电子科技大学

博士学位论文

形式语言与自动机理论若干问题研究

姓名：陈文宇

申请学位级别：博士

专业：计算机应用技术

指导教师：孙世新

20091125

摘 要

计算理论是研究使用计算机解决计算问题的数学理论。有三个核心领域：形式语言与自动机理论、可计算性理论和计算的复杂性理论。可计算性理论的中心问题是建立计算的数学模型，进而研究哪些是可计算的，哪些是不可计算的。计算的复杂性理论研究算法的时间复杂性和空间复杂性。在可计算性理论中，将问题分成可计算的和不可计算的；在复杂性理论中，目标是把可计算的问题分成简单的和复杂的。

形式语言与自动机理论论述计算的数学模型的定义和性质，这些模型在计算机科学的若干应用领域中起着重要作用，其应用范围已被扩展到生物工程、自动控制系统、图像处理与模式别等许多领域。

本文主要研究形式语言与自动机理论中的形式语言理论、自动机理论和形式语言与自动机之间等价性理论的若干问题。研究内容包括：

（1）形式语言与自动机理论中关于空串ε的一些问题。

（2）四类语言对联合、连接、克林闭包运算的有效封闭性问题。

（3）根据等价类构造一类有限状态自动机的方法。

（4）通用图灵机的统一、合理的编码系统。

（5）一种新的图灵机构造技术：扫描子串技术。

（6）非负整数的 k 元函数对于关系运算和算术运算的图灵可计算问题。

（7）非负二进制整数关于关系运算和算术运算的图灵可计算问题。

关键词：形式语言，自动机，有效封闭性，扫描子串技术，图灵可计算性

ABSTRACT

Computation theory is the mathematic theory for solving computational problems with computers. The three main research areas of the theory are: formal language and automata theory, computability theory and the computing complexity theory. The core issue of the computability theory is to establish mathematic model of the computation and then decide its computability. The computing complexity theory discusses the time complexity and the space complexity of the algorithms. The objective of the complexity theory is to group the computable problems into simple problems and complex ones.

Formal language and automata theory discusses the definition and properties of the computational models. These models play a significant role in many fields of computer science and the range of application has been extended to multiple fields like bioengineering, automatic control system, image processing and pattern recognition.

The dissertation discusses formal language theory, automata theory and the equivalency of formal language and automata. Research area includes:

(1)Issues regarding blank string ε of formal language and automata theory.

(2)The effective closer issue of four types of languages by join, linkage and Kleene closure operations.

(3)The method of creating a specific type of finite state automata by equivalence class.

(4)The universal reasonable coding system for Turing machines.

(5)A new technique to create Turing machines: substring scan technique.

(6)The Turing computable issue of k-variable functions of non-negative integers by relational operations and arithmetic operations.

(7)The Turing computable issue of non-negative binary integers by relational operations and arithmetic operations.

Keywords: Formal language, automata, closure effectiveness, substring scan technique, Turing computability

独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

签名：_____ 日期：2009年11月25日

论 文 使 用 授 权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

签名：_____ 导师签名：_____

日期：2009 年 11 月 25 日

第一章 绪 论

1.1 研究的目的与意义

计算机科学与技术学科强调 4 个方面的专业能力：计算（机）思维能力、算法设计与分析能力、程序设计与实现能力、计算机系统的认知、分析、设计和运用能力[1]。这也是计算机科学与技术学科与其他学科的重要区别。相关的理论是计算机学科的基础。理论方面的知识是计算机的真正灵魂。理论是从计算机应用当中抽象出来的，目的在于更好地指导实践。

计算机科学与技术学科要求具有形式化描述和抽象思维能力，要求掌握逻辑思维方法。这种能力就是计算思维能力[1]。

计算机学科系统地研究信息描述和变换算法，主要包括信息描述和变换算法的理论、分析、效率、实现和运用。学科的根本问题是有效自动化问题。学科的重要内容之一是研究计算领域中的一些普遍规律，描述算法的基本概念与模型[1,2]。

计算模型无论从方法还是从工具等方面，都表现出它在计算机上科学中的重要作用。建立物理符号系统并实施等价变换是计算机学科进行问题描述和求解的重要手段。

本文从形式语言与自动机的角度研究信息形式化描述方式的一些规律以及自动机模型的若干问题，对计算机科学与技术学科具有较强的理论意义。

1.2 研究的背景与现状

1.2.1 形式语言与自动机理论的发展

语言学家 Noam Chomsky(乔姆斯基)最早从产生语言的角度研究自然语言。1956 年，通过抽象，Chomsky 将语言形式地定义为由一个字母表的字母组成的一些字符串的集合：对于任意语言 L，有一个字母表Σ，使得$L\subseteq\Sigma^*$。语言中的字符串称为句子。可以在字母表上按照一定的形成规则定义一个文法 G（Grammar），该文法产生的所有的句子组成的集合就是该文法产生的语言 L（Language）[3]。判断一个字符串是否是一个语言的句子，需要判断该字符串是否能够由语言对应的文法推导产生[4]。1959 年，Chomsky 根据产生语言的文法的产生式的不同特点，

将文法分为三类，文法产生的语言对应也分为三类[5]，因此建立了 Chomsky 文法和语言的基本体系。

正则表达式 RE（Regular Expression）起源于人类神经系统的早期研究。神经生理学家 Warren McCulloch 和 Walter Pitts 研究出一种使用数学方式描述神经网络的方法，将神经系统中的神经元描述成小而简单的自动控制元[6]。1950 年，数学家 Stephen Kleene（克林）利用称之为正则集合的数学符号来描述神经网络的模型；1956 年，Stephen Kleene 正式引入了正则表达式的概念[6,7]。

Stephen Kleene 与数学家 Claude Shannon（香农）和 John McCarthy（麦卡锡）在 1951~1956 年间，从识别语言的角度来研究语言，给出了语言的另一种定义方式。Stephen Kleene 在研究神经细胞时建立了自动机模型[8]，使用该模型来识别（接收）一个语言：按照某种识别规则构造自动机，该自动机就定义了一个语言，该语言由自动机能够识别的所有字符串构成。Stephen Kleene 还研究了数学与自动机之间的关系[9,10,11]。

语言的两种不同的定义方式引起了科学家们的研究兴趣。一个语言，可以采取不同的定义方式：文法产生语言和自动机识别语言；两种方式是等价的，存在两种方式之间的等价的相互转换方法。

1959 年，Chomsky 将他本人对形式语言的研究成果和 Kleene 的（正则表达式与自动机）的研究成果结合起来，不仅确定了可以从文法产生语言的角度和自动机识别语言的角度进行语言的定义，而且证明了上下文无关文法 CFG（Context Free Grammar）与正则表达式 RE 和有限状态自动机 FA（Finite Automata）的等价性[4,12-14]。此时，形式语言与自动机理论才开始真正诞生。

形式语言与自动机理论出现后，迅速在计算机科学技术领域得到了应用。使用巴科斯-诺尔范式 BNF（Backus-Naur Form）[15]成功地对高级程序设计语言 ALGOL-60 的词法规则和语法规则进行了形式化的描述（实际上，巴科斯-诺尔范式就是 Chomsky 文法体系中的上下文无关文法 CFG 的产生式的另一种表示方式），使得形式语言与自动机理论得到了进一步的发展；尤其是上下文无关文法 CFG，被作为计算机程序设计语言的语法规则的最佳近似描述得到了较为深入的研究[16-23]。后来，人们又将上下文无关文法 CFG 应用到了模式匹配和模型化处理等方面[24-26]，这些研究成果是算法描述和分析、计算复杂性理论和可计算性理论的研究基础或扩展。

下推自动机 PDA（Push Down Automaton）是自动机理论中定义的一种抽象的计算模型。Anthony G. Oettinger 在二十世纪 50 年代，对计算机的指令系统进行了

研究[27,28]，并最早于1961提出了下推自动机PDA这一形式系统[29]。下推自动机PDA与上下文无关文法CFG的等价性由Chomsky于1962年发现[4]。下推自动机较有限状态自动机更为复杂，除了有限的状态组成部分外，还包括一个存储容量可以不受限制的栈。下推自动机的状态转换不但要参考有限状态部分，还要参考栈顶当前内容；状态转换不但包括了状态的改变，还包括一个出栈或入栈操作。下推自动机可以形象的理解为，把有限状态自动机扩展使之可以存取一个栈[20,31]。

接收语言能力最强的自动机是图灵机（Turing Machine），它是由A.Turing(图灵)于1936年提出的，用于描述（定义）算法，提出了图灵论题：一切算法可计算函数都是图灵可计算函数[32]。图灵机是作为一种可计算性的数学模型提出的，为计算机的发展奠定了理论基础。

二十世纪30年代，Church(丘奇)使用称为λ-演算的记号系统来定义算法[33,34]；提出丘奇论题：算法可计算的函数都是λ-可定义函数；算法可计算函数都是递归函数[35]。1937年，图灵拓广了丘奇论题，形成丘奇-图灵论题[36]：对于任何可以用有效算法解决的问题，都存在解决此问题的图灵机。丘奇-图灵论题对计算理论的严格化，对计算机科学的形成和发展都具有奠基性的意义。1985年，牛津大学的Deutsch D.（多奇）提出了丘奇-图灵论题的物理版本，即多奇原理(也称物理的丘奇-图灵原理)：每个有限可实现的物理系统，总能为一台通用模拟机器以有限方式的操作来完美地模拟[37]。

丘奇-图灵论题还没有被严格的证明，因为，只能列出算法的有限性、机械可执行性、确定性、终止性等特征，即有效算法说明的可解性概念是非形式化的、不严格的，而图灵机的概念却是形式化的和严格的。但是，图灵机作为人们普遍接受的计算模型，可以认为，可计算问题等同于图灵机的可计算问题[1,36]。

图灵机等同于各种存储指令的计算机系统，可以将图灵机用作算法定义的精确模型。描述算法有三种基本方式：形式描述、实现描述和高级描述。形式描述需要详细给出图灵机的状态定义和状态转换函数的定义；实现描述通常使用自然语言来描述图灵机的动作，如读/写头的移动，怎样存储数据等，不需要给出具体的状态转换函数的描述；高级描述采用自然语言（或其他描述方式）进行描述，忽略了图灵机模型，即通常意义上的算法描述。

自动机作为一种抽象的计算模型，对于理解计算和信息处理的概念与机制具有非常重要的作用[38]。Hopcroft J.E, Ullman J.D对传统的4类自动机，即有限状态自动机、下推自动机、线性有界自动机和图灵机（线性有界自动机实际上是一种特殊的图灵机）进行了规范定义，并研究了四类自动机的性质和特点[39]。随着自

动机理论的不断发展和实际应用的需要，许多新的（或变形的）的自动机模型被定义。作为一般的有限状态自动机的扩充，时间自动机时钟变量的取值是非负的实数，为实时系统的自动化分析和验证提供了形式化的理论模型[40,41]；树自动机是对传统自动机的扩展，树自动机状态转换前（或后）支持状态集合，这种特性使得树自动机能够更加自然地处理树状数据结构[42]；集合自动机也是对传统自动机的扩展，集合自动机采用一个集合作为存储带，所识别的语言对于空集问题是可判定的[43]；交替下推自动机是对传统的下推自动机的扩展[44-47]，并成为一种并行计算理论模型[48,49]。该类型存在一个特例，对于有界上下文无关语言可以由 2 路确定的下推自动机识别[50]。张继军将自动机进行扩展，使得自动机可以存取一个袋，这样的自动机就称为袋自动机，具有与传统的下推自动机不同的性质和特点[51]。Ginsgurg, S., Spanier, E.H.和 Kutrib, M., Malcher, A.研究了下推自动机的一种限制变形，即转向（turn）下推自动机，以空存储方式接收语言，使得转向下推自动机接收的语言类等同超线性语言类[52,53]。郭清泉，邵长春提出了转向下推自动机的一种构造方法[54]。Gudder S. 等人研究了量子计算及量子自动机[55-60]，邱道文和郭秀红研究了量子自动机[61,62,63]。

无限自动机理论是研究存储量无限的离散数字系统功能和结构以及两者关系的的理论，是自动机理论的次级学科。主要研究内容包括信息加工系统和计算过程的数学模型、模拟、计算、识别和限制等问题。吕映芝等研究了无限自动机[64,65,66]。

形式语言与自动机的理论得到了许多的应用[67-81]，新的运用也在不断的研究中。

1.2.2 本文研究的相关问题现状

1 空句子 ε 的产生和接收问题

Chomsky 的文法体系中，上下文无关文法 CFG 和正则文法 RG（Regular Grammar）没有对空串产生式（ε 产生式）进行限制，也就没有特别讨论语言中空句子的产生方法；蒋宗礼对文法中空串 ε 产生式进行了限制[1]，讨论了语言中空句子的产生方法，但没有特别讨论有限状态自动机 FA 对空句子的接收问题。Michael Sipser 和 Hopcroft J.E, Ullman J.D 提出扩展带空移动的不确定的有限状态自动机 ε-NFA（Non-Deterministic Finite Automaton with ε-moves）转换为不确定的有限状态自动机 NFA（Non-Deterministic Finite Automaton）的方法[38,39]，但该方法比较烦琐；周启海讨论了有穷自动机的规范化和 NFA 的转换方法[82,83]，但没有涉及到

ε-NFA 自动机的转换问题。

2 语言运算的封闭性

语言运算的封闭性问题，是形式语言语言研究中的重要问题之一，在理论和实践上都具有重要意义。

蒋宗礼和陈有祺证明对于不同的字母表，上下文无关语言 CFL（Context Free Language）和正则语言 RL（Regular Language）对语言的基本运算是有效封闭的[1,2]；Peter Linz 提出了上下文相关语言 CSL（Context Sensitive Language）对运算的串道（crosstalk）问题，提出了相应的解决方法[84]，但没有讨论上下文相关语言 CSL 和短语结构语言 PSL（Phrase Structure Language）对克林闭包运算的串道问题。Hopcroft J.E, Ullman J.D 和 Peter Linz 从自动机（特别是有限状态自动机）的角度分析了语言运算的封闭性问题[39,84]。

从形式语言的角度，Chomshy 四类语言对运算的有效封闭性，还没有统一的规范化的方法。

3 基于等价类构造确定的有限状态自动机

一类特定语言是由任意字母表上的 K 进制的数字串构成，而数字串（当作数字）能够整除任意非负整数 N（或对 N 的余数为 M，N≥2，N≥M≥0），蒋宗礼，吴哲辉针对任意字母表上的语言，提出了基于等价类构造确定的有限状态自动机 DFA（Deterministic Finite Automaton）的方法[1,85]，但对任意的 K、N 和 M，缺乏通用性的考虑。

4 通用图灵机的编码

蒋宗礼研究了通用图灵机（Universal Turing machine ）的编码方式，约定字母表为{0，1}，采用 0、1 组合的编码方式，对图灵机的状态转换函数进行编码[1]，但没有对图灵机的基本情况、任意的字母表和待接收的输入串进行编码。

5 构造图灵机的常用技术

图灵机成熟的构造技术包括：存储技术、移动技术、多道技术、查讫技术、子程序技术，陈晓亮提出了图灵机的递归技术[86]。在各种技术中，图灵机读/写头每次只能扫描一个符号。如果语言中的所有句子必须包含（或者不能包含）特定的于串；或者需要将语言中所有句子包含的特定子串进行替换，图灵机常用的构造技术就比较烦琐。

6 图灵计算模型

蒋宗礼、陈有祺、Hopcroft J.E 和 Ullman J.D 采用“一进制”方式表示一个非负整数，即使用 0^m，表示整数 m，分析了非负整数的加法、真减法和乘法的图灵

可计算问题[1,2,39]，但没有讨论关系运算、一般减法运算、除法和余数运算，也没有涉及到其他进制非负整数的运算。

1.3 本文的主要研究内容

本文主要研究形式语言与自动机理论中形式语言理论、自动机理论和形式语言与自动机之间等价性理论的若干问题。研究内容包括：

（1）形式语言与自动机理论中关于空串ε的一些问题。分析ε产生式对文法和语言分类的影响；从文法和有限状态自动机的角度，讨论开始符号 S 和开始状态 q_0 的作用，提出语言增加或减少ε句子的简单方法；研究ε-NFA 的ε状态转换函数的本质，提出ε-NFA 转换为 NFA 的新方法。

（2）四类语言对联合、连接、克林闭包运算的有效封闭性问题。分析上下文相关语言 CSL 和短语结构语言 PSL 对于连接、克林闭包运算可能存在的串道问题，提出了避免串道的方法；根据语言对联合，连接和迭代运算是封闭的特点，提出基于简单自动机构造复杂自动机的方法。

（3）针对特定的一类语言，提出根据等价类构造有限状态自动机的方法。该方法可以针对所有字母表和所有进制的数字串构成的语言，满足语言中的所有数字串能够对任意正整数 N 取任意余数，该类有限状态自动机是极小化的。

（4）图灵机的统一、合理的编码系统。研究基于 0、1 对图灵机的编码方法，可以对图灵机的基本情况、字母表、状态（包括开始状态、接收状态和可能的拒绝状态）、状态转换函数和待接收的输入串进行编码。

（5）针对特定的一类语言，提出一种新的图灵机构造技术：扫描子串技术，即将特定的子串当作一个整体，使得图灵机一次可以扫描多个符号，但读/写头每次仍然只移动一个单元。

（6）研究非负整数的 k 元函数对于关系运算（包括大于、小于、等于运算）和算术运算（包括加法、减法、乘法、除法和余数运算）的图灵可计算问题。

（7）研究非负的二进制整数关于关系运算和算术运算的图灵可计算问题。提出输入串长度图灵计数器模型，可以对图灵机输入带上符号串进行计数。

1.4 特色和创新

本文从理论的角度对形式语言与自动机理论中的若干问题进行了研究。

研究形式语言与自动机理论中关于空串ε的一些问题；提出ε-NFA 转换为 NFA 的新方法，该方法简单，易于理解和实现。

分析上下文相关语言 CSL 和短语结构语言 PSL 对于连接、克林闭包运算可能存在的串道现象，提出对应的解决办法，较完整地解决了语言运算中可能的串道问题；提出基于简单自动机构造复杂自动机的方法。

根据等价类构造有限状态自动机可以接收一类特定语言，对该方法进行了扩展，可以针对任意字母表和所有进制的数字串构成的语言，并且从构造的角度保证了有限状态自动机的极小化。

扩展基于 0、1 对图灵机进行编码的方法，可以对图灵机的基本情况、字母表、状态（包括开始状态、接收状态和可能的拒绝状态）、状态转换函数和待接收输入串进行了编码。

针对特定的一类语言，提出一种新的图灵机构造技术：扫描子串技术，即将特定的子串当作一个整体，使得图灵机一次可以扫描多个符号，具有较强的抽象性。

研究非负整数的 k 元函数对于关系运算（包括大于、小于、等于运算）和算术运算（包括加法、减法、乘法、除法和余数运算）的图灵可计算问题，提出非负二进制数与非负一进制数之间的相互转换方法。

创建非负的二进制整数关于关系运算和算术运算的图灵可计算模型，并扩展到 N 进制（$3 \le N \le 10$）；提出输入串长度图灵计数器模型，可以计算图灵机输入带上符号个数。

1.5 本文的章节安排

论文的内容是按照形式语言理论（第 3 章）、形式语言与自动机等价理论（第 3、4 章）和自动机理论（5-9 章）进行组织的。 自动机的研究的重点内容是构造技术，因此在第 5 章和第 7 章分别讨论了有限状态自动机和图灵机的构造技术；图灵机是功能最强大的自动机，论文的 6-9 章围绕图灵机进行研究；而图灵机最重要的价值是可计算的模型，论文的 8、9 章则重点研究了图灵的计算模型的一些问题。

本文共分为十章。

第一章绪论部分，概述本文的研究意义和研究内容。

第二章介绍形式语言与自动机的基本理论。

第三章研究形式语言与自动机中的关于ε的一些问题。

第四章研究形式语言运算的有效封闭性分析；讨论了根据运算的有效封闭性构造自动机的方法。

第五章研究利用等价类构造有限状态自动机的方法。

第六章研究通用图灵机的编码方案。

第七章研究图灵机扫描多个符号技术。

第八章研究非负 k 元函数的图灵计算模型。

第九章研究二进制非负整数的图灵计算模型，并扩展到 N 进制（$3 \leq N \leq 10$）。

第十章总结全文。

第二章 形式语言与自动机的基本理论

形式语言和自动机理论中的语言是一个广泛的概念。一个字母表上的语言就是该字母表的某些字符串的集合。 语言中的字符串称为该语言的句子。

语言的的定义可以从两个方面进行，从产生语言的角度；从接收(或识别)语言的角度。

产生一个语言，需要根据语言中的基本句子和其他句子的形成规则，产生该语言所包含的所有句子。这是形式语言理论研究的问题。

接收一个语言，需要使用某种自动机模型来接收句子，该模型所接收的所有句子，也形成一个语言。这是自动机理论研究的问题。

形式语言与自动机研究的目标都是语言，存在形式语言与自动机的等价性理论。

本章介绍形式语言和自动机的基本理论，主要内容来源于参考文献[1]、[2]、[38]、[39]、[84]和[85]。

2.1 形式语言基本理论

有穷语言的表示较容易，即使语言中的句子的结构上没有什么规律，也可以使用枚举的方式列出语言中的所有句子。

对于无穷语言，需要使用可能的有穷描述的方式表达。需要从语言包含的句子的一般构成规律去考虑问题。这种从语言的有穷描述来表达语言的方法对一般的语言都是有效的。尤其在使用计算机自动地判断一个字符串是否是某个语言的句子时，从句子和语言的结构特征上着手是非常重要的。

对于一类语言，可以在字母表上，按照一定的构成规则，根据语言的结构特点，定义一个文法。使用文法作为相应语言的有穷描述，不仅可以描述出语言的结构特征，而且可以产生这个语言的所有句子。

文法的作用就是产生一个语言。

2.1.1 文法和语言

文法 G 是一个四元式，即

$$G=(\Sigma,V,S,P)$$

其中，

Σ是一个有限字符的集合，叫做字母表，它的元素称为字母或者终结符；

V 是一个有限字符的集合，叫做非终结符集合，它的元素称为变量或者非终结符(一般用大写英文字母表示)；

S 是一个特殊的非终结符，即 $S\in V$，称为文法的开始符号；

P 是有序偶对(α，β)的集合，其中 α 是集合($\Sigma\cup V$)上的字符串，但至少包含一个非终结符；β 是集合$(\Sigma\cup V)^*$的元素。一般，将有序偶对(α，β)记为 $\alpha\rightarrow\beta$，称为产生式。

形如$\alpha\rightarrow\varepsilon$ 的产生式称为空产生式，也可称为 ε 产生式。

给定文法 G，y 和 z 是集合($\Sigma\cup V$)上的串，若 y，z 分别可以写成 pvr 和 pur(p 和 r 可能同时为空串)，而 $v\rightarrow u$ 是文法 G 的一个产生式，则称串 y 可以直接推导出串 z，记为 y =>z，或 pvr =>pur。

与之相对应，称串 pur 可以直接归约成串 pvr。

使用 $y=>^+z$ 表示 y 可以经过多步(至少一步)，推导出 z，即存在一个串的序列

$$\alpha_1,\alpha_2,\alpha_3,\cdots,\alpha_n$$

有

$$y=\alpha_1，\ z=\alpha_n$$

且

$$\alpha_i=>\alpha_{i+1}$$

对所有 $n>i\geq 1$。

使用 $y=>^*z$ 表示 y 可以经过任意步(包括 0 步)推导出 z，即

$$y=z$$

或

$$y=>^+z$$

对于文法 G，如果 $S=>^*\omega$，则称 ω 是文法的一个句型；若 ω 中包含的字符全是终结符，称 ω 是文法的一个是句子。

给定文法 G，有开始符号 S，把 S 可以推导出的所有的终结符串的集合(即所有句子的集合)，称为由文法产生的语言，记为 L(G)，即

$$L(G)=\{\omega|\ S=>^*\omega,且\ \omega\in\Sigma^*\}$$

一个文法 G 确实产生语言 L，必须同时满足：文法 G 推导产生的所有句子都在语言 L 中；语言 L 的任意一个句子都可以由文法 G 产生。

2.1.2 Chomsky 文法和语言体系

Chomsky 对文法进行了分类；而语言是由文法产生的，对语言的分类，是根据产生语言的文法的分类进行的。

对于任意文法 G=(Σ，V，S，P)，G 为 0 型文法，或短语结构文法 PSG（Phrase Structure Grammar）。对应产生的语言为 0 型语言或短语结构语言 PSL。

文法 G，如果对于任意$\alpha\rightarrow\beta\in P$，均有$|\alpha|\leq|\beta|$成立，则 G 为 1 型文法，或上下文相关文法 CSG（Context Sensitive Grammar）。对应产生的语言为 1 型语言或者上下文相关语言 CSL。

如果对于任意$\alpha\rightarrow\beta\in P$，均有$|\alpha|\leq|\beta|$且$\alpha\in V$ 成立，则 G 为 2 型文法，或上下文无关文法 CFG。对应产生的语言为 2 型语言或者上下文无关语言 CFL。

如果对于任意$\alpha\rightarrow\beta\in P$，$\alpha\rightarrow\beta$均具有形式 $A\rightarrow w$ 或 $A\rightarrow wB$，其中，$A,B\in V$，$w\in\Sigma^{+}$，则称 G 为 3 型文法，或正则文法 RG（Regular Grammar）。对应产生的语言为 3 型语言或正则语言 RL。

上下文无关文法 G 是 GNF 范式(Greibach Normal Form)的形式，若上下文无关文法 G 的每个产生式的形式为 $A\rightarrow bW$，其中，$b\in\Sigma$，$W\in V^{*}$。

四类文法和对应的四类语言之间具有真包含关系，如图 2-1 所示。

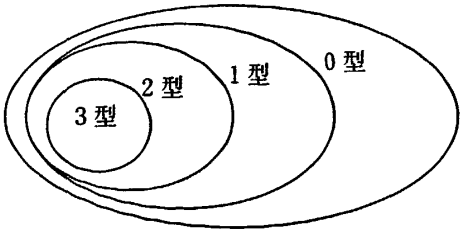

图 2-1 四类文法和四类语言之间的包含关系

根据文法分类的定义，在 CSG、CFG 和 RG 中，都不能含有空产生式，所以任何 CSL、CFL 和 RL 中都不包含空句子 ε。

2.1.3 正则表达式和正则集

计算学科讨论的是什么能够被有效地自动化，而实现有效自动化的基础首先是实现对问题进行恰当的形式化描述。

对于正则的语言，可以使用正则表达式进行表示。正则表达式对正则语言的表示具有特殊的优势：它更简单，更方便，更容易进行处理；而且，这种表达形式还更接近语言的集合表示和语言的计算机表示。

