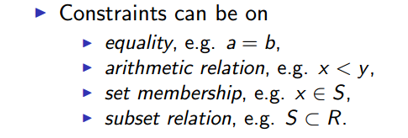
没有交集：限制





Z里面除法是div，区间是..,包括左右两边

Predicate 谓语，给Variable加上限制

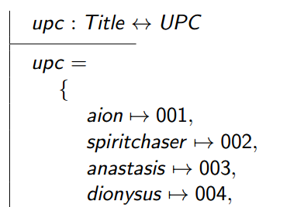


声明type:

声明简单type ，中括号一括

声明复合type，基于简单type用== 

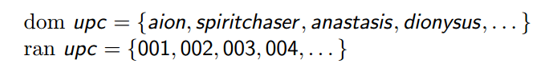
binary relation，由pair组成的set

,declare要写双箭头，具体要写单箭头

relational calculus关系演算,binary relation中的定义域与范围

dom，定义域，左边

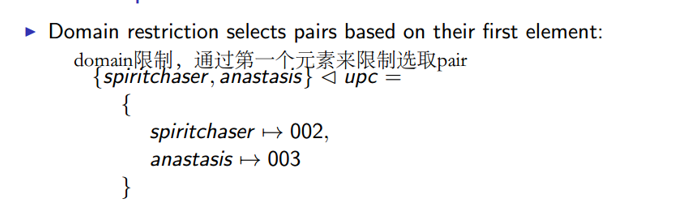
ran,右边

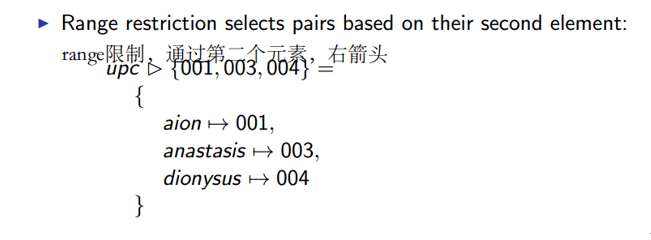


rational image， 左右两个D，



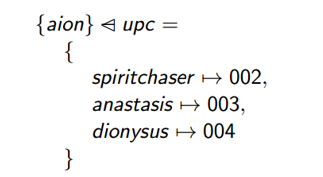
Domain限制

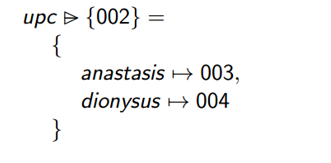




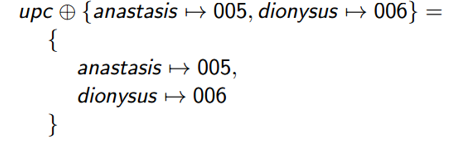
限制是提取我们指定的domain 与range

也可以用domain subtraction，他是一个符号,可以减掉我们指定的domian元素

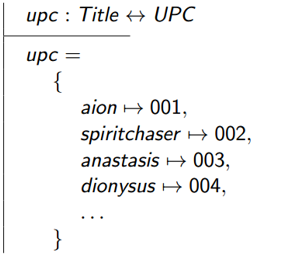
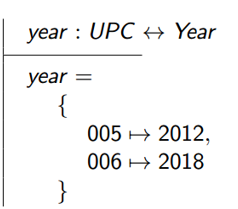


