大脑逻辑

首先，图可以用来表达逻辑，图确实可以表达一切，图可以比程序表达更加复杂的逻辑和流向，

将主动性，数据，逻辑集合在一起可以作为图中的一个节点，这应该就是大脑的运作模式

既然这样的话，那么自己写的那个2048如何用图来表示呢？

1. 手指向那个方向滑动，方块就向那个方向移动，
2. 如果移动的方向有方块进行阻碍，自己则不移动，移动的过程中，如果和相邻的方向的方块的数值相同，那么两者融合，并且随机的在空位置的地方产生一个方块。

这就是整个2048所包含的所有的逻辑。

将原子操作作为 图的一个节点，一个点在图上移动的过程，可以这样表达吗？

这个样子是不行的，

逻辑应该是图上的边，边拥有操纵节点的能力，

表达成逻辑图应该是这个样子

发送命令的时候，会激活边，（我现在不考虑减少复杂度的方法，只是设想将逻辑转换为图的方式），边会操纵节点，从而达到稳定状态，这个样子是正确的转换方式吗？好像也不是。

怎样将逻辑转换为图呢？

也就是说规律作用于事物这件事如何去表达呢？

事物彼此之间的复杂的联系如何去表示呢？

怎么搞呢？事物和事物之间的联系才是要表达的，

还是说2048，在这里面有几种事物？ 第一种，方块，第二种，方框，

方块和方块的关系如上所述。方块的一切运动必须依赖于给他定义的规则，

规则制定：

事物相互作用的图的表示

用节点表示事物，

事物内部具有各种属性，

被动函数：由于自身属性改变引发的函数体，

主动函数：不依赖与自身属性的改变，由具有某些能量的能量体催动引发的函数，

主动或者被动函数都是用于改变自身，或者其他事物属性的函数，并且属性的改变只能是增加或者减少。

转换为伪代码

Struct shuxing

{

String name;

Int value;

}

Delegate setshuxing(int val);

{

检测是否达到调用hanshuti的条件，

达到，调用函数体，否则不调用，

设置属性值

}

Strut hanshu

{

Int what;//静态指针，指向函数体要修改的事物

Int th；//指向要修改的属性号，属于自述型内容，共主控函数用于识别

Delegate hanshuti（int value）；//函数的具体内容

{

Setshuxing（value）；

}

}

类名字

{

Const Int what；//表示自己是什么的一个整形变量，属于自述型变量，

Shuxing【count】；

Setshuxing【count】//对应上面的属性

Beidonghanshu//被动函数体

Zhudonghanshu//主动函数，

}

//这就是类内部所有的东西，

事物相互关系对应的树状图

使用二位数组来表示

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 行列 | 0/事物对应的int值 | 1/被动函数序列号1 | 2/被动函数序列号2 | 3．。。。。。。 | 4主动函数1 | 5被动函数序列号2 |
| 0 |  | 对应的事物名称，属性号，以及对属性做的增加或者减少的值 |  |  | 主动函数标记，还加上被动函数的内容 |  |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |

具体事物对应的图，是否需要插入函数列，不插入好像也可以，有待考虑

2016/2/11 这应该是Condition 类做的事情，对于事物的状态进行一个记录。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 事物名字 | What标识符 | 属性1 | 属性2 | 属性3 | 属性4 |
| Name1 | 对应的值 | 对应的值 |  |  |  |
| Name2 |  |  |  |  |  |
| Name3 |  |  |  |  |  |
|  |  |  |  |  |  |

虚拟映像就是创建上面这个表的一个副本。

上述表也就是目前状态，

目标状态也是上面的一个表。

如何从目前状态到目标状态构建一条通路，构建通路的算法，构建之后的通路如何转换为程序流，

首先，由主控函数调用，从主动函数出发，任意次调用主动函数，催动整个体系的运转，现在不是想这个的时候，/

从目标状态到目前状态进行反推，或者从现在进行正推，要不要将what标识符和其中的某个属性结合在一起，组成一个结点，这个样子来构成图，就可以使用书上学过的最短路径算法，图中的节点的what标识符找到对应的事物，作为。。。。具体算法随后想。