L 是字母表Σ上的语言，若语言 L 是有限的，则语言 L 是正则的；或者语言 L 能够由下列运算递归地产生：

(1)若 L_1 和 L_2 是正则的，且 $L=L_1 \cup L_2$；

(2)若 L_1 和 L_2 是正则的，且 $L=L_1L_2$；

(3)若 L_1 是正则的，且 $L=L_1^*$；

则 L 也是正则的；语言 L 也称为正则集。

正则表达式 R 和对应表达的正则集 S(R)为

(1)ϕ 是一个正则表达式，$S(\phi)=\phi$；

(2)ε 是一个正则表达式，$S(\varepsilon)=\{\varepsilon\}$；

(3)若 $a \in \Sigma$，则 a 是一个正则表达式，$S(a)=\{a\}$；

(4)若 R_1 和 R_2 是正则表达式，则 R_1+R_2 是正则表达式 $S(R_1+R_2)=S(R_1) \cup S(R_2)$；

(5)若 R_1 和 R_2 是正则表达式，则 R_1R_2 是正则表达式，$S(R_1R_2)=S(R_1)S(R_2)$；

(6)若 R 是正则表达式，则 $(R)^*$ 是正则表达式，$S((R^*))=(S(R))^*$；

正则表达式中，克林闭包运算的优先级最高，联合运算的优先级最低。

2.2 自动机基本理论

在计算机科学中自动机是计算机和计算过程的动态数学模型，用来研究计算机的体系结构、逻辑操作、程序设计乃至计算复杂性理论。在语言学中则把自动机作为语言识别器，用来研究各种形式语言。在神经生理学中把自动机定义为神经网络的动态模型，用来研究神经生理活动和思维规律，探索人脑的机制。在生物学中有人把自动机作为生命体的生长发育模型，研究新陈代谢和遗传变异。在数学中则用自动机定义可计算函数，研究各种算法。

传统的自动机分为 3 大类：有限状态自动机、下推自动机和图灵机。

2.2.1 有限状态自动机

有限状态自动机 FA 是为研究有限存储的计算过程和某些语言类而抽象出的一种计算模型。有限状态自动机拥有有限数量的状态，每个状态可以迁移到零个或多个状态，有限状态自动机可以表示为一个有向图（状态转换图）。

有限状态自动机物理模型如图 2-2 所示。

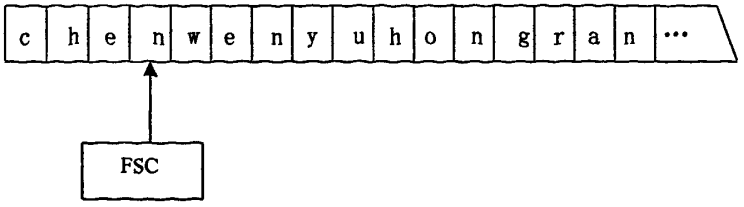

图 2-2 有限状态自动机的物理模型

有限状态自动机包括一个输入存储带，带被分解为单元，每个单元存放一个输入符号（字母表上的符号），整个输入串从带的左端点开始存放，而带的右端可以无限扩充；

有限状态自动机还包括一个有限状态控制器（FSC），该控制器的状态只能是有限多个；FSC 通过一个读头和带上单元发生耦合，可以读出当前带上单元的字符。初始时，读头对应带的最左单元，每读出一个字符，读头向右移动一个单元（读头不允许向左移动，在有限状态自动机的一类变性中，允许读头向左移动）。

有限状态自动机的一个动作为，读头读出带上当前单元的字符；FSC 根据当前自动机的状态和读出的字符，改变有限状态自动机的状态；并将读头向右移动一个单元。

有限状态自动机可以分为确定的有限状态自动机和不确定的有限状态自动机；有限状态自动机还可以带有 ε 动作。

1 确定的有限状态自动机

字母表Σ上的有限状态接收机 FA 是一个五元式，

$$FA=(Q,\Sigma,\delta,q_0,F)$$

其中，

Q 是一个有限状态的集合；

Σ是字母表，也就是输入带上的字符的集合；

$q_0 \in Q$ 是开始状态；

$F \subset Q$ 是接收状态（终止状态）集合；

δ 是 $Q\times\Sigma\rightarrow Q$ 的状态转换函数，即

$$\delta(q,x)=q'$$

代表自动机在状态 q 时，扫描字符 x 后到达状态 q′。

有限状态自动机的状态转换函数的个数应该为$|Q|*|\Sigma|$。因为对于 Q 中的每个状态，都应该定义扫描字母表Σ上的每个字母的状态转换函数。称这种有限状态自动机为确定的有限状态自动机 DFA。

给定 DFA，扩展的状态转换函数 δ^*为

$$Q\times\Sigma^{*}\rightarrow Q$$

即

$$\delta^{*}(q,w)=q'$$

即自动机在一个状态 q 时，扫描串 w 后到达唯一确定的状态 q′。δ^{*}可以递归为

$$\delta^{*}(q,\varepsilon)=q$$

$$\delta^{*}(q,x)=\delta(q,x)$$

$$\delta^{*}(q,w)$$

$$=\delta^{*}(q,x\alpha)$$

$$=\delta^{*}(\delta(q,x),\alpha)$$

其中，$w\in\Sigma^{+}$，$x\in\Sigma$，$\alpha\in\Sigma^{+}$。

对于字母表Σ上的有限状态自动机 DFA，它能识别的所有串的集合，称为有限状态自动机能识别的语言。记为 L(DFA)。

$$L(DFA)=\{w|w\in\Sigma^{*}\text{且}\ \delta^{*}(q_0,w)\in F\}$$

若语言 $L\subset\Sigma^{*}$，对于某个确定的有限状态自动机 DFA，有 L=L(DFA)，则称语言 L 为一个有限状态语言 FSL（Finite State Language）。

2 不确定的有限状态自动机

不确定的有限状态自动机NFA是一个五元式，

$$NFA=(Q,\Sigma,\delta,Q_0,F)$$

其中，

Q是一个有限状态的集合；

Σ是字母表；

$Q_0\subset Q$是开始状态集合；

$F\subset Q$是接收状态（终止状态）集合；

δ是$Q\times\Sigma\rightarrow 2^{Q}$的状态转换函数，即 $\delta(q,x)\subset 2^{Q}$；代表不确定的有限状态自动机在状态q时，扫描字符x后到达可能的下一状态集合。

不确定有限状态自动机与确定有限状态自动机的重要区别在于它们的转移函数不同。确定有限状态自动机对每一个可能的输入只有一个状态的转移。不确定有限状态自动机对每一个可能的输入可以有多个状态转移，当接收到某个输入时从这多个状态转移中不确定地选择一个。

在扫描一个串w时，经过NFA可能会有多条路径，某些可能会在接收状态时终止，某些可能会在非接收状态时终止；若至少存在一条路径可以使自动机在扫描串w后到达接收状态，则称串w能被不确定的有限状态自动机NFA所识别。

对于字母表Σ上的不确定的有限状态自动机NFA，能够识别的所有串的集合，称为自动机NFA能识别的语言。记为L(NFA)。

$$L(NFA)=\{w|w\in\Sigma^{*}且\ \delta^{*}(Q_0,w)\cap\neq\Phi\}$$

确定的有限状态自动机 DFA 与非确定有限状态自动机 NFA 是等价的。它们接收的语言为正则语言。

3 带空移动的有限状态自动机

对于一般的有限状态自动机，有

$$\delta(q,\varepsilon)=q$$

和

$$\delta^{*}(q,\varepsilon)=q$$

表示有限状态自动机不读入任何字符（即只扫描空串时），自动机的状态不发生改变，并且读头不进行移动，而仍然指向当前非空字符。

如果允许有限状态自动机在不读入任何字符时，自动机的状态可以发生改变，则称该有限状态自动机为带空移动的有限状态自动机 ε- NFA。

带空移动的有限状态自动机 ε -NFA 是一个五元式，

$$\varepsilon\text{- NFA}=(Q,\Sigma,\delta,Q_0,F)$$

其中，

Q，Σ，Q_0，F 的含义同 NFA。

δ 是 $Q\times\Sigma\cup\{\varepsilon\}\rightarrow 2^{Q}$ 的状态转换函数，$\delta(q,x)\subset Q$ 或 $\delta(q,\varepsilon)\subset Q$，具体地

$$\delta(q,x)=\{p_1,p_2,p_3,\cdots,p_m\}$$

表示自动机在读入字母 x 后，自动机的状态可以选择地改变为 p_1，p_2，p_3，…，或者 p_m，并将读头向右移动一个单元而指向了下一个字符；

$$\delta(q,\varepsilon)=\{p_1,p_2,p_3,\cdots,p_m\}$$

表示自动机在状态 q 时，不读入任何字母，自动机的状态可以选择地改变为 p_1，p_2，p_3，…，或者 p_m，并且读头不进行移动，而仍然指向当前非空字符。

带空移动的有限状态自动机，对于任意状态 q，从 q 接收空串 ε 后能够到达的状态集记为 ε-CLOSURE(q)。

$$\varepsilon\text{-CLOSURE}(q)=\{p|从\ q\ 到\ p\ 有一条标记为\ \varepsilon\ 的路\}$$

对于状态集合 P，定义

$$\varepsilon\text{-CLOSURE}(P)=\{\varepsilon\text{-CLOSURE}(q)|\ q\in P\ \}$$

ε -NFA 扩展状态转换函数为映射 $Q\times\Sigma^{*}\rightarrow 2^{Q}$，即

$$\delta^{*}(q,\varepsilon)=\varepsilon\text{-CLOSURE}(q)$$

$$\delta^*(q,a)=\varepsilon\text{-CLOSURE}(P)$$
$$=\varepsilon\text{-CLOSURE}(\delta(\delta^*(q,\varepsilon),a))$$
$$\delta^*(q,wa)=\varepsilon\text{-CLOSURE}(P)$$

其中，

$$P=\{p|\ p\in\delta(r,a)\text{且 } r\in\delta^*(q,w)\}$$

ε -NFA 最终扩展的状态转换函数为映射 $2^Q\times\Sigma^*\rightarrow 2^Q$，即

$$\delta^*(P,w)=\cup\delta^*(q,w)$$

其中，$q\in P$。

带空移动的有限状态自动机 ε -NFA 与不确定有限状态自动机 NFA 是等价的。

2.2.2 下推自动机

下推自动机 PDA 的的主要部分是一个后进先出的栈存储器，一般有两个操作，入栈：增加栈中的内容（作为栈顶）；出栈：将栈顶元素移出。将栈的操作用于下推自动机的的动作描述，加上状态和不确定的概念，可以构成完全识别上下文无关语言的自动机模型—下推自动机。

下推自动机物理模型如图 2-3 所示。

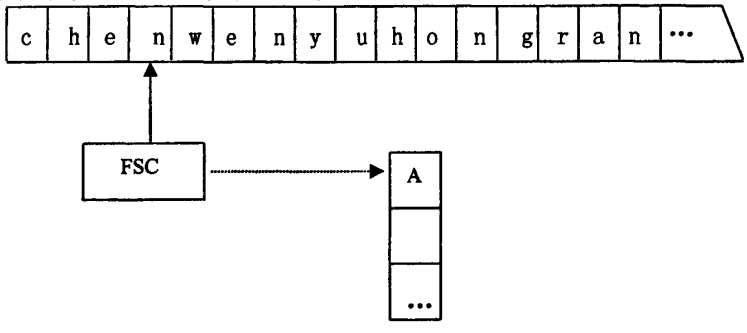

图 2-3 下推自动机的物理模型

下推自动机包括一个输入存储带，带被分解为单元，每个单元存放一个输入符号（字母表上的符号），整个输入串从带的左端点开始存放，而带的右端可以无限扩充；

下推自动机包括一个有穷状态控制器（FSC），FSC 通过一个读头和带上单元发生耦合，可以读出当前带上单元的字符。初始时，读头对应带的最左单元，每读出一个字符，读头向右移动一个单元（读头不允许向左移动）。

下推自动机还包括一个堆栈，存放不同于输入带上的符号，只能对栈顶元素

进行操作。

下推自动机的一个动作为，读头读出当前带上单元的字符；根据当前的状态、读出的字符以及栈顶符号，下推自动机改变状态，将一个符号压入栈或将栈顶符号弹出栈，并将读头向右移动一个单元。

下推自动机 PDA 是一个七元式，

$$PDA=(Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$$

其中，

Q 是一个有限状态的集合；

Σ 是输入串的字母集合；

Γ 是栈内符号集合；

$q_0\in Q$ 是开始状态；

$Z_0\in\Gamma$ 是初始的栈底符号

$F\subset Q$ 是接收状态（终止状态）集合；

δ 是 $Q\times(\Sigma\cup\{\varepsilon\})\times\Gamma\rightarrow Q\times\Gamma^*$的状态转换函数。

对于确定的 PDA，有 $\delta(q,x,Z)=(q',Z')$；一般用

$$<q,x,Z,q',Z'>$$

代表 δ 函数（也称为 PDA 的规则）。

PDA 的格局是一个三元式

$$(q,w,\sigma)$$

其中，q 为状态，$w=x_1x_2\cdots x_n$，是还没有被 PDA 扫描到的串，且将要扫描 x_1，$\sigma=Z_1Z_2\cdots Z_m$，是栈内的符号串，且 Z_1 在栈顶，Z_m 在栈底。一个格局代表了某个时刻 PDA 的情况。PAD 的初始格局为

$$(q_0,w,Z_0)$$

而接收格局为

$$(q,\varepsilon,\varepsilon)$$

格局的转换是由于状态转换函数的作用引起的。

对于确定的下推自动机 PDA，若 PDA 有规则

$$<q,x,A,q_1,A_1A_2\cdots A_k>$$

则引起的格局转换为

$$(q,xw,A\sigma)=>(q_1,w,A_1A_2\ldots A_k\sigma)$$

对于不确定的下推自动机 PDA，有两类不确定的情况。

1）若不确定的下推自动机 PDA 有规则

$$<q,x,A,q_1,A_1A_2\cdots A_k>$$

则引起的格局转换为

$$(q,xw,A\sigma) \Rightarrow (q_1,w,A_1A_2\ldots A_k\sigma)$$

或 PDA 有规则

$$<q,\varepsilon,A,q_2,B_1B_2\cdots B_j>$$

则引起的格局转换为

$$(q,xw,A\sigma) \Rightarrow (q_2,xw,B_1B_2\cdots B_j\sigma)$$

2）若不确定的下推自动机 PDA 有规则

$$<q,x,A,q_1,A_1A_2\cdots A_k>$$

则引起的格局转换为

$$(q,xw,A\sigma) \Rightarrow (q_1,w,A_1A_2\cdots A_k\sigma)$$

或 PDA 有规则

$$<q,x,A,q_2,B_1B_2\cdots B_j>$$

则引起的格局转换

$$(q,xw,A\sigma) \Rightarrow (q_2,w,B_1B_2\cdots B_j\sigma)$$

不确定的 PDA 对于相同的格局（q,xw,Aσ）可能会有不同的格局转换。

使用$\Rightarrow^*$来代表 PDA 格局的多次变换。

PDA 以空栈方式接收的语言为 L(M)，且

$$L(PDA)=\{w|(q_0,w,Z_0)\Rightarrow^*(q,\varepsilon,\varepsilon),\text{对任意 } q\in Q\}$$

接收情况与接收状态无关，只要当串 w 扫描结束，而栈为空，则串 w 就能够被 PDA 以空栈方式所接收。

PDA 经过有限状态接收的语言为 F(M)，且

$$F(PDA)=\{w|(q_0,w,Z_0)\Rightarrow^*(q',\varepsilon,\sigma),q'\in F,\sigma\in\Gamma^*\}$$

接收情况与栈内是否还有内容无关，只要当串 w 扫描结束，而 PDA 处于某个接收状态，则串 w 就能够被 PDA 经过有限状态所接收。PDA 经过有限状态所接收语言也称为 PDA 以终态方式接收语言。

PDA 接收语言两种方式是等价的。

一个广义的下推自动机也是七元式，（除了状态转换函数外，其余同一般的 PDA），状态转换函 δ 是

$$Q\times(\Sigma\cup\{\varepsilon\})\times\Gamma^+\rightarrow Q\times\Gamma^*$$

的映射，即

$$\delta(q,x,B_1B_2\cdots B_k)=(q',C_1C_2\cdots C_n)$$

规则的一般形式为

$$<q,x, B_1B_2\cdots B_k,q',C_1C_2\cdots C_n>$$

对于一般的下推自动机，栈顶是一个符号，而广义的下推自动机的栈顶可以为一个符号串。

广义的下推自动机与一般的下推自动机等价。

若一个下推自动机仅仅只有一个状态，称之为单态的下推自动机 PDA，

$$单态\ PDA=(\{*\},\Sigma,\Gamma,\delta,*,Z_0,\{*\})$$

单态下推自动机与具有多个状态的下推自动机等价。

2.2.3 图灵机

图灵机用于可计算性(可计算的特点是有穷、离散、机械执行、停机)的 研究。实际上，图灵机可以模拟现代的计算机的计算能力。使用图灵机可以解决计算机程序的可计算问题。

图灵机的物理模型结构类似于有限状态自动机的物理模型结构，如图 2-4 所示。

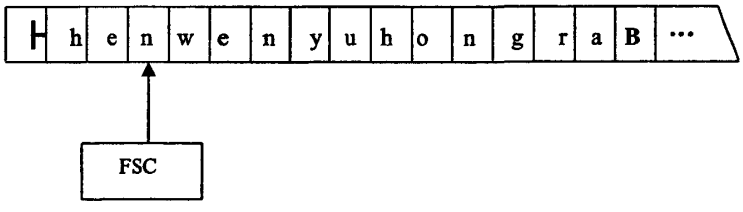

图 2-4 图灵机的物理模型

图灵机有一个有限状态控制器 FSC，一个外部的存储设备，即一条可以随机向右扩展的无限长度带(带上具有左端点，使用“ ⊢”表示)。带被分为单元，每个单元可以为空或者存放字母表上的字母符号，为方便起见，使用不属于字母表的特殊字符 B 来标记带上的空单元；有限状态控制器通过一个读/写头来和带进行耦合。一般在带的右边用 B 标记带上符号的右期间。

在任意时刻，有限状态控制器处于某个状态，读/写头将扫描带上的一个单元，依照此时的状态和扫描到的带上符号，图灵机将有一个动作为，有限状态控制器所处的状态进行改变；把扫描过的单元上符号擦除掉，并印刷上一个新的符号（有可能印刷上与原来符号相同的符号，使得带上该单元的内容不变）；把读/写头向左或者向右移动一个单元；或者读/写头不移动。

1 图灵机的定义

图灵机 TuringM（Turing Machine）是一个五元式，

$$TuringM=(Q,\Sigma, q_0, q_\alpha,\delta)$$

其中，

Q 是有限状态集合；

Σ是带上字母表的有限集合，用$\Sigma'=\Sigma\cup\{B\}$代表Σ的增广集合；

$q_0\in Q$，是开始状态；

$q_\alpha\in Q$，是接收状态；

δ 是 $Q\times\Sigma'\rightarrow Q\times\Sigma'\times\{L,R,N\}$的状态转换函数

$$\delta(q,x)=(q',W,\{L,R,N\})$$

一般，将状态转换函数（也称为图灵机的规则）记为

$$<q,x,q',W,\{L,R,N\}>$$

其中，$x,W\in\Sigma\cup\{B\}$，表示图灵机处于状态 q 时，若扫描到符号 x，则状态变换为 q′，印刷上新的符号 W，读/写头向左（L）、向右（R）或者不移动（N）。

图灵机有状态集 Q 和Σ的增广集合Σ'，则每个某个时刻图灵机的格局为

$$w_1qw_2\in(\Sigma')^*Q(\Sigma')^*$$

其中，

w_1 是读/写头左边已经印刷在带上的符号串；

q 是图灵机当前所处的状态；

w_2 是读/写头右边还未扫描到的带上的符号串。

若格局 $w_1qw_2=w_1qxw$ 对于 $\delta(q, x)$无定义，则图灵机在格局 w_1qw_2 时停机，。

若图灵机在格局 w_1qw_2 时不停机，则图灵机可以进行格局的转换。

使用符号=>表示格局的转换。某个时刻，图灵机处于格局

$$w_1qw_2=r_1yqxr_2$$

其中，

$r_1y=w_1$　（若 $w_1=\varepsilon$，则 y=B，　$r_1=\varepsilon$）

$xr_2=w_2$　（若 $w_2=\varepsilon$，则 $r_2=\varepsilon$，　x=B）

下一个格局为记为 $w_1'qw_2'$。

若

$$\delta(q,x)=(q',x',L)$$

则

$$w_1'qw_2'=r_1q'yx'r_2$$

若

$$\delta(q,x)=(q',x',N)$$

则

$$w_1'qw_2'=r_1yq'x'r_2$$

若

$$\delta(q,x)=(q',x',R)$$

则

$$w_1'qw_2'=r_1y\,x'q'r_2$$

使用$=>^*$代表图灵机格局的多次变换。

图灵机在字母表Σ上接收的语言为L(TuringM)，则

$L(TuringM)=\{w|w\in\Sigma^*$，且存在$w_1$，$w_2\in(\Sigma')^*$，有$q_0w=>^*w_1q_\alpha w_2\}$

图灵机M称为不确定的图灵机（或不确定的图灵机），除状态转换函数δ的定义外，它的其余部分的定义同确定的图灵机。即对于某个状态q和扫描到的带上符号x，图灵机可能有多个动作（即图灵机的状态转换函数δ可能将$Q\times\Sigma'$映射到$Q\times\Sigma'\times\{L,R,N\}$的一个子集上）。

图灵机接受的语言是短语结构语言。

2 线性有界的图灵机

一个不确定的图灵机，若它的所有操作都局限于带上预先设置的区间内，即读/写头不允许离开带上的某个区间，该不确定的图灵机是线性有界的图灵机LBA（Linear Bounded Aotomaton），一般，分别用\$和¢来标记带上的左、右边界。

一个线性有界的图灵机所接收的语言是上下文相关语言CSL。

3 图灵机的构造技术

1）图灵机的存储技术

图灵机的有限状态控制器（FSC）可以保存有限数量的信息，即保存多个状态。状态的表示方法可以多种多样，不仅仅是单个字母或者加上一些下标的字母的简单标记。实际上，状态可以使用比较复杂的结构进行表达，至少可以使用一个n元组表示一个状态，而n元组的不同的分量可以代表不同的含义。比较常用的是使用二元组表示单个状态，其中第一个分量仍然表示原来的状态；第二个分量是输入带上的符号串的子串，可以使用这种方式将输入带上的一个或多个符号“存储”到图灵机的有限控制器中。即使用一个分量实现控制，而另一个分量用于存储。

2）图灵机的移动技术

在解决比较复杂的问题时，图灵机经常需要将输入带上一组连续的非空符号

左移或者右移若干个单元，实现这一要求的图灵机也是使用状态存储一个或多个符号，直到某个阶段再将这些符号印刷到输入带上。

3）图灵机的多道技术

为了能够保存和处理更复杂的数据，可以将图灵机的一条输入带水平地划分为若干道，在各道上可以存放不同的符号。这样没有改变图灵机的基本模型，只是将输入带上的符号当做是一个向量的组合，其中每个符号可以是一个 K 维向量（将输入带划分为 K 道）。

4）图灵机的查讫技术

在图灵机的工作过程中，有时需要对输入带上已经扫描过的符号进行某种检查。为了区分带上的某个符号是否已经检查过，可以使用查讫符号“√”进行标记，即在该已经检查过的符号上方或下方印刷上特殊符号“√”。为了给查讫符号预留出位置，需要使用多道技术。

5）图灵机的子程序技术

和通常的程序设计技术相似，子程序的思想在图灵机的构造中也是一种十分重要的技术。子程序技术的使用，可以将复杂的问题进行分解（化简），同时，也可以将图灵机的构造“模块化”，更便于图灵机的设计。图灵机的子程序技术的基本思想是将图灵机中需要重复使用的功能分离出来，作为一个子程序。

设完成整个功能的图灵机为 M（作为主程序对待），完成某个特定功能的图灵机为 M′（即使用某个小的图灵机作为子程序），图灵机 M′从状态 q 开始（q 不是整个图灵机的开始状态），到一个固定的状态 f（f 不是整个图灵机的接收状态）结束；而状态 q 和 f 是图灵机 M 的两个一般状态；当图灵机 M 进入状态 q 时，就启动 M′（相当于调用子程序）；当 M′进入状态 f 时就返回到 M（相当于子程序结束）。

2.3 形式语言与自动机等价性基本理论

1 正则语言 RL 与有限状态自动机 DFA(或 NFA)等价

正则语言有 5 种等价模型：正则文法 RG，正则表达式 RE、确定的有限状态自动机 DFA，不确定的有限状态自动机 NFA，带 ε 动作的有限状态自动机 ε -NFA。

正则语言的 5 种等价模型的转换关系可以用图 2-5 表示。

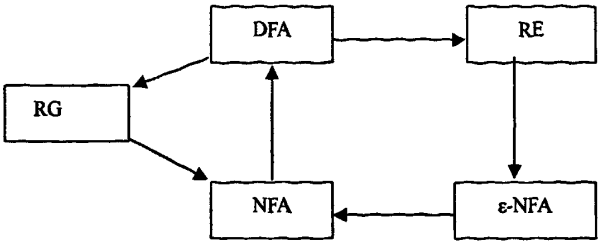

图 2-5 正则语言的5种等价模型的转换关系

正则语言的5种等价模型的直接转换可以归纳为6种情况。

1）确定的有限状态自动机DFA 转换为正则文法RG

假设L是有限状态语言FSL，且L=L(DFA)，令

$$DFA=(Q,\Sigma,\delta,q_0,F)$$

将自动机的状态当作是文法的非终结符，构造右线性文法

$$RG=(\Sigma,Q,q_0,P)$$

其中，P为

$$\{q\rightarrow xq'|\delta(q,x)=q'\}\cup\{q\rightarrow x|\delta(q,x)\in F\}$$

特别地，若开始状态也是接收状态，则有$q_0\rightarrow\varepsilon$。

2）正则文法RG转换为不确定的有限状态自动机NFA

假设L是正则语言，且L=L(G)，令

$$G=(\Sigma,V,S,P)$$

构造NFA，将文法的非终结符当作NFA的状态，并且增加一个接收状态q（若文法G中有$S\rightarrow\varepsilon$，即$\varepsilon\in L$，则开始状态S也是接收状态）使得

$$NFA=(Q,\Sigma,\delta,Q_0,F)$$

其中，

$$Q=V\cup\{q\}$$

$$Q_0=\{S\}$$

$$F=\{q\}$$

$$\delta(A,x)=\{B|B\in V,\text{且 } A\rightarrow xB \text{ 在 P 中}\}\cup\{q|A\rightarrow x \text{ 在 P 中}\}$$

3）确定的有限状态自动机DFA 转换为正则表达式RE

对DFA的状态转换图进行适当的处理：增加标记为X和Y的两个状态：X状态为新的开始状态，且入度为0；Y状态是新的惟一接收状态；然后，对状态图进行相应的处理，直到整个图最后只剩下X和Y的两个状态，以及从X状态到Y状态的可能的惟一一条弧；而这条弧上标记的正则表达式，就是所求的DFA所接收

语言对应的正则表达式；当该弧不存在时，DFA 所接收语言为Φ，对应的正则表达式为Φ。

4）正则表达式 RE 转换为带 ε 动作的有限状态自动机 ε -NFA

正则语言对于联合、连接和闭包三种运算是有效封闭的，则对于正则表达式 R，存在一个等价的带 ε 动作的有限状态自动机 ε -NFA。

5）带 ε 动作的有限状态自动机 ε -NFA 转换为不确定的有限状态自动机 NFA

假设语言 L 被一个带空移动的有限状态自动机 ε -NFA 接收，令

$$\varepsilon\text{-NFA}=(Q,\Sigma,\delta,q_0,F)$$

构造一个不带 ε 动作的有限状态自动机 NFA，

$$\text{NFA}=(Q,\Sigma,\delta_1,q_0,F_1)$$

其中，

$$\delta_1(q,a)=\delta^*(q,a)$$

$$F_1=\begin{cases} F\cup\{q_0\} & \text{如果 } F\cap\varepsilon\text{-CLOSURE}(q_0)\neq\Phi \\ F & \text{如果 } F\cap\varepsilon\text{-CLOSURE}(q_0)=\Phi \end{cases}$$

