博、硕学位论文保密声明

本论文因涉及国家基金 《引入模糊逻辑的数学公式结构分析与理解研究》项目的研究，有些内容为项目的保密部分。该项目正在进行，尚未结项。因此本论文全文部分需保密。

保密期限：☐ 6个月　☐ 12个月　☑ 24个月

特此声明！

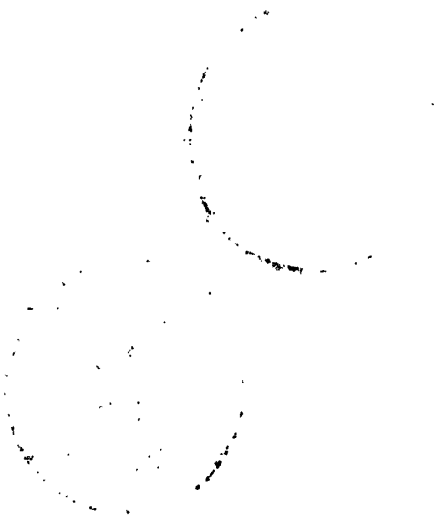
申请人：王思

导师签名：田学东

科技处意见：

学位委员会意见：

申请日期：2008年6月1日



河北大学

学位论文独创性声明

本人郑重声明： 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。尽我所知， 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的研究成果，也不包含为获得河北大学或其他教育机构的学位或证书所使用过的材料。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了致谢。

作者签名：王思 日期：2008年6月1日

学位论文使用授权声明

本人完全了解河北大学有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

本学位论文属于

1、保密 √，在2010年6月1日解密后适用本授权声明。

2、不保密 □ 。

（ 请在以上相应方格内打“√” ）

保护知识产权声明

本人为申请河北大学学位所提交的题目为（ 印刷体数学公式的语法语义分析 ）的学位论文，是我个人在导师（ 田学东 ）指导并与导师合作下取得的研究成果，研究工作及取得的研究成果是在河北大学所提供的研究经费及导师的研究经费资助下完成的。本人完全了解并严格遵守中华人民共和国为保护知识产权所制定的各项法律、行政法规以及河北大学的相关规定。

本人声明如下：本论文的成果归河北大学所有，未经征得指导教师和河北大学的书面同意和授权，本人保证不以任何形式公开和传播科研成果和科研工作内容。如果违反本声明，本人愿意承担相应法律责任。

声明人：王思 日期：2008年6月1日

作者签名：王思 日期：2008年6月1日

导师签名：田学东 日期：2008年6月1日

摘 要

目前主流的OCR（Optical Character Recognition，光学字符识别）技术能够将印刷体文字信息高速、自动地输入计算机，节省了大量的人力资源。然而，对于科技文献中大量存在的结构复杂、含义丰富的印刷体数学公式，OCR 仍然不能较好地进行识别与理解。因此，研究印刷体数学公式识别系统具有重要的理论意义和良好的应用前景。

本文针对印刷体数学公式的分析与理解进行了研究，在对公式结构类型进行归纳总结的基础上，结合数学公式的语法规则和语义知识，对公式结构进行辅助处理，并且设计了融入检错纠错规则的语法树遍历算法，自动纠正字符识别阶段产生的错误。此外，本文还引入了一种数学公式的语法语义分析算法，定义了符合LL(1)文法的表达式构成规则，根据公式上下文情况判别表达式识别的正确性。针对不同类型印刷体文档的对比实验表明，本文设计的具有检错纠错规则的结构分析方法和语法语义分析算法能够取得较高的正确率和较好的执行效率。

关键词 光学字符识别；数学公式识别；结构分析；语法分析；语义分析

Abstract

At present, the mainstream technology of the OCR(Optical Character Recognition) system can save lots of manpower because of its ability of reading printed characters into computer automatically and quickly. However, OCR can not get satisfying results in dealing with the mathematical expressions that exist in scientific and engineering documents because they always have complex structures and various of connotations. To this regard, it will generate important theoretical and practical values to do research on mathematical expressions recognition system.

In this paper, a study on analyzing and understanding of mathematical expressions has been done as follows. First of all, structure of expressions is disposed based on the summarization of all structure types of expressions and the combination of grammatical and semantic knowledge of expressions. At the same time, a traversal algorithm for syntax tree is designed to comply with the error detection and error correction rules so as to correct the errors occurred in the process of automatic recognition. Moreover, an algorithm used to analyze the syntax and semantic knowledge of mathematical expressions is also proposed. This algorithm check the errors of recognition according to the content-free grammar with expressions by through defining the rules of expressions which comply with LL(1) grammar. The results of experiments on various types of documents show that the methods proposed in this paper can achieve higher accuracy as well as better application efficiency.

Key words: OCR; Mathematical expression recognition; Structural analysis; Syntax analysis; Semantic analysis

目 录

第1章 引 言

1.1 研究背景和意义

近年来，随着网络的飞速发展，网络已经成为信息交换与传播的重要手段。数字图书馆和远程教育等领域也随之成为研究热点。而要推动这些领域的发展，关键就是开发出一种简单有效的、能将现有的纸质形式的文档转换成相应的电子文档的方法。只有这样，现在所拥有的大量信息才能够使用计算机进行处理，并使之能够在互联网上传播。现在文字识别的方法很多，也形成了一些比较成熟的软件，它们对印刷体文字（包括英文和汉字）的识别率较高，对手写体文字的识别率也在逐步提高，但是在处理数学公式时却显得稍逊一筹。数学表达式构成了大多数科技工程准则的基本部分，由于印刷体数学公式结构的二维嵌套特性、所包含符号的复杂性及其数学符号语法、语义的多样性，现有的 OCR 技术还不能有效地处理印刷体数学公式，只能将其以图像形式保存，无法实现对公式的编辑，而且以图像形式保存的公式占用存储空间较大，不利于科技文献的网上传输。因此，研究印刷体数学公式识别的自动输入技术对于科技文献的数字化，乃至社会的信息化建设，具有重要的理论意义和良好的应用前景。

本课题来源于河北省科学技术研究与发展计划项目(No.06213598)。

1.2 国内外研究现状

国外于 20 世纪 60 年代后期开始数学公式识别的研究。进入 90 年代，这个领域的研究热度逐渐增加[1]。

早在1968年，Anderson[2]最先采用了自顶向下的分析方法解析数学表达式。该方法从最终目标出发，将问题分为许多个子目标，直到所有的子目标都达到或都不能满足为止。这种目标分割策略的提出为数学表达式识别系统做出了巨大贡献。A. Belaid和J. P. Haton[3]使用自顶向下和自底向上两种方法。在识别出公式各符号后，用自顶向下的方法将表达式分解成子表达式，而用自底向上的方法将子结构连接成较大的结构。这种方法只适用于分析一些简单的数学表达式（如算术表达式和一些三角函数方程）。随后这

种句法解析方法广泛应用于语音识别及图像处理领域。Faure和Wang[4]提出了一种采用各图像基元的边界框直接分析各字符位置关系的方法。这种方法仅用边界框决定符号之间的位置关系是不充分的，有时会出现一些歧义或误判。

进入 90 年代，这个领域的研究热度逐渐增加，仅第一届到第五届 ICDAR（International Conference on Document Analysis and Recognition）大会上就有 12 篇与数学公式识别直接相关的文章。D. Blostein[4]在识别阶段之前设计了水平切分和垂直切分的预处理方法，提出了基于随机文法的二维文法来解析数学公式，并结合图改写方法检测公式分析过程中可能出现的语法错误。R. J.Fatman[5]设计了一种典型的系统，该系统能够成功地将排版好的数学公式转换成 Lisp 表达式。在符号识别阶段，采用了计算 Hausdorff 距离和符号灰度值等方法；在结构分析阶段，运用了句法解析的方法来分析二维的数学公式，通过对操作符的分析提取子表达式。Okamoto 和 Miyazawa[6]使用递归的投影轮廓切分方法并结合空白域的属性来分析数学公式的结构。Winkler[7]等人在识别手写数学表达式时采用了生成图的方法，利用软决策来处理符号关系的模糊性。

二十一世纪以来，随着文档图像处理技术的发展，相关技术得到了充分的研究。在公式处理方面，又有将近 20 篇论文发表，但是仍然没有完整的、满足实际需要的产品出现[8,9]。Huang 和 Kechadi[10]提出了基于基准线和属性串文法相结合的方法，处理联机手写体数学公式结构分析。算法包括结合特征抽取、解析结构树和表达式结构分析三步。并且采用了 MLP(Multilayer Perceptron Netural Network)和 SVM(Support Vector Machines)方法分析判断基线中心字符与其他字符关系，得到较好的效果。Ling Zhang[11]引入了模糊逻辑理论来处理上下标，结构分析结果有了较好的改善。

国内对数学公式识别的研究尚处于起步阶段，相关资料还很欠缺。主要研究机构有南开大学机器智能研究所和中科院自动化所。江红英和靳简明[12]提出了基于统计特征的印刷体数学公式上/下标关系判别方法。该方法将字符识别结果与位置信息相结合，很好的判断出上/下标关系。刘昌平和郭育生[13]提出了一种基于多候选方法的数学公式识别系统。该系统主要包括公式图像预处理,多候选公式符号分割和多候选公式结构分析三个部分。在公式符号切分中,使用三次动态规划方法对公式图像进行多候选公式符号切分。在公式结构分析中,采用层次结构分析方法分析数学公式中的矩阵和子表达式,然后使用LaTex格式和MathType格式表示数学公式的识别结果。哈尔滨工程大学、广西师

范大学、大连理工大学的研究人员也在关注此类研究，发表了相关的论文[14-17]。

总的看来，数学公式识别是一个新兴而富有挑战性的研究方向。如何超越特定对象，研究适应我国文献特点的、能够实际应用的公式识别的系统性解决方案，还有很多有待解决的问题。