还有就是从外界中获取的信息构建成自身逻辑的一部分的能力。找到事物的因和果，

同时给程序因和果，让程序分析其中的逻辑关系。

我似乎又想到了分析其中逻辑关系的方法。也是采用图的方法。

我现在觉得这应该是个逆向思维的过程，为什么呢？

例如说，有一些白色点的，白色点拥有一个在周围各个方向复制自己的能力，

在一个黑底的图纸上，如何出现一个字，在这种情况下如何构建通路呢？

如何在已有的逻辑上构建新的逻辑（逻辑：新的事物以及联系），

Part 1这一段的上面我自己写了，找到事物的因和果来构建彼此之间的联系，

Part 2：我在课上想到 现实世界是一个巨大的具体事物的联系的图，图中的一条连线就是一个具体的活动，

怎样向自己内部中添加新的逻辑呢，新的逻辑必然对应这着新的事物和事物之间的联系，加入到自己已有的逻辑图中时，保持图中各个节点联系的完整性就好了，

怎样区分单个事物和多个事物呢，根据联系是区分事物的标准这一准则，也就是说单个事物无法具备相同属性的两个值，如果检测到两个值，说明在这一相互作用下，他们是两个不同的事物，这是一个好办法。

在一个活动过程中，可以看到什么？可以看到各个事物各种属性值的变化，是否可以看到促使属性值发生改变的上一个被动函数呢？这个我不知道，通过单个变量变化来检测属性值的改变是可行的，是可以新事物之间的联系的。

如何根据事物联系，从目前状态到目标状态构建一条通路呢？（函数只有条件，属性值的增加和减少的其中一个）（原来这个算法才是真正难得地方）

情形一：目前状态和目标状态的事物没有增减，也就是说，从一个活动的开始状态到末状态，没有新事物的增加，也没有旧事物的减少。

每个活动都有对于这次活动来说的“活动主动函数”，也就是说这次活动中，完全由他进行动力输出，

首先，描述一下目前状态，有若干个事物，每个事物有若干个已经确定的属性，

自己某个属性的改变或许会牵涉到其他事物的某个属性的改变。

这里面应该说明一些问题，如果说，事物A的属性a的改变，会牵涉到事物B 的属性b的改变，那么就说属性a可以牵涉到属性b，记作“”（还没想好）

在这种情形下构建通路的第一步是对比目前状态和目标状态的属性之间的差异，并列一张表，

活动必须有一个发起者，进行一系列连锁反应，

怎样进行函数的调度，怎么调度?来恰好达到那种状态呢？怎样调度呢？，

利用定义的事物之间的相关关系，在具体事务之间画出一个具体事物的联系图，

如果这样的话，链接的线就会很多，而且不符合实际情况，应该给被动函数加上距离限制，使得具体事物之间的连线之和自己作用的连线。

有没有必要在同一事物的不同个体之间建立有一个标识符以区分不同个体。

事物的存储结构是一个以为列表 list<thing> what;静态的链表来实现树状结构。

可以通过访问列表来访问所有的事物，事物之间的逻辑结构通过属性来体现。

所有的东西一定要记录清楚，函数访问属性的时候首先要 在访问的对象体内留下自己的标记供可能的条件判断访问，删除工作有自己完成，或者其他之类的。

还有什么？

具体事物之间的相互作用应该如何实现，

如果按照之前想的那样，和任意两个可以发生相互作用的事物的属性之间建立链接的话，那样链接就会很多，依据作用的实际情况，只链接附近直接参与相互作用的，以自己所在的位置为起点，向周围搜索，而达到利用周围的事物，也就是说，一种属性需要建立一个网络。来连接同种属性，对吗？

崭新的模型（新想出来的更加符合实际的模型）

事物应该是继承自属性的。

定义属性的时候也应该同时定义修改属性值的原子操作。以及朝某个方向变化时应该执行的原子操作。

属性对象中应该包括的函数包括：

属性比较函数，根据比较结果决定调用的set属性函数，调用的次数，以及可能需要的参数值。

Set函数系列；

对象包含的内容

【

继承的属性对象集，

被动函数集，

】

由于属性改变而调用被动函数怎么实现。

路径连接设想：

找到发生变化的属性，利用属性对比函数，找到能使属性发生此变化的属性设置函数，再找到引发此属性设置函数的被动函数，再找到引发被动函数的属性变化，这样依次找下去，直到找到拥有主动函数的个体。