6）不确定的有限状态自动机 NFA 转换为确定的有限状态自动机 DFA

假设语言由 NFA 接收，令

$$\text{NFA}=(Q,\Sigma,\delta,Q_0,F)$$

构造

$$\text{DFA}'=(Q',\Sigma,\delta',q_0',F')$$

其中，

$$Q'=2^Q$$

$$\delta'(p,x)=\cup\{\delta(q,x)|q\in p\};\ p\in Q',x\in\Sigma$$

$$q_0'=Q_0\in Q'$$

$$F'=\{p'|p'\in Q'\text{且 }p'\cap F\neq\Phi\}\subset Q'$$

2 下推自动机接收上下文无关语言

下推自动机可以和上下文无关文法相互进行转换，它们都对应于上下文无关语言。

假设上下文无关文法是 GNF 范式（若不是，则先将文法改造为 GNF 范式的形式），可以构造一个单态的 PDA 来接收语言 L。GNF 范式有 3 种形式的产生式，它们分别对应 PDA 的 3 种状态转换函数。

若 GNF 范式有产生式

$$S\rightarrow\varepsilon$$

则单态的 PDA 有状态转换函数

$$<\varepsilon,S,\varepsilon>$$

若 GNF 范式有产生式

$$A\rightarrow b$$

则单态的 PDA 有状态转换函数

$$<b,A,\varepsilon>$$

若 GNF 范式有产生式

$$A\rightarrow bW$$

则单态的 PDA 有状态转换函数

$$<b,A,W>$$

其中：$A\in V$，$W\in V^{+}$。

3 线性有界的图灵机与上下文相关语言

线性有界的图灵机可以和上下文相关文法相互进行转换，它们都对应于上下文相关语言。详细转换过程请见参考文献[1]。

4 图灵机与短语结构语言

图灵机可以和短语结构文法进行相互转换，它们都对应于短语结构语言。详细转换过程请见参考文献[1] 。

2.4 本章小结

本章介绍了形式语言与自动机理论 3 个方面的基本内容，包括形式语言基本理论、自动机基本理论和形式语言与自动机等价性基本理论。

第三章 形式语言与自动机理论中的关于ε的一些问题

Chomsky 的文法体系中，上下文无关文法 CFG 和正则文法 RG 没有对空串产生式（ε 产生式）进行限制，也就没有特别讨论语言中空句子的产生方法；蒋宗礼对文法中空串 ε 产生式进行了限制，讨论了语言中空句子的产生方法，但没有特别讨论有限状态自动机 FA 对空句子的接收问题。Michael Sipser 等提出扩展带空移动的不确定的有限状态自动机 ε-NFA 转换为不确定的有限状态自动机 NFA 的方法，但该方法比较烦琐；周启海讨论了 NFA 的转换方法，但没有涉及到 ε-NFA 自动机的转换问题。

本章讨论形式语言与自动机理论中关于空串 ε 的一些问题。分析 ε 产生式对文法和语言分类的影响；从文法 G 和有限状态自动机 FA 的角度，讨论了文法开始符号 S 和有限状态自动机的开始状态 q_0 的作用，提出语言增加或减少 ε 句子的简单方法；研究了 ε-NFA 的 ε 状态转换函数的本质，提出 ε-NFA 转换为 NFA 的新方法，并给出实际例子进行了验证。

3.1 ε 产生式对文法和语言分类的影响

形式语言理论中，文法的基本分类原则只允许 0 型文法中出现 ε 产生式，而不允许 1、2 和 3 型文法中出现 ε 产生式[1]。通过限制开始符号 S 的作用，使得 1、2 和 3 型文法中可以出现关于开始符号 S 的 ε 产生式。

空串 ε 不属于任何字母表，但可以属于某个语言，称 ε 为该语言的空句子。文法要产生空句子 ε，最简单的方法是提供关于开始符号 S 的 ε 产生式

$$S\rightarrow \varepsilon$$

文法产生的语言如果不包括空句子 ε，则文法 G 可以没有任何的 ε 产生式。

为了简便，文法中可能包括 ε 产生式，例如语言

$$L=0(0+1)^*$$

产生语言 L 的文法可以为 G_1：

$$S\rightarrow 0|0A$$

$$A\rightarrow 0|1|0A|1A$$

其中，

$$A=>^{+}(0+1)^{+}$$

文法 G_1 等价于文法 G_2：

$$S\rightarrow 0B$$

$$B\rightarrow \varepsilon|0B|1B$$

其中，

$$B=>^{+}(0+1)^{*}$$

文法 G_2 中存在 ε 产生式，较文法 G_1 少了 2 个产生式。

如果文法的开始符号 S 不出现在文法 G 的任意产生式的右边，那么 S 仅作为特殊非终结符号使用，只负责推导过程的开始。在任意句子的推导过程中，所有句型都不包括开始符号 S，即

$$S=>^{+}\alpha A\beta$$

其中，

$$\alpha,\beta\in\Sigma;\ A\neq S$$

文法增加 ε 产生式

$$S\rightarrow \varepsilon$$

则该 ε 产生式仅只使用一次，产生 ε 句子。

如果文法的开始符号 S 出现在文法的某个产生式的右边，增加的 ε 产生式可以产生多余的句子。例如文法 G_3：

$$S\rightarrow ab|aS$$

产生语言

$$a^{+}b$$

增加 ε 产生式后的文法 G_4：

$$S\rightarrow ab|aS|\varepsilon$$

产生语言

$$a^{+}b+\varepsilon+a^{+}$$

增加了空句子 ε，但产生多余句子 a^{+}。

对于任意文法 G，如果 S 不出现在文法 G 的任何产生式的右边，若 G 是 1、2 或 3 型文法，则

$$G'=(\Sigma,V,S,P\cup\{S\rightarrow\varepsilon\})$$

$$G''=(\Sigma,V,S,P-\{S\rightarrow\varepsilon\})$$

仍然是 1、2 或 3 型文法。对应产生的语言仍然是 1、2 或 3 型语言。

3.2 FA 接收 ε 句子

如果 FA 需要接收空句子，最简单的方法是设置开始状态为接收状态。设 FA 接收语言 L，设置语言

$$L'=L\cup\{\varepsilon\}$$

或

$$L''=L-\{\varepsilon\}$$

可以方便地构造新的 FA 接收 L′和 L″。但存在一些情况，使得新的 FA 还可以接收多余的串。

3.2.1 DFA 接收 ε 句子

给定确定的有限状态自动机 DFA，令 L=L(DFA)，如果将 DFA 的开始状态 q_0 也设置为接收状态，则可以接收空句子。

但如果 DFA 存在

$$\delta(q,a)=q_0$$

形式的产生式，就可以接收不属于 L 的串。例如图 3-1 所示的 DFA_1

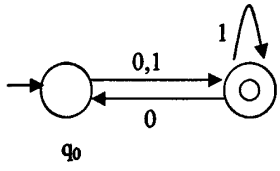

图 3-1 DFA_1 的状态图

开始状态 q_0 不属于接收状态，且其他状态可以转换为开始状态 q_0，如果直接将 DFA 的开始状态也设置为接收状态，则得如图 3-2 所示的 DFA_2

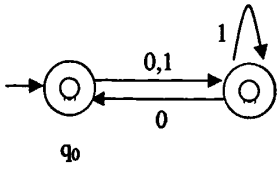

图 3-2 DFA_2 的状态图

可以接收空句子 ε，还接收 00 等多余串。

对 DFA 进行改进，增加开始状态 q_s，使得开始状态 q_0 只负责接收整个输入串的第一个符号（类似于文法的开始符号 S 不出现在文法的任何产生式的右边），即

开始状态的入度为 0，不允许任何状态转换为开始状态，那么开始状态不能够作为一般状态使用，仅仅负责接收工作的开始。

任意 $a\in\Sigma$，对于 DFA 所有关于开始状态 q_0 的状态转换函数

$$\delta(q_0,a)=q'$$

增加关于开始状态 q_s 的状态转换函数

$$\delta(q_s,a)=q'$$

改造 DFA_1 为等价的状态图如图 3-3 所示的 DFA_3

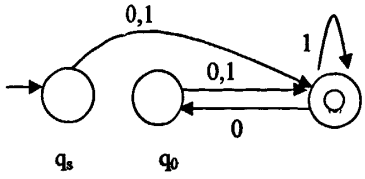

图 3-3 DFA_3 的状态图

可以直接将开始状态设置为接收状态，则仅仅多接受空句子 ε 。即改造 DFA_1 为状态图如图 3-4 所示的 DFA_4

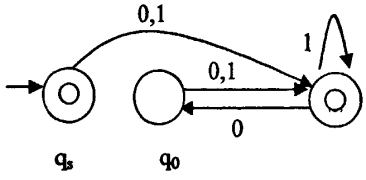

图 3-4 DFA_4 的状态图

3.2.2 NFA 接收 ε 句子

不确定的有限状态自动机 NFA 接收空句子的情况类似于 DFA；任意 $a\in\Sigma$，对于 NFA 所有关于开始状态 q_0 的状态转换函数

$$\delta(q_0,a)=Q'$$

增加新的开始状态 q_s 和关于 q_s 的状态转换函数

$$\delta(q_s,a)= Q'$$

即可。

3.2.3 ε -NFA 接收 ε 句子

对于带空移动的不确定的有限状态自动机 ε -NFA，改进 ε -NFA 的方法是增加

新的开始状态 q_s（也设置为接收状态）和一个 ε 状态转换函数

$$\delta(q_s, \varepsilon) = q_0$$

即可。

DFA、NFA 和ε-NFA 接收的语言需要除去空句子 ε 则只需要将开始状态 q_s 设置为非接收状态即可。

3.3 ε -NFA 转换为 NFA

ε-NFA 转换为 NFA′ 的传统方法是将 ε -NFA 的状态转换函数转换为 NFA′ 的状态转换函数，使得

$$\delta'(q,a)=\delta^*(q,a)$$

该方法需要计算出ε-NFA 每个状态 q 的扩展状态转换函数 $\delta^*(q, \varepsilon)$，比较烦琐。

可以将ε-NFA 先转换为扩展的正则文法，再转换为 NFA。对于

$$\varepsilon\text{-NFA} = (Q, \Sigma, \delta, q_0, F)$$

构造等价的文法

$$G=(\Sigma, Q, q_0, P)$$

其中，

$$\begin{aligned} P = &\{q \to aq' \mid q' \in \delta(q,a)\} \\ &\cup \{q \to q' \mid q' \in \delta(q,\varepsilon)\} \\ &\cup \{q \to a \mid \delta(q,a) \cap F \neq \Phi\} \\ &\cup \{q \to \varepsilon \mid \delta(q,\varepsilon) \cap F \neq \Phi\} \end{aligned}$$

该文法不属于正则文法。主要是包含有单产生式和 ε 产生式，称为扩展的正则文法，可以通过删除单产生式和 ε 产生式，得到规范的正则文法。

1）删除单产生式

对于单产生式 A→B 的推导作用，推导过程

$$S \Rightarrow^* \alpha A \Rightarrow \alpha B \Rightarrow \alpha w_I \Rightarrow \cdots$$

可以简化为

$$S \Rightarrow^* \alpha A \Rightarrow \alpha w_1 \beta \Rightarrow \cdots$$

即省略掉

$$A \Rightarrow B$$

的推导。

因此，需要使用

$$A\rightarrow w_1|w_2|w_3|\cdots|w_n$$

将A→B进行替换，而

$$B\rightarrow w_1|w_2|w_3|\cdots|w_n$$

是文法G中关于B的所有产生式。

2）删除ε产生式

对于C∈V，考虑产生式

$$C\rightarrow wA$$

增加产生式

$$C\rightarrow w$$

则能够体现出A→ε产生式的作用，则可以删除A→ε。

如果ε∈L(G)，则要增加新的开始符号S′和产生式

$$S'\rightarrow \varepsilon$$

对于

$$S\rightarrow r$$

增加

$$S'\rightarrow r$$

得到等价的规范的正则文法。

对于正则文法

$$G=(\Sigma,V,S,P)$$

将文法的非终结符当作NFA的状态，并增加一个接收状态q（若文法G中有S→ε，即ε∈L，则开始状态S也是接收状态），使得

$$NFA=(Q,\Sigma,\delta,Q_0,F)$$

其中，

$$Q=V\cup\{q\}$$

$$Q_0=\{S\}$$

$$F=\{q\}$$

$$\delta(A,x)=\{B|B\in V,\text{且 } A\rightarrow xB \text{ 在 P 中}\}$$

$$\cup\{q|A\rightarrow x \text{ 在 P 中}\}$$

则

$$L(G)=L(NFA)$$

例3-1 转换的实例。

状态图如图3-5所示的ε-NFA

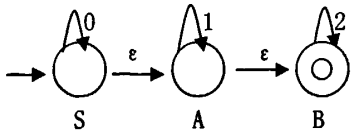

图 3-5 接收 $0^*1^*2^*$的ε-NFA

接收语言

$$0^*1^*2^*$$

转换为扩展的正则文法

$$S\rightarrow 0S|A$$

$$A\rightarrow 1A|B|\varepsilon$$

$$B\rightarrow 2B|2|\varepsilon$$

消除单产生式后得

$$S\rightarrow 0S|1A|2B|2|\varepsilon$$

$$A\rightarrow 1A|2B|2|\varepsilon$$

$$B\rightarrow 2B|2|\varepsilon$$

消除 ε 产生式后得

$$S\rightarrow 0S|0|1A||1|2B|2|\varepsilon$$

$$A\rightarrow 1A||1|2B|2$$

$$B\rightarrow 2B|2$$

增加 S′ 和相应产生式后得正则文法

$$S'\rightarrow \varepsilon|0S|0|1A||1|2B|2$$

$$S\rightarrow 0S|0|1A||1|2B|2$$

$$A\rightarrow 1A||1|2B|2$$

$$B\rightarrow 2B|2$$

直接转换为状态图如图 3-6 所示的 NFA。

3.4 本章小结

形式语言与自动机理论中，文法的基本分类原则只允许 0 型文法中出现 ε 产生式，而不允许 1、2 和 3 型文法中出现 ε 产生式。通过限制开始符号 S 的作用，使得 1、2 和 3 型文法中可以出现关于 S 的 ε 产生式，同时提出了文法产生 ε 句子的简单方法。相应地，限制了 FA 中开始状态的作用，使得开始状态仅仅只负责接收输入串的第 1 个字母。

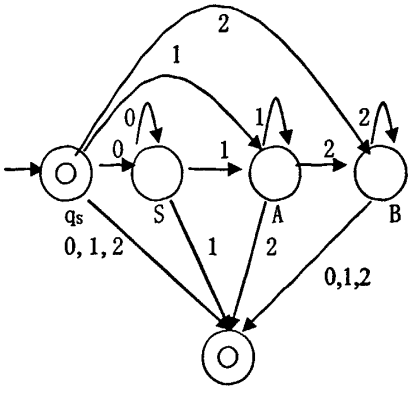

图 3-6 与图 3-5 代表的 ε -NFA 等价的 NFA 的状态图

文法 G 的开始符号可以作为一般的非终结符使用，有限状态自动机 FA 的开始状态也可以作为一般状态使用，G 产生的语言和 FA 接收的语言需要增加或减少空句子 ε 是困难的。本文通过限制开始符号和开始状态的作用，提出语言增加或减少空句子 ε 的简单方法。

通过研究 ε -NFA 的 ε 状态转换函数的本质，提出 ε -NFA 转换正则文法和 NFA 的简便方法，即先将 ε -NFA 转换为扩展的正则文法形式，消除 ε 产生式和单产生式后，得到规范正则文法，再将正则文法转换为 NFA，该方法简单，易于理解和实现。

第四章 语言运算的有效封闭性分析

语言运算的有效封闭性问题,是语言研究中的重要问题之一，在理论和实践上都具有重要意义。

语言对运算有效封闭可以证明一个语言属于某个语言类；也可以从简单语言构造复杂的同类语言。本章介绍 Chomsky 语言体系中语言之间的基本运算；讨论四类语言对联合、连接、克林闭包运算的有效封闭性问题。如果两个语言的字母表有相同元素时，分析上下文相关语言和短语结构语言对于连接、克林闭包运算可能存在的串道问题，提出了对应的解决办法；讨论通过正则语言的有效封闭性形成正则表达式的方法；针对上下文无关语言，讨论上下文无关语言对于上下文无关置换运算的有效封闭性问题。

本章也讨论了利用简单自动机构造复杂自动机的方法，请见参考文献[87]。

4.1 引言

设字母表为Σ，Ψ 是Σ上的某一类语言（0，1，2 或 3 型语言类），L_1 和 L_2 是 Ψ 的任意两个语言，若对 L_1 和 L_2 进行某种运算后产生的语言也是 Ψ 的语言，则称语言类 Ψ 对于该运算封闭。

对于产生语言类Ψ中语言的文法，若可以构造出给定运算下所获得的同类语言的文法，则语言类Ψ对给定运算有效封闭。

文献[1，2，84，85]证明了不同字母表情况下，正则语言和上下文无关语言对语言的基本运算是有效封闭的；也提出了上下文相关语言运算的串道问题，但没有讨论语言对克林闭包运算的串道问题。

从形式语言的角度，针对 Chomshy 四类语言对运算的有效封闭性，还没有统一的规范化的方法。

构造接收复杂语言的自动机是困难的，而复杂语言可以通过简单的语言进行语言间的运算而产生，根据正则语言 RL、上下文无关语言 CFL 和上下文相关语言 CSL 对联合，连接和迭代运算是封闭的特点，提出了利用简单自动机构造复杂自动机的方法。

4.2 语言的基本运算和运算的有效封闭

语言 L_1 和 L_2 分别是字母表 Σ_1 和 Σ_2 上的任意两个语言。

设置 $\Sigma=\Sigma_1\cup\Sigma_2$。$L_1$ 和 L_2 的联合运算定义为

$$L=L_1\cup L_2=\{w|w\in L_1 \text{ 或者 } w\in L_2\}$$

语言 L 的字母表为 Σ。

L_1 和 L_2 的连接运算定义为

$$L=L_1L_2=\{\,w\mid w=w_1w_2,w_1\in L_1,w_2\in L_2\}$$

语言 L 的字母表为 Σ。

L_1 的克林闭包运算（或星运算）定义为

$$L=L_1^*=\{w|w=w_1w_2\cdots w_m,w_i\in L_1,m\geq 0\}$$
$$=\cup L_1^n \qquad \text{对 } n\geq 0$$

语言 L 的字母表为 Σ_1。

有效封闭性问题可以简单描述为，存在相同类型文法 G_1 和 G_2

$$L_1=L(G_1)$$
$$L_2=L(G_2)$$

需要构造同类型的文法 G，使得

$$L(G)=\alpha(L_1,L_2)$$

或

$$L(G)=\beta(L_1)$$

4.3 四类语言对基本运算有效封闭

设参加运算的语言 L_1 和 L_2 分别是字母表 Σ_1 和 Σ_2 上的语言，产生语言 L_1 的文法为

$$G_1=(\Sigma_1,V_1,S_1,P_1)$$

产生语言 L_2 的文法为

$$G_2=(\Sigma_2,V_2,S_2,P_2)$$

则

$$S_1=>\alpha=>^*w_1\in L_1$$
$$S_2=>\beta=>^*w_2\in L_2$$

假定

$$\Sigma_1 \cap \Sigma_2 = \Phi$$

$$V_1 \cap V_2 = \Phi$$

$$S \notin V_1$$

$$S \notin V_2$$

设置

$$\Sigma = \Sigma_1 \cup \Sigma_2$$

$$V = V_1 \cup V_2 \cup \{S\}$$

4.3.1 联合运算的有效封闭

联合运算需要考虑既能够产生语言 L_1 的所有句子，又能够产生语言 L_2 的所有句子。可以考虑增加产生式

$$S \rightarrow S_1 | S_2$$

产生语言 L_1 的句子或语言 L_2 的句子。

构造文法

$$G_3 = (\Sigma, V, S, P_3)$$

其中，

$$P_3 = \{ S \rightarrow S_1 \} \cup \{ S \rightarrow S_2 \} \cup P_1 \cup P_2$$

对于 i=0、1、2， 若 G_1 和 G_2 是 i 型文法，则 G_3 也是同类型文法。

G_3 可以利用

$$S => S_1 => \alpha => {}^{*} w_1 \in L_1$$

产生语言 L_1 的所有句子；或利用

$$S => S_2 => \beta => {}^{*} w_2 \in L_2$$

产生语言 L_2 的所有句子，即

$$L(G_3) = L_1 \cup L_2$$

结论，0、1、2 型语言类对联合运算有效封闭。

如果 G_1 和 G_2 是 3 型文法，而 G_3 不是 3 型文法，构造 3 型文法

$$G_4 = (\Sigma, V, S, P_4)$$

其中，

$$P_4 = \{S \rightarrow \alpha | S_1 \rightarrow \alpha \in P_1\} \cup \{S \rightarrow \beta | S_2 \rightarrow \beta \in P_2\} \cup P_1 \cup P_2$$

则 G_4 是 3 型文法。

G_4 可以利用

$$S\Rightarrow \alpha \Rightarrow^* w_1 \in L_1$$

产生语言 L_1 的所有句子；或利用

$$S\Rightarrow \beta \Rightarrow^* w_2 \in L_2$$

产生语言 L_2 的所有句子，即

$$L(G_4)=L_1 \cup L_2$$

结论，3 型语言对联合运算有效封闭。

实际上，G_4 的构造方法也适合于构造 0、1、2 型文法。

4.3.2 连接运算的有效封闭

连接运算需要考虑语言 L_2 的句子能够出现在语言 L_1 的句子后面。可以考虑增加产生式

$$S\rightarrow S_1S_2$$

产生语言 L_1 的句子和语言 L_2 的句子的连接。

构造文法

$$G_5=(\Sigma,V,S,P_5)$$

其中，

$$P_5=\{S\rightarrow S_1S_2\} \cup P_1 \cup P_2$$

对于 i=0、1、2， 若 G_1 和 G_2 是 i 型文法，则 G_5 也是同类型文法。

G_5 利用

$$S\Rightarrow S_1S_2\Rightarrow \alpha\beta\Rightarrow^* w_1w_2 \in L_1L_2$$

得到 L_1 和 L_2 的连接，即

$$L(G_5)=L_1L_2$$

结论，0、1、2 型语言对连接运算有效封闭。

如果 G_1 和 G_2 是 3 型文法，而 G_5 不是 3 型文法，构造 3 型文法

$$G_6=(\Sigma,V_1 \cup V_2,S_1,P_6)$$

其中，

$$P_6=\{A\rightarrow wS_2 | A\rightarrow w \text{ 在 } P_1 \text{ 中}\}$$
$$-\{A\rightarrow w\} \cup P_1 \cup P_2$$

文法 G_6 利用

$$S_1 =>^+ r_1r_2\cdots r_kA$$

$$=>r_1r_2\cdots r_kwS_2=>^* w_1w_2\in L_1L_2$$

其中，$r_1r_2\cdots r_kw \in L_1$，产生语言 L_1 和 L_2 的连接，即

$$L(G_6)=L_1L_2$$

结论，3 型语言对连接有效封闭。

4.3.3 连接运算的串道问题

G_1 和 G_2 是 0 型或 1 型文法，若

$$\Sigma_1\cap\Sigma_2\neq\Phi\ （包括\Sigma_1=\Sigma_2）$$

则构造的文法 G_5 可能存在问题[84]。

例 4-1 串道的发生。

文法 G_1：

$$S_1\rightarrow a$$

文法 G_2：

$$S_2\rightarrow aS_2$$

$$aS_2\rightarrow bc$$

则

$$L_1=a$$

$$L_2=a^*bc$$

$$L_1L_2= a^+bc$$

而 G_5 可以进行

$$S=>S_1S_2 =>aS_2=>a^+S_2=>^+ a^+bc$$

的推导，也可以进行

$$S=>S_1S_2=>aS_2=>bc$$

的推导，即文法 G_5 产生的语言为

$$a^*bc\neq L_1L_2=a^+bc$$

该问题产生的原因是 S_1 和 S_2 产生的句型发生了串道，即 S_1 产生的句型可能将 S_2 产生的句型作为下文，S_2 产生的句型可能将 S_1 产生的句型作为上文。

因为两个文法的非终结符不允许有相同的符号，所以只是终结符导致了串道。

解决串道的问题，必须区别两个文法的终结符，将 Σ 复制为 Σ' 和 Σ''

$$\Sigma'=\{x'\,|x\in\Sigma\}$$

$$\Sigma''=\{x''\ |x\in\Sigma\}$$

将 P_1 中的 x 用 x′代替，得到 P′，将 P_2 中的 x 用 x″代替，得到 P″；使得推导过程中，能够将 G_1 和 G_2 的终结符进行区分。最终，还需要将 x′和 x″还原为对应的终结符。

构造文法

$$G_7=(\Sigma,V\cup\Sigma'\cup\Sigma'',S,P_7)$$

其中，

$$P_7=\{S\rightarrow S_1S_2\}\cup P'\cup P''$$
$$\cup\{x'\rightarrow x|x\in\Sigma\}$$
$$\cup\{x''\rightarrow x|x\in\Sigma\}$$

G_7 利用

$$S=>S_1S_2=>^+w_1'\ w_2'$$
$$=>^+w_1\ w_2'$$
$$=>^+w_1w_2\in L_1L_2$$

得到 L_1 和 L_2 的连接，即

$$L_7=L_1L_2$$

通过区分两个文法的终结符，解决了串道问题。

例 4-2 串道的解决。

对于例 4-1 的文法 G_1 和 G_2，P_7 为

$$S\rightarrow S_1S_2$$
$$S_1\rightarrow a'$$
$$S_2\rightarrow a''S_2$$
$$a''S_2\rightarrow b''c''$$
$$a'\rightarrow b$$
$$a''\rightarrow a$$
$$b''\rightarrow b$$
$$c''\rightarrow c$$

G_7 利用推导

$$S=>S_1S_2$$
$$=>a'S_2 \quad \text{//不能够使用 } a''S_2\rightarrow b''c'' \text{ 继续推导}$$
$$=>a\ 'a''S_2$$

$$=>^{+} a\,'a''^{*}b''c''$$

$$=>^{+} a^{+}bc$$

产生 L_1 和 L_2 的连接语言 $a^{+}bc$。

4.3.4 克林闭包运算有效封闭

克林闭包运算为一元运算。克林闭包运算必需考虑 ε 句子的产生和句子任意次的连接。可以考虑增加新的开始符号 S 和产生式

$$S\rightarrow \varepsilon \mid SS_1$$

产生空句子和语言 L_1 句子的任意次的连接。

但由于 S 出现在产生式右边，不满足封闭性原则，也可能导致产生多余的其他串，可以增加新的非终结符 S′ 以解决问题。

新增加的产生式改写为

$$S\rightarrow \varepsilon \mid S'$$

$$S'\rightarrow S_1 \mid S_1S'$$

则 S 只推导出 ε 和 $S_1^{n}(n\geq 1)$。

构造文法

$$G_8=(\Sigma, V_1\cup\{S,S'\}, S, P_8)$$

其中，

$$P_8=\{S\rightarrow \varepsilon \mid S'\}\cup\{S'\rightarrow S_1 \mid S_1S'\}\cup P_1$$