1.3 本文的工作

全文的组织结构可以概括如下：

第1章 引言。介绍数学公式识别的研究背景、意义和国内外研究现状。

第2章 印刷体数学公式识别系统。介绍数学公式识别系统的组成，并对结构分析阶段的关键问题进行讨论。

第3章 印刷体数学公式结构分析。设计了基于DBSCAN思想的分层方法，针对公式特点对印刷体数学公式进行结构分析。

第4章 印刷体数学公式语法语义分析。引入LL(1)文法，设计了一种数学公式语法、语义分析方法，定义了符合LL(1)文法的表达式构成规则，根据表达式上下文情况分析判别表达式识别的正确性。另外，结合数学公式的语法规则和语义知识，设计了融入检错纠错规则的语法树遍历算法，自动纠正字符识别阶段产生的错误。设计了带有回溯机制的子表达式提取算法，分析含有绝对值符号的公式的语义。

第5章 实验结果分析。对算法改进前后的结构分析方法进行对比，并对实验结果进行分析。

第6章 结论与展望。对所做的研究工作进行总结，并对今后的研究工作提出建议。

第 2 章 印刷体数学公式识别系统

2.1 印刷体数学公式识别系统的功能与组成

印刷体数学公式识别系统可以完成印刷体文档到电子文档的转变，并实现对印刷体数学公式的自动识别与理解，最终形成可供用户编辑的电子文件。该系统的输入是经过扫描得到的包含数学公式的文档图像，输出为识别出来的数学公式的排版语言(如 Latex)，印刷体数学公式识别系统流程，如图 2-1 所示。

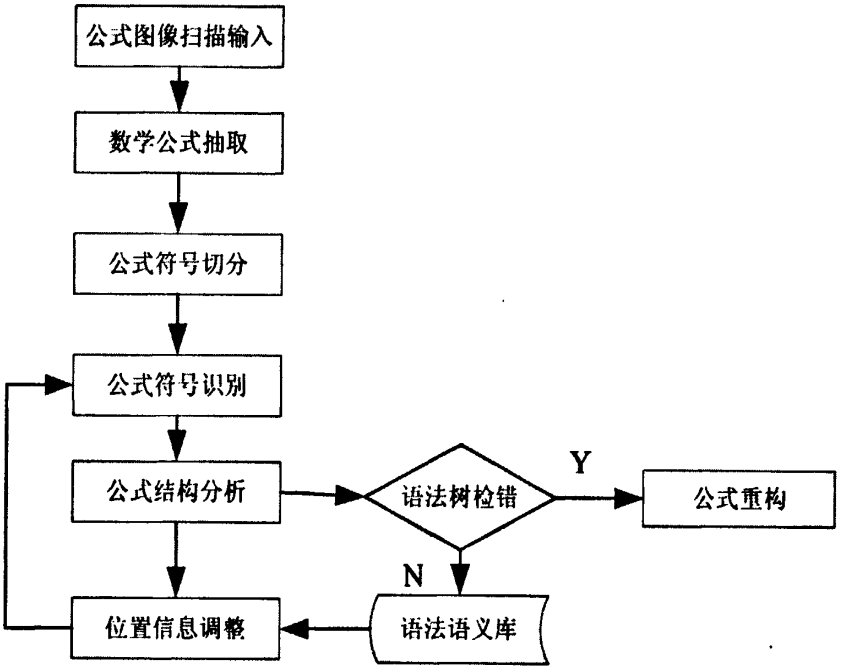


图 2-1 印刷体数学公式识别系统

(1) 数学公式抽取：该模块完成文档中数学公式的定位，以便对数学公式进行单独处理。抽取工作可以减少公式对其他文本的影响，而且还可以提高数学公式字符识别的准确率。公式抽取模块主要分为两部分：对独占一行的孤立公式抽取和混杂在文本行中的内嵌公式抽取。

(2) 公式符号识别：对数学公式抽取模块得到的公式图像进行符号切分，得到待识别单个符号的图像；提取符号特征，在分类器中将提取的特征向量与标准字典中的特征

比较，得到符号图像所对应的代码，即识别结果。如果识别阶段不但能够给出符号的识别结果，还能给出符号的字体、风格、字号等更多信息，将为下一步的结构分析带来更多的便利。

(3) 公式结构分析：结构分析包括版面分析和语法语义分析，该阶段主要利用符号的固有属性，如大小、位置和相应的代码信息，结合公式语法、语义特征，确定公式符号之间空间关系和逻辑关系，并以关系树或分析树表示出来。

(4) 公式重构：根据字符识别与结构分析的结果，构造出来能够重现公式原貌的通用格式文件，并设计实现公式编辑器，方便对公式的编改。

2.2 印刷体数学公式结构分析与理解的关键问题

数学公式的意义是通过公式符号来表达的，数学语言是一种非日常和非自然的语言，其中一部分是被规定或定义的。公式符号有以下特点：

(1) 数学公式包含的符号种类很多，如数字、英文字母、希腊字母以及运算符号，不同类型的符号在公式中充当着不同的角色，并且随着公式的不同而变化，不容易分析。

(2) 公式符号包含的笔画少，结构简单；符号大小不一、相似性高，不易区分。

(3) 数学公式中有很多约定俗成的规范，同样的符号在不同的位置表示不同的数学含义。例如：圆点可能表示乘，也可能表示小数点或者噪声点。又如 dx 可用来表示微分符号或者 d 与 x 相乘。

(4) 公式中包含二维嵌套结构，各符号有着不同的组织规则。对于一些含有特殊符号的公式如 $\int_{+\infty}^{-\infty} i$ 、$\int_a^b f(x)dx$ 中积分符号绑定了不同的符号，具有不同的逻辑关系。

(5) 一些公式字符出现的位置是随机的，需要依赖上下文来分析判断具体的语法与意义。比如：lim，cos，log 等组合符号，在英文论文中很难把它们同其他英文字符区分开。

(6)有些书籍印刷质量偏差，有污点或者出现字符粘连现象影响识别效果，导致结构分析的错误。

(7) 目前对数学公式尚没有规范的排版规则，教科书、报纸、期刊、杂志等不同的

书籍有不同的公式排版规则，这给过度依赖空间关系的结构分析带来了很大困难。

上述这些特点，影响了对印刷体数学公式的结构分析与理解。这些问题如果得不到有效地解决，就会使结构分析阶段产生很多错误，并且进一步影响重构结果。

第 3 章 印刷体数学公式结构分析

3.1 结构分析方法综述

现有的数学公式结构分析方法主要有三种[12]：

第一种是基于几何特征的分析方法。该方法直接根据字符的内容、大小、相对位置以及字符间的空白域等属性判断相邻字符的关系、生成符号组、合并子表达式，从而达到分析数学公式的目的。

Ha[18]提出了解决二维表达式的识别与理解系统，算法有三步：首先提取表达式外接矩形所提供的字符属性信息，根据位置关系对字符进行分解或合并，建立层次结构的表达式，构造初始表达式树，然后通过通用的语义规则检测初始表达式，最后修订层次结构，重构初始表达式树。

Twaakyondo[19]提出了自顶向下和自底向上的结构分析策略。使用较大的字符和空白域水平或者垂直分割表达式，递归的处理每个子表达式的结构，降低了结构分析的复杂性。

Okamoto[20]等采用根据表达式类型进行结构分析的方法。将表达式分为两类：一类是含有分式、根号、括号或数组的子表达式，另一类是带有上下标或上下部的子表达式。对于第一类的子表达式定义了一些核心符号，如“(”，“-”，“$\sqrt{\ }$”，“[”，“{”，“|”等，然后通过查找这些核心符号来分析表达式；对于第二类子表达式则通过查找上/下标和上/下部的符号区域来分析表达式。

Lee 和 Lee[21]利用传统的统计方法识别单个字符和数字，然后用程序导向方法把二维结构的数学公式转化为一维的字符串表示。

Lee 和 Wang[22]采用了启发式规则来检测识别错误，规则如下：（1） 任何一个双目运算符都必须有两个操作数；（2）数学函数名的纠错规则；（3）数字没有下标；（4）在同一操作数中的符号具有相同的性质。

Fukuda[23]提出了“数学元件”的概念。数学元件是一组数学符号，它包括一个母字符和若干子数学元件。根据元件之间的位置关系，计算出每种关系的惩罚值，具有最小

惩罚值的关系被认为是正确的关系。

第二种是基于文法分析的结构分析方法。通过定义文法来分析数学公式的含义。文法分析的语义识别能力较强，但定义文法是很复杂和困难的。传统的产生式文法虽然可以描述一维的文本，但是不能很好地表示具有二维结构的数学公式。研究人员采用不同的方法扩展传统的产生式文法，使其具有表达二维数学公式的能力。

Pfeiffer[9]设计了采用上下文无关文法的分析器，提高了结构分析的独立性，该分析器仅仅具有理论意义，没有实际结果。

Chan 和 Yeung[8]提出了公式分析中，常伴随发生的 4 类错误：词法错误、句法错误、语义错误和逻辑错误。词法错误主要由于手写或者排版乱引起，这类错误很难纠正。句法错误主要是指括号不匹配，缺少操作数等。语义错误是指操作数应用于不合适的操作域、函数名识别错误。逻辑错误类似于 1+2=5，文中没给出相应的解决方法。

Dimitriadis[24]除了定义句法规则外，还定义了语义规则。句法规则用来将复杂表达式分解成简单表达式，而语义规则定义了句法规则各部分在坐标空间的相对位置关系。

Toumit[25]提出了初始化模型来分析自动识别带有公式的文本，并使用树结构表示抽取出的数学公式。采用算符优先级规则结合递归算法逐级分析复杂的数学公式，简化了分析的复杂度。

Grbavec[26]采用图改写的方法分析二维数学表达式。图改写方法依据数学表达式操作码优先级别和操作码作用域的运算规则来分析表达式，并且能够检查出某些不合语法的公式，比单纯依靠句法和基于结构的分析方法更加有效。图改写方法对分析二维模式的图形提供了坚实的理论基础和模型。

第三种是几何特征和文法相结合的结构分析方法。Pagallo[27]应用约束属性文法分析，把重要字符作为关键字，然后运用关键字的特性进行分析，避免了回溯，但没有测试结果。