【属性】 将struct ，此struct的对比函数，以及根据对比结果所要调动的各种减小或者增大函数。（struct指属性集中的单个属性）

因为我还没有设想到循环结构如何实现，只是有顺序和条件判断，啊啊啊啊啊！

想一个能够应用这种逻辑的比较实际的例子。

属性使用单个数值表示的，就举一个在一条线上的物体碰撞试验吧，ok就这样了

具体介绍隔相等间距放置一个小物块，与地面之间有摩擦力，

目前状况，各个小木块不动，

目标状态，与具有主动函数相隔3个的小木块向右移动一定距离。

===============================================================================

属性集

属性一：位置 属性对比函数 :比较远近 set位置

属性二：速度 对比函数， set速度

属性三：摩擦力 set摩擦力

结束了，怎样设置他们之间的连接关系， 速度可以改变位置，速度改变位置的同时会因为摩擦力而减小速度的值直到速度为0.

这个过程没有考虑到物体的碰撞问题。每个木块可以访问所有的木块的位置，就可以实现碰撞了，根据动量守恒原理，设置碰撞之后的木块的速度值。

好像无论如何都少不了对附近位置的物体有无的判断，好吧。

如何控制循环次数，以及当木块获得速度进入了一种

在设计的时候，一个设计不好，就需要全部重新来进行设计。

事物是有主动函数，被动函数，属性的集合

属性: 值，set函数，check函数，所连接的passivefun组成，

世界: 事物，主动函数，因为某些需要而设置的特定函数的集合。

Alltheway: 对于事物发展的描述，是condition的集合。

事物之间的关系是被怎样描述的？ 存在于类的具体定义之中。

Passivefun : 包含哪些信息呢？

Condition类 是针对事物来讲的。怎样将对状态的描述转换为对事物的实时的一种状态，必须有这样的一种方法。

重新到一个白纸上画一遍，还有命名系统，怎么定义呢？

就采用c++的代码设计，还有就是值的计算，图的构造，状态类，世界必须有一个状态类，事物必须有一个函数，实现状态与状态之间的转换，还有世界也是，

还有必须实现路径类，状态与状态之间的转换可以产生路径，还有什么，

可以实现探测，可以预测事物的发展，可以准确知道。对事物有感知，

应该在world里面实现内部类，box存储事物地址，

假设我已经完成了目前的设计，接下类的就是路径设计了，

启动程序，根据判断标准来启动，

什么是路径， 函数的文本标识，

现在就只剩下两个问题了

问题一：被动函数的执行问题

问题二：路径构建问题

属性分为积极属性，和消极属性，积极属性可以改变其他的属性的值，所以在构建路径的过程中先设法满足积极属性，然后再满足消极属性，满足改变就好了，积极属性药提供一种机制来描述 自己改变所引发的其他属性值的改变情况，并且是可访问的。

大致过程应该是怎么样的？建立路径的时候，首先是对比两个属性，找出属性之间的差距，（找到一种描述属性之间差距的方法，需要吗？）

主动属性是直接通过调用属性的设置函数来进行调用的？（这个是否正确，正确）

如何描述属性之间复杂的关联关系，主动函数自己要附加一个表，有时候属性之间的关联非常复杂， 这样描述 为了达到积极属性所关联属性的某一种变化，而应该将积极属性改变的值的多少？ 这样只能描述

存在好多个集合，首先是 主体可以调用的属性设置函数（主体并非可以调用所有事物属性的设置函数）

目前状态，目标状态，

还有 主动属性，还有被动属性，我需要构建一个游戏模型。

完成这两个问题，我的包就完成了，我应该好好想一想。

这个文件到此结束。

谁可以解决这个图片中的数学问题？或者需要学习那些数学知识才能解决这个问题，问题可能有点笼统。

游戏中存在两种点，一种叫死点，一种叫活点，

每个点都包含有一个数值，对于死点和活点的集合叫做域，

你拥有其中一部分点，叫做拥有点，并且拥有点中至少包含一个活点，你只能改变拥有点的数值。

死点和活点不同的是改变死点的时候，其他点的数值不会发生改变。

活点在发生改变的时候，可能会改变你拥有点之外的点的数值，当你能按照自己的期望去改变有拥有点之外的点的数值，这就是学习的过程，就是你的拥有点在扩张。

但是去改变活点的时候，其他活点或者死点的值就会发生变化，具体怎样变化你可以获取，你也可以获取活点中所包含有的逻辑，

（活点本身包含一种逻辑或者说是函数，这种逻辑可以是很简单，也可以很复杂，具体情况是那个点的改变可能跟其他点的状态有关，也可以没有关系，但是你可以获取在某种情况下活点对于其他点值的改变情况，也可能改变拥有点的情况，这种改变情况是任意的）

