第十章
图灵机的其他模型

姚　刚

中国科学院信息工程研究所

目录

图灵机的定义

我们给出了标准图灵机$M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$的定义，这并不是唯一可行的，还存在着若干等价的定义。我们所能得出的关于图灵机能力的结论在很大程度上独立于为之选定的特定结构。我们将介绍几个更加复杂的图灵机，在某种意义上讲，它们与标准图灵机的能力是等价的。

较小的改动

先考虑对标准图灵机做较小的改动，并探讨这些改动是否导致了一般概念上的差异。每次改动后，我们要解决一些问题：新的自动机和改动前的自动机是否有实质上的区别？一类自动机与另一类子动机的“实质差别”是什么？有时虽然定义上有明显的差异，但是这些差异不一定能导致令我们感兴趣的结果。我们需要在一般意义上定义自动机类的等价性和不等价性。

自动机类的等价性

定义 (自动机类的等价性)

两个自动机是等价的，如果他们接受同样的语言。考虑两个自动机类(*classes of automata*)C_1和C_2，如果对于C_1中的任意自动机M_1，存在C_2中的自动机M_2，满足$L(M_1) = L(M_2)$，我们就称C_2具有至少和C_1相同的能力。如果对于C_2中的任意自动机M_2，也都存在C_1中的自动机M_1，满足$L(M_1) = L(M_2)$，我们就称C_1和C_2是等价的。

模拟(simulation)

设M是一个自动机，我们称自动机$\hat{M}$可以模拟M的一个计算，如果$\hat{M}$可以按照如下的方式模拟M的计算。设d_0, $d_1, \cdots$是M计算的瞬时描述序列，即
$d_0 \vdash_M d_1 \vdash_M \cdots \vdash_M d_n \vdash_M \cdots$,
则称$\hat{M}$模拟这个计算，如果$\hat{M}$可以执行和M类似的如下计算

模拟(续)

$\hat{d}_0 \overset{*}{\vdash}_{\hat{M}} \hat{d}_1 \overset{*}{\vdash}_{\hat{M}} \cdots \overset{*}{\vdash}_{\hat{M}} \hat{d}_n \overset{*}{\vdash}_{\hat{M}} \cdots$,
其中$\hat{d}_0, \hat{d}_1, \cdots$是瞬时描述，并且每个瞬时描述都对应于$M$的唯一一个格局。换句话说，如果我们知道$\hat{M}$执行的计算以及初始格局，我们就能准确地确定$M$所执行的计算。

带不动选择的图灵机

在标准图灵机的定义中，读写头要么向左移动，要么向右移动，而有时为了方便，我们引入第三种选择，即读写头重写带上单元后位置保持不动。于是，我们修改标准图灵机的定义，得到新的带不动选择的图灵机(Turing machine with stay-option)的定义。

带不动选择的图灵机

将标准图灵机的定义中的转移函数替换为$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$。这里，$S$表示读写头的位置保持不动。

定理

带不动选择的图灵机类与标准图灵机类等价。

多道图灵机

在标准图灵机中，每个带上的符号可以是一些符号的组合体，而不仅仅是单个符号。如果我们将标准图灵机中带上的每个单元划分为多个部分，每个部分都称为道(track)，每个道包含多元组的一个成员。基于这样的直观认识，这样的图灵机有时被称为多道图灵机(Turing machine with multiple track)。

这样的改变并没有扩展标准图灵机的能力。

单向无穷带图灵机

如果将标准图灵机的定义中的带改为单向无穷的，即带存在一个左边界，这样的图灵机称为单向无穷带图灵机(Turing machine with semi-infinite tape)。这一模型与标准图灵机的区别在于：当读写头位于带的左边界时，不能向左移动。

定理

单向无穷带图灵机类与标准图灵机类等价。

离线图灵机

在自动机的一般定义中包含一个输入文件和一个临时存储空间。

如果在标准图灵机中加入输入文件，我们所得到的就是离线图灵机(off-line Turing machine)。在离线图灵机中，每一步转换都是由内部状态、从输入文件当前读到的符号以及读写头所见到的带上的符号控制的。

定理

离线图灵机类与标准图灵机类等价。

具有复杂存储结构的图灵机

在标准图灵机$M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$的定义中，存储装置是如此简单，以至于人们可能以为通过将存储装置复杂化有可能增强图灵机的能力。

事实上，将图灵机的存储装置复杂化并没有增强图灵机的能力。

多带图灵机

多带图灵机(multitape Turing machine)是一个有多条带的图灵机，每一条带都有一个被独立控制的读写头。多带图灵机的转移函数不同于标准图灵机，一般我们如下定义$n-$带图灵机：$M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$，其中$Q$，$\Sigma$，$\Gamma$，$q_0$，定义与标准图灵机相同，不同的是转移函数定义了发生在所有带上的转移：$\delta : Q \times \Gamma^n \to Q \times \Gamma^n \times \{L, R\}^n$。