Chaudhuri 和 Garain[28]采用符号识别的方法来检测嵌入式表达式，如果存在某个特殊符号如$\sum$、$\int$、∞等，则说明存在数学表达式，进而使用启发式算法来分析整个表达式。

综上所述，基于几何特征的分析方法过分依赖符号的排版信息，而基于文法的结构分析方法则需要利用公式符号识别结果，文法总结困难。将几何特征和文法相结合的方

法弥补了两种方法的不足，具有更好的鲁棒性。

3.2 结构分析预处理

在对公式进行结构分析之前，公式符号识别阶段已得到了数学表达式中每个符号的固有属性，如大小、位置和相应符号的 ASCII 码。而结构分析阶段是要在此基础上分析得到符号的排列层次，并将其层次结构表示成为结构关系树。

结构分析阶段所需要的符号信息不能完整无误的通过识别阶段得到，因此需要先进行结构分析预处理。预处理是对识别阶段得到的符号特征进行加工，以适应结构分析的需要，包括符号位置特征的提取、字符的水平膨胀处理、空间关系的分层。

3.2.1 符号位置特征提取

印刷体数学公式识别阶段提供每个字符的外接矩形的坐标来描述字符属性信息，结构分析阶段就是利用这些信息来确定相关字符的位置关系。如图 3-1 所示。

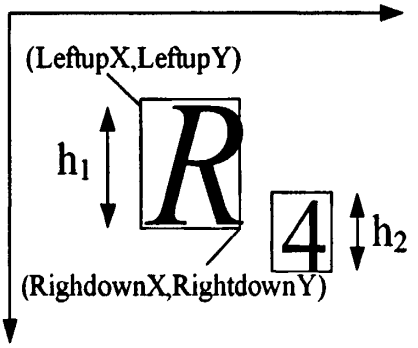


图 3-1 符号位置特征定义

其中，(Leftup$X$,Leftup$Y$) 表示字符外接矩形的左上角坐标，(Rightdown$X$,Rightdown$Y$)是字符外接矩形的右下角坐标，h_1 和 h_2 分别表示两个符号外接矩形的高。

3.2.2 空间关系分层

数学公式具有二维嵌套结构，相邻符号大小和位置的相关性也不明确。为了获得更多的公式属性信息，便于高效的进行版面分析，采取以下两步预处理：

第一步：对抽取后的公式进行符号水平膨胀处理[29]。设 A 为抽取后得到的二值图像集合，B 为水平动态因子，A 被 B 水平膨胀记为 $A \oplus B$，则水平膨胀后的二值图像为 Z，则 $\{Z \mid (\hat{B})_Z \cap A \neq \phi\}$，且 $\{Z \mid Z = a+b, a \in A, b \in B\}$。

第二步：利用基于 DBSCAN 聚类思想的方法对图像 Z 进行分层处理，锁定公式中处于同一水平层次的符号坐标范围。首先引入 DBSCAN 算法的数学建模和相关概念[30]：

定义 3-1（密度）空间中任意点的密度是以该点为圆心，以 EPS 为半径的圆区域内包含点数目。

定义 3-2（邻域）空间中任意一点的邻域是以该点为中心、以 EPS 为半径的圆区域内包含的点集合，记作 $NEps(p) = \{q \in D, dist(p,q) \leq Eps\}$。这里 D 为数据库。

定义 3-3（核心点）空间中某一点的密度如果大于某一给定阈值 $MinPts$，则称该点为核心点。

定义 3-4（直接密度可达到）点 p 从点 q 直接密度可达，若它们满足：

(1) p 处于 q 的邻域中，即 $p \in NEps(q)$；

(2) q 是核心点，即 $NEps(q) \geq MinPts$，即 $NEps(q) \geq MinPts$。

定义 3-5（密度可达到）点 p 从点 q 密度可达，若 $(p_1, p_2, \ldots p_n)$，其中 $p_1=p$，$p_n=q$，且有 p_i 从 p_{i+1} 直接密度可达。

定义 3-6（密度连接）点 p 和点 q 是密度连接的，若对任意的 o，使 p 和 q 都从 o 密度可达。

定义 3-7（基于密度的簇）是基于密度可达性的最大的密度相连对像的集合。

DBSCAN 算法主要反映数据点集的空间分布以及各数据点之间的关系，从而为聚类操作提供支持和便利。我们基于 DBSCN 聚类算法思想，对膨胀后的公式进行密度可达簇的合并，得到公式的层次信息，使得公式化繁为简。

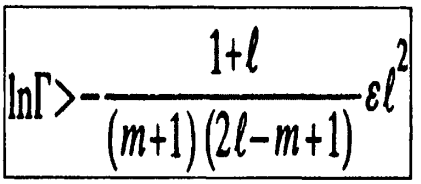
算法基本思路如下：对给定图像自上而下，从左向右依次扫描，遇到第一个黑色像

素点就设其为核心点（CorePoints）或种子点，设 CoreP=1（如图 3-2 所示）。通过区域扩张查询得到该点的八邻域上的密度可达的黑像素点，把这些密度可达的点作为下一轮考察对象（即种子点），并通过不断地对种子点进行区域扩张查询，直到找到一个完整的密度可达簇，比较密度可达簇中所有点的位置信息，得到各个密度可达簇的外接矩形用 (Leftup X, Leftup Y, Rightdown X, Rightdown Y) 表示。然后依此过程寻找其他的密度可达簇，最后剩下的不属于任何簇的点即为噪声点。

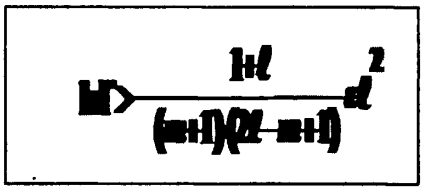
0	1	2
7	CoreP	3
6	5	4

图 3-2 种子点八邻域密度可达示例

该算法只需对整个图像数据扫描一遍就可以得到公式各个层次所包含字符的范围，为下一步分层处理的结构分析方法奠定了基础。算法的计算时间复杂度为 $N*\log(N)$，执行效率较高。分层效果如图 3-3 所示。



（a）公式图片



（b）分层效果

图 3-3 基于 DBSCAN 思想的子表达式范围锁定

3.3 结构分析方法

各类公式空间位置的复杂程度和嵌套方式各不相同，印刷体数学公式结构分析方法就是要用形式化语言来描述这些公式符号的空间位置关系。首先建立一个结构分析模型，抽象出符号各属性特征和逻辑关系，结合语义知识完成公式的结构分析。

3.3.1 结构分析模型

结构分析的输入是符号序列，其中每一个符号都有 ASCII 码值和外接矩形坐标信息

(Leftup X, Leftup Y, Rightdown X, Rightdown Y)。经过预处理，每一个符号的信息位扩展为七位，其 VC++描述如下：

```
struct SymbolNode{
        int     code;//符号的码值
        int     CentroidX;//中心坐标
        int     CentroidY;
        int     LeftupX;//符号左上角坐标
        int     LeftupY;
        int     RrghtdownX;//右下角坐标
        int     RightdownY;  };
```

结构分析输出的是一个结构关系树（以下简称语法树）。语法树中的每一个节点表示一个符号，除了符号自身的属性信息外，还包括 6 个位置指针，分别代表 6 种不同的空间关系，up、super、right、subsc、down、inclusion 分别表示上部、上标、水平、下标、下部以及包含关系。如图 3-4 所示。

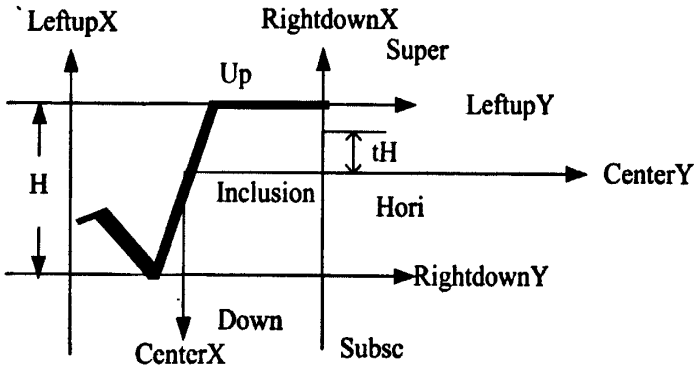


图 3-4 空间关系语法树

例如公式 $S_{n^2}=n^3$ 通过结果分析后最终生成的语法树如图 3-5 所示。

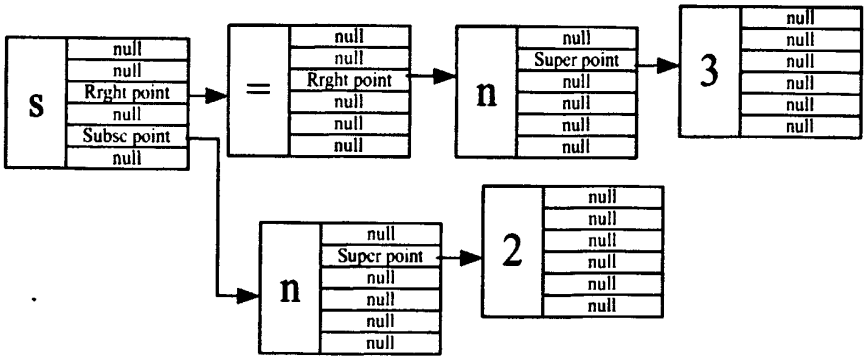


图 3-5 公式语法树

语法树中节点结构用 VC++语言描述如下：

```
struct structualTreeCode{
        int code;
        int CentroidX;
        int CentroidY;
        int LeftupX;
        int LeftupY;
        int RightdownX;
        int RightdownY;
        structualTreeCode* up;
        structualTreeCode* superscript;
        structualTreeCode* right;
        structualTreeCode* inclusion;
        structualTreeCode* subscript;
        structualTreeCode* down;
};
```

3.3.2 结构分析算法

利用以上预处理过程获得的公式字符属性和层次信息，结构分析算法主要分为五个步骤：

(1) 根据识别阶段获得的公式所有字符的属性，按照 Leftup$X(s_i)$ 升序排列得到数组{Array}，结合空间关系分层处理数学公式中各模块所含的字符，把公式化分为各个子表达式，分别写入数组{$Array_1, Array_2, ...Array_n$}；