若 G_1 是 2 型文法，则 G_8 也是 2 型文法；且

$$L(G_8)=L_1^{*}$$

结论，2 型语言对克林闭包封闭。

若 G_1 是 0、1 型文法，文法 G_8 也可能会有同样的串道问题。由于

$$S=>^{+} S_1\cdots S_1S_1\cdots S_1$$

每个 S_1 只能单独按照 P_1 的产生式出 L_1 的句子，而任意连续的 2 个 S_1 产生的句型可能互相为上下文，从而产生串道。

为避免串道，首先将Σ复制为Σ′和Σ″，并构造 P′和 P″；将 S_1 改写 S′，构造文法

$$G'=(\Sigma, V_1\cup\Sigma'\cup S'\}-\{S_1\}, S', P')$$

将 S_1 改写 S″，构造文法

$$G''=(\Sigma, V_1\cup\Sigma''\cup\{S''\}-\{S_1\}, S'', P'')$$

构造文法

$$G_9=(\Sigma,V_1\cup\Sigma'\cup\Sigma''\cup\{S,S'',S_1,S_2\},S,P_9)$$

其中，

$$P_9=\{S\to\varepsilon|S_1|S_2\}$$
$$\cup\{S_1\to S'|S'S_2\}$$
$$\cup\{S_2\to S''|S''S_1\}$$
$$\cup P'\cup P''$$
$$\cup\{x'\to x|x\in\Sigma\}\cup\{x''\to x|x\in\Sigma\}$$

为避免文法 G_1 自己的串道现象，需要 S′ 和 S″交替出现，使得

$$S\Rightarrow^+S_1\Rightarrow^+S'S''S'S''\cdots S'S''$$

或

$$S\Rightarrow^+S_1\Rightarrow^+S'S''S'S''\cdots S'$$

以及

$$S\Rightarrow^+S_2\Rightarrow^+S''S'S''S'\cdots S''S'$$

或

$$S\Rightarrow^+S_2\Rightarrow^+S''S'S''S'\cdots S''$$

则 S_1 的连续出现，变换为 S′ 和 S″ 交替出现，每个 S′ 或 S″只能单独按照 P′或 P″的产生式继续推导下去，避免了串道。

S′和 S″分别产生以Σ′和Σ″为字母表的语言（它们的句子的结构完全等同于语言 L_1 的句子结构），再经过还原，得到 L_1^*，即

$$L(G_9)=L_1^*$$

结论，0 型和 1 型语言对克林闭包有效封闭。

如果 G_1 是 3 型文法，G_8 不是 3 型文法，增加新的开始符号 S 和

$$S\to\varepsilon$$

产生空串 ε（若在 P_1 中有 $S_1\to\varepsilon$，则删除 $S_1\to\varepsilon$），增加

$$S\to r$$

其中，

$$S_1\to r\in P_1$$

以便开始推导(r=wB 或 r=w)。

对于每个形如 A→w 的产生式，增加

$$A\to wS_1$$（不删除 A→w）

从 S 开始，可以推导出句型

$$r_1r_2\cdots r_kA$$

其中，

$$r_1,r_2,\cdots,r_k \in L_1$$

可以在推导出

$$r_1r_2\cdots r_kw$$

时停止，也可以从

$$r_1r_2\cdots r_kwS_1$$

开始推导出另一个句子， 直至 L_1^*。

G_1 是 3 型文法，构造 3 型文法

$$G_{10}=(\Sigma,V_1\cup\{S\},S,P_{10})$$

其中，

$$P_{10}=\{S\rightarrow\varepsilon\}\cup\{P_1-\{S_1\rightarrow\varepsilon\}\}$$
$$\cup\{S\rightarrow r\mid S_1\rightarrow r\in P_1\}$$
$$\cup\{A\rightarrow wS_1\mid A\rightarrow w\in P_1\}$$

则

$$L(G_{10})=L_1^*$$

结论，3 型语言对克林闭包有效封闭。

总之，不论字母表

$$\Sigma_1\cap\Sigma_2=\Phi$$

或

$$\Sigma_1\cap\Sigma_2\neq\Phi$$

0 型、1 型、2 型和 3 型语言类对联合、连接和克林闭包运算是有效封闭的。

对于正闭包运算的有效封闭性问题，可以参考对克林闭包有效性封闭的分析，只是，正闭包运算不需要考虑 ε 句子的产生。

一个复杂语言可能是由多个简单语言经过多次的联合、连接或闭包（包括克林闭包和正闭包）产生的，可以利用联合、连接和克林闭包运算的有效封闭性最终构造出产生复杂语言的文法。

4.4 正则表达式的形成

对于正则语言，可以利用该方法得到正则表达式。

正则语言 L_1 和 L_2 的正则表达式分别为 R_1 和 R_2，若

$$L=L_1 \cup L_2$$

则语言 L 的正则表达式为

$$(R_1)+(R_2)$$

若

$$L=L_1L_2$$

则语言 L 的正则表达式为

$$(R_1)(R_2)$$

若

$$L=L_1^*$$

则语言 L 的正则表达式为

$$(R_1)^*$$

若

$$L=L_1^+$$

则语言 L 的正则表达式为

$$(R_1)^+$$

一个复杂语言可能是由多个简单语言经过多次的联合、连接或闭包（包括克林闭包和正闭包）产生的，可以利用联合、连接和克林闭包运算的有效封闭性最终产生语言完整的正则表达式。

4.5 上下文无关语言对上下文无关置换有效封闭

设 X 和 Y 是两个字母表，映射

$$g: X^* \rightarrow Y^*$$

若

$$g(\varepsilon)=\varepsilon$$

且对 $n \geq 1$

$$g(x_1x_2\cdots x_n)=g(x_1)g(x_2)\cdots g(x_n)$$

其中，

$$x_i \in X$$

$$g(x_i)=y \in Y^*$$

或

$$g(x_i)=\{y_1, y_2, \cdots\}$$

其中，$y_i \in Y^*$

则 g 是一个上下文无关置换[84]。

若 L 是字母表 X 上的一个语言，则

$$g(L)=\cup g(w)$$

其中，$w \in L$。

上下文无关文法

$$G=(X,V,S,P)$$

产生上下文无关语言 L，g 是一个上下文无关置换

$$g(x)=L_x$$

其中，$x \in X$。

将 X 复制为 X′

$$X'=\{x' \mid x \in X\}$$

将 P 中的每个产生式的右边的终结符 x 替换为 x′，得到 P′。

文法 G 改造为

$$G'=(X',V \cup X',S,P')$$

文法 G 产生的语言基于字母表 X，文法 G′ 产生的语言是基于字母表 X′ 的。文法 G′ 产生语言的句子与文法 G 产生语言的句子结构完全一致（仅字母表不一致）。

对于每个 x′，增加一组上下文无关的产生式，使得

$$x' \Rightarrow^{+} L_x$$

得到 P″

构造上下文无关文法

$$G''=(Y,V \cup X',S,P'')$$

文法 G 产生

$$x_1x_2\cdots x_n$$

文法 G″先利用 P′产生

$$x_1'x_2'\cdots x_n'$$

再利用新增加的产生式得到

$$L_{x_1} L_{x_2} \cdots L_{x_n}$$

文法 G″ 产生语言 g(L)，也是上下文无关的语言。

例 4-3 上下文无关置换的封闭性。

上下文无关文法 G 为

$$S \rightarrow aSb$$

$$S\rightarrow ab$$

产生语言

$$a^nb^n$$

上下文无关置换为

$$g(a)=0^+=L_a$$

$$g(b)=101^*=L_b$$

构造文法 G′

$$S\rightarrow a'S\,b'$$

$$S\rightarrow a'b'$$

产生

$$a'^nb'^n$$

增加产生式

$$a'\rightarrow 0|0a'$$

产生

$$0^+$$

增加产生式

$$b'\rightarrow 10|10A$$

$$A\rightarrow 1|1A$$

产生

$$101^*$$

构造文法 G″

$$S\rightarrow a'S\,b'$$

$$S\rightarrow a'b'$$

$$a'\rightarrow 0|0\,a'$$

$$b'\rightarrow 10|10A$$

$$A\rightarrow 1|1\,A$$

产生语言

$$0^+(101^*)^+$$

4.6 利用语言运算的封闭性构造自动机

若语言 L 是由两个简单语言 L_1 和 L_2 通过联合或连接运算生成的；或者由一

个简单语言 L_1 通过克林闭包（或正闭包）运算生成的，就可以将 L 分解为简单语言 L_1 和 L_2，分别构造出接收 L_1 和/或 L_2 的自动机，基于四类语言对联合、连接和克林闭包运算的有效封闭性原理，可以构造出接收语言 L 的自动机。

若语言是由多个简单语言经过多次的联合、连接或闭包（包括克林闭包和正闭包）产生的，可以先构造接收简单语言的自动机，最终得到接收复杂语言的自动机。

4.6.1 带空移动的有限状态自动机 ε-NFA 的构造

设 L_1 和 L_2 是任意两个正则语言，接收它们的 ε-NFA 分别为 M_1 和 M_2，其中

$$\varepsilon\text{-NFA}M_1=(Q_1,\Sigma_1,\delta_1,q_1,\{f_1\})$$

$$\varepsilon\text{-NFA}M_2=(Q_2,\Sigma_2,\delta_2,q_2,\{f_2\})$$

假设

$$Q_1\cap Q_2=\Phi$$

1 ε-NFA 接收基于联合运算产生的语言

构造 ε-NFA 为

$$\varepsilon\text{-NFA}=(Q_1\cup Q_2\cup\{q_0,f_0\},\Sigma_1\cup\Sigma_2,\delta,q_0,\{f_0\})$$

其中，状态转换函数 δ 为

(1) 对于 ε-NFA 的开始状态 q_0，定义

$$\delta(q_0,\varepsilon)=\{q_1\}$$

$$\delta(q_0,\varepsilon)=\{q_2\}$$

(2)对于 $q\in Q_1$，$x\in\Sigma_1\cup\{\varepsilon\}$，定义

$$\delta(q,x)=\delta_1(q,x)$$

(3)对于 $q\in Q_2$，$y\in\Sigma_2\cup\{\varepsilon\}$，定义

$$\delta(q,y)=\delta_2(q,y)$$

(4) 对于 Q_1 中接收状态 f_1，对于 Q_2 中的接收状态 f_2，定义

$$\delta(f_1,\varepsilon)=\{f_0\}$$

$$\delta(f_2,\varepsilon)=\{f_0\}$$

对于 ε -NFA，可以形象地使用状态图 4-1 表示。

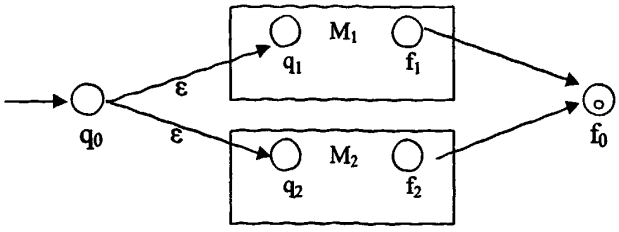

图 4-1 接收联合运算语言的 ε -NFA

ε -NFA 增加 4 个 ε 动作；从 ε -NFA 的开始状态 q_0 出发，通过

$$\delta(q_0,\varepsilon)=\{q_1\}$$

或

$$\delta(q_0,\varepsilon)=\{q_2\}$$

进入 q_1 或 q_2，使用 δ_1 或 δ_2 对应的 δ 函数，可以到达 f_1 或 f_2，通过

$$\delta(f_1,\varepsilon)=\{f_0\}$$

或

$$\delta(f_2,\varepsilon)=\{f_0\}$$

进入 ε -NFA 的接收状态 f_0，实现了联合。

$L(\varepsilon\text{-NFA})=L_1\cup L_2$ 。

2 ε-NFA 接收基于连接运算产生的语言

构造 ε-NFA 为

$$\varepsilon\text{-NFA}=(Q_1\cup Q_2,\Sigma_1\cup\Sigma_2,\delta,q_1,\{f_2\})$$

其中，状态转换函数 δ 为

(1)对于 $q\in Q_1-\{f_1\}$，$x\in\Sigma_1\cup\{\varepsilon\}$，定义

$$\delta(q,x)=\delta_1(q,x)$$

$$\delta(f_1,\varepsilon)=\{q_2\}$$

(2)对于 $q\in Q_2-\{f_2\}$，$y\in\Sigma_2\cup\{\varepsilon\}$，定义

$$\delta(q,y)=\delta_2(q,y)$$

对于 ε -NFA，可以形象地使用状态图 4-2 表示。

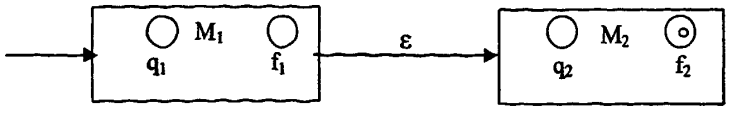

图 4-2 接收连接运算语言的 ε -NFA

ε -NFA 增加 1 个 ε 动作（以便进行连接）；从 ε -NFA 的开始状态 q_1（M_1 的开

始状态作为 ε -NFA 的开始状态）出发，使用 δ_1 对应的 δ 函数，可以到达 f_1（接收语言 L_1 的句子），通过

$$\delta(f_1,\varepsilon)=\{q_2\}$$

进入 q_2，使用 δ_2 对应的 δ 函数，可以到达 M_2 的接收状态 f_2（f_2 也是 ε -NFA 的接收状态），接收语言 L_2 的句子，，实现了连接。

$L(\varepsilon\text{-NFA})= L_1L_2$ 。

3 ε-NFA 接收基于迭代运算产生的语言

构造 ε-NFA 为

$$\varepsilon\text{-NFA} =(Q_1 \cup \{q_0,f_0\},\Sigma_1,\delta,q_0,\{f_0\})$$

其中，状态转换函数 δ 为

(1) 对于 ε-NFA 的开始状态 q_0，定义

$$\delta(q_0,\varepsilon)=\{q_1\}$$

$$\delta(q_0,\varepsilon)=\{f_0\}$$

(2) $q\in Q_1-\{f_1\}$，$x\in \Sigma_1 \cup \{\varepsilon\}$，定义

$$\delta(q,x)=\delta_1(q,x)$$

(3) 对于 f_1，定义

$$\delta(f_1,\varepsilon)=\{q_1,f_0\}$$

对于构造出的 ε-NFA，可以形象地使用状态图 4-3 表示。

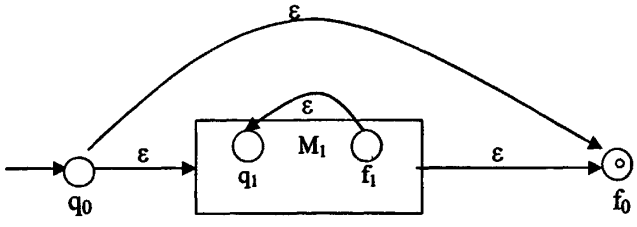

图 4-3 接收迭代运算语言的 ε -NFA

ε -NFA 增加 4 个 ε 动作；从 q_0 出发，可以通过

$$\delta(q_0,\varepsilon)=f_0$$

直接到达 ε -NFA 的接收状态 f_0（接收空句子 ε）；或者通过

$$\delta(q_0,\varepsilon)=\{q_1\}$$

进入 q_1，使用 δ_1 对应的 δ 函数，到达 M_1 的接收状态 f_1，此时，可以通过

$$\delta(f_1,\varepsilon)=\{ f_0,q_1\}$$

或者到达 ε -NFA 的接收状态 f_0，或者将状态转换为 q_1，从而进行迭代。

$L(\varepsilon\text{-NFA})= L_1^*$ 。

对于正闭包运算，只需要删除

$$\delta(q_0,\varepsilon)=\{f_0\}$$

即可。

4.6.2 下推自动机 PDA 的构造

L_1 和 L_2 是任意的两个上下文无关语言，接收它们的 PDA 分别为 M_1 和 M_2，其中

$$PDAM_1=(Q_1,\Sigma_1,\Gamma_1,\delta_1,q_1,Z_{01},\Phi)$$

$$PDAM_2=(Q_2,\Sigma_2,\Gamma_2,\delta_2,q_2,Z_{02},\Phi)$$

假设，PDA 以空栈的方式接收语言，且

$$Q_1\cap Q_2=\Phi$$

$$\Gamma_1\cap\Gamma_2=\Phi$$

1 PDA 接收基于联合运算产生的语言

构造 PDA 为

$$PDA=(Q_1\cup Q_2\cup\{q_0\},\Sigma_1\cup\Sigma_2,\Gamma_1\cup\Gamma_1\cup Z_0,\delta,q_0,Z_0,\Phi)$$

其中，状态转换函数 δ 为

(1) 对于开始状态 q_0，定义

$$\delta(q_0,\varepsilon,Z_0)=(q_1,Z_{01})$$

$$\delta(q_0,\varepsilon,Z_0)=(q_2,Z_{02})$$

(2) 对于 $q\in Q_1$，$x\in\Sigma_1\cup\{\varepsilon\}$ ，定义

$$\delta(q,x,Z)=\delta_1(q,x,Z)$$

其中，Z 为 M_1 任意的栈顶元素。

(3) 对于 $q\in Q_2$，$y\in\Sigma_2\cup\{\varepsilon\}$ ，定义

$$\delta(q,y,Z)=\delta_2(q,y,Z)$$

其中，Z 为 M_2 任意的栈顶元素。

PDA 增加 2 个 ε 动作；从 PDA 的开始状态 q_0 开始，通过

$$\delta(q_0,\varepsilon,Z_0)=(q_1,Z_{01})$$

或

$$\delta(q_0,\varepsilon,Z_0)=(q_2,Z_{02})$$

分别可以到达 q_1 或 q_2，并将栈底符号变换为 Z_{01} 或 Z_{02}，使用 δ_1 或 δ_2 对应的 δ 函数，最终，到达 M_1 或 M_2 的空栈接收格局，实现了联合。

$L(PDA)= L_1 \cup L_2$ 。

2 PDA 接收基于连接运算产生的语言

构造 PDA 为

$$PDA=(Q_1 \cup Q_2 \cup \{q_0\}, \Sigma_1 \cup \Sigma_2, \Gamma_1 \cup \Gamma_1, \delta, q_1, Z_{01}, \Phi)$$

其中，状态转换函数 δ 为

(1) 对于 $q \in Q_1$，$x \in \Sigma_1 \cup \{\varepsilon\}$，定义

$$\delta(q,x,Z)=\delta_1(q,x,Z)$$

(2) 将 M_1 最终的

$$\delta_1(q,\varepsilon,Z_{01})=(q', \varepsilon)$$

修改为

$$\delta(q,\varepsilon,Z_{01})=(q_2, Z_{02})$$

(3) 对于 $q \in Q_2$，$b \in \Sigma_2 \cup \{\varepsilon\}$，定义

$$\delta(q,b,Z)=\delta_2(q,b,Z)$$

PDA 从 q_1 出发，先接收语言 L_1 的句子，通过

$$\delta(q,\varepsilon,Z_{01})=(q_2, Z_{02})$$

到达 q_2(栈底变换为 Z_{02})，使用 δ_2 对应的 δ 函数，接收 L_2 的句子，实现了连接。

$L(PDA)= L_1 L_2$ 。

3 PDA 接收基于迭代运算产生的语言

构造 PDA 为

$$PDA=(Q_1 \cup Q_2, \Sigma_1, \Gamma_1, \delta, q_1, Z_{01}, \Phi)$$

其中，状态转换函数 δ 为

(1) 对于 PDA 开始状态 q_1 ，定义

$$\delta(q_1,\varepsilon,Z_{01})=(q_1,\varepsilon)$$

(2) 对于 $q \in Q_1$，$x \in \Sigma_1 \cup \{\varepsilon\}$，定义

$$\delta(q,x,Z)=\delta_1(q,x,Z)$$

(3) 对于 M_1 最终的

$$\delta_1(q,\varepsilon,Z_{01})=(q', \varepsilon)$$

相应地增加

$$\delta(q,\varepsilon,Z_{01})=(q_1, Z_{01})$$

PDA 从开始状态 q_1 出发，使用

$$\delta(q_1,\varepsilon,Z_{01})=(q_1,\varepsilon)$$

接收空句子 ε；或者使用 δ_1 对应的 δ 函数，接收语言 L_1 的句子；或者通过

$$\delta(q,\varepsilon,Z_{01})=(q_1, Z_{01})$$

可以到达 q_1，从而进行迭代。

L(PDA)= L_1^*。

对于正闭包运算，只需要删除

$$\delta(q_1,\varepsilon,Z_{01})=(q_1,\varepsilon)$$

即可。

4.6.3 图灵机 TuringM 的构造

若语言 L_1 和 L_2 是任意两个上下文相关语言，接收它们的 TuringM 分别为 M_1 和 M_2，其中

$$TuringM_1=(Q_1,\Sigma_1, q_1,q_{\alpha 1},\delta_1)$$

$$TuringM_2=(Q_2,\Sigma_2, q_2,q_{\alpha 2},\delta_2)$$

假设

$$Q_1\cap Q_2=\Phi$$

$$\Sigma_1\cap\Sigma_2=\Phi$$

1 TuringM 接收基于联合运算产生的语言

将 TuringM 输入带的第 2 个单元设置为 B。

设置

$$TuringM=(Q_1\cup Q_2\cup\{q_0\},\Sigma_1\cup\Sigma_2,q_0,q_{\alpha 1}\cup q_{\alpha 2},\delta)$$

其中，δ 为

(1) 对于 TuringM 开始状态 q_0，L_1 和 L_2 的句子的第一个字母 x 和 y，定义

$$\delta(q_0,x)=(q_0,x,L)$$

$$\delta(q_0,y)=(q_0,y,L)$$

$$\delta(q_0,B)=(q_1,x,R)$$

$$\delta(q_0,B)=(q_2,y,R)$$

(2) 对于 $q\in Q_1$，$a\in$(Σ_1 的增广集合)，定义

$$\delta(q,a)=\delta_1(q,a)$$

(3) 对于 $q\in Q_2$，$b\in$(Σ_2 的增广集合)，定义

$$\delta(q,b)=\delta_2(q,b)$$

TuringM 从 q_0 出发，通过

$$\delta(q_0,B)=(q_1,x,R)$$

或

$$\delta(q_0,B)=(q_2,y,R)$$

到达 q_1 或 q_2，使用 δ_1 或 δ_2 对应的 δ 函数，到达 $q_{\alpha 1}$ 或 $q_{\alpha 2}$，实现了联合。

$L(TuringM)=L_1 \cup L_2$ 。

2 TuringM 接收基于连接运算产生的语言

为区别 L_1 和 L_2 的句子，在句子之间增加一个 B，作为分隔符号。

构造

$$TuringM=(Q_1 \cup Q_2 \cup \{q_3\}, \Sigma_1 \cup \Sigma_2, q_1, q_{\alpha 2}, \delta)$$

其中，状态转换函数 δ 为

(1) 对于 $q \in Q_1$，$x \in$(Σ_1 的增广集合)，定义

$$\delta(q,x)=\delta_1(q,x)$$

(2) 对于 a，$b \in$(Σ_1 的增广集合)，将 M_1 最终的

$$\delta_1(q,a)=(q_{\alpha 1},b,D)$$

转换为

$$\delta(q,a)=(q_\beta,b,D)$$

$$\delta(q_\beta,x)=(q_\beta,x,R)$$

$$\delta(q_\beta,B)=(q_2,B,R)$$

(3) 对于 $q \in Q_2$，$y \in$(Σ_2 的增广集合)，定义

$$\delta(q,y)=\delta_2(q,y)$$

TuringM 从 q_1 开始，接收语言 L_1 的句子后，可以使用

$$\delta(q,a)=(q_3,b,D)$$

到达 q_β，通过

$$\delta(q_\beta,x)=(q_\beta,x,R)$$

$$\delta(q_\beta,B)=(q_2,B,R)$$

到达 q_2，接收 L_2 的句子，实现了连接。

$L(TuringM)=L_1L_2$ 。

3 TuringM 接收基于迭代运算产生的语言

构造

$$TuringM=(Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, q_1, q_{\alpha 1}, \delta)$$

其中，状态转换函数 δ 为

(1) 对于 TuringM 的开始状态 q_1，定义

$$\delta(q_1,B)=(q_{\alpha 1},B,R)$$

(2) 对于 $q\in Q_1$，$a\in(\Sigma_1$ 的增广集合)，定义

$$\delta(q,a)=\delta_1(q,a)$$

(3) 对于 a，$b\in(\Sigma_1$ 的增广集合)，将 M_1 的最后的状态转换函数

$$\delta_1(q,a)=(q_{\alpha 1},b,D)$$

增加

$$\delta(q,a)=(q_\beta,b,D)$$

$$\delta(q_\beta,x)=(q_\beta,x,R)$$

$$\delta(q_\beta,B)=(q_1,B,R)$$

TuringM 使用

$$\delta(q_1,B)=(q_{\alpha 1},B,R)$$

可以接收空句子 ε；或者从 q_1 出发，接收 L_1 的句子后，停机，也可以通过

$$\delta(q,a)=(q_\beta,b,D)$$

$$\delta(q_\beta,x)=(q_\beta,x,R)$$

$$\delta(q_\beta,B)=(q_1,B,R)$$

到达 q_1，从而进行迭代。

$L(TuringM)=L_1^*$。

对于正闭包运算，只需要删除

$$\delta(q_1,B)=(q_{\alpha 1},B,R)$$

即可。

4.7 本章小结

复杂语言通常可以分解为多个简单的同类语言经过联合、连接和闭包（包括克林闭包和正闭包）运算产生。

本章从形式语言的角度完整地证明了 Chomshy 四类语言对于联合、连接、克林闭包运算是有效封闭性的。给出了构造产生复杂语言文法的通用性方法。针对上下文相关语言和短语结构语言对于连接、克林闭包运算可能产生的串道问题，提出了对应的解决办法。讨论了上下文无关语言对于上下文无关置换运算的有效封闭性问题。

0 型、1 型、2 型和 3 型语言类对语言的其他运算如交、补运算的封闭性不在本章讨论的范围中。

基于语言运算有效封闭的特点，分析构造复杂自动机的基本方法。如果语言

是由多个简单语言经过多次的联合、连接或闭包产生的，可以先构造接收简单语言的自动机，最终得到接收复杂语言的自动机。

第五章 利用等价类构造确定的有限状态自动机

一类特定语言是由任意字母表上的K进制的数字串构成，而数字串（当作数字）能够整除任意非负整数N（或对N的余数为M，N≥2，N≥M≥0），蒋宗礼，吴哲辉针对指定字母表上的语言，提出了基于等价类构造确定的有限状态自动机DFA的方法，但对任意的K、N和M，缺乏通用性的考虑。

本章研究根据等价类构造接收该类语言的确定的有限状态自动机的通用方法。该方法可以针对任意的字母表和某种进制的数字串构成的语言，并满足该语言中的所有句子（当作整数）能够整除任意正整数N（或对N的余数为任意的M），同时，构造出的有限状态自动机是极小化的，请见参考文献[88]。