现在有这些点的两种状态下的两组值，分别命名为状态1和状态2，

问题：如何改变状态1中各种点的属性值来达到状态2，在哪些情况下不能通过改变状态1中的属性值来达到状态2，即 两个状态之间无法互相变化，

上述问题，我觉得自己的描述本身就有问题，由于函数本身可能包含复杂的逻辑，我应该仔细想一想，或许可以进行统一的描述，

除了点和线以外，自己今晚决定在基本表达元素中加入圈，来表达一种集合关系，就像是 学校这个名词 一样，他可以提供 教书育人这个函数，他有一些属性，但是同时他又是一个集合，有内部的逻辑和子元素，

逻辑，规则，知识，定理 都是对 某一个范围内 事物的函数，和属性的某种关系的描述。这些规则可以跨越函数所建立起来的链接，不需要进行活动 推进 来进行的可能的部分的验证，也可以说是规定。

对事物进行重新定义，事物可以进行剖析， 事物 具有函数和属性这两个值，这个是事物的对外表现， 事物本身可以叫做一个集合，事物是 由若干具有属性的事物和事物之间的关系，组成，（逻辑被定义为事物之间函数链接关系），

我需要更加深刻的理解函数，函数是什么？我之前的理解是函数是对目的属性的改变，是否这样太过狭义了，

对事物的拓展不必基于已有的知识网络，一个网络可以代表一个世界。

一个事物具有属性和函数，这是事物的表层概念，事物本身是由子事物和函数组成，事物本身的函数都可以表达为子事物的某个活动的执行序列。

如果这是一个正确理论的话，下一步应该怎么做呢？

我并没有想到如何构建路径的问题，以及构建节点的问题，如果这是真正的人工智能的表示方法的话，那么下一步应该怎么做。

下一步或许应该进行学习，如何进行学习，从基本事物开始，

从“我” 开始，从自然语言理解开始，来构建知识网络，

由于语言中，包含很多的抽象的逻辑，所以说应该怎么办呢？先行搁置，

从单个事物的网络拓展开始，

因为我的结构表达中，包含哪些结构元素呢？

我的逻辑世界包含的结构元素：

节点:

属性：

函数：

节点本身所包含的事物逻辑：

活动：在实例世界中，若干函数和节点的连线，

规则：某个节点内部，子节点之间属性所包含的某种关系，

如果要进行自然语言理解，必须区分出节点，活动，属性，以及对于属性的描述，这是一个开放的逻辑世界，

自然语言一般就是进行那几方面的描述，从语句中分离出来那些成分，

由于语言本身中的一些词汇是由复杂的结构网络构成，所以无法对其进行理解，

应该怎样开始？

世界本身不包含任何的节点，函数以及路径。

需要做的就是向其中添加东西。

就像婴儿一样，应该从最最基本的进行学习，学习他们的关系，然后慢慢拓宽这个链接的网络，

属性分为可直接更改的属性，也就是函数直接连接的属性，或者说是 属性不能够

取决于我对于属性的定义，属性可以定义为 构成节点的子事物的描述，

我现在最需要解决的问题是：关系图的描述以及实例图的描述问题，

关系图 节点的集合，主要是函数的描述，函数所针对的节点，函数，我写不出来了。

在java中应该怎样体现，将类分为两类，一种是关系类，一种是实例类，关系类要能够提供一个函数返回一个实例类，类一定是实例化其中的函数才可以调用，所以类应该被分为两种，前者返回后者的实例。

现在最难的问题是关系类的存储问题，动态生成问题，世界里面类的关系问题，程序的动态执行问题，应该教给他去学习语言。

事物不再是 属性和函数的集合，事物代表的是一个系统，一个事物是若干事物和规则的集合体，属性是一个很宽泛的概念，（是对构成事物的子事物的某些特征的描述），函数是子事物活动的描述，这样来讲的话，和路径其实差不多是一个概念，