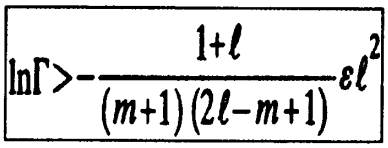
(2) 通过 StructuralAnalysis()函数处理包含公式最左侧字符的子表达式{$Array_i$}中各字符位置关系，并令其为整个公式的主导子表达式；

(3) 循环调用处理函数 StructuralAnalysis()处理其余数组中所含字符的位置关系；根据子公式模块与主导子表达式模块的位置关系进行匹配，组合完成整体结构分析；

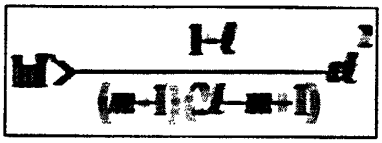
(4) 对公式进行空间关系分层处理后存在的不属于任何子表达式的单个字符，采用最近邻原则对应到已处理字符的五种位置关系指针中，（除去水平指针）最终生成公式结构语法树。

分析过程说明如图 3-6 所示，首先处理簇一所包含字符的位置关系，作为公式主导子表达式，然后分析簇二和簇三与主导子表达式的位置关系，进一步组合完成整体结构

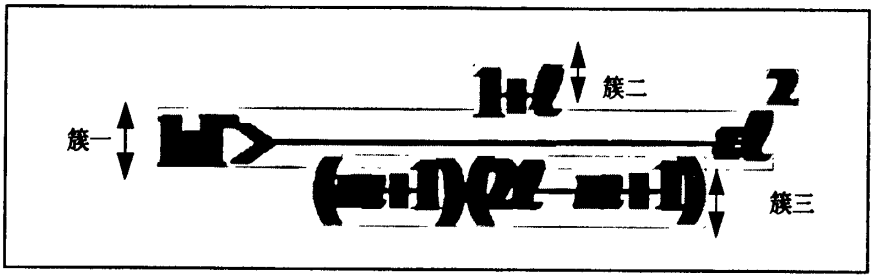
分析。



(a) 原扫描图片



(b) 水平膨胀处理效果图



(c) 分层处理

图 3-6 公式分层处理过程

第 4 章 印刷体数学公式语法语义分析

4.1 数学公式语法理论

数学语言是表达数学思想的专门语言，具有抽象性、准确性、简约性和形式化等特点，是一种以符号表达为主的特殊语言。具体可分为符号语言，文字语言和图表语言三类[31]。符号语言是数学中通用的、特有的简练语言。按照感知规律，数学符号分为三种：象形符号、缩写符号和约定符号。象形符号是由数学对象的空间位置结构或数量关系经抽象概括得到的各种数学图形或图式，再经缩小或改造而形成的一类数学符号，如∠,∇,Θ⊥等都是原形压缩改造，属于象形符号。缩写符号是由数学概念的西文词汇缩写或加以改造而成的符号，比如函数 f（function）、极限 lim(limit)、最大 max(maximal)、存在∃（exist）、任意∀（any）、等符号均为此类。约定符号是数学共同体约定的，具有数学思维合理性、流畅性的数学符号，如运算符号+、-、*，≅(全等)、≈(约等)、>(大于)、<(小于)，等均属于此类[31]。由各种符号按照数学的逻辑意义和规则组合成各种符号串或式子，从而构成数学语言或数学句子。所谓的逻辑意义和规则是指数学中的一些规律或者原理法则。

数学语言的语法是指一组规则，它赋予语言以结构系统，必须从语义的角度看语法及结构，从而找出可遵循的原则。语法研究中，实现语义、表达与语法结构三者的结合，使得语义成为广义的语义，是和语法结构组合有关的意义，和语法结构项之间的组合有关的意义以及和表达有关的语用意义[31]。数学语言是不断发展变化的，某些数学句子是否合乎“语法”，有时也受语义特征的影响。因此，要注意语义特征的发展和变化。

阐明语法的一个工具是文法，这是形式语言理论的基本概念之一。使用文法作为工具，不仅为了严格地定义句子的结构，也是为了能用适当数目的规则把语言的全部句子描述出来，是以有穷的集合刻画无穷的集合的工具。

自从乔姆斯基建立形式语言的描述以来，形式语言的理论发展很快。乔姆斯基把文法分成四种类型，即 0 型、1 型（上下文有关文法）、2 型（上下文无关文法）、3 型（正则文法）。这几类文法的差别在于对产生式施加不同的限制。以下主要介绍上下文无关

文法[32]：

定义 4-1 文法 G 定义为四元组（V_N, V_T, P, S），其中 V_N 为非终结符号集；V_T 为终结符号集；P 为产生式（也称规则）的集合；V_N、V_T 和 P 是非空有穷集。S 称为识别符号或者开始符号，它是一个非终结符，至少在一条规则中作为左部出现。若 P 中的每一个产生式 $\alpha \to \beta$ 满足：α 是一个非终结符，$\beta \in (V_N \cup V_T)^*$，则此文法为上下文无关文法。

定义 4-2：设 G(V_N, V_T, P, S)是上下文无关文法，

$\mathrm{FIRST}(\alpha) = \{a \mid \alpha \overset{\bullet}{\Rightarrow} a\beta, a \in V_T, \alpha, \beta \in V^*\}$，若 $\alpha \overset{\bullet}{\Rightarrow} \varepsilon$，则规定 $\varepsilon \in \mathrm{FIRST}(\alpha)$。

定义 4-3：设 G(V_N, V_T, P, S)是上下文无关文法，$A \in V_N$，S 是开始符号，

$\mathrm{FOLLOW}(A) = \{a \mid S \overset{\bullet}{\Rightarrow} \mu A \beta, a \in V_T, \text{且} a \in V_T, a \in \mathrm{FIRST}(\beta), \mu \in V_T^*, \beta \in V^+\}$，

若 $S \overset{\bullet}{\Rightarrow} \mu A \beta$，且 $\beta \overset{\bullet}{\Rightarrow} \varepsilon$，则 $\# \in FOLLOW(A)$。

也可以定义为：$\mathrm{FOLLOW}(A) = \{a \mid S \overset{\bullet}{\Rightarrow} \ldots Aa \ldots, a \in V_T, \text{且} a \in V_T\}$，若有 $S \overset{\bullet}{\Rightarrow} \ldots A$，则规定 $\# \in \mathrm{FOLLOW}(A)$。

定义 4-4：给定上下文无关文法的产生式 $A \to \alpha, A \in V_N, \alpha \in V^*$，

若 $\alpha \overset{\bullet}{\Rightarrow} \backslash\varepsilon$，则 $\mathrm{SELECT}(A \to \alpha) = \mathrm{FIRST}(\alpha)$，如果 $\alpha \overset{\bullet}{\Rightarrow} \varepsilon$，则

$\mathrm{SELECT}(A \to \alpha) = (\mathrm{FIRST}(\alpha) - \{\varepsilon\}) \cup \mathrm{FOLLOW}(A)$。

定义 4-5：一个上下文文法是 LL(1)文法的充分必要条件是，对每个非终结符 A 的两个不同产生式，$A \to \alpha, A \to \beta$，满足 $\mathrm{SELECT}(A \to \alpha) \cap \mathrm{SELECT}(A \to \beta) = \phi$，其中 α, β 不能同时能 $\overset{\bullet}{\Rightarrow} \varepsilon$。LL(1)文法的含义是：第一个 L 表明自顶向下分析是从左向右扫描输入串，第二个 L 表明分析过程中将用最左推导，1 表明只需向右看一个符号便可决定如何推导即选择那个产生式（规则）进行推导。

4.2 基于 LL(1)文法的语法识别算法

LL(1)文法是一种自上而下的语法分析方法，广泛应用于描述自然语言和程序设计语言的语法分析。文献[33]提出了一种引入 LL(1)文法的数学公式结构分析方法，该方

法定义了符合 LL(1)文法的数学表达式构成规则，把无序的符号串通过语法分析器映射为符合表达式构成规则的语法关系树，最终被转变成可编辑的 LATEX 公式格式，在公式重组方面取得了良好的效果。本文基于 LL(1)文法设计了数学公式语法识别算法，定义了符合 LL(1)文法的数学公式语法、语义规则，通过对数学公式的语法、语义分析，实现了对数学公式的语法检错功能。依据数学公式语法识别算法构建的语法分析器包括以下四个部分：

(1) 预测分析程序

(2) 预测分析表

(3) 分析栈

(4) 符号栈

语法检错过程为：首先建立相应的语境无关语法及词典，如表 4-1 所示，然后引入一种自顶向下，自左向右的非确定性识别方案，最后针对一个公式讨论其语法分析步骤。

表 4-1 一部语境无关语法及其配合的字符串集合

(0) $S->ET$ (1) $T->OET$ (2) $T->\varepsilon$ (3) $E->FM$ (4) $M->OFM$ (5) $F->${函数名} (6)$M->\varepsilon$ (7)$F->(E)$ (8)$O->${操作符} (9)$M->$字母|数字}	
sin	F
a	M
b	M
*	O
(	(
)	)

上述文法，终结符 $V_n=\{S,E,T,O,F,M\}$，S 代表表达式初始态，E 代表子表达式，T 代表嵌套子表达式，O 代表操作符，F 代表函数名，M 代表字母或者数字；非终结符 $V_t=\{Operater,Compare,Symbol,Number,(,),[,],|,=\}$，其中：*Operater* 代表操作符集合：{±, ∓, ×, *, +, −}；*Symbol* 代表字符集合：{*a*, *b*, ...*z*, *A*, *B*, ...*Z*}；*Compare* 代表逻

辑操作符集合：{≤, ≥, ≪, ≫, ≈, ~, ≃, ≅, ≠, ≡, ≜}；*Number* 代表数字集合：{0, 1, ...9}。自顶向下，自左向右判别公式语法的方案[34]如表 4-2 所示。

