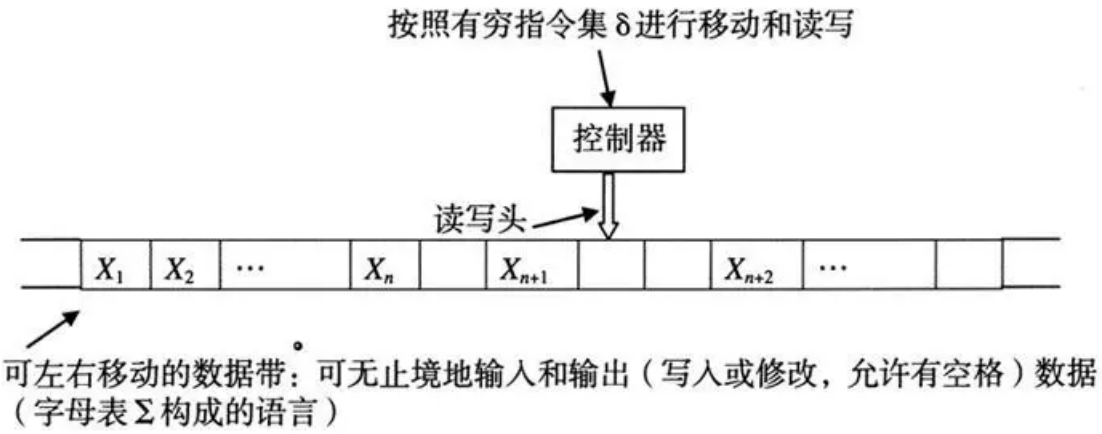
图灵机与停机问题

原创 张忠诚 LOA算法学习笔记 2021-02-07 14:57

01 什么是图灵机?

图灵机是串行计算的数学模型，他的结构非常简单，主要包含以下几部分：

1. 一个无限长的纸带，可以理解为存储器，纸带被划分为一个个单元，每个单元可以存储一个字符。
2. 一个读写头，可以理解为指针，它能够纸带上左右移动，每次可以移动一个单元，每次移动可以读写当前所在单元的符号。
3. 一个带字母表，表示纸带上可能出现的所有字符，其中包含一个特殊的空白字符。
4. 一个输入字母表，它是除去空白字符的带字母表。
5. 一个状态集合，可以理解为状态寄存器，记录着机器所处的状态，其中包含一个开始状态和一个终止状态。
6. 一个转移函数集合，可以理解为指令集，它记录着读写头在特定状态下的行为，给出了图灵机的操作规则。它的输入是当前图灵机的状态以及即将读入的带符号，它的返回结果是一个图灵机状态（可以和当前状态相同）、一个替换读入带符号的新符号(可以与读入带符号相同)，以及一个迁移符号L或者R，表示读写头向左移还是向右移。



上述图灵机的结构如果大家细心观察，很容易发现他跟电子计算机是不等价的，因为上述图灵机的转移函数的集合一但确定，它就只能完成特定的计算任务。而我们都知道电子计算机是可编程的，针对这个问题，我们可以通过构造一个通用图灵机来解决，通用图灵机以某一图灵机M的编码和输入串w作为输入，模拟M在输入串w的运行，所谓的可编程就是针对某一图灵机进行编码的过程。至于如何构造一个通用图灵机，大家如果感兴趣可以参考自动机的相关教材，本文的重点不在于此。

02 停机问题及其证明

什么是停机问题？一言以蔽之，停机问题是说计算机是否可以判断一个程序是死循环还是正常结束，对应到图灵机中就是，是否可以利用图灵机判断一个图灵机可以在有限时间停机。

那么怎么证明停机问题呢？我们可以利用反证法,假设存在一个足够牛逼的程序它可以判断一个程序P在输入情况为I的情况下是否停机，由于这么牛逼程序一定是出自上帝之手，所以它的定义如下所示：

```
int God(P, I) {
If (P停机)
return 0;
else
return 1;
}
```

由于程序本身也是一种数据，比如编译器的自举，所以程序P也可以作为输入，即上帝程序应该可以判定将P作为P的输入时，P是否会停机，函数原型为int God(P,P),接下来就是弑神的过程，我们从上帝程序出发构造这么一个程序NoGod(P),该程序的定义如下所示：

```
int NoGod(P) {
if (God(P,P) == 1)
return 0;
else
while(1) {}
}
```

这个程序是说，如果上帝程序判断结果为不停机，NoGod程序就停机，否则就进入死循环，接下来让我们把NoGod程序作为他自身的输入，会得到什么结果呢？以下是替换后的程序定义：

```
int NoGod(NoGod) {
if (God(NoGod, NoGod) == 1)
return 0;
else
while(1) {}
}
```

分析这段程序，奇怪的事情出现了，当上帝程序判断NoGod程序不停机的时候，NoGod程序却返回0停机了，当上帝程序判断NoGod程序停机的时候，它却进入了死循环。所以上帝程序根本就不存在，即停机问题是不可判定的。

03 证明方法怎么想出来的?