这样的话，事物包含的内容:子事物集合，必要的子事物逻辑集合，从子事物中抽象出来的属性集合，子事物活动所代表的函数集合。子事物活动抽象出来的函数集合。

初始化一个事物不必要填充他的子事物集合和子事物逻辑集合，可以直接填充他的属性集合和函数集合，没有前面的两者，那么对于这样事物的认识称为表象认识，包含了前面的两者称为深刻认识，（暂时可以这样去想），

这样的话，一个事物包含4或者5个域，

在我的模型里面暂时还无法表示，活动，函数，子事物的逻辑集合。我也想不到应该怎

样在程序中进行体现，

活动是对象函数的执行序列，不管他分叉不分叉，

我仍然将函数设定为对目的事物属性的更改吗？，那样的话，我必须解释属性，

是的，还那个样子进行定义，属性通过某种方式可以进行更改。是否所有的属性通过某种方式都可以被更改，好像是的，好的还是那个样子进行定义，

活动就是各种事物函数的按时间进行执行的序列，

函数的表示方法是 本事物， 执行的函数，受影响的目的事物以及属性。

不应该将函数解释为特定的程序语言。

可是这样的话，还有缺陷，条件是什么，例如在我前方5米处有一个石头，如果我不走到石头的旁边，我就不能把石头捡起来，为了捡起石头，我必须走到石头的旁边，

捡起石头是我的目的，我要对捡起石头这句话进行解释， 就是我要将石头放在我的手里，涉及的是 石头和我空间位置相对位置关系的改变，为了达到这个目的我应该怎么做，

还有一个问题，石头它有空间的位置属性，位置并不是石头本身的组成成分，位置的含义是 石头和其他事物的相对位置关系，也就是说和其他事物的相对位置关系也构成了属性的一部分，这样能定义为属性吗？不能，位置只能定义为关系，不能定义为属性，

如何表达关系呢？

如果我想表达关系，我必须明白什么是关系？什么是关系？关系就是事物之间的相互联系，这样的话，位置是什么？我处在一个世界之中，我有一个位置，那么这个位置是什么东西呢？

这个理论中无法表达条件，是吗？

首先，我和石头的关系是，位置相聚较远，我有搬石头这个函数，但是这个函数执行有一个条件，相距较近，所以说，我必须达到这个条件，条件并不是什么新的东西，条件是一种状态，可以使用函数的状态，函数的使用所具有的前提，条件就是状态的含义。

但是石头不会告诉人说，你要改变我的位置，你必须离我很近，我的手可以改变小石头的位置属性是由于物质本身之间的相互决定的。为了捡起石头，必须在现在的状态和函数可以执行的状态之间构建通路，就是这样的。

虽然构建通路的算法暂时还没有想出来，但是客观上他是存在的，等我更加理清楚这之间的关系的时候，我也可以想出来。

我还是没有讲空间位置究竟是什么东西，

**自然语言理解，**

我想在要建立一个自然语言理解系统，可以达到和人正常的交互，

缺乏自己的认识，靠别人说，可以得到真正的知识吗？

需要几个步骤：

* 完善自己的系统
* 手动构建初级的事物逻辑关系，
* 构建初级的为了获取知识所必须的所有初级准备，
* 自然语言理解，
* 扩大的自然语言理解
* 构建外界事物之间的逻辑关系。
* 完成系统，

自然语言理解的目的是建立事物之间的联系网络。

完善自己的系统第二次

重新规划设计，

世界

关系事物

一个数据库，记载了事物以及事物之间的关系

实例事物：

空位，表达子事物，以及具备的某些关系，

属性，(经过子事物构建，也可以不构建，那么就是浅度理解)