表 4-2 自顶向下判别公式语法的方案

说明	检验某一公式字符串在所定义的语境无关语法中是否合法	
背景条件	1. 一部语境无关语法 2. 字符串集合	
输入	待分析公式所包含的字符串	
工作结构	当前位置	输入公式字符串某一符号，初始时为 1
	规则串	由句法范畴所组成的字符串，初始时由本文法起始符号这一单个字符号组成，即 S
算法步骤	重复做： 规则串的第一符号是 1. 一个终结符，或者 2. 一个语法范畴 则 如此符号与当前位置的字符串相匹配，则将规则串的第一字符消去 如当前位置为输入数学表达式字符串的最终符号， 如规则串为空，则成功；否则，失败。 否则，做 在语法的规则中选择任一其左端为此符号的规则，并构造余串。 余串由下列两部分组成 1. 所选定规则的右端； 2. 前一规则串消去第一字符号所剩下的子字符串	
中止条件	如果余串为空，而当前位置尚未达到最后，则失败。	

设输入的、待识别词串为 T，其当前位置用 n 表示，T(n)即为待识别公式字符串中当前正在加工的字符，公式字符串的长度约定为 Maxn 。当自顶向下操作时，规则串一

般用R_s表示；并以$R_s(1)$表示规则串的第一个字符，而R_{s-1}表示规则串中消去$R_s(1)$后所得的词串。数学公式语法识别的规则集合$R=\{R_1,R_2,...R_n\}$表示，一个规则 R 的左、右两端则分别用 L(R)、R(R)来表示。如规则R_i为规则集合中的一条，则用$R_i \in R$来表示。空字符用ε表示，#分析栈判空标志。语法分析流程[34]如图 4-1 所示。

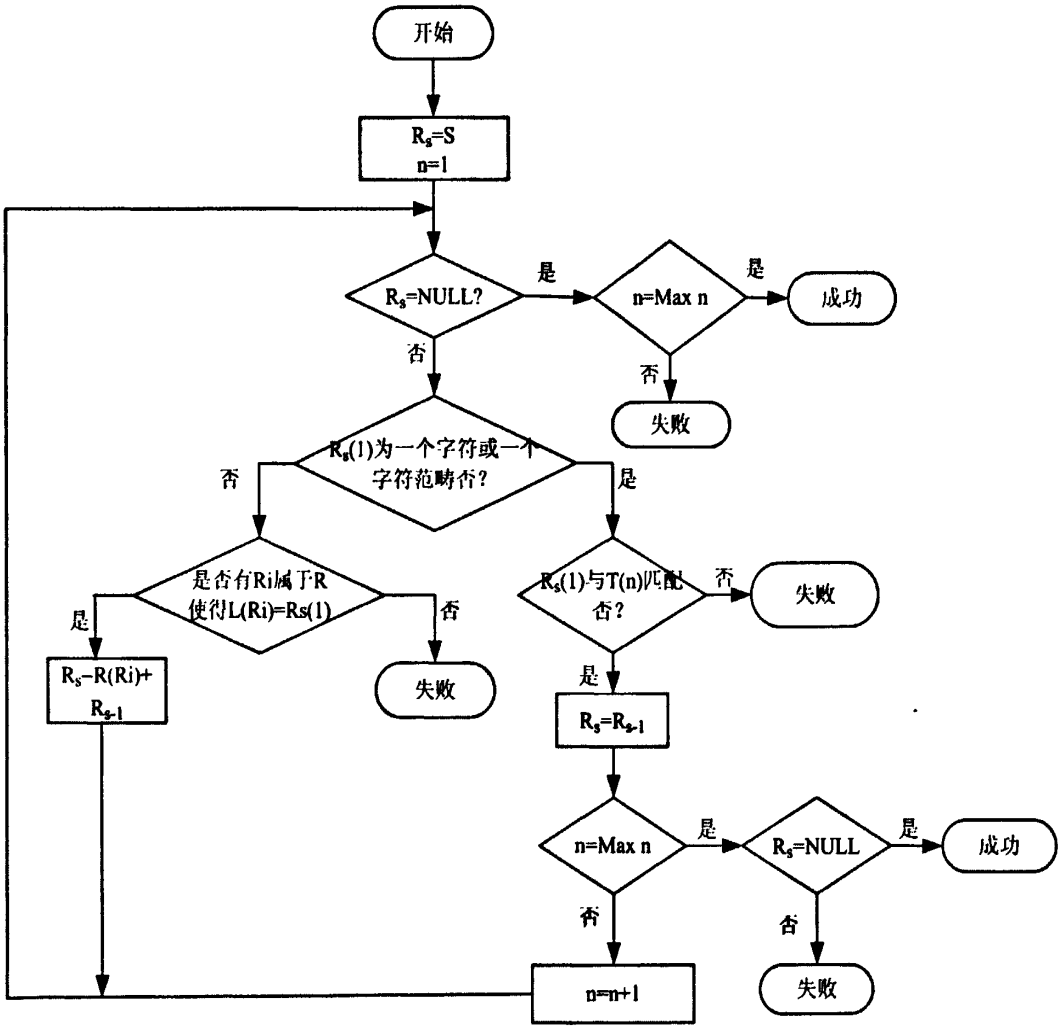


图 4-1 公式语法判别流程图

4.3 语法识别文法的应用

1.合法公式应用举例

检验公式 lna* ln(lna)，分析步骤如下：

(1) 文法 G(*S*)：

(0) *S*->*ET*

(1) *T*->*OET*

(2) *T*->ε

(3) *E*->*FM*

(4) *M*->*OFM*

(5) *F*->ln

(6) *M*->ε

(7) *F*->(*E*)

(8) *O*->+

(9) *O*->*

(10) *M*->*a*

(2) 所得 FIRST 集为：

FIRST(*S*)={l ,(}； FIRST(*E*)={l ,(}； FIRST(*T*)={ * ,+ , #}；

FIRST(*O*)={* , +}； *FIRST*(*F*)={l ,(}； FIRST(*M*)={+ , *, # , a}

(3) 所得 FOLLOW 集为：

FOLLOW(*S*)={#}； *FOLLOW*(*E*)={* , +, #,) }； FOLLOW(*T*)={ #}；

FOLLOW(*O*)={l ,(}； *FOLLOW*(*F*)={*, +, a,)}； FOLLOW(*M*)={* , +, # ,)}

构造预测分析表如表 4-3 所示，公式语法识别过程如表 4-4 所示。

表 4-3 预测分析表

	l	n	a	+	*	(	)	#
S	R(0)					R(0)		
E	R(3)			R(1)		R(3)		
T				R(9)	R(1)			R(2)
O					R(8)			
F	R(6)					R(7)		
M			R(10)	R(5)	R(5)		R(5)	R(5)

注：其中空白处是 ERROR。

表 4-4 公式 lna* ln(lna)分析过程

步骤	分析符号栈	当前指示字符	使用产生式序列号
0	#S	l	--
1	#TE	l	0
2	#TMF	l	3
3	#TMnl	l	6
4	#TMn	n	出栈、后移
5	#TM	a	出栈、后移
6	#Ta	a	10
7	#T	*	出栈、后移
8	#FEO	*	1
9	#FE*	*	9
10	#FE	l	出栈、后移
11	#FMF	l	3
12	#FMnl	l	6
13	#FMn	n	出栈、后移
14	#FM	(	出栈、后移
15	#F	(	5
16	#)E(	(	7
17	#)E	l	出栈、后移
18	#)MF	l	3
19	#)Mnl	l	6
20	#)MN	n	出栈、后移
21	#)M	a	出栈、后移
22	#)a	a	10
23	#)	)	出栈、后移
24	#	#	出栈、后移
25	#	#	分析成功
lna* ln(lna)#符合给定文法			

2. 非法公式语法检测举例

在数学公式中，不同的字符以某个核心字符为基点，根据一定的联系，按照一定的原则，从不同的范畴进行不同的组合，从而构成一个完整的义项，体现出表达式的意义。数学公式中常用核心字符的统计如表 4-5 所示。

表 4-5 数学公式中常用核心字符的统计

sin（正弦）	arcsec（反正割）	arth（反双曲正切）	arch（反双曲余弦）
cos（余弦）	arccsc（反余割）	arcth（反双曲余切）	arcsh（反双曲正弦）
tan（正切）	arctan（反正切）	sech（双曲正割）	arsech（反双曲正割）
ctg（余切）	arcctg（反余切）	csch（双曲余割）	arcsch（反双曲余割）
sec（正割）	arccos（反余弦）	lg（常用对数）	ln（自然对数）
csc（余割）	arcsin（反正弦）	th（双曲正切）	rot（旋度）
grad（梯度）	div（散度）	exp（指数）	lim（极限）
log（对数）	arg（幅角）	||（包含符号）	$\sum$（求和符号）
$\int$（积分符号）	$\sqrt{}$（根式）	$/$（分式）	$\prod$（求积符号）

将以上的统计信息作为文法的初始化词典，当分析具体数学表达式的时候进行模式匹配和语法检错。常见语法错误类型统计如表 4-6 所示。

表 4-6 常见语法错误类型统计表

粘连字符导致误识	$\sin x \rightarrow sim(nx$粘连$)$;
字体影响	$\arcsin - arC\sin;\cos - cOs$
括号匹配	$(l\ +a) \rightarrow ((\ +a)$
语义错误	$w/z \rightarrow w\|z$
复合字符串误识	$\sin \rightarrow 5in;\lim \rightarrow 1im;\lim \rightarrow l1m;\sec \rightarrow soc;\cos \rightarrow ccs$ $\log \rightarrow loa;\ln \rightarrow 1n;\tan \rightarrow +an$
操作数应用于不合适的操作域	$\cos? = b^2 + c^2 - a/2bc$