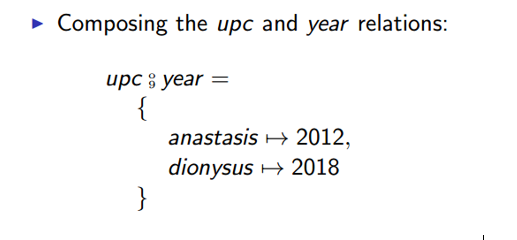


override，可以更新数据库，



compose：注意这里upc已经改了·，只是为了看原始模型

注意是两个小写variable的compose，竖着写09，舍去共用元素既UPC

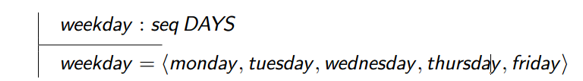
如果你描述的binary relation是function（特殊情况）

那么每个domain element都是一个独立Key，只能出现一次

relational calculus:Sequence:与枚举不同，我们可以用seq 来给enum出来的基本数据制定一个顺序



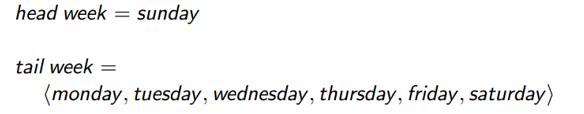
加上seq关键词



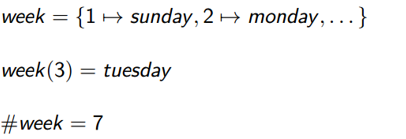
也可以用连接符

，左右List，中间连接符

还是用head tail的方式得到东西list数据



注意：sequence是可以被看做function的，有点像JS的array，只要Key自动生成12345

井号就是array length

Z specification中的数学符号演算

基本表示法

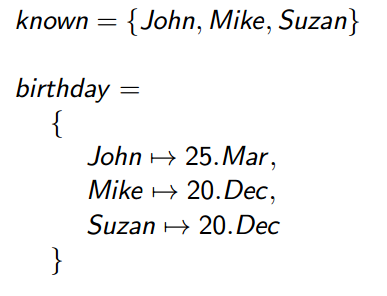
但也可以把它做成一个function 按这种形式

写Z specification

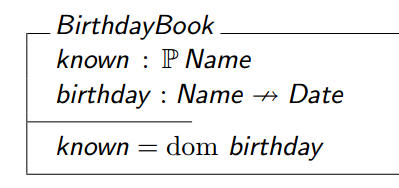
1.用中括号列出 基本type



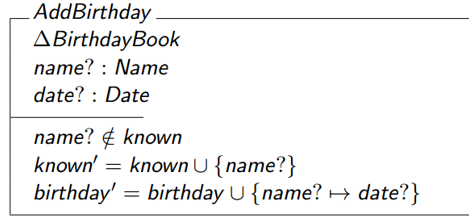
2.构建database



3.构建基本CLASS，这一步叫state schema，初始状态

因为Name很多，known不能完整包括，所以用的是不完整箭头，birthday就是一个function，最后加个domain

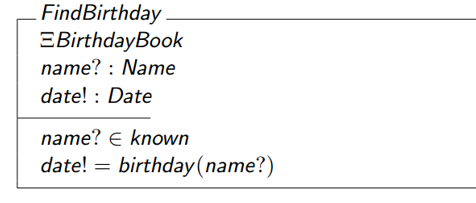
然后用操作schema来描述这个system不同动作



三角形会改变birthdayBook，问号是我们要输入的变量以及他的形式

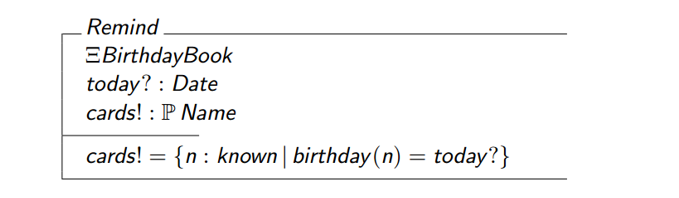
name?这里是限制，如果满足限制

后面的Known'就会被改变，这样database就被改变了



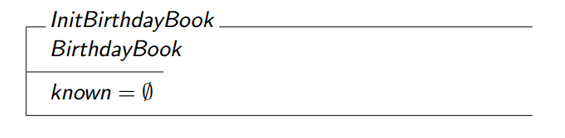
这个符号是代表着这个操作本身不会改变system的state

感叹号就是输出



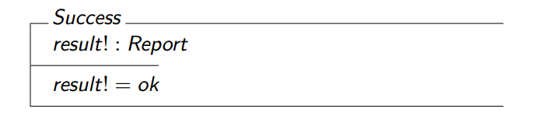
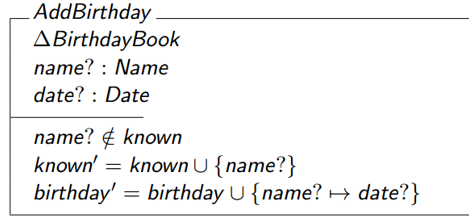
无先决条件，会return所有等于today的card

初始化（必须有）

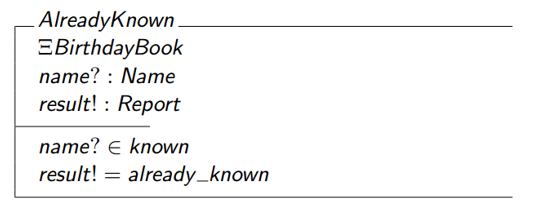


handle errors，我们必须要handle没满足限制条件的情况

加入新的type

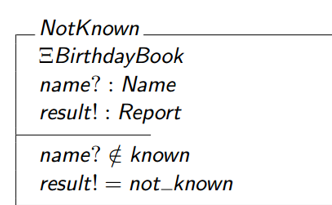
加入新success，这个是最简单的

然后加入Handleerror部分

输出Already known\_



得到R，也就是我们的final

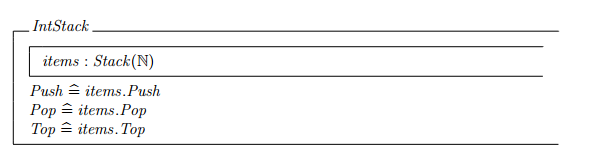
objectZ



state scheme中描写的变量是global变量，任意人可以使用既element,count,element',count'