5.1 等价关系和等价类

集合A和B的笛卡儿乘积使用A×B表示，

$$A\times B=\{(a,b)\mid a\in A \text{ 且} b\in B\}$$

设A和B为任意的两个集合，则A到B的二元关系是笛卡儿乘积A×B的任意一个子集；若A=B，则称为A上的关系[89]。

设R是集合A上的关系，则

若对A中的任一元素a，都有aRa，则称R为自反的关系。

若对A中的任何元素a和b，从aRb能够推断出bRa，则称R为对称的关系。

若对A中的任何元素a，b和c，从aRb和bRc能够推断出aRc，则称R为传递的关系。

如果关系R同时是自反的、对称的和传递的关系，则关系R为等价关系。

集合A上的一个等价关系R可以将集合A划分为若干个互不相交的子集，每个子集称为一个等价类。对A中的任意元素a，使用[a]表示a的等价类，即

$$[a]=\{b\mid aRb\}$$

该等价关系R将集合A划分为等价类的数目，称为等价关系R的指数[90]。

例5-1 等价关系。

非负整数集合（记为集合N）上的模3同余关系R是等价关系，

$$R=\{(a,b)\mid a,b\in N, \text{且 } a \bmod 3 = b \bmod 3\}$$

可以将 N 划分为 3 个子集；集合

$$\{0,3,6,\cdots,3n,\cdots\}$$

是一个等价类，可以记为[0]；集合

$$\{1,4,7,\cdots,3n+1,\cdots\}$$

是一个等价类，可以记为[1]；集合

$$\{2,5,8,\cdots,3n+2,\cdots\}$$

是一个等价类，可以记为[2]。

等价关系 R 的指数为 3，而 N=[0]∪[1]∪[2]。

5.2 有限状态自动机的 set 集合

任意确定的有限状态自动机 DFA，对于状态 q∈Q；定义能将 DFA 从开始状态转换到 q 状态的所有字符串的集合为

$$set(q)=\{w|w\in\Sigma^*,\delta(q_0,w)=q\}$$

对于确定的有限状态自动机 DFA，语言 L=L（DFA），可以定义关系 R，若 x,y∈L，则

$$xRy \Leftrightarrow x\in set(q)且\ y\in set(q)$$

其中，q ∈Q。

关系 R 是集合Σ^*上的一个等价关系，利用该等价关系，可以将Σ^*划分为不多于|Q|个的等价类。

按照语言的特点可以对字母表Σ^*进行划分，每个划分相当于Σ^*上的一个等价分类，而 DFA 的每个状态实际上就对应着一个等价类。因此，利用一个状态去表示一个等价类是构造有限状态自动机的一条有效思路[1,2,39]。

对有限状态自动机，要求是极小化的。MALIK S、孙玉强、朱征宇和雷红轩等人研究了有限自动机的极小化算法[91-94]；韩光辉给出了有限自动机极小化算法的实现[95]，他们都是从已存在的有限自动机的约简角度进行极小化；而根据等价类构造的有限状态自动机，基于等价类的性质，能够从构造的角度保证有限状态自动机是极小化的。

5.3 经典问题

例 5-2 构造 DFA，识别{0,1,2}上的语言，该语言的每个句子（当作十进制整数）能够整除 3。

若一个十进制整数的各位的数字和能够整除 3，该十进制整数就能够整除 3。一个十进制整数除以 3，余数只能为 0、1 和 2（余数为 0，表示该数能够整除 3）；

使用 3 个状态分别代表已经读入的数字的和除以 3 的余数情况（即等价类）

q_0：已经读入的数字的和除以 3，余数为 0 的等价类；

q_1：已经读入的数字的和除以 3，余数为 1 的等价类；

q_2：已经读入的数字的和除以 3，余数为 2 的等价类；

扫描串 w 后，DFA 处于某个状态，读入当前数字 x，引导 DFA 到达下一状态的输入串为 wx。

1） 对于状态 q_0

读入 0，w0 的各位数字和属于 q_0 对应的等价类，应该继续保持 q_0 状态；

读入 1，w1 的各位数字和属于 q_1 对应的等价类，应该到达 q_1 状态；

读入 2，w2 的各位数字和属于 q_2 对应的等价类应该到达 q_2 状态；

2） 对于状态 q_1

读入 0，w0 的各位数字和属于 q_1 对应的等价类，应该保持 q_1 状态；

读入 1，w1 的各位数字和属于 q_2 对应的等价类，应该到达 q_2 状态；

读入 2，w2 的各位数字和属于 q_0 对应的等价类，应该到达 q_0 状态；

3） 对于状态 q_2：

读入 0，w0 的各位数字和属于 q_2 对应的等价类，应该保持 q_2 状态；

读入 1，w1 的各位数字和属于 q_0 对应的等价类，应该到达 q_0 状态；

读入 2，w2 的各位数字和属于 q_1 对应的等价类，应该到达 q_1 状态；

为避免接收空串 ε 和以 0 开始的数字串，还需要设置开始状态 q_s，并定义

$$\delta(q_s,1)=q_1$$

$$\delta(q_s,2)=q_2$$

DFA 的状态图如图 5-1 所示。

如果要求对 3 的余数为 1 或 2，则修改 q_0 为非接收状态，而将 q_1 或 q_2 设置为接收状态即可。

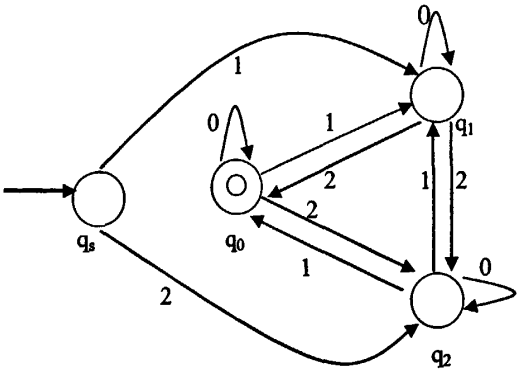

图 5-1 识别{0,1,2,}上能整除 3 的数字串形成语言的 DFA（省略关于 q_s 的陷阱状态）

例 5-3 构造 DFA，识别{0，1，2，4，5，6，7，8，9}上的语言，该语言的每个句子（当作十进制整数）能够整除 3。

一个十进制整数除以 3，余数只能是 0、1 和 2，所以，仍然只使用 3 个状态分别代表已经读入的数字的和除以 3 的余数情况（即等价类）。

该 DFA 的状态图类似于图 5-1 的状态图，只是将标记 0 改为 0、3、6、9，将标记 1 改为 1、4、7，将标记 2 改为 2、5、8。

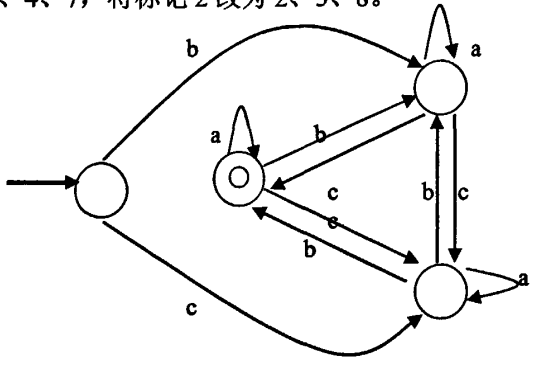

图 5-2 识别{0,1,…,9}上能整除 3 的数字串形成语言的 DFA（省略关于 q_s 的陷阱状态）

其中：a-0,3,6,9;b=1,4,7;c=2,5,8

5.4 通用性考虑

对于二进制整数，或十进制整数（甚至任意 base 进制整数），要求能整除任意的非负整数 N（N≥3），没有统一的数学规律。

一类语言由任意字母表上的某种进制的数字串构成，要求该语言中的所有数字串能够整除正整数 N（或对 N 的余数为 M），构造有限状态自动机接收该类语言，

该问题可以描述为：构造 DFA，识别字母表 $\Sigma=\{x_1,x_2,x_3,\cdots,x_{base-1}\}$上的语言，该语言的每个字符串（挡成 base 进制数）能整除 N（或对 N 的余数为 M）。

将 base 进制的数转换为十进制数，除以 N 的余数只能为 0、1、2、3、…和 N-1，可以使用 N 个状态分别代表已经读入的串（挡成 base 进制数）除以 N 的不同余数的等价类：

q_0：已经读入的串（挡成 base 进制数）除以 N 余数为 0 的等价类；该类数可以记为$(N*n+0)_{10}$；

q_1：已经读入的串（挡成 base 进制数）除以 N 余数为 1 的等价类；该类数可以记为$(N*n+1)_{10}$；

q_2：已经读入的串（挡成 base 进制数）除以 N 余数为 2 的等价类；该类数可以记为$(N*n+2)_{10}$；

…

q_{N-1}：已经读入的串（挡成 base 进制数）除以 N 余数为 N-1 的等价类；该类数可以记为$(N*n+N-1)_{10}$；

问题的实质是，已知 $q_i(i=0,1,2,\cdots)$ 和 $x=0,1,2,\cdots$，对于状态转换函数

$$\delta(q_i,x)=q_j$$

应该如何确定 j？

状态 q_i 对应已经读入的串 w（挡成 base 进制数）除以 N，余数为 i 的等价类；该类数十进制可以记为 N*n+i；继续读入的字符为 x，则串 wx（挡成 base 进制数）的十进制数为

$$\begin{aligned}&base*(N*n+i)+x\\&=base*N*n+base*i+x\end{aligned}$$

除以 N 的余数就是 base*i+x 对于 N 的余数；若余数为 j，则状态就应该从 q_i 变换为 q_j。

为了不接收空句子 ε，需要一个开始状态 q_s，当读入 x 时，直接进入对应状态 $q_x(x\neq 0)$。

例 5-4 构造 DFA，识别{1,2,3}上的语言，该语言的每个句子（当作十进制整数），能整除 4。

除以 4 的余数只能为 0、1、2 和 3，使用 4 个状态分别代表已经读入的数除以 4 的不同的余数的等价类：

q_0：已经读入的数除以 4，余数为 0 的输入串的等价类；

q_1：已经读入的数除以 4，余数为 1 的输入串的等价类；

q_2：已经读入的数除以 4，余数为 2 的输入串的等价类；

q_3：已经读入的数除以 4，余数为 3 的输入串的等价类；

因为不能接收空串，所以，还需要一个开始状态 q_S。

设 w 是当前已经读入的输入串；继续读入字母 x，则引导 DFA 到达下一状态的输入串为 wx。

q_S：读入 1，进入状态 q_1；读入 2 时，进入状态 q_2；读入 3，进入状态 q_3；

q_0：能引导 DFA 到达 q_0 状态的 w 除以 4 的余数为 0，$(w)_{10}=4n+0$；

读入 1，则

$$(w1)_{10}=base*i+x=10*0+1=1$$

即 w1 属于 q_1 对应的等价类，应该到达 q_1 状态；

读入 2，则

$$(w2)_{10}=base*i+x=10*0+2=2$$

即 w2 属于 q_2 对应的等价类，应该到达 q_2 状态；

读入 3，则

$$(w3)_{10}=base*i+x=10*0+3=3$$

即 w3 属于 q_3 对应的等价类，应该到达 q_3 状态；

q_1：能引导 DFA 到达 q_1 状态的 w 除以 4 余数为 1，因此，$(w)_{10}=4n+1$；

读入 1，则

$$(w1)_{10}=base*i+x=10*1+1=11$$

即 w1 属于 q_3 对应的等价类，应该到达 q_3 状态；

读入 2，则

$$(w2)_{10}=base*i+x=10*1+2=12$$

即 w2 属于 q_0 对应的等价类，应该到达 q_0 状态；

读入 3，则

$$(w3)_{10}=base*i+x=10*1+3=13$$

即 w3 属于 q_1 对应的等价类，应该到达 q_1 状态；

q_2：能引导 DFA 到达 q_2 状态的 w 除以 4 余数为 2，因此，$(w)_{10}=4n+2$；

读入 1，则

$$(w1)_{10}=base*i+x=10*2+1=21$$

即 w1 属于 q_1 对应的等价类，应该到达 q_1 状态；

读入 2，则

$$(w2)_{10}=base*i+x=10*2+2=22$$

即 w2 属于 q_2 对应的等价类，应该到达 q_2 状态；

读入 3，则

$$(w3)_{10}=base*i+x=10*2+3=23$$

即 w3 属于 q_3 对应的等价类，应该到达 q_3 态；

q_3：能引导 DFA 到达 q_3 状态的 w 除以 4 余数为 3，因此，$(w)_{10}=4n+3$；

读入 1，则

$$(w1)_{10}=base*i+x=10*3+1=31$$

即 w1 属于 q_3 对应的等价类，应该到达 q_3 状态；

读入 2，则

$$(w2)_{10}=base*i+x=10*3+2=32$$

即 w2 属于 q_0 对应的等价类，应该到达 q_0 状态；

读入 3，则

$$(w3)_{10}=base*i+x=10*3+3=33$$

即 w3 属于 q_1 对应的等价类，应该到达 q_1 状态。

完成 DFA 的构造，状态转换图略。

例 5-5 构造 DFA，识别{0,1,2}上的语言，该语言的每个句子（当作三进制整数），能整除 5。

除以 5 的余数只能为 0、1、2、3 和 4，使用 5 个状态分别代表已经读入的数除以 5 的不同的余数的等价类：

q_0：已经读入的数除以 5，余数为 0 的输入串的等价类；

q_1：已经读入的数除以 5，余数为 1 的输入串的等价类；

q_2：已经读入的数除以 5，余数为 2 的输入串的等价类；

q_3：已经读入的数除以 5，余数为 3 的输入串的等价类；

q_4：已经读入的数除以 5，余数为 4 的输入串的等价类；

因为不能接收空串，所以，还需要一个开始状态 q_S。

设 w 是当前已经读入的输入串；继续读入字母 x，则引导 DFA 到达下一状态的输入串为 wx。

q_S：读入 0，进入状态 q_0；读入 1 时，进入状态 q_1；

q_0：能引导 DFA 到达 q_0 状态的 w 除以 5 的余数为 0，$(w)_{10}=5n+0$；

读入 0，则

$$(w0)_{10}=base*i+x=2*0+0=0$$

即 w0 属于 q_0 对应的等价类，应该到达 q_0 状态；

读入 1，则

$$(w1)_{10}=base*i+x=2*0+1=1$$

即 w1 属于 q_1 对应的等价类，应该到达 q_1 状态；

读入 2，则

$$(w2)_{10}=base*i+x=2*0+2=2$$

即 w2 属于 q_2 对应的等价类，应该到达 q_2 状态；

q_1：能引导 DFA 到达 q_1 状态的 w 除以 5 的余数为 1，$(w)_{10}=5n+1$；

读入 0，则

$$(w0)_{10}=base*i+x=2*1+0=2$$

即 w0 属于 q_2 对应的等价类，应该到达 q_2 状态；

读入 1，则

$$(w1)_{10}=base*i+x=2*1+1=3$$

即 w1 属于 q_3 对应的等价类，应该到达 q_3 状态；

读入 2，则

$$(w2)_{10}=base*i+x=2*1+2=4$$

即 w2 属于 q_4 对应的等价类，应该到达 q_4 状态；

q_2：能引导 DFA 到达 q_2 状态的 w 除以 5 的余数为 2，$(w)_{10}=5n+2$；

读入 0，则

$$(w0)_{10}=base*i+x=2*2+0=4$$

即 w0 属于 q_4 对应的等价类，应该到达 q_4 状态；

读入 1，则

$$(w1)_{10}=base*i+x=2*2+1=5$$

即 w1 属于 q_0 对应的等价类，应该到达 q_0 状态；

读入 2，则

$$(w2)_{10}=base*i+x=2*2+2=6$$

即 w2 属于 q_1 对应的等价类，应该到达 q_1 状态；

q_3：能引导 DFA 到达 q_3 状态的 w 除以 5 的余数为 0，$(w)_{10}=5n+3$；

读入 0，则

$$(w0)_{10}=base*i+x=2*3+0=6$$

即 w0 属于 q_1 对应的等价类，应该到达 q_1 状态；

读入 1，则

$$(w1)_{10}=base*i+x=2*3+1=7$$

即 w1 属于 q_2 对应的等价类，应该到达 q_2 状态；

读入 2，则

$$(w2)_{10}=base*i+x=2*3+2=8$$

即 w2 属于 q_3 对应的等价类，应该到达 q_3 状态；

q_4：能引导 DFA 到达 q_4 状态的 w 除以 5 的余数为 0，$(w)_{10}=5n+4$；

读入 0，则

$$(w0)_{10}=base*i+x=2*4+0=8$$

即 w0 属于 q_3 对应的等价类，应该到达 q_3 状态；

读入 1，则

$$(w1)_{10}=base*i+x=2*4+1=9$$

即 w1 属于 q_4 对应的等价类，应该到达 q_4 状态；

读入 2，则

$$(w2)_{10}=base*i+x=2*4+2=10$$

即 w2 属于 q_0 对应的等价类，应该到达 q_0 状态；

完成 DFA 的构造，状态转换图略。

5.5 本章小结

本章提出了根据等价类构造一类有限状态自动机 DFA 的通用方法，该方法可以针对任意字母表和某种进制的数字串构成的语言，并满足该语言中的所有句子（当作 base 进制数）能够整除任意正整数 N（或对 N 的余数为任意的 M），同时，构造出的有限状态自动机是极小化的。

第六章 通用图灵机编码方案的扩展

图灵机是一个算法的实现装置。直观地，一台通用的计算机，如果不受存储空间和运行时间的限制的话，它应该能够实现所有的有效算法。按照丘奇-图灵论题，图灵机应该是现代计算机的形式化的模型。因此，应该存在一个图灵机，它可以实现对所有图灵机的模拟，也就是说，该图灵机可以实现所有有效的算法，称该图灵机为通用图灵机。

安立新将图灵机转移函数 $\delta(qi,aj)=(q_k,a_l)$编码为$(i,Unicode(a_j),k,Unicode(a_l))$,并将此编码方案应用于所设计的通用图灵机模型[96]。刘俭等基于计算机病毒领域提出了一种通用图灵机模型[97]。马龙等针对单带图灵机模型设计了一个完整的图灵机模拟系统[98]。文献[99]描述了小型的通用图灵机（基于小范围的应用领域）。

为了使通用图灵机能够模拟所有的图灵机，需要对图灵机设计一个统一的、合理的编码系统。对图灵机的编码方案可以有多种。

基于 0、1 的编码方案是简单的。使用 0、1 的组合可以表示图灵机的整个情况，也可以对图灵机的符号（除了空白符号以外的其他符号）进行编码；同时，可以使用 0、1 对图灵机的状态转换函数进行编码。实际上，通用图灵机只是改变了一般图灵机的字母表的元素和状态转换函数的表示方法，即仅仅只利用 0 和 1 来代表图灵机，使得通用图灵机可以方便地模拟任何图灵机。

6.1 编码的目的

为了使通用图灵机模拟某个图灵机 M，需要将图灵机 M 作为通用图灵机的输入信息对待。这就需要对图灵机 M 有一个统一的、合理的编码，便于通用图灵机识别该图灵机 M 的动作。如果图灵机 M 接收给定的输入串，则通用图灵机就 N 能够识别图灵机 M 和它所接收的输入串。换句话说，为了使通用图灵机模拟某个图灵机 M，需要设计一个编码系统，它在对图灵机 M 进行编码的同时，可以对该图灵机处理的句子的进行编码。当考察一个输入串是否被一个给定的图灵机 M 接收时，就可以将该图灵机 M 和该输入串作为通用图灵机的输入，由通用图灵机模拟该图灵机的运行。

对图灵机的统一编码的方案可以有多种。一种最简单的思路是：用 0、1 对图

灵机的符号进行编码。具体而言，就是用 0 对图灵机的输入上的符号（即输入带符号集合的增广集合的符号）和状态进行编码，这样，图灵机的输入带上的符号集合可以为某个字母表（可能包含多个字母），而通用图灵机的输入带上的符号集合仅仅为{0}。同时，也使用 0 对图灵机的状态转换函数进行编码。为区别符号和状态，使用 1 表示符号和状态对应的编码之间的分隔。

蒋宗礼研究了通用图灵机的编码方式，约定字母表为{0，1}；采用 0、1 组合的编码方式，对图灵机的状态转换函数进行编码[1]，但没有对图灵机的基本情况、任意的字母表和待接收的输入串进行编码。

6.2 编码方法

设图灵机

$$TuringM=(Q,\Sigma,q_0,q_\alpha,\delta)$$

其中，

$$Q=\{q_1,q_2,\cdots,q_n\}$$

$$\Sigma=\{x_1,x_2,\cdots,x_k\}$$

$$\Sigma'=\{x_1,x_2,\cdots,x_k,x_{k+1},\cdots,x_m\}$$

$$q_0=q_i$$

$$q_\alpha=q_j$$

代表任意一个图灵机。

为了使通用图灵机能够模拟该图灵机 M，将图灵机 M 进行编码。

编码由三部分构成，编码开始部分；状态转换函数编码；输入带上的符号串 w 的编码。

1 编码的开始部分

编码开始部分表示了图灵机 M 的基本情况。

图灵机 M 有 n 个状态 q_1，q_2，…，q_n ，可以使用对应的编码分别表示为

$$q_1:0$$

$$q_2:00$$

$$\cdots$$

$$q_n:0^n$$

图灵机 M 有 m 个带上符号（其中前 k 个为输入符号），可以使用编码分别表示为

x_1:0

x_2:00

…

x_k:0^k

…

x_m:0^m

则，图灵机 M 编码的开始部分为

$$0^n10^m10^k10^s10^t10^r10^u10^v$$

其中，

0^s，0^t，0^r，分别代表开始状态、接收状态和拒绝状态对应的编码；

0^u 代表输入带左端点 ⊢的编码；此处，令 u=k+1；

0^v 代表输入带空白符号 B 的编码；此处，令 v=m。

2 状态转换函数编码

同样，使用用 0，1 对图灵机的状态转换函数进行编码。

图灵机读/写头的移动方向为，R，L，N，可以使用对应的编码分别表示为

$R(D_1)$:0

$L(D_2)$:00

$N(D_3)$:000

图灵机的状态转换函数的一般形式为

$$\delta(q_i, x_j)=(q_k, x_l, D_m)$$

使用对应的编码

$$0^i10^j10^k10^l10^m$$

表示，使用

$$111\delta_111\delta_211\cdots11\delta_r111$$

表示图灵机 M 的 r 个状态转换函数编码（使用 11 进行分隔，使用 111 作为开始和结束的标记）。

3 输入带上符号串编码

图灵机 M 输入带上符号串为 $w=y_1y_2\cdots y_n$，有 m 个带上符号（其中前 k 个为输入符号），可以使用对应的编码分别表示为

y_1:0^{z_1}

y_2:0^{z_2}

…

$$q_n:0^{z_n}$$

其中，z_i 表示 y_i 的序号，0^{z_i} 表示 y_i 的编码。

图灵机 M 的输入带符号的编码为

$$0^{z_1}10^{z_2}1\cdots10^{z_n}$$

最后，得到图灵机 M 的编码和输入带上的符号串 w 的编码为

$$0^n10^m10^k10^s10^t10^r10^u10^v111\delta_111\delta_211\cdots11\delta_r111w$$

即

$$0^n10^m10^k10^s10^t10^r10^u10^v111\delta_111\delta_211\cdots11\delta_r1110^{z_1}10^{z_2}1\cdots10^{z_n}$$

6.3 图灵机编码实例

例 6-1 图灵机的编码。

存在图灵机

$$TuringM=(\{q_1,q_2,q_3,q_4\},\{a,b\},q_4,q_3,\delta)$$

其中，δ 的定义为

$$\delta(q_4,a)=(q_4,a,R)$$
$$\delta(q_4,b)=(q_1,b,R)$$
$$\delta(q_1,a)=(q_1,a,R)$$
$$\delta(q_1,b)=(q_2,b,R)$$
$$\delta(q_2,a)=(q_2,a,R)$$
$$\delta(q_2,b)=(q_3,\#,N)$$

图灵机 M 的编码的开始部分为

$$0^n10^m10^k10^s10^t10^r10^u10^v$$

其中，

状态数 n=4；

带上符号数目 m=5；

输入符号数目 k=2；

开始状态：q4；

接收状态：q3；

左端点符号 ├为带上符号的第 3 个，即 u=3；

#为带上符号的第 4 个，即 v=3；

空白符号 B 为带上符号的第 5 个；

没有拒绝状态。

则图灵机 M 的编码的开始部分就是

000010000010010000100011000100000

输入符号 a 对应代码 0；

输入符号 b 对应代码 00；

左端点符号 ⊢对应代码 000；

带上符号#对应代码 0000；

带上空白符号 B 对应代码 00000；

状态转换函数

$$\delta(q_4,a)=(q_4,a,R)$$

对应

000010100001010

状态转换函数

$$\delta(q_4,b)=(q_1,b,R)$$

对应

00001001010010

状态转换函数

$$\delta(q_1,a)=(q_1,a,R)$$

对应

010101010

状态转换函数

$$\delta(q_1,b)=(q_2,b,R)$$

对应

010010010010

状态转换函数

$$\delta(q_2,a)=(q_2,a,R)$$

对应

00101001010

状态转换函数

$$\delta(q_2,b)=(q_3,\#,N)$$

对应

00100100010001000

图灵机 M 的状态转换函数编码为

111000010100001010110000100101001011010101010110100100100101100101001010110010010001001000111

如果需要识别的串为

abaab

对应编码

01001010100

则图灵机 M 的编码和输入带上的符号串的编码为

000010000010010000100011000100000111000010100001010110000100101001101010101101001001001011001010010100101100100100010001000111010010101001

将上述编码作为通用图灵机的输入，则通用图灵机能够分辨出图灵机 M 的基本情况（包括图灵机的状态、字母表等信息）、所有的状态转换函数和当前需要识别的串。

6.4 本章小结

对图灵机的编码方式可能有多种，基于 0、1 的编码方法是最方便的一种。

通用图灵机只是改变了一般的图灵机输入带上的符号和状态转换函数的表示方法，即仅仅只利用 0 和 1 来代表图灵机。正如任何的高级程序语言的程序，最终都可以转换为使用 0 和 1 的机器语言的程序表示（当然，机器语言中，1 不是作为编码的分隔符号）。

本章扩展了基于 0、1 对图灵机进行编码的方法，可以对图灵机的基本情况、字母表、状态（包括开始状态、接收状态和可能的拒绝状态）、状态转换函数和待接收的输入串进行编码。

第七章 图灵机扫描多个符号技术

图灵机成熟的构造技术包括：存储技术、移动技术、多道技术、查讫技术和子程序技术，陈晓亮提出了图灵机的递归技术。在各种技术中，图灵机读/写头每次只能扫描一个符号。

语言由字母表上的字符串构成，而语言中的所有句子必须包含有或者不能包含有特定的子串；或者需要将语言中句子所包含的特定子串进行替换；构造图灵机识别该类语言是困难的，传统的方法是利用图灵机的存储技术和移动技术进行实现，过程比较烦琐。

本章提出一种新的构造图灵机的技术：扫描子串技术，即将特定子串当作一个整体，使得图灵机一次可以扫描多个符号，可以方便地构造图灵机识别该类语言，请见参考文献[100]。

7.1 图灵机的动作

在任意时刻，有限状态控制器处于某个状态，读/写头将扫描带上的一个单元，依照此时的状态和扫描到的带上一个符号，图灵机将有一个动作如下：

有限状态控制器所处的状态进行改变；

把刚刚扫描过的单元上符号擦除掉，并印刷上一个新的符号；

把读/写头向左或向右移动一个单元；或者读/写头不移动。

图灵机的一个动作可以使用五元式

$$<q,x,q',x',\{L,R,N\}>$$

表示。

7.2 利用存储技术识别包含子串的语言

语言由字母表上的字符串构成，某类语言要求语言中的所有句子必须包含或不能包含特定的子串。

利用图灵机接收该类语言，不需要对特定的子串进行修改。传统的方法是利用存储技术，将输入带上的一个或多个符号“存储”到图灵机的有限状态控制器

中[1]。可以使用一个 n 元组表示一个状态，而 n 元组的不同的分量可以代表不同的含义。

7.3 利用扫描子串技术识别包含子串的语言

图灵机每次只能扫描输入带上的一个符号，如果对于字母表Σ，要求图灵机接收语言 L 的每个句子必须包含子串

$$a_1a_2\ a_3\cdots a_k$$

其中，K>=2，就可以将该子串当作一个整体，使得图灵机可以一次扫描输入带上的多个符号。

例如要求语言的每个字符串必须包含子串 100，可以将输入带上的三个符号当作一个单元，整体进行扫描，但是，将图灵机的读/写头每次还是只能向右仅移动一个单元。例如串 010011，它包含有子串 100，而如果图灵机的读/写头每次向右移动三个单元，则该串分解为 010 和 011，都不包含子串 100。