通过基于 LL(1)的语法识别算法可以检测出以上六类语法错误。举例说明：输入公

式 sina*5inb，语法检错过程如表 4-7 所示。

表 4-7 sina*5inb 语法检错过程

步骤	分析符号栈	当前指示字符	使用产生式序号
0	#S	s	--
1	#TE	s	0
2	#TMF	s	3
3	#TMnis	s	5
4	#TMi	i	出栈、后移
5	#TMn	n	出栈、后移
6	#TM	a	出栈、后移
7	#Ta	a	9
8	#T	*	出栈、后移
9	#TEO	*	1
10	#TE*	*	8
11	#TE	5	出栈、后移
无可用产生式，此公式不是合法表达式			

注：预测分析表见表 4-3 此处略。

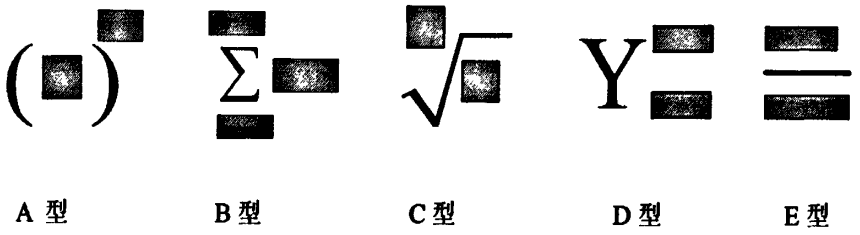
4.4 语法树检错纠错算法

4.4.1 表达式语义特征提取

研究数学语言的意义和理解要从语义范畴入手来寻找语法形式，使语法成为语义支配下的自觉行为。要注意数学语言的语义转换，数学语言符号引入的自然性，以及数学语言句法特点分析[35]。因此，研究语法必须研究语义，研究语法必须结合语义[36]。

国外关于语义特征的分类对我们研究数学公式的语义特征具有启发作用。根据这种思路，我们可将数学公式的“语义特征”分为两种：自然性语义特征和组合性语义特征。

自然性语义特征是从公式结构类型和逻辑意义分解出的语义特征。根据常见公式结构特征，我们分为五个类型如图 4-2 所示。



A 型　　B 型　　C 型　　D 型　　E 型

图 4-2 公式结构特征

组合性语义特征是从组合角度划分出来的语义特征，它影响到某个字符跟表达式中有联系的其它字符搭配的语义特征。这样的语义特征以字符的自然性意义为基础，但只有与其它词语相联系时才显现出来，因此称为组合性语义特征。例如，函数名的组合，绑定字符的空间位置关系。

4.4.2 语法检错纠错算法

在公式识别与分析阶段常出现各种类型的错误，识别中的检错与纠错也是印刷体数学公式识别系统中一个重要的组成部分。根据以上数学公式的语义特征，我们在结构分析后期设计了公式语法检错纠错算法，提高了分析的正确率。算法流程如图 4-3 所示。

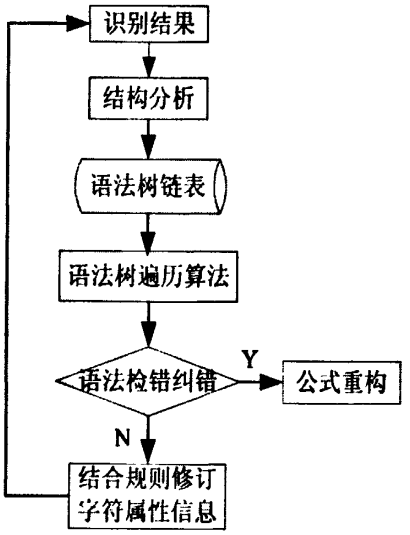


图 4-3 融入语法检错纠错的结构分析方法流程图

算法步骤如下：

第一步：输入公式语法树，调用 RecursionDeep(StructureTreeCode* *p*,int *flag*) 语法

树遍历函数处理公式语法树中的各个结点。其中：StructureTreeCode* *p* 表示语法树链表 m_list 的头指针 *P*，*flag* 值用来标记表示遍历七个方向的标志，初始值为 0。如果 *flag=0*，则调用 rowdeep(*q*)函数处理处于同一层的字符。

第二步：结合上节总结的数学公式语义特征，构造语法检错纠错规则，在 rowdeep(*q*)函数遍历符号的同时，调用 CheckErrors(structualTreeCode *q) 函数，对公式结构树进行语法分析。如果发现语法错误，则根据上下文环境，结合识别阶段提供的信息修改字符属性，反馈到结构分析阶段重新分析。

第三步：判断语法树中各个结点的属性值，分别是上部、上标、包含、下标、下部和左上部六个方向的值是否为空，不为空的则转入处理各个方向子表达式的函数。up!=NULL 转入处理上部子表达式的函数 updeep(*q)。同理，supscrip!=NULL 转入处理上标子表达式的函数 supdeep(*q)；include!=NULL 转入处理包含子表达式的函数 indeep(*q)；subscript!=NULL 转入处理下标子表达式的函数 subdeep(*q)；down!=NULL 转入处理下部子表达式的函数 downdeep(*q)；leftup!=NULL 转入处理左上部子表达式的函数 leftupdeep(*q)。

第四步：字符各个方向的子表达式处理完毕，返回到递归函数 RecursionDeep(StructureTreeCode* *p*,int *flag*)，继续处理下一个结点，直到结点右部为空，没有链接属性表明整个语法树遍历完毕。

4.4.3 语法检错纠错规则

为了提高算法的适应性，我们采用大量不同类型的公式进行了实验，针对实验中出现较多的问题，不断丰富语法检错纠错规则，有效提高了结构分析的准确率。检错纠错规则如下：

一、字母右部语法判决规则

目前对数学公式尚没有规范的排版规则，排版的不规则或者扫描样张倾斜都会给过多依赖位置信息的公式结构分析带来困难，容易造成上/下标与水平之间位置关系的混乱，导致结构分析结果错误。图 4-4 所示的扫描的样张中，由于排版的不规则造成字符“d”、“t”与各自的平方项“2”的质心位置关系模糊。错误的结构分析结果如图 4-4

所示。

$$m\frac{d^2y}{dt^2}=-\frac{kmM}{y^2}$$

图 4-4 公式字符外接矩形

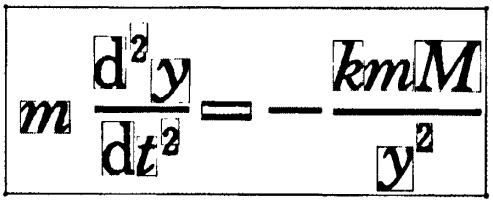
$$m\frac{d2y}{dt2}=-\frac{kmM}{y^2}$$

图 4-5 语法错误的结构分析结果

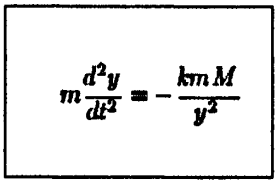
为了解决上述问题，加入字母右部语法判决规则如下：

```
if ((*q).code>='a'&&(*q).code<='z'||(*q).code>='A'&&(*q).code<='Z')
{
   if ((*q).right!=NULL&&(*(*q).right).code>='1'&&(*(*q).right).code<='9')
    {
       Search(*q.right); //在识别结果中寻找与q的右部字符具有相同minx的字符
      if (buffsymbol.leftupy<(*q).miny)//提升字符的Y坐标
      {
          bufsymbol.leftupy=bufsymbol.leftupy-Δt;
         bufsymbol.rightdowny=bufsymbol.rightdowny-Δt;
      }
       if (buffsymbol.leftupy>(*q).maxy)//降低字符的Y坐标
       {
          bufsymbol.leftupy=bufsymbol.leftupy+Δt;
         bufsymbol.rightdowny=bufsymbol.rightdowny+Δt;
        }
        m_symbol.SetAt(i,bufsymbol); //返回修改后的字符位置信息，重新分析
    }
}
```

根据本文大量测试结果，规则中动态调整了相关字符的原始位置信息，动态阈值 Δt 取值为 ±5 可以取得较好的分析结果。返回调整后的字符的位置信息，重新进行结构分析，得到新的公式结构树，确保公式语法的正确性。修订后的正确结果如图 4-6 所示。



(a) 修订后的公式字符外接矩形



(b) 修订后的正确结果

图 4-6 加入规则后的分析结果

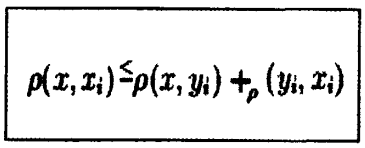
二、操作符、运算符的语法判决规则

常见的操作符与运算符分别有±、∓、×、*、·、+、−、/，≤、≥、≃、≅、≠、≜、=，由于排版问题可能导致操作符或运算符带有上/下标的错误分析。语法树检错纠错算法实现如下：

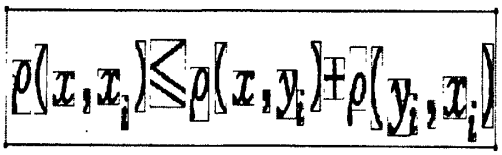
```
if ((*q).code=='+'||(*q).code=='-'||(*q).code=='*'||(*q).code=='/'||(*q).code=='≤')
{
 if((*q).superscript!=NULL))
  {//循环查找在识别结果中由于结构分析导致出错的字符
    Search((*(*q).superscript).minx==bufsymbol.leftupx);
    bufsymbol.leftupy=bufsymbol.leftupy+Δt;//降低字符的Y坐标
    bufsymbol.rightdowny=bufsymbol.rightdowny+Δt;
  }
 if ((*q).subscript!=NULL)
  {//循环查找在识别结果中由于结构分析导致出错的字符
  Search((*(*q).subscript).minx==bufsymbol.leftupx);
  bufsymbol.leftupy=bufsymbol.leftupy-Δt;//提升字符的Y坐标
  bufsymbol.rightdowny=bufsymbol.rightdowny-Δt;
  }
 m_symbol.SetAt(i,bufsymbol); //返回修改后的字符位置信息，重新分析
}
```