函数，对于目的事物，目的属性的更改，以及子事物的活动序列（反映本质），

还有什么？暂时没有了，

应该还有规则，应用规则可以建立路径，是吗？有待思考

完成代码设计

主动函数，反映上帝，哈哈，为了某些特殊需求而设立，

状态转换之间的路径构建，我需要对事物有一个更加清晰的认识，定义和了解。

事物的变化之处在于那里。事物的属性以及函数的改变。由于事物属性改变而导致函数改变的。

对事物做出限定，所有的属性和函数已知。已知属性和函数 ，未知属性和函数。

对子事物了解的程度。

先做一个简单的，所有的属性和函数已知，在此情况下进行路径构建工作。

如何进行路径构建工作。

建立框架。

关系世界

实例世界

建立一个世界，

添加事物，需要一个name属性，

向世界获取某种事物的实例，需要一个id属性。

如何构建路径。依据事物之间的关系，然后根据实例事物来构建一个图形。

所有的属性都是可以被更改的。属性的更改，必须沿着路劲走才可以。

目标状态有一个事物的一个属性需要被达到。可以走的路径有很多。当有多个目标属性需要被达到的时候，取所有路径的交集就可以了。

如何将微积分的有关内容植入到这个系统中呢，

二维向量，三维向量。

在可能有事物数量变化的情况下，怎么到达目标呢？

world的状态对应的是各个事物的属性集合，

怎么到达目标状态呢？那是个什么东西啊？

事物就是一个个孤立的小球， 彼此之间的联系也就是各种函数就是就是桥梁，首先要触及到各个目标属性，

每个事物都包含有限的桥梁，事物的本质在某些情况下还有可能会改变，这怎么设计呢？

现象，本质，研究一个事物的属性改变的事物，对其他事物的他们之间是否拥有桥梁。

这个怎么做呢？

每个事物包含有限的桥梁，桥梁可通可闭，事物的属性变化范围非常巨大，

设计到桥梁的开通，属性的变化，属性的变化是桥梁开通的关键，桥梁的开通是属性变化的前提。

每个桥梁的开通可能会涉及到一系列的事物的内部关系的维持，这是一个集合，这个集合可大可小，

目标就是多个事物的属性的共同集合，各个集合之间的关系就是能否到达目标的，集合之间的关系如何对应 函数的调用准则呢？所包含的事物是否存在交集，如果存在交集，那么就不可能同时开启 两个桥梁了，如果没有交集，就可以同时开启， 假设存在一个桥梁需要同时上面两个桥梁的存在，那么这个桥梁就不可以打开了。

事物的每个函数对应一个启发条件，启发条件是对世界状态的一种描述（），这就是一种状态，也可以设置为一种目标，这就是为了达到终极目标中间不得不实现的子目标。

如何去设定这些子目标，主动去设定吗？

设定子目标的过程涉及到学习的过程，暂时不讨论去如何学习

如何由事物的集合关系（桥梁A 是桥梁B 启发条件的子集，那么A必须在B之前完成，集合之间的成员和关系决定了函数的调用顺序）决定函数的调用序列，前驱关系，差不多想到了，但是我想的还不足以完全准确的描述。

学习的过程是怎样的？因为函数的目标是改变物体的属性，前提条件又是属性的改变，尝试去测定函数改变和属性之间变化的关系就是学习的过程。

数学才是描述世界存在的正确的语言，

我需要有逻辑转换程序，

函数的判定执行函数需要更改为程序可以理解的外物的状态，

（现在想一想，语言无非可以分为两类：第一类：对事物属性的描述，第二类：对事物函数的描述，对吗？不对，）

\\\\\\\\\\\\\\\\\\\胡思乱想分解线开始\\\\\\\\\\\\\\\\\\\\\\\\\\\\

我的模型存在问题，存在的问题如下，描述的内容有限，描述的是事物和事物之间的联系，好像缺少点什么？并不能描述一切，描述清楚什么并不重要？是吗重要的是什么？

\\\\\\\\\\\\\\\\\\\胡思乱想分界线结束\\\\\\\\\\\\\\\\\\\\\\\\\\\\

好多东西的逻辑是等价的，

世界中存在的所有东西都可以使用数学进行描述，学习的过程，建立节点，连线，更改已有连线的过程，为什么要更改已有的连线，

模型事物，实例事物，目标状态的路径（算法就是不断尝试，尝试各种不同的情况），

可以进行学习的特点是，改变行为时，可以获取状态，笨的方法就是记录下行为改变和状态的关系，死记住，聪明的方法就是对数据建立模型，

被动函数的感知能力必须局限在本事物的属性集合之内

对属性1进行改变，属性1的变化激起被动函数启动，被动函数的变化导致一系列其他连带属性的变化， 属性1和其他连带属性之间存在的关系(属性1的变化程度和连带属性的变化程度的关系)