定理

考察转移函数

$$\delta(q_0, a, e) = (q_1, x, y, L, R)$$

执行之前和执行之后的情况。

定理

多带图灵机类与标准图灵机类是等价的。

考虑语言$\{a^n b^n\}$，试给出识别该语言的多带图灵机。

多维图灵机

多维图灵机(multidimensional Turing machine)是一种其带在多个维度上都可以无限扩展的图灵机。在二维图灵机的形式化定义中，转移函数具有如下形式：

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, U, D\}$,

其中，U和D分别表示读写头向上移动和向下移动。

定理

多维图灵机类与标准图灵机类是等价的。

引入非确定性

虽然根据图灵论题似乎可以得出特定的带结构并不会影响图灵机能力的结论，但是这一结论对于非确定型图灵机来说则是不成立的。因为非确定型引入了选择，从而具有某种非机械化的特性，此时不能再用图灵论题了。

为了证明非确定型图灵机与标准图灵机的能力相同，需要更仔细地考察非确定性带来的影响。用模拟技术证明非确定性行为可以通过确定的方式加以处理。

非确定型图灵机

定义 (非确定型图灵机)

非确定型图灵机类似于标准定义中的图灵机，只是转移函数δ变为

$$\delta : Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L,R\}}。$$

只要含有非确定性，δ的值域就是一个由可能发生的转移构成的集合。对每一步转移，图灵机都可以从集合中任意选择一个。

符号串的接受器

因为非确定性在计算函数中的作用并不明显，所以我们通常将非确定型图灵机看作符号串的接受器。

非确定型图灵机接受符号串w，如果存在某个可行的迁移序列满足

$$q_0 w \overset{*}{\vdash} x_1 q_f x_2,$$

其中$q_f \in F$。

例子

如果一个图灵机有如下转移函数：

$$\delta(q_0, a) = \{(q_1, b, R), (q_2, c, L)\},$$

那么这个图灵机就是非确定型的。

定理

非确定型图灵机类与确定型图灵机类是等价的。

模拟非确定型图灵机

非确定性可以被看作确定型的回溯算法。确定型图灵机只要能记录回溯过程中的状态，就可以模拟非确定型图灵机。可以换一种观点看非确定性：一个非确定型图灵机可以被看作是一个可以在任何必要的时刻都能复制自身的图灵机。当有多种选择时，图灵机会复制多个自身，然后每个图灵机去执行一种可能的迁移。

与图灵论题相悖的论断

“一个标准图灵机是一台服务于特殊目的的计算机，一旦δ确定了，这台机器就被限制于执行某种特定的计算。而数字计算机却是通用的，人们可以对它编程，使其在不同的时间作不同的工作，因此，图灵机与数字计算机是不等价的。”

我们通过设计一台可重编程的图灵机(reprogrammable Turing machine)来解决这个问题，我们把这类图灵机称为通用图灵机(universal Turing machine)。

建造通用图灵机

通用图灵机M_u以任意一台图灵机M的描述和符号串w作为输入，并可以模拟M在w上的计算。

为了建造这样的M_u，先选择一种图灵机的标准描述方式。我们不失一般性地假定$Q = \{q_1, q_2, \cdots, q_n\}$，其中$q_1$是初态，$q_2$是唯一的终态。假设$\Gamma = \{a_1, a_2, \cdots, a_m\}$，其中$a_1$表示空白符。

然后我们选择一种编码方式，使得q_1被编码为1，q_2被编码为11，以此类推。

建造通用图灵机(续)

类似地，a_1被编码为1，a_2被编码为11，以此类推。符号0作为1串之间的分隔符。

有了如上对初态、终态以及空格符的定义，我们就可以用δ完全描述任何一个图灵机了。于是任何一个图灵机都可以被编码为一个有限的$\{0,1\}^+$上的符号串，并且对任给的一个图灵机M的编码，我们都可以按唯一的方式解码。

建造通用图灵机(续)

图灵机的其他模型

姚刚

目录

对图灵机的较小修改

具有更复杂存储的图灵机

非确定型图灵机

通用图灵机

线性有界自动机

一个通用图灵机M_u包括一个$\{0,1\}$上的字母表和一个多带图灵机结构。

对于任意给定的输入M和w，1带用于记录M定义的编码，2带记录M的带内容，3带记录M的内部状态。M_u首先查看2带和3带的内容以决定M的当前格局，然后查看1带以决定M 在此格局下的动作，最后修改2带和3带以反映此次迁移的结果。