加入语法检错纠错规则前/后试验对比，如图 4-7 所示。

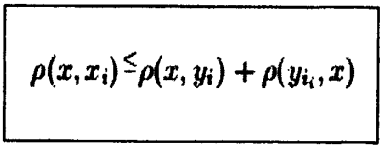
(a) 公式字符外接矩形



(b) 语法错误的结构分析结果



(c) 修订后的公式字符外接矩形



(d) 修订后的正确结果

图 4-7 修订字符信息前/后的结构分析结果对比示例

三、根号的语法判决规则

根号是数学公式中经常出现的符号，其特点是其它符号可以与之形成内部关系和左上关系，但没有左下、右下、右上关系，如图4-8所示。

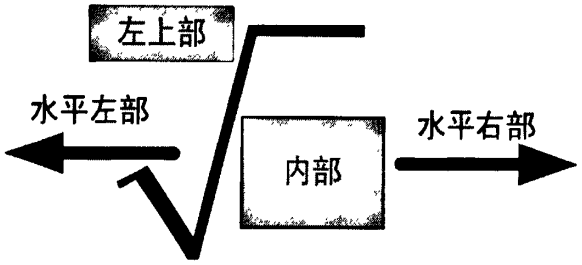


图4-8 根号区域划分

根式语法检测的难点在于内部区域和左上部区域符号的遍历。

用VC++描述根号处理过程如下：

RecursionDeep(*R*,int *flag*=0) //遍历含有符号为根号语法树

{

if (RITree=RecursionDeep(*R*,int *flag*=3))//转入遍历 R 内部子表达式的语法树；

CheckErrors(*RITree*)；//调入水平纠错函数分析根式符号内部的子表达式；

if (RLTree=RecursionDeep(*R*,int *flag*=6))//转入遍历 R 的左上部子表达式的语法树；

CheckErrors(*RLTree*)；//调入水平纠错函数分析根式符号左上的子表达式；

}

其中 R 为根号，RITree，RLTree 为根式内部与左上部的语法树链表，函数 RecursionDeep(*R*,int *flag*=3) 的作用是转入遍历根式的内部子表达式的语法树，函数 RecursionDeep(*R*,int *flag*=6) 的作用是转入遍历根式左上部子表达式的语法树，函数 CheckErrors(*RITree*)、CheckErrors(*RLTree*) 的分别是进行两个方向子表达式的检错纠错。

四、绑定字符组合关系判决规则

常见的绑定符号如 lim,$\prod$,$\iint$,$\sum$ 位置关系复杂，各种绑定符号组合后的公式空间关系更加不容易分析。如图 4-9 求和符号的组合，两个绑定符号由于位置很靠近，容易造成绑定符各自下部所属关系的混乱，分析结果如图 4-10 所示。

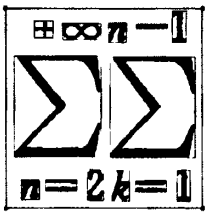


图 4-9 公式字符外接矩形

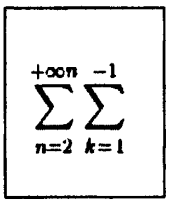
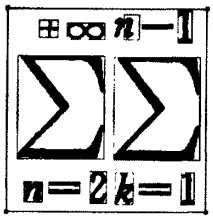


图 4-10 语法错误的结构分析结果

在遍历语法树的过程中，利用已知的符号信息加入如下语法规则，正确分析结果如图 4-11 所示。

```
if((*q).code==8734) {
    if((*q).right!=NULL))
    {//循环查找在识别结果中由于结构分析导致出错的字符
     Search(*(*q).right).minx==bufsymbol.leftupx);
     bufsymbol.leftupx=bufsymbol.leftupx+Δt;  //水平向右移动X坐标
     bufsymbol.rightdownx=bufsymbol.rightdownx+Δt;
     }
  m_symbol.SetAt(i,bufsymbol); //返回修改后的字符位置信息，重新分析
}
```



(a) 修订后公式字符的外接矩形

$$\sum_{n=2}^{+\infty}\sum_{k=1}^{n-1}$$

(b) 修订后的正确结果

图 4-11 加入规则后的分析结果

五、数字字符的语法判决规则

根据符号相对于水平线的偏移不同可把符号分为中心型、下降型、上升型三种。如图 4-12 所示。

图 4-12 不同符号的中心类型

公式中常见的下降型符号有：μ、η、ξ、ζ、γ、g、q、p、y，上升型符号有：b、d、f、h、i、k、l、t、0-9。不同类型的字符其水平域、上标域、下标域的定义有所不同，容易导致符号上\下标与水平位置关系混淆。如图 4-13 所示，字符“μ”与数字“3”是水平位置关系，但是由于“μ”是下降型符号字符中心偏低，容易分析成上升型字符“3”的下标，错误的重构结果如图 4-14 所示。

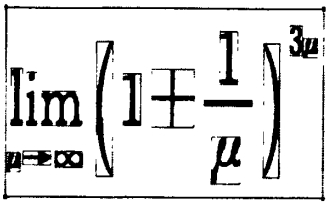


图 4-13 公式字符外接矩形

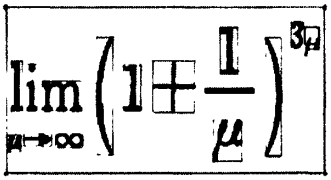
$$\lim_{\mu\to\infty}\left(1+\frac{1}{\mu}\right)^{3\mu}$$

图 4-14 结构分析错误结果

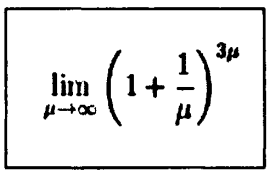
因此，我们加入如下语法规则，对不同类型的字符进行重新修订。正确分析结果

如图 4-15 所示。

```
if ((*q).code>='0'||(*q).code<='9')
{
    if ((*q).subscript!=NULL)
    {//循环查找在识别结果中由于结构分析导致出错的字符
    Search((*(*q).subscript).minx==bufsymbol.leftupx);
    bufsymbol.leftupy=bufsymbol.leftupy-Δt;//提升字符的Y坐标
    bufsymbol.rightdowny=bufsymbol.rightdowny-Δt;
    }
  m_symbol.SetAt(i,bufsymbol); //返回修改后的字符位置信息，重新分析
}
```



(a) 修订后的公式字符外接矩形



(b) 修订后的正确结果

图 4-15 加入规则后的分析结果

4.5 绝对值符号的语义分析算法

对于带绝对值符号的表达式，由于绝对值符号不同于括号，不具有方向信息，因此对含有绝对值符号的公式的语义特征提取具有一定的难度。通过采集各种绝对值表达式的样张，总结出作为绝对值起始符号“|”与作为绝对值终结符号“|”各自产生的 FIRST(“|”)与 FOLLOW(“|”)[37]，如表 4-8 所示。

表 4-8 绝对值符号产生的二元组

	FIRST()集	FOLLOW()集
作为绝对值起始符号 “|”	$0-9,a-z,A-Z,+,-,*,/,(,\le,=,\|$	$0-9,a-z,A-Z,+,-,(,\|$
作为绝对值终结符号 “|”	$0-9,a-z,A-Z,),\|$	$0-9,a-z,A-Z,+,-,*,/,(,\le,=,\|$

根据上表，可以看出起始绝对值符号与终结绝对值符号各自产生的二元组存在交

集： $W=\{(f,s)|f\in\{0-9,a-z,A-Z,+,-,(,|\},s\in\{0-9,a-z,A-Z,|\}$ 。例如：公式 $|c-d|-b*a|-5|$ 可理解为两种含义，即绝对值符号匹配为“$(|c-d|)-b*a(|-5|)$”，或者“$(|c-d(|-b*a|)-5|)$” 。所以，绝对值表达式存在二义性，无法利用 LL(1)文法进行语法检错。我们采用带回溯机制的子公式提取算法来分析带有绝对值表达式的语义。

具体步骤如下：

步骤一： RecursionDeep(StructureTreeCode* *p*,int *flag*) 递归函数遍历公式结构树，遇到一个绝对值符号时候进行存储 abs_stack.push(*node*)，找到第二个绝对值符号后，记录下两个绝对值符号之间的子串链表存入 *sub_list* 。

步骤二：调用 CheckErrors(structualTreeCode *q) 判断子串链表是否合法。若合法，则从 *sub_list* 出栈，成为匹配字符串。

步骤三：不合法，则在第二个绝对值基础上继续寻找下一个绝对值符号，把新字符连接到上一非法子串列表中 creat a new sub list ； 返回步骤二。

统计大量含有绝对值符号的公式，总结了常见含有绝对值符号的公式类型有以下四种。并且经实验证明，上述算法能够正确分析这四类含有绝对值符号的表达式的语义。

（1）|expression1+|expression2||;

（2）| expression1|+| expression2|;

（3）|| expression1 |+ expression2 |;

（4）|| expression ||;

用VC++描述子公式提取算法处理过程如下：

```
while() {
 if (node=="|")
  {
  abs_stack.push(node); //存储绝对值的"|"符号栈
  }
else if (abs_stack is not Empty)
  {
```

```
    sub_list.insert(); //存储两个绝对值符号之间子串链表的符号栈
  }
}
while{
   if (node=="|")
    {
    abs_stsck.push(node);
    creat a sub_list;
    sub_list_stack.push(sub_list);
    }
  if (sub_list_stack is not Empty&&cur_sub_list is legal)
    {
    sub_list_stack.pop ; //合法子串链表输出
    }
  else
   {
    creat a new sub list; //重新创建子串链表
    sub_list_stack.push();
   }
  else if (cur_sub_list is not Empty )
    {
     cur_sub_list.insert();
     }
}
```

第 5 章 实验结果分析

5.1 评价标准

评价标准是在测试过程中所使用的一些用来评价版面分析与语法、语义准确度的量化标准。这里引入识别率和纠错率两个指标来描述算法的性能。

(1) 识别率：用来衡量引入 LL(1)文法的语法分析方法在识别输入公式是否合乎语法方面的效果。其数学公式表示如下：

$$R_r = \frac{RN}{EN} \times 100\% \tag{5-1}$$

其中，*RN* 表示通过该文法能够正确识别的公式个数。所谓正确识别包含两种情况：第一种情况为语法正确的数学公式通过该文法检测后，能够判别为符合文法的数学公式；第二种情况为存在语法错误的数学公式通过该文法检测后，能够判别为不符合文法的数学公式。*EN* 表示被识别的数学公式总数。