在我的脑海里面，现在就是一些列发毛的点点，事物本身就是点，而函数就是外面的一根根毛，我现在要将这些毛链接起来，

要根据自己设定的目标来对这些毛进行链接，

多个事物链接在一起的时候，多个事物也可以看成一个单一的事物进行处理，单一的事物也可以分解为多种事物。

对毛进行连接就要依赖于之前建立的属性和其连带属性之间建立的关系图纸。

需要反馈，怎么进行反馈？自己进行反馈或者功能性反馈。

学习。区分事物的能力。

被动函数体的设定存在一些问题，因为我不能直接去调用那个事物的被动函数，我之前的设定是更改属性才会去调用链接的函数，要更改这个设定，也可以调用函数，调用函数的时候，也可以设定链接的函数，要仔细设定一下。属性和函数都可以设定是否启动函数

要对函数进行函数链接

少了很多的东西，该死，

我对被动函数的设定，被动函数的设定存在问题。函数不能传入参数，

主动函数呢，

属性是否需要设定类似神经网络的东西来进行预测判断。

我必须假设存在这样的一个东西。

我要开始进行路径构建了。

怎么进行路径构建呢，集合运算。首先是提出要求。

逐级反推到启动点上，启动点范围的扩大试验，

反推的过程其实就是启动点的变大过程。

主动函数什么时候开始执行啊，一个事物要给另外一个事物传递一些参数怎么办？

其实内部已经 内置参数传递方式了，被动函数都有目的属性和源属性，

事物都有唯一性标识。

这个不应该是我现在需要探讨的，我需要探讨的是路径构建工作。这个才是我的模型的最大特点和有点，其他一切程序所没有的特点。

梳理一下逻辑，小明学习，需要课本，并且灯光为亮，目的是为了改变成绩。

小明学习，如果课本为亮，则小明的成绩提升，若课本的亮度为暗，则小明的成绩不变。

灯光可以改变书的亮度由暗变亮。

需要怎么写呢？学习可以提高成绩，学习改变的目标属性是自己的成绩，在学习过程中，

因为需要书本的亮度为亮，所以在学习的时候需要达成子目标（书本的亮度为亮）。此刻就需要寻找可以改变书本亮度的事物。

寻找被动函数中目的属性包括书本亮度的事物。

这个时候就需要映射体，每个事物都应该有一个影响产生的映射表，每个事物都应该有一个，此表由系统维护，事物自身可以提供一份作为系统参考(需要建立模拟环境，系统可以对事物本身进行测试。

压缩，事物简化，链接所有。（这个系统都可以拥有这些特性，这才是操作系统应该有的样子）

我现在需要什么？子目标达成函数，小明在学习的时候，必须先达成目标，我应该怎么编写这个代码呢？（我的系统最重要的特点来了），设定目标，以及去达成目标。

所以说事物需要介绍自己，（介绍自己的方法就是，自己可以产生的影响集合）

某个事物在执行函数的时候，首先会将目标达成函数交给自己进行处理（请求别人帮忙，必须将目标给其他人，这个过程中也必须遵守之前定的各种规则），

读书需要一个源属性，被动函数是如何处理源属性的

只监听 自己的属性变化是不是正确的，源属性存在的意义是什么？源属性是函数启动的根本，(如果函数可以监听自身以外的属性的变化，那么事物存在将变的没有意义)

依然是函数必须监听自身的属性变化，可以怎么向事物传递参数呢？（添加额外的属性又违背了）

现在规定可以监听事物自身属性之外的属性，

那么源属性的改变才会启动激发被动函数进行检验，进而改变目的属性（这样的话，那岂不是函数必须拥有源属性，这样的是正确的吗？）

暂时就这样吧，

被动函数这样的话，那岂不是一定不可能首先启动（源属性不发生变化的话，被动函数是没有执行的机会的，这个时候就需要主动函数了），我就需要重新定义主动函数了。

好的，就这样，事物存在不稳定状态，check是一定要执行的。那么源属性改动然后启动check的不是就变得没有意义了吗!

而且不得check的顺序会导致整个世界朝完全不同的方向发展，这个是不允许的。

我的整个模型还存在一些问题：

问题一：源属性改变启动被动函数变得没有意义。

问题二：启动函数存在问题，改成设想的那样会导致世界一开始就必须是稳定状态，但是这样就存在问题，

必须由主动函数去启动世界的运行。

分割线

所有事物都需要实现一个函数，用于获取映射表，这个就是介绍自己的方式，也不是必须实现，

映射表包含什么内容。

自己的标识。映射表主要是介绍自身属性改变和被动函数启动之间的关系，

被动函数怎么介绍呢？ 默认的事物图形构建界面，原生的支持图形和命令行式

源属性和目的属性之间复杂的关系图。这个时候可以采用神经网络，简单或者是复杂的。

原来事物之间真的是 这种线状结构，事物自己可以提供一个。系统也可以自己构建。

对于函数的描述要通过事物的属性，因为函数有源属性和目的属性。所以对于属性的描述必须像这个样子，

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 源属性1 | 源属性2 | 源属性3 | 源属性4 | 目的属性 |
| Value1 | Value2 | Value3 | Value4 | Destination X |
|  |  |  |  |  |