实例化stack，说白了这时stack(T),T是没有具体指代的

intstack，用整数来实例化他



state:

变量，所有的item都是stack类型用整数来实例化



操作就是items.XX,

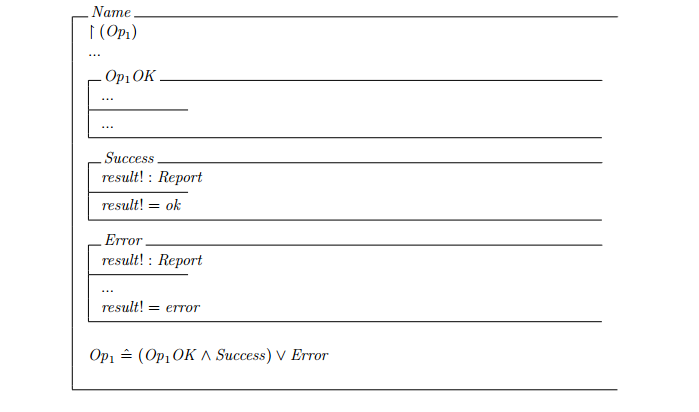
继承

继承一切，除了visibility list

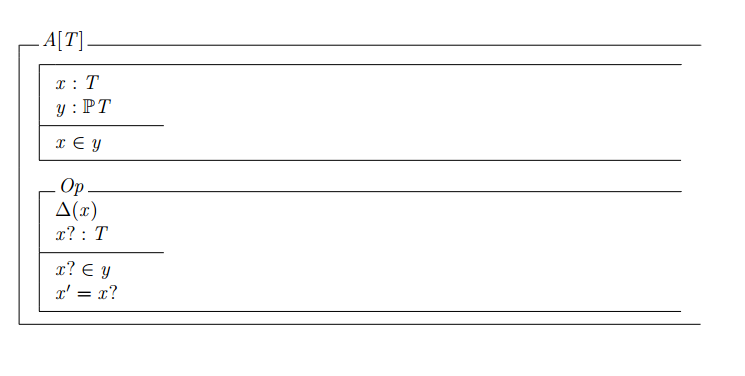
state定义的variable可以覆盖父类的variable。例如count:N,你可以改成count:Z,或者countN+,但是不能改成COUNG:STRING

如果继承operation,只需要写改动过的部分

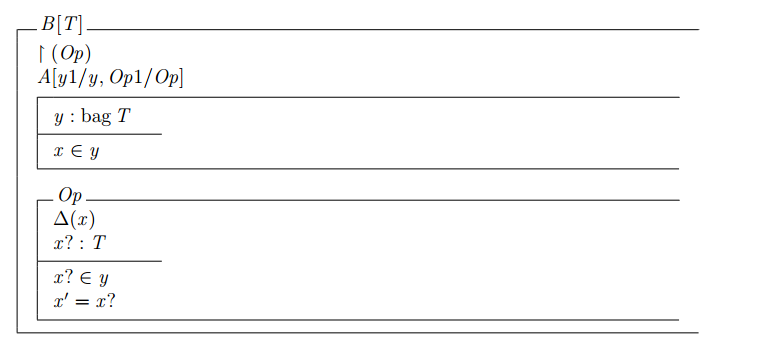
怎么控制error



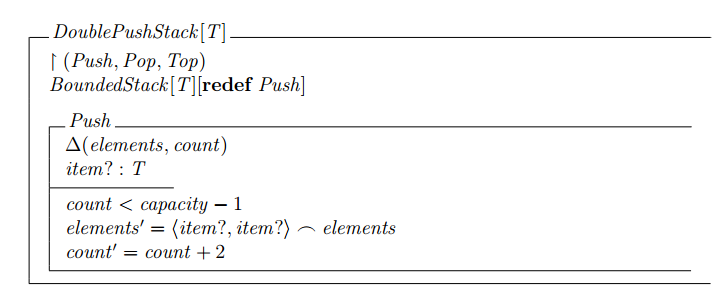
成功交success，失败并error，



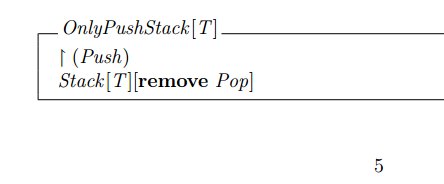
原来A[T]



继承B【T】，在继承这一步，我们可以把y改名成y1,Op改编成OP1，因为OP1与Y1都没被囊括在VISIBILITY LIST 里，就相当于删除了原来的y与OP，然后我们就可以完全重新定义Y与OP



我们可以指定要重新描述哪个operation



或者remove

看课件