卜老师上课经常问的一个问题，算法是怎么想出来的呢？那么我们也来探究一下这个绕来绕去的证明过程是怎么构造出来的？到底是妙手偶得还是有迹可循。

几千年来，人类科学不断发展，但在面对无穷时总还是有点束手无策。比如针对无穷集合的“大小”问题，人们始终不知道应该采用什么方式表示无穷集合的“大小”。直到19世纪70年代康托尔引入了一一对应的概念来表示无穷集合的“大小”，比如自然数集合和偶数集合“大小”是相同的，因为他们总可以采用f(n) = 2n这个函数来一一映射，虽然偶数集合从直观上是自然数的真子集。此外有理数集和自然数集也是一一对应的，那么实数集合和自然数集合是不是一一对应的呢？也是康托尔，在1874年天才般的创造了对角线方法从而证明了它。并且在之后对角线的思想被广泛的应用，比如本文的主角停机问题就可以利用对角线的思想给出证明，你会发现从对角线方法出发构造那个证明过程是多么的自然。为了便于理解，下面我将从实数集合与自然集合是否一一对应的证明出发阐述对角线方法的思想，然后阐述如何从对角线方法出发证明停机问题。

为了证明方便，我们取(0,1)之间的实数集合，如果这个小区间内的实数集合都无法和自然数集合一一对应，那么扩展到整个实数集合的结果是显而易见的，采用反证法，假设（0，1）区间内的实数集合跟自然数集合可以一一对应，由于自然数集合是可列的，那么在该假设下实数集合也是可列的，将自然数集合和实数集合一一对应列出可得：

1 $0.A_{11}A_{12}A_{13}A_{14}...$

2 $0.A_{21}A_{22}A_{23}A_{24}...$

3 $0.A_{31}A_{32}A_{33}A_{34}...$

4 $0.A_{41}A_{42}A_{43}A_{44}...$

...

我们重点关注实数集合小数点之后的部分，我们取其所有的对角线元素，并且选择一个与其不等的数从而构成一个新实数的小数部分，假设我们用B_{11}表示这个数与A_{11}不同，那么我们构成的新实数即为$0.B_{11}B_{22}B_{33}B_{44}...$，显然这个新的实数与上面我们列出的所有实数都不同，因为至少有一个小数位不相等，所以我们无法列出所有的实数与自然数集合一一对应，与假设矛盾，故实数集合和自然数集合不是一一对应的，但是有理数集确是可以一一对应的，大家可以思考一下为什么采用这种方法不能证明有理数集合不是一一对应的？

上面是对角线方法的基本思想，那么怎么使用对角线方法想出那个诡异的程序，从而证明停机问题呢？首先所有图灵机编码构成的集合是个可列集合，即我们可以枚举出所有的图灵机（这是一个定理，具体证明可以参考自动机相关教材），所以我们可以利用对角线的思想构造出如下所示的矩阵，用来表示图灵机在特定输入下的输出，其中Mi表示图灵机，表示其对应的编码，S表示停机，N表示不停机。至于图灵机的编码为什么可以作为输入可以参考第一节通用图灵机的定义。

	<M1>	<M2>	<M3>	<M4>	<M5>	...
M1	S	N	N	S	S	...
M2	S	N	S	N	S	...
M3	N	N	S	S	S	...
M4	N	N	N	N	N	...
M5	S	S	S	S	S	...
...						

同样的，运用对角线方法，我们可以构造出一个行为与上述图灵机均不相同的图灵机M，即该图灵机在输入上的输出与Mi不同，该图灵机的行为可以用如下所示的代码表示：

```
If (God(i,i) == 1) {//如果Mi在输入上不停机
return S;
} else {//如果Mi在输入上停机
return N;
}
```

上面的程序大家应该会感到非常熟悉，他其实跟我们的NoGod程序表达的语义是相同的，根据上一节我们可以知道如果我们把其自身作为输入就会推出矛盾，就可以证明停机问题了。那么为什么要把自身作为输入呢？我们也能从对

角线方法中得到启发，首先如果只有上述矩阵所列出的情况我们是能判断是否停机的，查表就可以了嘛，但我们偏偏构造了一个与上述图灵机均不相同的异类出来，他到底能不能被判断呢，我们当然要试一下。

04 总结

当初选课的时候看卜老师设置了图灵机以及NP问题相关的课程大纲，加上自己对这方面比较感兴趣就写了这篇文章，文章主要是先简单介绍了图灵机的构造，但为了便于理解我并没有给出形式化的定义，图灵机形式化的定义是一个七元组，如果感兴趣大家可以参考自动机的相关教材，随后介绍了停机问题的证明，以及如何从对角线方法出发证明停机问题，文章重点在第三部分，对角线方法的用途非常广泛，罗素悖论以及著名的哥德尔不完全性定理都是建立在对角线方法之上的，所以大家可以抽时间好好理解一下。最后，由于我对图灵机的理解也不是多么深刻，如果证明过程存在错误，欢迎大家批评指正。