定义 7-1 扫描多个符号的图灵机

扫描多个符号的图灵机是七元式

$$TuringM=(Q,\Sigma,q_0,q_\alpha,\delta)$$

其中，

Q、Σ、q_0、q_α的定义同扫描一个符号的图灵机；

δ 是

$$Q\times\Sigma^*\rightarrow Q\times\Sigma^*\times\{L,R,N\}$$

的一个映射，即

$$\delta(q,w)=(q',w,\{L,R,N\})$$

记为

$$<q,w,q',w,\{L,R,N\}>$$

7.4 扫描多个符号的图灵机与扫描一个符号的图灵机等价

定理 7-1 扫描多个符号的图灵机与扫描一个符号的图灵机是等价的

证明：

假设图灵机不改变子串的内容，读/写头每次都向右移动。

扫描多个符号的图灵机与是扫描一个符号的图灵机的结构是一致的，不同之

处在于状态转换函数的定义。

扫描多个符号的图灵机的状态转换函数为

$$<q,a_1a_2\ a_3\cdots a_k,q',a_1a_2\ a_3\cdots a_k,R>$$

扫描一个符号的图灵机的状态转换函数为

$$<q,x,q',x',R>$$

可以将扫描多个符号的图灵机的一个状态转换函数等价的改造为扫描一个符号图灵机的多个状态转换函数

$$<q,a_1,q_1,a_1,R>$$

$$<q_1,a_2,q_2,a_2,R>$$

$$<q_2,a_3,q_3,a_3,R>$$

$$\cdots$$

$$<q_{k-1},a_k,q_k,a_k,L>$$

$$<q_k,a_{k-1},q_{k+1},a_{k-1},L>$$

$$\cdots$$

$$<q_{2k-3},a_2,q',a_2,N>$$

其中，状态q_1、q_2、…、q_{2k-3}是新增加的状态。

扫描多个符号的图灵机实际上是扫描输入带上的多个符号，而读/写头仅移动一个单元；扫描多个符号的图灵机的一个状态转换函数实际上是由扫描一个符号的图灵机的多个状态转换函数组合而成的。

扫描多个符号的图灵机与扫描一个符号的图灵机是等价的。

证毕。

例 7-1 构造图灵机，接收语言 L，L 中每个句子必须包含 100 子串。

构造图灵机

$$TuringM=(Q,\Sigma,start,accept,\delta)$$

其中，

$$Q=\{start,q,refuse,accept\}$$

$$\Sigma=\{0,1\}$$

$$\Sigma'=\{0,1,B\}$$

(1) 输入带上的符号个数不足 3 个

$$<start,BBB,refuse,BBB,N>$$

$$<start,0BB,refuse,0BB,N>$$

$$<start,1BB,refuse,1BB,N>$$

<start,01B,refuse,01B,N>

<start,10B,refuse,10B,N>

<start,00B,refuse,00B,N>

<start,11B,refuse,11B,N>

(2) 扫描前三个符号

<start,000,q,000,R>

<start,001,q,001,R>

<start,010,q,010,R>

<start,011,q,011,R>

<start,101,q,101,R>

<start,110,q,110,R>

<start,111,q,111,R>

<start,100,accept,100,R>

<3> 每次扫描三个符号

<q,000,q,000,R>

<q,001,q,001,R>

<q,010,q,010,R>

<q,011,q,011,R>

<q,101,q,101,R>

<q,110,q,110,R>

<q,111,q,111,R>

(4) 扫描到子串 100 时结束

<q,100,accept,100,R>

(5) 整个输入串中没有 100

<q,BBB,refuse,BBB,N>

(6) 输入带上剩余的符号个数不足 3 个

<q,0BB,refuse,0BB,N>

<q,1BB,refuse,1BB,N>

<q,01B,refuse,01B,N>

<q,10B,refuse,10B,N>

<q,00B,refuse,00B,N>

<q,11B,refuse,11B,N>

7.5 扫描多个符号的图灵机的扩展

将扫描多个符号的图灵机的状态转换函数

$$<q,a_1a_2\ a_3\cdots a_k,q',a_1a_2\ a_3\cdots a_k,R>$$

扩展为

$$<q,a_1a_2\ a_3\cdots a_k,q',b_1b_2\ b_3\cdots b_m,R>$$

表示可以将特定的子串进行替换。

将扩展的状态转换函数等价地改造为扫描一个符号图灵机的多个状态转换函数，可能存在三种情况。

1) k=m

$$<q,a_1,q_1,b_1,R>$$

$$<q_1,a_2,q_2,b_2,R>$$

$$<q_2,a_3,q_3,b_3,R>$$

$$\cdots$$

$$<q_{k-1},a_k,q',b_k,R>$$

刚好替换子串的 k 个符号

2) k<m

先将子串的 k 个符号进行替换

$$<q,a_1,q_1,b_1,R>$$

$$<q_1,a_2,q_2,b_2,R>$$

$$<q_2,a_3,q_3,b_3,R>$$

$$\cdots$$

$$<q_{k-1},\ a_k,\ q_{k+1},\ b_k,\ N>$$

将$a_1a_2\ a_3\cdots a_k$子串的后面部分利用移动技术或子程序技术右移m-k个位置

$$\cdots$$

复制剩余的m-k个符号

$$\cdots$$

3) k>m

先将子串的前 m 个符号替换

$$<q,a_1,q_1,b_1,R>$$

$<q_1,a_2,q_2,b_2,R>$

$<q_2,a_3,q_3,b_3,R>$

…

$<q_{m-1},a_m,q_m,b_m,N>$

将$a_1a_2\ a_3\cdots a_m$串的后面部分利用移动技术或子程序技术左移k-m个位置

…

最后，读/写头都处于$b_1b_2\ b_3\cdots b_m$的右边第一个单元处。

特别地，如果m=0，即扫描多个符号的图灵机的状态转换函数为

$<q,a_1a_2\ a_3\cdots a_k,q',\varepsilon,R>$

表示可以删除$a_1a_2\ a_3\cdots a_k$子串。

整个过程以图灵机的移动方向为右移为标准，左移的情况可以类似讨论。

7.6 扩展图灵机状态转换函数实例

例 7-2 构造 TuringM，要求将输入串包含的子串 100 替换为 abcde 子串。

令

$$x,y,z\in\{0,1\}$$

(1) 输入带上的符号个数不足 3 个

<start,BBB,refuse, BBB,N>

<start,xBB,refuse,xBB,N>

<start,xyB,refuse,xyB,N>

(2) 扫描前三个符号

<start,xyz,q,xyz,R> //此处 ：xyz 不是 100

<start,100,q,abcde,R>

(3) 每次扫描三个符号

<q,xyz,q,xyz,R> //此处 ：xyz 不是 100

<q,100,q,abcde,R>

(4) 输入带上剩余的符号个数不足 3 个

<q,xBB,q,xBB,N>

<q,xyB,q,xyB,N>

(5) 结束

<q,BBB,halt,BBB,N>

例 7-3 构造 TuringM ，要求 M 将输入串的 100 子串替换为 ab 子串。

令

x,y,z∈{0,1}

(1) 输入带上的符号个数不足 3 个

<start,BBB,halt,BBB,N>

<start,xBB,halt,xBB,N>

<start,xyB,halt,xyB,N>

(2) 扫描前三个符号

<start,xyz,q,xyz,R>　　//此处 ： xyz 不是 100

<start,100,q,ab,R>

(3) 每次扫描三个符号

<q,xyz,q,xyz,R>　　//此处 ： xyz 不是 100

<q,100,q,ab,R>

(4) 输入带上剩余的符号个数不足 3 个

<q,xBB,halt,xBB,N>

<q,xyB,halt,xyB,N>

(5) 结束

<q,BBB,halt,BBB,N>

7.7 本章小结

针对特定的一类语言：语言中的所有句子必须包含有或者不能包含特定的子串；或者需要将语言中句子所包含的特定子串进行替换，提出一种新的图灵机构造技术：扫描子串技术，即将特定的子串当作一个整体，使得图灵机一次可以扫描多个符号，具有较强的抽象性。

第八章 非负整数的k元函数图灵计算模型

图灵可计算问题，是图灵机研究中的重要问题之一，在理论和实践上具有重要意义。

图灵机和其他自动机都具有具有识别语言的功能，还具有“计算”功能，可以在指定的数的表示法情况下，实现非负整数的函数求值[1,2,84,85]。

8.1 引言

设有k元函数

$$f(n_1,n_2,\cdots,n_k)=m$$

如果图灵机接受输入

$$0^{n_1}10^{n_2}1\cdots10^{n_k}$$

而输出符号串

$$0^m$$

则称k元函数f为图灵机计算的函数，也称f是图灵可计算的；但若k元函数f无定义，则图灵机没有恰当的输出。

蒋宗礼、陈有祺分析了非负整数的加法、真减法和乘法的图灵可计算问题[1,2]，但没有讨论关系运算、一般减法运算、除法和余数运算的图灵可计算问题。本章较完整地讨论k元函数对于关系运算和算术运算的图灵可计算问题。

采用 “一进制”方式表示一个非负整数，即使用0^m，表示整数m。

k元函数是多个2元函数的组合，本章仅讨论2元函数的情况。

若需要计算

$$f(x,y)$$

则可以在输入带上存放

$$0^x10^y$$

通过图灵机的动作，将带上串改写为

$$0^{f(x,y)}$$

的形式，即表示出计算的结果。

为简化描述，没有给出图灵机的完整定义，只给出图灵机的状态转换函数或图

灵机的实现描述。

8.2 非负整数图灵关系运算

对于非负整数 n 和 m，图灵机输入 0^n10^m，对 1 左、右的 0 依次使用 2、3 进行标识，并根据接收状态确定 n 和 m 之间的关系为大于、小于或等于。

<start,0,del_01,2,R>

<del_01,0, del_01,0,R>

<del_01,1, del_010,1,R>

<del_010,3, del_010,3,R>

<del_010,0,seek_2,3,L>

<seek_2,3,seek_2,3,L>

<seek_2,1,seek_2,1,L>

<seek_2,0,seek_2,0,L>

<seek_2,2,seek_0,2,R>

<seek_0,0,del_01,0,R>

结束情况

<del_010,B,GT,N>

<seek_0,1,L,R>

<L,3,L,3,R>

<L,0,LT,0,N>

<L,B,EQ,B,N>

接收状态 GT、LT 和 EQ 分别代表 n>m、n<m 和 n=m。

8.3 非负整数图灵算术运算

8.3.1 非负整数图灵加法运算

图灵机输入 0^n10^m，需要输出 0^{n+m} 。

当 n=m=0 时，输入为 1，输出为 B，需将 1 改写为 B。

当 n=0，m>0 时，输入为 10^m，输出为 0^m，需将 1 改写为 0，将最后一个 0 改写为 B。

当n>0，m=0时，输入为0^n1，输出为0^n，需将1改写为B。

当n>0，m>0时，输入为0^n10^m，输出为0^{n+m}，需将1改写为0，将最后一个0改写为B。

设计统一的动作为将1改写为0，将最后一个0(可能是1改写成的0)改写为B。

<start,1,accept,B,N>　　//串为10^m

<start,0,q_1,0,R>　　//扫描第一个0

<q1,0,q1,0,R>　　//跳过剩余的0

<q1,1,q1,0,R>　　//遇到1,改为0

<q1,B,q2,B,L>　　//遇到B,向左找0

<q2,0,accept,B,N>　　//最后一个0改为B

其中，

<q_1,0,q_1,0,R>

可以跳过1左边和右边的所有的0。

8.3.2 非负整数图灵减法运算

非负整数减法分为真减法和一般减法。

1 非负整数真减法图灵机

非负整数真减法定义为

$$n \dot{-} m = \begin{cases} n-m & n>m \\ 0 & n \le m \end{cases}$$

字符串的形式为0^n10^m，寻找1左边的一个0，删除1右边的一个0；可能在寻找1左边的0时结束（m<=n），或者在删除1右边的0时结束（m>n）。

(1) 扫描第一个0

<start,0,q_1,B,R>

(2) 跳过1左边的剩余的0

<q_1,0,q_1,0,R>

<q_1,1,q_2,1,R>

(3) 将1后面的第一个0变为1

<q_2,0,q_3,1,L>

<q_2,1,q_2,1,R>

(4) 向左找0

<q_3,1,q_3,1,L>

<q_3,0,q_3,0,L>

<q_3,B,start,B,R>　//转(1) 重复上述动作

(5) 遇到最右边的B，表示1右边已经没有0

<q_2,B,q_4,B,L>

<q_4,1,q_4,B,L>　//将1改写为B

<q_4,0,q_4,0,L>

<q_4,B,accept,0,N>　//n>m，补写一个0，结束

(6) 当n<=m时， start将遇到第一个1

<start,1,q_5,B,R>

<q_5,1,q_5,B,R>　//将后面的1改写为B

<q_5,0,q_5,B,R>　//将后面的0改写为B

<q_5,B,accept,B,N>　//结束

在第(5)步开始时，输入带上的串形式为

BBB…B000…011…11B

根据图灵机的指令，先在左边消除一个0，再去1的右边找0，当发现1的右边已经没有0时，此时减法工作应该结束，但左边多消除了一个0，因此，第(5)步，在q_4的的控制下除了将1改写为B外，还应该将最左边的一个0补写回来，才能结束减法工作。

当n<=m时，整个减法的结果应该为0。输入带全为B；由第(6)部分完成。

2 非负整数一般减法图灵机

第(1)、(2)、(3)、(4)部分的规则同非负整数真减法对应部分规则；将(6)修改为

(6) 当n<=m时， start将遇到第一个1

<start,1,q_5,B,R>

<q_5,1,q_5,B,R>　//将后面的1改写为B

<q_5,B,EQ,B,N>

<q_5,0,q_6,0,R>

<q_6,0,q_6,0,R>

<q_6,B,negative,B,N>

接收状态negative表示减法的结果为负数。

8.3.3 非负整数图灵乘法运算

乘法可以转换为多次加法，

$$n*m=(1+1+\cdots 1)*m$$

图灵机的输入带上存放串 0^n10^m，　运算后，使得带上的串变为 0^{n*m} 形式。

图灵机处理该问题的最一般的方法为，当从 1 的左边消去一个 0 后，应该在 0^m 的后面增加 m 个 0(补 1 作为分隔标记)；当 1 左边的所有的 0（共有 n 个）消完后，再消去多余的符号（两个 1 和原来的 0^n），就得到了 0^{n*m} 形式。

在 0^m 后面添加 n 个 0 的过程是重复的，可以使用子程序技术。

在某个时刻，TM 输入带上的符号为

$$B^h0^{n-h}10^n10^{h*m}$$

表示已经完成 h*m。

图灵机的指令分为 3 部分：初始化，主控程序和子程序。

1) 初始化：

将第一个 0 变为 B，并在最后一个 0 后面设置标记为 1（该标记表明了增加 0 的开始位置）；使得增加的 0 在第二个 1 的后面；并将读/写头移动到 m 个 0 中的第一个处。则格局变换为

$$start0^n10^m =>^* B0^{n-1}1sub\_start0^m1$$

此时，只是消去了第一个 0，并设置标记 1，但还没有在标记 1 后面增加 0；图灵机即将扫描原来 0^m 的第一个 0。

2) 主控程序：

初始化后，状态变为 sub_start，相当于子程序图灵机的开始状态。

子程序的功能是将 m 个 0 增加到第二个 1 的后面。

当退出子程序时，状态为 sub_end（sub_end 也就是子程序图灵机的结束状态），此时需要将读/写头移动到前面 n 个 0 中剩余 0 的第一个处；并将这个 0 改为 B，准备进入下次循环。对应的格局转换为

$$B^hstart0^{n-h}10^m10^{(h-1)*m}$$
$$=>^* B^h0^{n-h}1sub\_start0^m10^{(h-1)*m}$$
$$\cdots \quad //进入子程序$$
$$=>^* B^h0^{n-h}1sub\_end0^m10^{h*m}$$
$$=>^* B^{h+1}start0^{n-h-1}10^m10^{h*m}$$

当找不到前面 n 个 0 中剩余的 0 时，表示乘法计算工作已经结束，需要消去

多余的符号（两个 1 和原来的 0^n），得到最后的结果串。对应的格局转换为

$$B^n find\_left\_B10^m10^{n*m} =>^* B^{n+m+1+1} end0^{n*m}$$

其中，状态 end 是接收状态。

3) 子程序

子程序 TM 从它的开始状态 sub_start 启动， 进入接收状态 sub_end 时完成一次工作，并返回到主控程序。

进入图灵机子程序时，格局为

$$B^h0^{n-h}1sub\_start000\cdots010^{(h-1)*m}$$

子程序对应的格局转换为

$$B^h0^{n-h}1sub\_start0^m10^{(h-1)*m} =>^* B^h0^{n-h}1sub\_end0^m10^{h*m}$$

综上所述，整个图灵机的格局转换情况分别为

初始化：

$$start0^n10^m =>^* B0^{n-1}1sub\_start0^m1$$

主程序消除 0

$$B^h start0^{n-h}10^m10^{(h-1)*m} =>^* B^h0^{n-h}1sub\_start0^m10^{(h-1)*m}$$

子程序增加 0^n

$$B^h0^{n-h}1sub\_start0^m10^{(h-1)*m} =>^* B^h0^{n-h}1sub\_end0^m10^{h*m}$$

主程序消除 0

$$B^h0^{n-h}1sub\_end0^n10^{hn} =>^* B^{h+1} start0^{n-h-1}10^m10^{h*m}$$

主程序消除多余符号

$$B^n find\_left\_B10^m10^{n*m} =>^* B^{n+m+1+1} end0^{n*m}$$

图灵机的状态转换函数为

(1) 初始化

<start,0,find_1,B,R>
<find_1,0,find_1,0,R>
<find_1,1,find_B,1,R>
<find_B,0,find_B,0,R>
<find_B,B,add_1,1,L> //在最后增加标记 1
<find_B,1,add_1,1,L> //串的最后只能放一个 1
<add_1,0,add_1,0,L>
<add_1,1,q_1,1,R>

此时，将扫描原来 0^n 的第一个 0

(2) 子程序

<sub_start,0,sub_find_B,2,R>

//将0标记为2，以方便复制0^n

<sub_find_B,0,sub_find_B,0,R>

//向右寻找B

<sub_find_B,1,sub_find_B,1,R>

//遇到标记1（第二个1）

<sub_find_B_2,B,sub_find_2,0,L>

//复制一个0;向左寻找0^n中剩余的0(寻找标记2)

<sub_find_2,0,sub_find_2,0,L>

<sub_find_2,1,sub_find_2,1,L>

<sub_find_2,2,sub_find_2,2,R>

//准备复制下一个0

<sub_start,1,sub_2_to_0,1,L>

//n个0都已复制

<sub_2_to_0,2,sub_2_to_0,0,L>

//将2恢复为0

<sub_2_to_0,1,sub_end,1,R>

子程序结束，读/写头仍然在0^n的第一个0处。

(3) 主程序

<sub_end,0,find_left_0,0,L>

<find_left_0,1,find_left_B,1,L>

<find_left_B,0,end_one_loop,0,L>

<end_one_loop,0,end_one_loop,0,L>

<end_one_loop,B,start,B,R>

//上一次循环结束,本次循环开始

<find_left_B,B,end_loop,B,R>

//n个0都消完,循环结束

<end_loop,1,change_to_B,B,R>

<change_to_B,0,change_to_B,B,R>

<change_to_B,1,end,B,N>

此时，图灵机带上的符号形式为

$$B^{n+m+1+1}0^{n*m}$$

图灵机共有 15 个状态，子程序图灵机使用了 5 个状态：

sub_start、sub_find_B、sub_find_2、sub_2_to_0、sub_end

主程序图灵机使用了 12 个状态：

start、sub_start、sub_end、find_1、find_left_0、find_left_B、end_one_loop、end_loop、change_to_B、find_B、add_1、end

8.3.4 非负整数图灵除法运算

非负整数除法定义为

$n \div m=0$，当 $n=0, m\neq 0$ 时；

$n \div m=1$，当 $n=0, m=0$ 时；

$n \div m=\infty$，当 $n\neq 0, m=0$ 时；

$n \div m=k$，当 $n\neq 0, m\neq 0$ 时(k 取整数)；

图灵机的输入带上存放串 0^n10^m，运算后，带上的串应该变为 $0^{n \div m}$ 形式。

除法可以转换为多次减法。

每次在 1 左边消去 m 个 0；该过程是重复的，可以使用子程序技术。

设置子程序对应的格局转换为

$$0^i(5^{m-1}3)^j1q_i0^m=>*0^{i-m}(5^{m-1}3)^{j+1}1q_f0^m$$

子程序的指令为

<start,0 seek_left_0,2,L>

<seek_left_0,1,seek_left_0,1 ,L>

<seek_left_0,3,seek_left_0,3 ,L>

<seek_left_0,5,seek_left_0,5 ,L>

<seek_left_0,0,seek-right_0,3,R>

<seek-right_0,5,seek-right_0, 5,R>

<seek-right_0,3,seek-right_0, 3,R>

<seek-right_0, 2,seek-right_0, 2,R>

<seek-right_0, 1,seek-right_0, 1,R>

<seek-right_0,0,seek_left_more_0,2,L>

<seek_left_m_0, 2,seek_left_m_0,2,L>

<seek_left_m_0, 1,seek_left_m_0,1,L>

<seek_left_m_0, 3,seek_left_m_0,3,L>

<seek_left_m_0, 5,seek_left_m_0,5,L>

<seek_left_m_0,0, seek-right_0,5,R>

正常完成一次减法结束

<seek-right_0,B,sub_end,B,L>

<sub_end ,2,sub_end,0,L>

<sub_end,1,end_sub,1,R>

向左找 0,未找到结束，且 n 能够整除 m；

<seek_left_0, ⊢, aliquot, ⊢,R>

<aliquot,5, aliquot,5,R>

<aliquot,3, aliquot,3,R>

<aliquot,1, aliquot,1,R>

<aliquot,2, end_aliquot,0,L>

向左找 0，未找到结束，且 n 不能够整除 m；

<seek_left_m_0, ⊢,non_ aliquot, ⊢,R>

<non_ aliquot,5, non_ aliquot,6,R>

<non_ aliquot,3,delete_3,6,R>

<delete_3,5, delete_3,5,R>

<delete_3,3, delete_3,3,R>

<delete_3,1, delete_3,1,R>

<delete_3,2, delete_3,2,R>

<delete_3,B, non_ aliquot_end,B,L>

<non_ aliquot_end,2, non_ aliquot_end,0,l>

<non_ aliquot_end,1,end_non_ aliquot,1,R>

最终，3 的个数代表商；6 的个数代表余数。

子程序的结束状态可能为

end_sub,end_Aliquot,end_non_ aliquot

8.3.5 非负整数图灵余数运算

如果只要求余数，则

<start,0 seek_Left_0,2,L>

<seek_left_0,1, seek_left_0,1 ,L>

<seek_left_0,2, seek_left_0,2 ,L>

<seek_left_0,3, seek_left_0,3 ,L>

<seek_left_0,A, seek_left_0,A ,L>

<seek_left_0,0,seek-right_0,3,R>

<seek-right_0,3,seek-right_0, 3,R>

<seek-right_0,2,seek-right_0, 2,R>

<seek-right_0,1,seek-right_0, 1,R>

<seek-right_0,0,seek_left_0,2,L>

正常完成一次减法结束，恢复 1 右边的 m 个 2 为 0，向左将 m 个 3 改写为 A

<seek-right_0,B,sub_end,B,L>

<sub_end ,2,sub_end,0,L>

<sub_end,A,sub_end ,B,L>

<sub_end,3 sub_end,B,L>

<sub_end 0,sun_end_R.0.R>

<sun_end_R,A, sun_end_R,A,R>

<sun_end_R,1,end_sun,1,R>

<sun_end, ⊢,aliquot_end,R> //余数为 0

<aliquot_end,A,aliquot_end ,B,R>

<aliquot_end,1,aliquot_end ,B,R>

<aliquot_end,0,aliquot_end ,B,R>

<aliquot_end,B, end_aliquot,B,N>

向左找 0,未找到结束，且 n 不能够整除 m；

<seek_L_0, ⊢,non_ aliquot, ⊢,R>

<non_ aliquot,3, non_ aliquot,0,R>

<non_ aliquot,A,delete_A,B,R>

<delete_B,1, delete_B,B,R>

<delete_B,2, delete_B,B,R>

<delete_B,B, end_non_ aliquot,B,N>

最终，0 的个数为余数。

8.4 非负二进制整数与一进制数之间的转换

8.4.1 非负二进制整数转换为一进制数

非负二进制整数转换为一进制数，用二进制数各位置的数按权展开后相加即可。

利用三道图灵机来实现，则第一道存放要转化的二进制形式的整数，第二道存放二进制各位置的权值即 2^n（n>=1），第三道存放结果。

首先读写头移动到最右端一位即二进制的最低位，第二道先输入一个 0，即 2^0，第三道开始全是 B，代表还是 0。图灵机一开始就进行判断，如果第一道是 0，则不做任何操作，继续向左移动；如果是 1，则将第二道的 0 全部复制到第三道，不论第一道是 0 还是 1，都要将第二道的数值翻倍，翻倍的算法是将 2^n 改变为 2^{n+1}，在第二道数值的最后设置一个标记符，将标记符前面的 0 都复制到标记符后面，然后将标记符改为 0 并将最后一个 0 改为标记符，此时第二道已完成了 2^n 改变为 2^{n+1} 的成功转变，同时第三道上的值是否改变依赖于第一道上的数值，判断第一道该位是 0 还是 1，是 0 则第三道上不做改动，是 1 的话则将第二道上的 0 再全部复制到第三道。这样依次向左移动读写头，直到遇到带的开始符号则停机，此时第三道为 0^m，m 的个数即是转换后的整数。

初始时，第二道为

0BB…BBB

即 2^0

第三道为

BBB…BBB

实现过程为

(1) 读/写头移动到第 1 道最后，若为 1，则将 2 道的 0 全部复制到第 3 道(即做一次加法)，并将 1 道的最后 1 位改为 B（循环后，若第 1 道的非 B 的最后为 ⊢，则结束）。