遗漏一项重要内容，关于被动函数的启动问题。怎样描述被动函数的启动问题呢，

有怎样去描述被动函数的启动。被动函数如何启动，之前的设定是检测自己属性的变化，然后，被动函数自己启动，不需要其他事物的介入。

认为或者自动 尝试进行规律化或者压缩这个表格，

我需要对这个过程进行处理。设定一些类。

首先需要可以表示上面的表格。

这样就可以表示了，在这个基础上就可以

对函数的描述是什么意思？ 可以允许对事物本身进行测试，

意识到一个问题，规律的静态性，在世界中，一般情况下，相同的环境中一般会出现相同的结构。人类就是这样认知规律的，规律如果不稳定的话就会很难进行认知。

一种可以描述做什么的语言。

建立那个表格的目的是什么？

我怎么对目的进行解析，

被动函数的启动问题。坚持一点，被动函数一定只能监听自身的属性改变。只有事物被影响到了，才可以启动相应的被动函数，

遵循客观规律设计，由此产生的程序效率问题，由提升效率的专门的程序解决。

小明需要开灯，但是，小明是不能监听灯的亮度变化，小明自己也需要添加明暗属性，

路径构造函数应该在什么层面上面执行啊， 路径构造暂时定在 最高层面上进行，

这个并没有背离客观规律，所以说可以。只有最高层才能纵览全局，调度所有的事物。

这样好像是不对的呢，

分割线

做一个简单的小小的总结：

首先我的思想是正确的，我的模型暂时还不能完全表达我的思想，现有的编程设计程序是我的一个子集，现有的编程设计缺少程序本身对于程序自身的认知，程序的运行完全依赖于底层的字节码文件。

今天我觉得我应该想到如何将现有的程序模型使用我的框架进行表达。

有产生了一个新的问题，即使我知道怎么进行路径构造，状态装换，但是我却不知道为什么要那么做，为什么要那么做呢， 我的程序好像缺少某些东西，缺少类似于哲学的一些东西。也不是，一些在更高层面上表现出来的规律性。

我知道了，我的程序缺少对于知识本身的总结和认知，对于事物本身接近与本质的表达能力和认知能力。我的程序虽然可以进行状态转换，但是缺少那些，所以

将现有模型归于自己的模型的时候，产生了一个问题，各种属性和函数之间存在不可沟通的问题，缺少必要的沟通机制，属性和函数之间不是分离的，我的模型存在问题，事物本身才具有状态的函数，本非所有类都是事物。

构想分割线，11/12/2017 2:08:34 PM

间隔很久之后，重新构想我的计划，这次的目标是克服之前的evolution存在的诸多问题：

* 不能动态表示事物以及子事物之间的关系，
* 除了事物之外，作为掌管事物运行的“自己”的功能太弱，没有对算法进行抽象，算法是真是存在的比较复杂的部分，我的本身的算法是不可能构建出其他的比较复杂的算法的，“自己”仅仅是一个路劲构建函数而已（正确吗），
* 自己的基础的问题，自己必须具有从符合逻辑的事物上构建出自己的能力，
* 逻辑的运算，可以对逻辑进行表达，运算。对执行路径进行求解。

目标：目标是构建出这么运行体，夸语言，系统的逻辑执行体，

目标系统应该包含这几部分，

自己：对事物进行管理，包括创建和销毁，事物之间的关系，

逻辑构建体：构建从现在到目标的路径，

自己：脑子：逻辑构建体

手：指导事物进行运作

眼睛：事物本身所带有的侦测函数，再加上对于自己的认知。

程序中我自己需要表达的部分，由于认知方式和人类一致，所以也具有不可认知本质的特性。所以程序本身需要一种表述事物之间关系的语言，将认知的所有事物转换为那种语言的表述。

还需要的一点是，自己运行所需要的逻辑基础部件。