通用计算机的模型

有理由建造一台实际的通用图灵机，但这一建造过程并无乐趣。相比之下我们更喜欢使用图灵假设。我们可以用某种程序设计语言实现这一点，给出一台通用图灵机在高级语言上的实现。因此，我们完全可以期望用一台标准图灵机完成这样的工作。于是，我们断言下述图灵机存在：对于任意给定的一个程序，它都可以完成此程序规定的计算。这种图灵机就是通用计算机的模型。

可数集和不可数集

有些集合是有限的，有些集合是无限的。对于无限集合，我们将其划分为可数集(countable) 和不可数(uncountable)集。如果一个集合的元素可以一一映射到正整数集，就称这个集合可数。这也就是说我们可以按照某种顺序写出这个集合的元素。如果我们能按照某种方法顺序地写出集合中的元素，就证明该集合是可数的。我们将这类方法称为枚举过程(enumeration procedure)。

枚举过程

定义 (枚举过程)

令S为字母表Σ上的符号串的集合。则S的枚举过程就是一个执行下列计算步骤的图灵机：

$$q_0\square \overset{*}{\vdash} q_s x_1\# s_1 \overset{*}{\vdash} q_s x_2\# s_2 \overset{*}{\vdash} \cdots,$$

其中，$x_i \in \Gamma - \{\#\}$，$s_i \in S$。用此方式，S中的每一个符号串s都会在有限步内被产生。状态q_s用于表示S的成员状态。即，每当机器进入状态q_s时，$\#$后面的符号串一定是S中的成员。

定理

令$\Sigma = \{a, b, c\}$。找到某种顺序枚举出集合$S = \Sigma^+$中的所有元素，从而证明集合S是可数的。

定理

所有图灵机构成的无穷集合是可数的。

受限制的图灵机

虽然我们不能通过使图灵机的带结构复杂化而使其功能变强，但是我们可以通过限制对带的使用而限制图灵机的能力。比如，下推自动机可以看作是有一条带的非确定型图灵机，而且这条带必须以栈的方式使用。有穷自动机可以看成一个离线自动机，输入只能从左到右地读一次，并且不能重写，带上至多可以使用有限个多余的单元作为工作空间，空间长度对所有输入是固定的。

一种感兴趣的限制

如果我们要求图灵机只能工作于带的输入部分，那么，较长的输入符号串就意味着较多的工作空间。这样的限制定义了一种自动机：线性有界自动机(linear bounded automata, LBA)。

线性有界自动机也有一条无限长的带，但带上能够使用的部分的长度是输入部分的函数。

左端标记和右端标记

为了限制使用部分的长度，我们将输入部分包含在两个特殊的符号之间：左端标记(left-end marker)([)和右端标记(right-end marker)(])。

对于一个输入w，图灵机的初始格局由瞬时描述$q_0[w]$给出。两个端点标记([和])所在的单元不能被重写，读写头也不能移动到[的左边或]的右边。

线性有界自动机

定义 (线性有界自动机)

一个线性有界自动机是一个非确定型图灵机$M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$，具有如非确定型图灵机的定义中所述的限制，并且字母表Σ必须包含两个特殊的符号“$[$”和“$]$”，并且$\delta(q_i, [)$只能包含形如$(q_j, [, R)$的元素，$\delta(q_i,])$只能包含形如$(q_j,], L)$的元素。

线性有界自动机接受的语言

定义 (线性有界自动机接受的语言)

符号串w被一个线性有界自动机接受，如果对于某个$q_f \in F$，$x_1, x_2 \in \Gamma^*$，存在一个可行的迁移序列

$$q_0[w] \overset{*}{\vdash} [x_1 q_f x_2]。$$

被线性有界自动机接受的语言就是所有被接受的符号串构成的集合。

例子

注

我们定义的线性有界自动机是非确定型的，这不仅是出于方便的考虑，而且本质上对于讨论线性有界自动机是必需的。

语言$L = \{a^n b^n c^n : n \geq 1\}$被某个线性有界自动机接受。

找到一个线性有界自动机，使其接受语言$L = \{a^{n!} : n \geq 0\}$。

说明

我们也可以定义确定型的线性有界自动机，但是我们目前还不清楚它是否与非确定型的线性有界自动机等价。

线性有界自动机比下推自动机功能更强大。为此，我们还需要证明任何上下文无关语言都可以被线性有界自动机接受。

我们还需要明确图灵机与线性有界自动机之间的关系。

谢谢！

主 讲 人： 姚 刚
电子邮箱： yaogang@iie.ac.cn