(2) 第 2 道 0 的个数*2，得到 2^{n+1}

(3) 回到第 1 步

例 8-1 将二进制数 1011 转换为一进制数。

初始

├1011

├0BBB

├BBBB

1 道最后 1 位为 1，则复制 2 道 0 到 3 道，将 1 道最后 1 位改为 B

├101B

├0BBB

├0BBB

2 道 0 个数*2

├101B

├00BB

├0BBB

1 道最后 1 位为 1，则复制 2 道 0 到 3 道，将 1 道最后 1 位改为 B

├10BB

├00BB

├000B

2 道 0 个数*2

├10BB

├0000B

├000BB

1 道最后 1 位为 0，不复制，将 1 道最后 1 位改为 B

├1BBB

├0000B

├000BB

2 道 0 个数*2

├1BBB

├00000000B

├000BB…

1 道最后 1 位为 1，复制，将 1 道最后 1 位改为 B

├BBBB

├00000000B

├000000000000B

第1道的非B的最后为├，结束，第3道为结果

8.4.2 非负一进制整数转换为二进制数

非负整数的一进制数转换为二进制数，除以2统计余数即可。

利用二道图灵机来实现，第一道存放要转化的一进制形式的整数，第二道存放转化后的二进制数。

每次除以2，可以利用本章7.2.4小节的方法进行，3的个数为商，6的个数为余数；然后将3改写为0（6和5改写为B），得到本次除法的商。

第2道依次（从右到左）保留每次除法的余数。

例8-2 将一进制数00000000000转换为二进制数。

初始

├00000000000100

├BB… BBBB

除以2

├65353535353100

├BB… BBBB

保留余数

├65353535353100

├BB… BB1

得商

├BBBBBB00000100

├BB… BB1

除以2

├BBBBBB65353100

├BB… BB1

得余数

├BBBBBB65353100

├BB… B11

得商

├BBBBBBBBB00100

├BB… B11

除以2，得余数

├BBBBBBBBB53100
├BB…　　BB011

得商

├BBBBBBBBBB0100
├BB…　　BB011

除以 2

├BBBBBBBBBB6100
├BB…　　BB011

得余数

├BB…　　BB100
├BB…　　B1011

8.5 本章小结

本章分析非负整数的 k 元函数对于关系运算（包括大于、小于、等于运算）和算术运算（包括加法、减法、乘法、除法和余数运算）的图灵可计算问题，并设置不同的接收状态区分非负整数的大小关系、一般减法和真减法以及整除的情况；提出非负整数的图灵计算模型，并根据该模型实现了非负整数的二进制数与非负整数的一进制数之间的相互转换。

第九章 非负二进制数的图灵计算模型

本章讨论非负整数的二进制数对于关系运算（包括大于、小于、等于运算）和算术运算（包括加、减、乘、除运算）的图灵可计算问题。以二进制为基础，将部分运算扩充到 N(3≤N≤10)进制。

参加运算的两个操作数分别存放于第一道和第二道，利用图灵机的存储和移动技术可以将每一道的第 2 个单元存放 B，并将第一道和第二道的两个操作数右对齐，且读/写头已经移动到最后，即数的最低位。

9.1 非负二进制整数图灵关系运算

对于二进制数 n 和 m，根据接收状态确定 n 和 m 之间的关系为大于、小于或等于。

构造 2 道 TM 进行二进制数的关系运算。

先考虑高位的情况；再考虑低位。

<start,[0,0],q,[0,0],L>

<start,[0,1],q,[0,1],L>

<start,[1,0],q,[1,0],L>

<start,[1,1],q,[1,1],L>

<q,[0,0],q,[0,0],L>

<q,[0,1],q,[0,1],L>

<q,[1,0],q,[1,0],L>

<q,[1,1],q,[1,1],L>

<q,[0,B],GT,[0,B],N>

<q,[1,B],GT,[1,B],N>

<q,[B,0],LT,[B,0],N>

<q,[B,1],LT,[B,1],N>

<q,[B,B],E,[B,B],R>

<E,[0,0],E,[0,0],R>

<E,[1,1],E,[1,1],R>

<E,[0,1],LT,[0,1],R>

<E,[1,0],GT,[1,0],R>

<E,[B,B],EQ,[B,B],N>

接收状态 GT、LT 和 EQ 分别代表 n>m、n<m 和 n=m。

对于 N(3≤N≤10)进制数，设 a,b=0,1,2,…，N-1

<start,[a,b],q,[a,b],L>

<q,[a,b],q,[a,b],L>

<q,[a,B],GT,[a,B],N>

<q,[B,b],LT,[B,b],N>

<q,[B,B],E,[B,B],R>

<E,[a,a],E,[a,a],R> //a=a

<E,[a,b],LT,[a,b],R> //a<b

<E,[a,b],GT,[a,b],R> //a>b

<E,[B,B],EQ,[B,B],N>

接收状态 GT、LT 和 EQ 分别代表 n>m、n<m 和 n=m。

9.2 非负二进制整数图灵二进制算术运算

9.2.1 非负二进制整数图灵加法运算

构造 3 道 Turing M 进行正的二进制数的加法运算。第三道存放计算结果（第一、二道的第 2 个单元存放 B）。

<[q,0],[0,0,B], [q,0],[0,0,0],L>

<[q,0],[0,1,B], [q,0],[0,1,1],L>

<[q,0],[1,0,B], [q,0],[1,0,1],L>

<[q,0],[1,1,B], [q,1],[1,1,0],L>

进位

<[q,1],[0,0,B], [q,0],[0,0,1],L>

<[q,1],[0,1,B], [q,1],[0,1,0],L>

<[q,1],[1,0,B], [q,1],[1,0,0],L>

<[q,1],[1,1,B], [q,1],[1,1,1],L>

两个数长度不一致

<[q,0],[0,B,B],[q,0],[0,B,0],L>

<[q,0],[1,B,B],[q,0],[1,B,1],L>

<[q,0],[B,0,B],[q,0],[B,0,0],L>

<[q,0],[B,1,B],[q,0],[B,1,1],L>

<[q,1],[0,B,B],[q,0],[0,B,1],L>

<[q,1],[1,B,B],[q,1],[1,B,0],L>

<[q,1],[B,0,B],[q,0],[B,0,1],L>

<[q,1],[B,1,B],[q,1],[B,1,0],L>

结束条件

<[q,0],[B,B,B],E,[B,B,B],L>

<[q,1],[B,B,B],E,[B,B,1],L>

<E, ⊢, ⊢⊢],END,N>

对于 N(3≤N≤10)进制数，设 a,b=0,1,2, …,N-1。

若 a+b<N；无进位

<[q,0],[a,b,B], [q,0],[a,b,a+b],L>

若 a+b>=N；有进位

<[q,0],[a,b,B], [q,1],[a,b,a+b-N],L>

若 a+b<N-1；无进位

<[q,1],[a,b,B], [q,0],[a,b,a+b+1],L>

若 a+b>=N-1；继续进位

<[q,1],[a,b,B], [q,1],[a,b,a+b-N+1],L>

两个数长度不一致

<[q,0],[a,B,B],[q,0],[a,B,a],L>

<[q,0],[B,b,B],[q,0],[B,b,b],L>

若 a<N-1

<[q,1],[a,B,B],[q,0],[a,B,a+1],L>

若 a=N-1

<[q,1],[a,B,B],[q,1],[a,B,0],L>

若 b<N-1

<[q,1],[B,b,B],[q,0],[B,b,b],L>

若 b=N-1

<[q,1],[B,b,B],[q,1],[B,b,0],L>

结束条件

<[q,0],[B,B,B],END,[B,B,B],N>

<[q,1],[B,B,B],END,[B,B,1],N>

9.2.2 非负二进制整数图灵减法运算

构造 3 道 Turing M 进行正的二进制数的减法运算。第 3 道存放计算结果。

先进行两个操作数的关系运算，若第 1 个操作数小，则利用第三道将第一、二道的操作数进行互换（结果取负即可）；使得第一道的操作数比较大。

<[q,0],[0,0,B],[q,0],[0,0,0],L>

<[q,0],[0,1,B],[q,1],[0,1,1],L>

<[q,0],[1,0,B],[q,0],[1,0,1],L>

<[q,0],[1,1,B],[q,0],[1,1,0],L>

<[q,1],[0,0,B],[q,1],[0,0,1],L>

<[q,1],[0,1,B],[q,1],[0,1,0],L>

<[q,1],[1,0,B],[q,0],[1,0,0],L>

<[q,1],[1,1,B],[q,1],[1,1,1],L>

两个数长度不一致

<[q,0],[0,B,B],[q,0],[0,B,0],L>

<[q,0],[1,B,B],[q,0],[1,B,1],L>

<[q,1],[0,B,B],[q,1],[0,B,1],L>

<[q,1],[1,B,B],[q,1],[1,B,0],L>

结束条件

<[q,0],[B, B, B],END,[⊢, B, B],N>

对于 N(3≤N≤10)进制数，设 a,b=0,1,2, …,N-1。

<[q,0],[a,b,B],[q,0],[a,b,a-b],L> //a>=b

<[q,0],[a,b,B],[q,1],[a,b,a+N-b],L> //a<b

<[q,1],[a,b,B],[q,0],[a,b,a-b-1],L> //a>b

<q,1],[a,b,B],[q,1],[a,b,a+N-1-b],L> //a<=b

两个数长度不一致

<[q,0],[a,B,B],[q,0],[a,B,a],L>

<[q,1],[0,B,B],[q,1],[0,B,N-1],L>

继续借位

<[q,1],[a,B,B],[q,0],[a,B,a-1],L>　// a≠0

结束条件

<[q,0],[B, B, B],END,[⊢, B, B],N>

9.2.3 非负二进制整数图灵乘法运算

如果乘数中至少有一个为 0，则结果为 0。乘数中两个都不为 0，利用多道图灵机实现二进制乘法。

1 五道图灵机

若需要保存原来的两个操作数，则利用 5 道图灵机来实现。

第一道存放第一个乘数，第二道存放第二个乘数，第三道存放上一次计算结果，第四道存放当前要加的结果，第五道存放乘法运算的结果。

二进制乘法运算时，是从第二个乘数的最低位开始，每一位去乘第一个乘数，是 0 则全为 0，1 则将第一个乘数复制，最后将每次的结果相加即可，每一位去乘第一个乘数也可以这样理解，每次乘时，0 则不管，1 则是以该位为最低位，复制第一个乘数即可。

读写头开始位于最右端即乘数的最后一位，假设刚开始两个乘数是右对齐的，图灵机开始就进行判断，如果第二道该位是 0，则第三道复制和第一道相同位数的 0；如果是 1，则将第一道上的二进制数复制到第三道。同时将第三道上的结果复制到第五道。第一道计算完毕后，继续左移依次计算，每次左移计算具体的方法是：先将第五道结果全部复制到第三道，然后开始先判断此时第二道第二位的值，0 则不管继续左移；第二道第二位是 1，则将第一道上的二进制数全部复制到第四道，并且最低位和当前计算的第二道该位位置一样，然后第三道和第四道做二进制加法，结果保存在第五道，完成后继续左移。直到左移到带的最左端则计算结束。

需要注意，每次左移后，把第五道的结果复制到第三道，即把已经计算的结果复制到第三道，并且把第五道全部改为 B。同时，已经计算过的第二道的要改为 B。

2　三道图灵机

若不需要保存原来的两个操作数，则可以利用三道图灵机实现二进制乘法。

第 1 道利用 2 作为分隔标记，左边全存放两个操作数长度加 1 个的 0，右边存放乘数（即第 2 个操作数）；第 2 道以第一道的 2 的左边一位为最低位，存放被乘

数（即第 2 个操作数）；第三道全存放 B。

算法：将读/写头移动到第一道的最后；

1)若第一道的最后一位为 0，将第一道的最后一位改为 B，转 3)；

2)若第一道的最后一位为 1；将 1 道和第二道的数相加；结果存放在第 3 道，然后将第 3 道的数复制到第一道；将第一道的最后一位改为 B，

3)若第一道的最后一位为 0，将二道的数左移一位；将第一道的最后一位改为 B，转 3)

4)第一道的最后一位为 1，将二道的数左移一位；将 1 道和第二道的数相加；结果存放在第 3 道，然后将第 3 道的数复制到第一道；将第一道的最后一位改为 B；转 3)

5)若第一道的最后一位为 2，结束；第三道的数为结果（第一道也为结果）。

例 9-1　计算 1010*101。

初始时，三道存放情况

├000000002101

├BBBB1010BBBB

├BBBBBBBBBBBB

第一道最后一位为 1，将 1 道和第二道的数相加；结果存放在第 3 道，然后将第 3 道的数复制到第一道；将第一道的最后一位改为 B

├00001010210B

├BBBB1010BBBB

├BBBB1010BBBB

第一道的最后一位为 0，将二道的数左移一位；将第一道的最后一位改为 B

├0000101021BB

├BBB10100BBBB

├BBBB1010BBBB

第一道的最后一位为 1，将二道的数左移一位；

├0000101021BB

├BB101000BBBB

├BBBB1010BBBB

然后，将 1 道和第二道的数相加；结果存放在第 3 道，然后将第 3 道的数复制到第一道；将第一道的最后一位改为 B

⊢001100102BBB

⊢BB101000BBBB

⊢BB110010BBBB

第三道为结果。

9.2.4 非负二进制整数图灵除法运算

假设被除数为 M，除数为 N，商为 y。使用三道图灵机处理二进制除法运算，初始，第 1 道存放被除数，第 2 道存放除数，第 3 道存放商。最终，第 1 道结果为余数，第 3 道为商。

若 N=0，0 作为分母，则结果无意义；

若 M=0，N 不为 0，则商为 0 ；

若 M<N，被除数 M 即为余数，商为 0。

若 M≥N，初始值 y=0,则 y=y+1，M=M-N；当 M<N，结束。

首先，对除数为 0 的情况的处理；然后，除数不为 0，被除数为 0 的情况的处理；再处理被除数小于除数的情况；最后处理除数大于被除数的情况。若位数相等，需要做二进制减法运算；若 M 位数大于 N，移动 N 与 M 右对齐，初始化商为 0，做二进制减法运算；根据减法运算后返回状态，若未溢出，做二进制的加法运算：商加 1；否则，二进制除法结束。

M≠0,N = 0 时

<start,[⊢, ⊢, ⊢],q0,[⊢, ⊢, ⊢],R>

<q0,[a,B,B],qinf,[B,B,B],R>

<qinf,[a,B,B],qinf,[B,B,B],R>

<qinf,[B,B,B],accept,[B,B,B],N>

M=0，N≠0，商为 0，第一道余数为空，第二道清空，第三道商为 0

<q0,[B,1,B],qzero,[B,B,0],R>

<qzero,[B,a,B],qzero,[B,B,B],R>

<qzero,[B,B,B],accept,[B,B,B],N>

M<N，第一道输出为 M 作为余数，第二道清空，第三道商为 0

<q0,[a,b,B],q0,[a,b,B],R>

<q0,[B,b,B],qless1,[B,b,B],R>

<qless1,[B,b,B],qless1,[B,B,B],R>

<qless1,[B,B,B],qless2,[B,B,B],L>

<qless2,[B,B,B],qless2,[B,B,B],L>

<qless2,[a,b,B],qless2,[a,B,B],L>

<qless2,[S,S,S],qless3,[S,S,S],R>

第一道存放的为余数，第三道存放结果 0，清空第二道，除数结束

<qless3,[a,B,c],accept,[a,B,c],N>

<qless3,[a,B,B],accept,[a,B,B],N>

M≥N ，第 1 道结果为余数，第 3 道为商

<q0,[a,b,B],q0,[a,b,B],R>

<q0,[a,B,B],qmore,[a,B,B],L>

<qmore,[a,b,B],qmove,[a,B,B],R>

<qmove,[a,B,B],qmove,[a,B,B],R>

<qmove,[a,B,B],qmodify,[B,B,B],L>

<qmodify,[a,B,B],qback,[a,b,B],L>

<qback,[a,B,B],qback,[a,B,B],L>

<qback,[a,b,B],qmove,[a,B,B],R>

所有位都已经移动为右对齐

<qback,[S,S,S],qminus1,[S,S,S],R>

<qminus1,[a,B,B],qminus1,[a,B,B],R>

<qminus1,[a,b,B],qminus1,[a,b,B],R>

<qminus1,[a,b,c],qminus1,[a,b,c],R>

读头移动到右对齐的最后一位，准备二进制减法

<qminus1,[B,B,B],qminus,[B,B,B],L>

无借位状态的减法运算标记

<qminus,[0,0,B],qminus,[0,0,B],L>

<qminus,[1,0,B],qminus,[1,0,B],L>

<qminus,[1,1,B],qminus,[0,1,B],L>

<qminus,[0,1,B],overflow,[1,1,B],L>

<qminus,[0,0,c],qminus,[0,0,c],L>

<qminus,[1,0,c],qminus,[1,0,c],L>

<qminus,[1,1,c],qminus,[0,1,c],L>

<qminus,[0,1,c],overflow,[1,1,c],L>

借位状态的减法运算标记

<overflow,[0,0,B],overflow,[1,0,B],L>

<overflow,[0,1,B],overflow,[0,1,B],L>

<overflow,[1,0,B],qminus,[0,0,B],L>

<overflow,[1,1,B],overflow,[1,1,B],L>

<overflow,[0,0,c],overflow,[1,0,c],L>

<overflow,[0,1,c],overflow,[0,1,c],L>

<overflow,[1,0,c],qminus,[0,0,c],L>

<overflow,[1,1,c],overflow,[1,1,c],L>

减法运算结束条件，处理减法结束的状态，并将读头移动到右对齐位置

<qminus,[a,B,B],qmin_over,[a,B,B],R>

<qminus,[a,B,c],qmin_over,[a,B,c],R>

<overflow,[1,B,B],flow_over,[0,B,B],R>

<overflow,[0,B,B],overflow,[1,B,B],L>

<overflow,[1,B,c],overflow_over,[0,B,c],R>

<overflow,[0,B,c],overflow,[1,B,c],L>

<qmin_over,[a,b,c],qmin_over,[a,b,c],R>

读头移动到右对齐位置，准备商加 1

<flow_over,[a,b,c],flow_over,[a,b,c],R>

<qmin_over,[B,B,B],qm_return,[B,B,B],L>

<flow_over,[B,B,B],flow_return,[B,B,B],L>

商加 1 运算考虑有无进位的情况

<qm_return,[a,b,B],add_over, [a,b,1],R>

无进位直接开始返回

<qm_return,[a,b,0],add_over,[a,b,1],R>

<qm_return,[a,b,1],carry,[a,b,0],L>

进位状态处理标记

<qm_return,[a,B,B], add_over, [a,b,1],R>

<qm_return,[a,B,0],add_over,[a,b,1],R>

<qm_return,[a,B,1],carry,[a,b,0],L>

<flow_return,[a,b,B],add_over,[a,b,1],R>

无进位直接开始返回

<flow_return,[a,b,0],add_over,[a,b,1],R>

<flow_return,[a,b,1],carry,[a,b,0],L>

进位状态处理标记

<carry,[a,b,B],add_over,[a,b,1],R>

进位状态的处理

<carry,[a,b,0],add_over,[a,b,1],R>

<carry,[a,b,1],carry,[a,b,0],L>

商加 1 结束返回到右对齐处

<add_over,[a,b,c],add_over,[a,b,c],R>

商加 1 返回，开始下一次的判断

<add_over,[B,B,B], q0,[B,B,B],L>

使用二进制减法和商加 1 实现二进制的除法运算。

9.3 串长度图灵计数器

利用四道图灵机来实现图灵计数器，可以计数输入带上符号个数。

第一道存放待记数的字符串；第二道存放上一次记数的结果（十进制，初始为 0），第三道存放 1（十进制）；第四道存放计数结果（十进制）。

从右向左，每次扫描第一道上的一个输入符号，利用 2 小节的规则，将第二道和第三道的数进行十进制加法运算，结果存放在第四道，然后将第四道的结果复制到第二道。重复，直到第一道上输入符号扫描结束。

例 9-2 对串“abcdefghijk”长度进行计数。

初始

```
├abcdefghijk
├BB…  B0
├BB…  B1          // 每次加 1
├BB…  BB
```

第 1 次计数

```
├abcdefghijB
├BB…   B0
├BB…   B1          // 每次加 1
├BB…   B1
```

4 道复制到 2 道

⊢abcdefghijB
⊢BB… B1
⊢BB… B1 // 每次加 1
⊢BB… B1

计数

⊢abcdefghiBB
⊢BB… B1
⊢BB… B1
⊢BB… B2

4 道复制到 2 道

⊢abcdefghiBB
⊢BB… B2
⊢BB… B1
⊢BB… B2

中间过程（省略）

…

最后一个符号

⊢aBBBBBBBBB
⊢BB… B10
⊢BB… BB1
⊢BB… B10

计数

⊢BBBBBBBBBB
⊢BB… B10
⊢BB… BB1
⊢BB… B11

4 道复制到 2 道

⊢BBBBBBBBBB
⊢BB… B11
⊢BB… BB1
⊢BB… B11

第一道为 ⊢（非 B）；则结束，第四道为结果。

9.4 本章小结

本文分析非负的二进制整数关于关系运算（包括大于、小于、等于运算）和算术运算（包括加法、减法、乘法、除法和余数运算）的图灵可计算问题。定义了二进制的 1 位数的计算规则和进位（或借位）规则，从而进行多位二进制数的运算。基于多道图灵机描述了算法。对于部分运算，将二进制扩展到 N 进制（$3 \leq N \leq 10$）。利用图灵加法运算规则，实现输入串长度图灵计数器，可以计算图灵机输入带上符号个数。

本章的下步工作是对算法的复杂度和 M（$M > 10$）进制数的图灵计算模型进行分析。

第十章 结束语

本文主要研究形式语言与自动机理论中形式语言理论、自动机理论和形式语言与自动机之间等价性理论的若干问题。主要工作与结论包括

（1）研究了形式语言与自动机理论中关于空串ε的一些问题。分析了ε产生式对文法和语言分类的影响；提出语言增加或减少ε句子的简单方法；特别从有限状态自动机的角度，讨论了开始状态 q_0 的作用；分析了ε-NFA 的ε状态转换函数的本质，提出ε-NFA 转换为 NFA 的新方法，即先将 ε-NFA 转换为扩展的正则文法形式，消除 ε 产生式和单产生式后，得到规范的正则文法，再将规范正则文法转换为 NFA，并通过实例进行了验证。该方法简单，易于理解和实现。

（2）针对 Chomshy 四类语言，讨论了四类语言对联合、连接、克林闭包运算的有效封闭性问题。基于蒋宗礼、陈有祺教授和 Peter Linz 等人的研究成果，分析了上下文相关语言和短语结构语言对于连接、克林闭包运算可能存在的串道问题，提出对应的解决办法：对字母表进行复制，避免句型中的来自不同文法的内容互相作为上下文。并根据语言对联合，连接和迭代运算封闭性的特点，提出基于简单自动机构造复杂自动机的方法。

（3）根据等价类构造有限状态自动机可以接收一类特定语言：语言由任意字母表上的某种进制 K 的数字串构成，而数字串（当作数字）能够整除某个非负整数 N（或对 N 的余数为 M，$N \geq 2$，$M \geq 0$），对该方法进行了扩展，可以针对任意字母表和所有进制的数字串构成的语言，而且满足语言中的所有数字串能够对任意正整数 N 取任意余数，从构造的角度保证了有限状态自动机的极小化。

（4）扩展基于 0、1 对图灵机进行编码的方法，可以对图灵机的基本情况、字母表、状态（包括开始状态、接收状态和可能的拒绝状态）、状态转换函数和待接收的输入串进行编码。

（5）针对特定的一类语言：语言中的所有句子必须包含有或者不能包含特定的子串；或者需要将语言中句子所包含的特定子串进行替换，提出一种新的图灵机构造技术：扫描子串技术，即将特定子串当作一个整体，使得图灵机一次可以扫描多个符号，但读/写头仍然只移动一个单元，具有较强的抽象性。

（6）研究非负整数的 k 元函数对于关系运算（包括大于、小于、等于运算）和算术运算（包括加法、减法、乘法、除法和余数运算）的图灵可计算问题，讨

论非负整数的二进制数与非负整数的一进制数之间的相互转换方法。

（7）研究非负的二进制整数关于关系运算和算术运算的图灵可计算问题，并扩展到 N 进制（3≤N≤10）；实现输入串长度图灵计数器，可以计算图灵机输入带上符号个数。

在论文中，对成果的验证主要采用理论推导和形式化验证的方法。在研究过程中，也采用了仿真程序（基于面向对象的 C++语言）对各项成果进行了正确性的程序验证。由于论文的篇幅的原因，没有进行详细的描述。

研究成果具有实际应用的价值，已经着手模式识别中的文本过滤的算法的改进、自动分词算法的改进等实际运用工作。

本文主要针对形式语言与自动机理论中的基础理论进行一些问题的分析与讨论。在各类自动机模型中，没有分析自动机状态转换函数的效率问题，下一步工作中，可以研究自动机的一个（或多个）状态转换函数的执行效率；自动机变形的研究已经取得了很大的成果，本文没有涉及到各类自动机变形的新模型的建立，下一步研究，可以针对通用或特定领域的具体需求，建立各类自动机的新模型。本文研究的自动机属于有限自动机，目前，也已经有了无限自动机模型和简单的应用，下一步工作中，可以深入研究无限自动机的模型新的应用；本文讨论了非负的二进制整数关于关系运算和算术运算的图灵可计算问题，并扩展到 N 进制（3≤N≤10），没有深入到 M（M＞10）进制整数关于关系运算和算术运算的图灵可计算问题，本文也没有涉及到关于实数的图灵可计算问题；下一步工作中，可以加强对应的研究内容。

致谢

本文的完成，首先要感谢我的导师—孙世新教授、博导。

孙老师凭着对理论研究的敏锐，以及对本人情况的了解，给我指明了研究的方向，并悉心指导了整个研究过程。

孙老师治学严谨、学识渊博，在言传身教中，他对理论研究的执着潜移默化地影响着我，引领我走进丰富多彩的理论世界。

孙老师对工作一丝不苟，待人平和又不失严厉，充满创造性的思想。在职攻读博士学位的过程中，孙老师始终无微不至地关心我的成长，给予我莫大的帮助。从孙老师身上，我不仅学到了丰富的理论知识，而且更让我懂得了作为一个科研工作者应有的治学态度，以及做人的基本道理。另外，在撰写学术论文和学位论文的过程中，孙老师总是严格要求，不断提出修改意见，在此再次表示衷心的感谢。

感谢我指导的研究生同学（特别是程小鸥同学，她帮助我完成了英文学术论文的撰写工作），他们对我的博士学业提供了大力的支持。

感谢电子科技大学计算机学院的王晓斌副院长、刘贵松副教授、屈鸿博士、余盛季老师和陈昆老师，我们经常在一起讨论学术和人生，使得我的博士生生活不仅充实，而且充满了乐趣。

感谢培养我的电子科技大学和计算机科学与工程学院，为我营造了和谐的学习、工作和生活的环境。

感谢本文参考文献的作者，正是由于他们前期大量辛苦、卓杰的工作，才有了本文的研究基础。

感谢发表和录用我的学术论文的期刊编委会和会议组委会，肯定了我的研究工作。

感谢辛勤评阅本论文的各位专家，中肯的评阅意见，将引领相关领域的研究达到一个新的高度。

最后，感谢我的父母以及爱人-曾红和我的小女-陈青然对我完成学业所给予的理解、支持、鼓励和包容。

我将继续努力，将毕生精力贡献给党的教育事业，回报我的祖国和人民。

参考文献

[1] 蒋宗礼, 姜首旭. 形式语言与自动机理论(第 2 版). 北京:清华大学出版社 2007

[2] 陈有祺. 形式语言与自动机. 北京:机械工业出版社, 2008

[3] Noam Chomsky. Syntactic Structures.北京:中国社会科学出版社, 1979,28-35

[4] Noam Chomsky, George A. Miller, Introduction to the Formal Analysis of Natural Languages. Mouton/Gauthier-Villars (Paris) ,1963,45-58

[5] Noam Chomsky, Aspects of the Theory of Syntax. MIT Press (Cambridge, MA), 1966,127-145

[6] Warren S. McCulloch, Walter Pitts, A Logical Calculus of the Ideas Immanent in Nervous Activity. Journal of . Symbolic Logic, 1944,Issue 2(9): 49-50

[7] Stephen Cole Kleene .Introduction to Metamathematics. New York : Van Nostrand, 1952,56-66

[8] Stephen Cole Kleene, Claude Shannon , John McCarthy. Representation of Events in Nerve Nets and Finite Automata in Automata Studies. Princeton, Princeton University Press,1956

[9] Stephen Cole Kleene,Richard Eugene Vesley. The Foundations of Intuitionistic Mathematics. Amsterdam : North-Holland Pub. Co., 1965,120-135

[10] Stephen Cole Kleene . Mathematical Logic. John Wiley.1967,Dover reprint, 2001 ,56-67

[11] Stephen Cole,Alonzo Church .Origins of Recursive Function Theory.Annals of the History of Computing 3(1). 1981,134-145

[12] Noam Chomsky Topics in the Theory of Generative Grammar, Paris: Mouton, 1972.

[13] Noam Chomsky The Logical Structure of Linguistic Theory, New York: Plenum Press, 1975

[14] Noam Chomsky Reflections on Language, New York: Pantheon Books, 1975

[15] Backus J.W., Bauer F.L., Green J., Katz C. , McCarthy J. , Naur P., Perlis A.J., etc. Report on the Algorithmic Language Algol 60

[16] 陈意云, 马万里. 语法制导编辑器的生成器. 计算机研究与发展, 1991, 28(7):

21-26

[17] 龚为众，尤晋元. 语法制导编辑器的自动生成系统. 计算机研究与发展, 1990, 27(9):48-53

[18] 丁忠俊. 基于模板式语法制导编辑器的程序转换系统. 计算机研究与发展, 1995, 32(1):28-32

[19] 朱明远，柳斌. ADA 语法制导编辑器的设计与实现. 计算机学报, 1990, 13(6):429-435

[20] Ibarra, O.H., Jiang, T., Wang, H. A characterization of exponential-time languagesby alternating context-free grammars. Theoretical Computer Science 1992,99(2): 301-313

[21] Cohen, J.M.The equivalence of two concepts of categorical grammar. Informationand Control 1967,10:475-484

[22] Pentus, M. Lambek grammars are context free. In: Proc. of 8th Ann. IEEE Symp.on Logic in Computer Science, 1993:429-433

[23] 陈力行. 上下文无关语言的数学理论. 济南:山东大学出版社, 1986

[24] 刘迎健，戴汝为. 识别在线手写汉字的模糊属性自动机. 自动化学报, 1988, 14(2):95-104

[25] 戴汝为. 模式识别的一类属性文法. 自动化学报, 1983, 9(2):45-49

[26] 张继军，吴哲辉. 广义有界上下文无关语言与 Petri 网语言. 系统仿真学报, 2005, 17(s):26-29

[27] Anthony G. Oettinger.Account Identification for Automatic Data Processing. J. ACM ,1957,4(3): 245-253

[28] Vincent E. Giuliano, Anthony G. Oettinger: Research on automatic translation at the Harvard Computation Laboratory. IFIP Congress 1959: 163-182

[29] Anthony G. Oettinger, Howard H. Aiken: Retiring computer pioneer. Commun. ACM 1962,5(6): 298-299

[30] Lisovik L P, Koval D A. Language Recognition by Two-Way Determinstic Pushdown Automata.Cybernetics and Systems Analysis, 2004, 40(6) :939-942

[31] 张继军，吴哲辉. 下推自动机的状态转换图与下推自动机的化简. 计算机科学, 2006, 33(3):271-274

[32] Turing A.On Computable Numbers, with an Application to the Entscheidungs problem. Proceedings of the London Mathematical Society, Series 2, 1936:230-265,

Available online at http://www.abelard.org/turpap2/tp2-ie.asp(2004年 7 月 12 日)

[33] Church, A.An Unsolvable Problem of Elementary Number Theory. American Journal of Mathematics, 1936,58:345-363

[34] Church, A. 1932. A set of Postulates for the Foundation of Logic. Annals of Mathematics, second series, 1932,33:346-366

[35] Church, A . A Note on the Entscheidungsproblem. Journal of Symbolic Logic, 1936, 1: 40-41

[36] Turing A..Computability andλ-definability. Symbolic Logic, 1937,vol. 2:153-163

[37] Deutsch,D.Quantum theory,the Church-Turing principle and Universal quantum computer.Proceedings of the Royal Society of London, 1985:97-116

[38] Michael Sipser. Introduction to the theory of computation. Boston, USA: PWS Publishing Company, 1997

[39] Hopcroft J.E, Ullman J.D, Introduction to Automata Theory, Languages and Computation. Addison-Wesleg, 1979

[40] Rajeev Alur , David L. Dill A theory of timed automata .Theoretical Computer Science archive, 1994,126(2): 183-235

[41] 朱维军, 刘保罗, 周清雷. 时间自动机与信号自动机的互模拟算法. 华南理工大学学报, 2008, 36(5) :130-135

[42] Neven F .Automata、logic and XML . University of Limburg press .2002

[43] Klans Jorn Lange.Klans Reinhardt .Set automata. Combinatorics, Complexity and Logic; Proceedings of the DMTCS'96, Springer ,321-329

[44] Ladner, R.E.. Lipton, R.J., Stockmeyer, L.J..Alternating pushdown and stack automata. SIAM Journal on Computing ,1984,13(1):135-155

[45] Moriya, E.A grammatical characterization of alternating pushdown automata.Theoretical Computer Science ,1989,67(1):75–85

[46] Ibarra, O.H., Jiang, T., Wang, H. A characterization of exponential-time languagesby alternating context-free grammars. Theoretical Computer Science,1992, 99(2): 301-305

[47] Okhotin, A. Efficient automaton-based recognition for linear conjunctive languages. International Journal of Foundations of Computer Science2003,14(6):1103-1116

[48] Okhotin, A. On the equivalence of linear conjunctive grammars and trellis

automata. RAIRO Theoretical Informatics and Applications 2004,38(1):69-88

[49] Culik II, K., Gruska, J., Salomaa, A. Systolic trelllis automata, i and ii. International Journal of Computer Mathematics ,1984,15(3): 3–22, 16(4):195–212

[50] Lisovik L P .D A Koval Language recognition by two-way determinstic pushdown automata。Cybernetics and Systems Analysis archive，2005.40(6): 939-942

[51] 张继军, 董卫, 吴哲辉等. 袋自动机. 计算机研究与发展, 2008, s1:200-204

[52] Ginsgurg, S., Spanier, E.H.: Finite-turn pushdown automata. SIAM Journal on Control 4(3),1996,4(3): 429-453

[53] Kutrib, M., Malcher, A.: Finite-turn pushdown automata. Discreet Applied Mathematics 2007,155: 2152–2164

[54] 郭清泉, 邵长春。接受超线性语言的有穷转向的 PDA 构造法和超线性语言秩的性质. 山东大学学报, 1988, 22(01):28-35

[55] Moore C, Crutchfield JP. Quantum automata and quantum grammars. Theoretical Computer Science, 2000,237(2):275-306.

[56] Gudder S. Basic Properties of Quantum Automata. International Journal of Theoretical Physics, 1999,38(9):2261-2282.

[57] Kondacsi, A. Watrous, J. On thei powe of quantum finite state automata. Foundations of Computer Science, 1997. Proiceedings. 38th Annual Symposium on 20-22 Oct. 1997 :66-75

[58] Dzelme-Brzia.Mathematical logic and quantum finite state automata. Theoretical Computer Science .2009,20(410):453-460

[59] Bertoni A, Carpentieri M. Analogies and differences between quantum and stochastic automata. Theoretical Computer Science, 2001,262(1):69-81.

[60] Benioff P.The computer as a physical system: a microsopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. Physical Review Letters,1982, vol.23:1581-1585

[61] 郭秀红. 基于量子逻辑的自动机理论的拓扑性质. 软件学报, 2007, 18(6):1282-1285

[62] 邱道文. 量子自动机的刻画. 软件学报, 2003, 4(1):7-14

[63] 邱道文. 基于量子逻辑的自动机和文法理论. 软件学报, 2003, 14(1):23-27

[64] 吕映芝. 上下文无关文法与无限状态自动机. 电子学报, 1996, 24(08):23-27

[65] 吴青娥, 舒兰. 模糊无限状态自动机及其收敛性. 模糊系统数学, 2004, 18(z1):240-245

[66] Minsky M.L.,Computation.Finite and Infinite Ma-chines, Prentice Hall,Englewood Cliffs,New Jersey,1967

[67] 于之硕, 杨静. 用于描述网络拓扑的形式语言的研究. 软件学报, 1999, 10(2):112-117

[68] Khatib L, Muscettola N, Havelund K. Mapping Temporal Planning Constraints into Timed Automata. New York :IEEE Press, 2007

[69] Pierre K, Qin M. A Lightweight Approach for Defining the Formal Semantics of a Modeling Language.2008, 5301 :690-704

[70] Jackson D, Wing J.Lightweight Formal Methods. IEEE Computer, 2006,29(4),16-30

[71] James W. A Bialgebraic Approach to Automata and Formal Language Theory. Berlin :Heidelberg Springer, 2009

[72]Ying M S.A formal model of computing with words.IEEE Transactions on Fuzzy Systems,2002,10:642-652

[73] 陈龙, 陈武. 基于时间有限状态自动机的事件重建推理算法. 重庆邮电大学学报(自然科学版) 2009, 3(21):393-397

[74] 陈倩. 一种基于有限自动机的快速串匹配算法. 计算机技术与发展. 2009, 1(19):131-138

[75] 李钢, 吴燎原, 张仁斌. 基于有限自动机的模式匹配算法及其应用研究. 系统仿真学报, 2007, 19(12):2772-2775

[76] Eberbach E,Burgin, M. Evolution of evolution: Self-constructing Evolutionary Turing Machine case study. Evolutionary Computation, 2007. CEC 2007. IEEE Congress on Volume , Issue , 25-28 Sept. 2007:4599-4605

[77] McUmberi, W.E.; Cheng, B.H.C A general framework for formalizing UML with formal languages. Software Engineering, 2001. ICSE 2001. Proceedings of the 23rd International Conference on 12-19 May 2001:433-442

[78] Makoto Murata, Dongiwon Lee,Muralii Mani,Kohisuke Kawaguchi. Taxonomy of XML schema languages using formal language theory. ACM Transactions on Internet Technology (TOIT) 2005,4(5):660-704

[79] Haitaoi Ma, Zhongxiao Hao,Yinghui Zhou.Active XML Schema Containment

Checking Based on Tree Automata Theory .Computeir and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on 16-19 Oct. 2007:11-6

[80] Wombacher, A.Fankhauser, P. Neuhold, E Transforming BPEL into annotated deterministic finite state automata for service discovery. Web Services, 2004. Proceedinigs. IEEE International Conference on 6-9 July 2004 :316-323

[81] Fan, Huyck, C. Implementation of finite state automata using LIF neurons. Cybernetic Intelligent Systems, 2008. CIS 2008. 7th IEEE International Conference on 9-10 Sept. 2008:1- 5

[82] 周启海. 有穷自动机的规范化. 计算机科学, 2004, 增刊, 156-157

[83] 周启海. NFA→FA→GFA 的自动转换算法. 电子科技大学学报, 2005, 34(3): 234-236.

[84] Peter Linz, An introduction to formal languages and automata. Boston :Jones and Bartlett,2005

[85] 吴哲辉, 吴振寰. 形式化语言与自动机理论. 北京:机械工业出版社, 2007

[86] 陈晓亮, 卢朝辉, 宋文. 基于图灵机的递归技术的实现. 计算机工程与学. 2008, 10(30):153-156

[87] CHEN Wen-yu , SUN Shi-xin.Constructing Automaton with the Closure of language calculation. International Computer Conference on Wavelet Active Media Technology and Information Processing,2006,232-239

[88] 蒋龙龙, 陈文宇. 利用等价类构造有限状态自动机. 计算机科学, 2006, 33(11): 272-274, 277

[89] 傅彦, 顾小丰, 王庆先等. 离散数学及其应用. 北京:高等教育出版社, 2007, 56-58

[90] 邵秀丽, 王孝喜. 离散数学. 北京:科学出版社, 2005

[91] 孙玉强, 李玉萍 王海燕. 确定有限自动机最小化算法的并行处理. 计算机科学, 2008, 35(1):298-300

[92] 朱征宇, 王术, 赵银春. 基于矩阵模型表示的有限自动机极小化方法. 计算机工程与应用, 2004, 40(35):47-49

[93]Campeanu C, Santean N. Minimal Cover-automata for Finite Languages. Theoretical Computer Science,2003,306:373-390

[94] Rafael C. Carrasco, Mikel L. Forcada. Incremental construction and maintenance of minimal finite-state automata. Computational Linguistics, 2002,2(28):323-327

[95] 韩光辉，段国丽. 有限自动机最小化算法的实现. 湖北工业大学学报, 2006, 21(2):69-71

[96] 安立新. 通用图灵机的计算机仿真设计. 中国计量学院学报, 2008, 19(3):246-250

[97] 刘俭，王剑，唐朝京等. 基于扩展通用图灵机的计算机病毒传染模型. 计算机研究与发展, 2003, 40(9): 21-27

[98] 马龙，梁意文. 图灵机模拟系统的设计与实现. 计算机工程与应用, 2005 41(8):101-104

[99] YuriiRogozhin.Small universal Turing machines.Theoretical Computer Science; 1996

[100] CHEN Wen-yu,CHENG Xiao-ou,SUN Shi-xin.Technology of Scanning Substring by Turing Machine.电子科技大学学报,2009,38(2), 270-274

攻博期间取得的研究成果

一、论文

[1] Chen Wenyu, Wang Xiaobin, Cheng Xiaoou, Sun Shixin.Turing Compute Model for Non-negative Binary Numbers. Journal of Software (EI 刊物,ISSN:1796-217X,已录用)

[2] Chen Wenyu,Wang Xiaobin,Cheng Xiaoou,Sun Shixin.Analysis of Valid Closure Property of Formal Language. Journal of Computers (EI 刊物, ISSN:1796-203X,已录用)

[3] CHEN Wen-yu,CHENG Xiao-ou,SUN Shi-xin.Technology of Scanning Substring by Turing Machine.电子科技大学学报，2009,38(2), 270-274 (EI: 20091812064287)

[4] Chen Wenyu, Wang Xiaobin, Cheng Xiaoou,Sun Shixin.Turing Compute Model for Non-negative Numbers. Journal of Electronic Science and Technology（已录用）

[5] 陈文宇, 王晓斌, 程小鸥, 孙世新. 形式语言与自动机关于ε的一些问题. 计算机科学（已录用）

[6] CHEN Wen-yu , SUN Shi-xin.Constructing Automaton with the Closure of language calculation. International Computer Conference on Wavelet Active Media Technology and Information Processing, 2006, 232-239(ISTP: 000240252100036)

[7] CHEN Wen-yu , CHENG Xiao-ou,SUN Shi-xin.Code of Universal Turing Machine, 第20届全国计算机新科技与计算机教育学术会议(昆明), 全国计算机新科技与计算机教育论文集（已录用）

[8] 陈文宇. 有限自动机理论(978-7-81114-415-4). 成都：电子科技大学出版社, 2007

[9] 蒋龙龙, 陈文宇. 利用等价类构造有限状态自动机. 计算机科学, 2006, 33(11)：272-274, 277

[10] 陈文宇, 刘井波, 孙世新. 层次分析的神经网络集成方法. 电子科技大学学报, 2008, 37(3)：432-436(EI082711352150)

[11] Wen-Yu Chen,Bin-Wei Yang,Shi-Xin Sun.Section Language Model for Information Retrieval Dynamics of Continuous, Discrete and Impulsive Systems, Series A, Vol.14:503-508

[12] Chen Wen-Yu, Sang Nan, Qu, Hong. Object-oriented distributed debugging event model.电子科技大学学报,2005,34(3):377-380（EI05279197894）

[13] CHEN Wen-yu,WANG Xiao-bin,Sun Shi-xin,LIU Jing-bo.A Method of Remote Fault Diagnosis based on Multilayer SOM. Process CIS-RAM'2008,2008,11:18-21 (EI085211805550)

[14] 陈文宇, 熊志斌, 黄卫华等. 具备智能感知的富文本制作方法. 中国, 发明专利, 2008. 6, 申请号: 200810045515. 8

[15] 王晓斌, 陈文宇, 余盛季等. 编译原理精品课程. Intel-教育部精品课程建设项目

二、参加科研项目

[1] 桑楠, 陈文宇, 左志宏等. 嵌入式构件的语义模型及其运行支撑技术的研究. 863 项目（2007AA01Z131）

[2] 李毅, 詹瑾瑜, 罗克露, 陈文宇等. 实时可信服务软件的构件化研究. 863 项目（2006AA01Z173）

[3] 雷航, 桑楠, 罗克露, 陈文宇等. 面向 PDA 嵌入式 Linux 操作平台. 863 项目（2003AA1Z2210）

[4] 张树人, 腾颖, 陈文宇等. 电子科技大学企业科技创新服务平台. 国家火炬计划（2007GH540017）

[5] 陈文宇, 王晓斌, 屈鸿等. 数据库管理通用平台. 四川省科技厅应用基础研究项目（2006J13-068）

[6] 王晓斌, 陈文宇, 扬宾伟等. 基于 PUA 编码的藏文 Windows 平台解决方案. 四川省科技厅应用基础研究项目（jy029－0692）

三、教学获奖

[1] 2003 年电子科技大学本科课程 A 级教师（形式语言与自动机课程）

[2] 2008 年电子科技大学本科课程 A 级教师（编译原理课程）

[3] 2005 年电子科技大学研究生课程优秀青年教师

[4] 2008 年电子科技大学研究生课程优秀青年教师

四、教材

[1] 王晓斌, 陈文宇. 程序设计语言与编译（第 3 版, 十一五国家规划教材）. 电子工业出版社, 2009

[2] 陈文宇. 面向对象程序设计语言 C++（第 1 版）. 机械工业出版社, 2004

[3] 陈文宇, 白忠建, 戴波. 面向对象程序设计语言 C++(第 2 版). 机械工业出版社, 2008

[4] 陈文宇, 白忠建, 吴劲. 面向对象技术与工具. 电子工业出版社（研究生教材）, 2008

形式语言与自动机理论若干问题研究

作者： 陈文宇
学位授予单位： 电子科技大学

相似文献(10条)

1.期刊论文 刘光武.石晓龙.许进.Liu Guangwu.Shi Xiaolong.Xu Jin 赋权型自动机的不同模型研究 -计算机工程与应用2006,42(11)

自动机理论是理论计算机科学的基础理论之一,在很多领域自动机有着广泛的应用,有穷状态自动机是正则语言的识别机器,通常分为确定型与非确定型两种模型,其识别语言的能力是等价的.赋权自动机是另一类重要的自动机模型,自动机的每条转移规则和状态可以赋以某一代数结构上的某一数值,从而可以计算输入字符串的权值.任何有穷状态自动机都可以视为一特殊赋权自动机,因此赋权自动机功能更强大,应用更为广泛.

2.学位论文 麻勇军 下推自动机的半环方法 2009

当今自动机理论及其相关的形式语言的理论得到了高度的发展，由其衍生出的知识也层出不穷。但经典自动机和语言理论也存在某方面的不足，特别是一些证明从数学的角度看仍不够完美，一个典型的例子就是在自动机的理论中，自动机状态转换的描述，仅仅是定义了状态转换，从数学运算的角度看还不够严密。本文在下推自动机概念的基础上给出了其在半环上的定义，下推转换矩阵的引入，使下推自动机的行为和半环代数理论上的等式建立了联系。从而使下推自动的讨论更加简洁。主要研究内容如下：

⑴介绍了课题研究的对象下推自动机的概念、下推自动机的及时描述、下推自动机所接受的语言等相关概念与半环、半模、收敛等概念。并重点讨论了偏序半环和关于偏序半环上等式的一些性质。最后说明了课题实现的具体目标和意义。

⑵引入了与语言理论密切相关的形式幂级数的概念，并将半环的概念转换到形式幂级数，同时在引入矩阵概念的基础上也将半环的概念转移到矩阵，重点证明了布尔形式幂级数矩阵半环和形式幂级数布尔矩阵半环的子半环同构，进一步结合形式幂级数布尔矩阵半环和分块矩阵的相关理论给出了：与矩阵星运算求解相关的线性系统，并证明了线性系统与矩阵星运算相关的一些定理。

⑶提出了语言半环的概念，证明了它和布尔形式幂级数半环是同构的。在语言半环到语言矩阵半环扩展的基础上定义了下推转移矩阵，进而定义了下推自动机和下推自动机行为。特别是下推转换矩阵的引入，通过一系列矩阵半环同构将下推自动机行为的研究转移到布尔形式幂级数矩阵半环上，最终使下推自动机的计算转变为矩阵半环上下推转换矩阵的乘法和加法运算。

3.期刊论文 严兵.卢朝晖.YAN Bing.LU Zhao-hui 一类有穷自动机的设计 -西华大学学报（自然科学版）2005,24(4)

讨论了一类有穷自动机与形式语言二者之间的关系,给出了一类语言与对应的自动机(包括确定型的有穷自动机与不确定型的有穷自动机)二者之间相互转换的方法,最后指出了这个方法可以适用于类似问题的求解.

4.期刊论文 邱丽萍.朱平 自动机和形式语言结构的理论研究 -江南大学学报(自然科学版)2003,2(5)

利用半群代数理论进一步讨论了自动机和形式语言的理论结构.首先构造了一个简单的有穷自动机的么半群,用代数理论分析了其性质,并推广到非确定性有穷自动机的情况.然后对字母表关于连接运算构成的么半群的性质作了进一步探讨,并给出了二进制串生成的半环,及研究了它的结构理论.

5.期刊论文 杨萌.张琼瑶.苏姣.赵苗.李嵩嵩.YANG Meng.ZHANG Qiong-yao.SU Jiao.ZHAO Miao.LI Song-song 基于超级自动机的语言识别及逻辑推理 -电脑知识与技术2009,5(24)

该文根据自动机的定义,对超级自动机作了特定的定义,并给出了自动机对语言的识别功能的具体步骤.接着由超级自动机的定义,以一个特殊的例子给出了超级自动机的语言识别和逻辑推理功能.

6.学位论文 程伟 非经典自动机与形式语言理论研究 2005

本文主要研究非经典自动机和形式语言理论中的几个基本问题。具体研究内容如下：

在第二章中，我们对量子计算理论方面的研究进展作了一个全面的介绍。就我们所知，这是现有文献中第一篇关于量子计算理论的综述性文章。

经典自动机理论和经典形式文法理论之间有密切的关系。受启于此，在第三章中，从应明生教授提出的基于量子逻辑的自动机理论出发，我们建立了基于量子逻辑的文法理论的一般框架，并且证明了1-值量子正规文法生成的1-值量子正规语言类和1-值量子自动机识别的1-值量子语言类是一致的。为了研究量子计算，Moore和Crutchfield提出了量子有限状态自动机、量子下推自动机、量子正规文法以及量子上下文无关文法的概念。在第四章里，我们证明了正规量子文法生成的语言与正规文法生成的语言是等价的。因此，从生成语言的角度看，正规量子文法并不比经典正规文法强。这意味着可能需要更合适地定义量子文法甚至量子自动机。

在第五章中，我们介绍了一种新的模糊有限状态自动机。对应于经典的Mealy型有限状态自动机，称之为新Mealy型模糊有限状态自动机。从引入的两类状态等价关系出发，我们定义了该自动机的最小化形式，最后得到了该自动机的一种状态最小化约简算法。

在第六章中，我们研究了Mizumoto型模糊有限自动机的状态最小化约简算法。与第五章中的方法类似，我们也首先引入了两种状态等价关系，然后根据这两种等价关系定义了该模糊有限自动机的状态最小化形式，并且给出了一个状态最小化约简算法。

在现有的文献中，除了经典自动机模型外，还存在各种各样的非经典自动机模型，例如概率(或者随机)自动机模型、模糊自动机模型、量子自动机模型等。在第七章中，我们试图抽取出这些自动机模型共有的基本性质，并给出了两种统一表示经典自动机模型和非经典自动机模型的框架，分别为矩阵表示框架和格值表示框架。

7.期刊论文 邱道文 基于量子逻辑的自动机和文法理论 -软件学报2003,14(1)

初步建立了基于量子逻辑的自动机和文法理论的基本框架.引入了量子文法(称为1值文法),特别是证明了任意1值正规文法生成的语言(称为量子语言)等价于某种基于量子逻辑且含动作(的自动机(称为1值自动机)识别的语言,反之,任意1值自动机识别的语言等价于某1值正规文法生成的语言.建立了1值泵引理,并得到量子语言的判定性刻画.最后简要讨论了正规文法与量子文法(即1值正规文法)的关系.因此,为进一步研究更复杂的量子自动机(如量子下推自动机和Turing机)和量子文法(如量子上下文无关文法和上下文有关文法)奠定了基础.

8.学位论文 严奇琦 理论计算机科学中的若干下界结果 2006

本文对理论计算机科学中的下界问题及其意义进行了简要的综述，并阐述了作者在ω-自动机换的状态复杂性和形式语言中starheight问题上的两项研究工作。

在ω-自动机转换上，首选提出了一　证明自动机状态转换复杂性下界的技巧，即full自动机技巧，然后将这种技巧应用到非确定ω-自动机的操作上。具体地，证明了一个Buchi自动机求补的Ω（(0.76n)n）的下界，并且证明了这个下界对于几乎所有ω-自动机的求补和确定化操作都有效。也证明了一个广义Buchi自动机求补的Ω（(nk)）n的下界，而这个下界对于Streett自动机的求补也有效。该项工作发表在了欧洲顶级的ICALP理论会议上，并获得最佳学生论文奖。

关于star height问题，引入了split游戏，一种逻辑中EhrenfeuchtFraisse游戏的变种，并证明了这种游戏能用于分析广义正则表达式的表达能力。也把split游戏推广到了广义ω-正则表达式。为了理解这种游戏如何能被用来攻克著名的困难的star height 2问题，提出并且解决了star height 2问题在ω-语言理论中的一个类似的但较为容易驾驭的变种，即omega power问题。实际上，证明了omega power算子和布尔算子以及连接算子一起无法表达整个ω-正则语言类。这项工作已被著名的Theoretical Computer Science杂志接受。

9.期刊论文 蔡国永.钱俊彦.CAI Guo-yong.QIAN Jun-yan 关于形式语言与自动机理论的教学方法探讨 -高教论坛 2008,""(4)

形式语言与自动机理论是计算机科学与技术专业的一门重要专业基础理论课,搞好本课程的教学,不但能为学生学好本专业后续课程奠定坚实的理论基础,而且有利于培养学生的计算机抽象思维.形式化设计构造的能力,从而适应进一步的学习和工作中本学科专业发展的需要.本文首先对当前该课程教学中存在的一些现象进行分析,然后介绍作者总结出的一些行之有效的教学策略与方法.

10.期刊论文 宋文.严兵.潘世永 对自动机与形式语言中几个问题的思考 -四川工业学院学报2002,21(4)

对连续两个0或连续两个1的语言,给出了它的抽象函数定义,并以Mealy机器作为这一问题的机器模型;给出了(0+1)* 中不含有任何子串ω,ω连续重复出现三次这一结论的证明.