(2) 纠错率：用来衡量融入检错纠错规则的结构分析方法对语法树检错纠错的效果。数学公式表示如下：

$$R_c = \frac{CN}{AN} \times 100\% \tag{5-2}$$

其中，*CN* 表示算法能够检错并纠错成功的语法结构树数目，*AN* 表示结构分析错误的公式数目。

5.2 试验结果分析

5.2.1 引入 LL(1)语法语义识别算法的结果分析

由于基于 LL(1)文法的语法识别算法在规则选取顺序上的局限性，该方法不能保证把实际上是本语法所产生语言中的合法公式都识别出来，依赖选择一组正确的步径序列，要求每一步都选择得对，任何一步发生错误，都会导致识别错误。分析结果如表 5-1 所示。

表 5-1 基于 LL(1)语法识别算法结果统计

类型	公式总数（EN）	正确识别个数（RN）	识别正确率(R_r)
I 型	500	451	90.20%
II 型	400	363	91.25%
III 型	400	359	89.75%

注：待识别公式，I 型表示含有复合字符串的公式，II 型表示含有小、中、大括号的一维公式，III 型代表连加连乘函数。

5.2.2 融入检错纠错规则的语法树遍历算法的结果分析

在 Windows 环境下使用 Visual C++ 6.0 开发平台完成了融入检错纠错规则的语法树遍历算法的程序设计，针对以下四种类型的数学公式进行了测试，试验结果如表 5-2 所示。

表 5-2 语法检错纠错结果统计

类型	分析错误的公式个数	纠错的个数	纠错率(R_c)
简单二维公式	107	95	88.78%
积分公式	61	54	88.52%
根式	64	56	87.50%
绑定符号组合公式	29	25	86.20%

从上表的统计结果可以看出，融入检错纠错规则的语法树遍历算法针对带有根式、积分、极限、求和等特殊符号的公式，语义分析效果明显，减少了结构分析阶段对版面信息的过度依赖性，提高了结构分析中的语义处理能力。

第6章 结论与展望

6.1 工作总结

本文针对印刷体数学公式语法、语义进行了研究，引入了LL(1)文法，设计了基于分层处理的结构分析方法，并且根据实验中出现的问题对算法进行了改进，增加了语法检错纠错的功能。实验表明该系统具有较高的准确率和适应性。

所完成的主要工作包括：

(1) 引入水平膨胀方法对数学公式进行预处理，设计了基于DBSCAN思想的公式符号切分算法，有效地提取了公式子表达式范围，为下一步公式结构分析打下了基础。

(2) 针对公式结构分析阶段常出现的语法错误，本文在结构分析后期设计了公式语法树检错纠错算法。结合数学公式的语法、语义规则及公式字符的识别结果，采用公式语法树深度遍历算法对公式字符间的逻辑关系进行语法检错与纠错。最后，根据修订后的字符信息重新进行结构分析，从而提高结构分析结果的正确率。

(3) 统计大量一维公式语义信息和语法规律，定义了符合LL(1)文法的表达式构成规则，有效判别表达式的语法正确性。

(4) 统计绝对值符号的位置关系信息，设计提取绝对值语义特征的算法，有效地识别含有绝对值符号的公式。

6.2 后续工作展望

由于本人研究水平和研究时间有限，本课题的研究尚存在不足之处，有待进一步提高。下一步的具体工作主要集中在以下几个方面：

(1) 对于不同的数学公式，所包含的字符有很大的差异，需要进一步找到一个合适的训练集，进一步优化水平膨胀阈值。

(2) 数学公式二维嵌套关系，符号间的位置信息容易混乱，需要建立一套完备的检错纠错规则，提高结构分析中的语义处理能力，进一步加强数学公式结构分析的人工纠正与自我学习能力，减少对版面信息的依赖性。

(3) 数学公式中包含大量具有二义性的符号和数学习惯用语，造成符号所表达含义的理解错误，文法规则总结难度大，需要进一步扩充与完善。

综上所述，印刷体数学公式识别是一项具有很大研究价值和应用前景的课题，还需要进行更深入的研发，尤其是开发与设计通用性好，鲁棒性高的数学公式识别系统，期待更多的科技开发者从事这方面的理论研究和实践工作。

参考文献

[1] K. F. Chan and D. Y. Yeung. Mathematical expression recognition: A survey. On Document Analysis and Recognition, 2000,13(1): 1-11.

[2] R. H. Anderson. Syntax directed recognition of hand-printed two-dimensional mathematics. Interactive Systems for Experimental Applied Mathematics. New York: Academic Press, 1968, 436-459.

[3] Belsid and J. P. Haton. A syntactic approach for handwritten mathematical formula recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1984, 6(1): 105-111.

[4] D. Blostein and A. Grbavec. Recognition of mathematical notation. Handbook of Character Recognition and Document Image Analysis. Singapore, 1997, 557-582.

[5] R. J. Fateman and T. Tokuyasu, B. P. Berman, et al. Optical character recognition and parsing of typeset mathematics. Journal of Visual Communication and Image Representation, 1996(7): 2-15.

[6] M. Okamoto and A. Miyazawa. An experimental implementation of document recognition system for papers containing mathematical expressions. In Structured Document Image Analysis, Berlin, Springer Verlag, 1992, 36-53.

[7] H. J. Windler, H. Fahrner and M. Lang. A soft decision approach for structural analysisi of handwritten mathematical expressions. International Conference on Acoustics, Speech and Signal Processing, Detroit, USA, May 1995, 4: 2459-2462.

[8] K. F. Chan and D. Y. Yeung. A novel application of on-line mathematical expression recognition technology. Proceedings of 6th International Conference on Document Analysis and Recognition, Seattle, Washington, USA, September 2001, 1059-1063.

[9] J. J. Pfeiffer. Parsing graphs representing two dimernsional figures. Proceeding of IEEE Workshop on Visual Language, Washinton, USA, September 1992, 200-206.

[10] B. Q. Huang and M. T. Kechadi. A structural analysis approach for online handwritten mathematical expressions. International Journal of Computer Science and Network Security, 2007, 7(7): 47-56.

[11] L. Zhang, D. Blostein and R. Zanibbi. Using fuzzy logic to analyze superscript and subscript relations in handwritten mathematical expressions. Proceedings of 8th International Conference on Document Analysis and Recognition, Seoul, South Korea, August 2005, 2: 972-976.

[12] 江红英, 勒简明, 王庆人. 基于统计的印刷体数学公式上/下标关系判别. 计算机工程与应用, 2003, 28: 75-78.

[13] 郭育生, 黄磊, 刘昌平. 基于多候选的数学公式识别系统. 计算机研究与发展, 2007,44(7): 1144-1150.

[14] 陈洪波, 王强, 许晓蓉, 等. 数学表达式的自动识别. 广西科学, 2004, 11(1): 20-26.

[15] 林桂芳. 印刷体数学公式识别中符号识别技术的研究. 硕士学位研究生学位论文, 哈尔滨工业大学, 2004, 22-30.

[16] 吴俊飞. 基于特征字符的印刷体公式识别研究. 硕士学位研究生学位论文, 哈尔滨工业大学, 2006, 36-49.

[17] 李宁. 印刷体数学表达式识别实现方法研究. 硕士学位研究生学位论文, 广西师范大学, 2005, 11-58.

[18] J. Ha, R. M. Haralick and I. T. Philips. Understanding mathematical expressions from document images. International Conference on Document Analysis and Recognition, Montreal, Canada, August 1995, 2: 956-959.

[19] M. Twaakyondo. Structure analysis and recognition of mathematical expressions. International Conference on Document Analysis and Recognition, Montreal, Canada, August 1995, 2: 430-438.

[20] M. Okamoto and B. Miao. Recognition of mathematical expressions by using the layout structures of symbol. International Conference on Document Analysis and Recognition, St. Malo, France, 1991, 1: 242-250.

[21] H. J. Lee and M. C. Lee. Understanding of mathematical expression using procedure-oriented transformation. Pattern Recognition, 1994, 27(3): 447-457.

[22] H. J. Lee and J. S. Wang. Design of a mathematical expressions recognition system. Proceedings of the Third International Conference on Document Analysis and Recognition, Montreal, Canada, August 1995, 2: 1084-1087.

[23] R. Fukuda, SIF. Tamari and X. Ming. M Suzuki. A technique of mathematical expressions for

automatic reading of mathematical documents. International Conference on Document Analysis and Recognition, Bangalore, India, September 1999, 1: 131-134.

[24] Y. A. Dimitriadis and J. L. Coronado. Towards an art based mathematical editor that uses on-line handwritten symbol recognition. Pattern Recognition, 1995, 28(6): 807-822.

[25] J. Y. Toumit, S. G. Salicetti and H. Emptoz. A hierarchical and recursive model of mathematical expressions for automatic reading of mathematical documents. International Conference on Document Analysis and Recognition, Bangalore, India, September 1999, 119-122.

[26] Grbavec and D. Blostein. Mathematics recognition using graph rewriting. International Conference on Document Analysis and Recognition, Montreal, Canada, August 1995, 2: 417-421.

[27] G. M. Pallao. Constrained attribute grammars for recognition of multidimensional objects. Pattern Recognition. 1998, 359-369.

[28] B. Chaudhuri and U. Garain. An approach for recognition and interpretation of mathematical expressions in printed document. Pattern Analysis and Applications, 2000(3): 120-131.

[29] 章霄，董艳雪，赵文娟，等. 数字图像处理技术. 北京: 冶金工业出版社, 2005 年.

[30] H.Jianwei, K Micheline. 数据挖掘概念与技术. 北京: 机械工业出版社, 2006 年.

[31] 汪敏里. 语法研究与语义. 电大教学, 1999, 6: 43-45.

[32] C.L.Kennech. Compiler construction: principles and practice. PSW Publishing Company, 1997.

[33] 吴微,侯利昌. 基于 LL(1)文法的印刷体数学公式结构分析方法. 大连理工大学学报, 2006, 46(3): 454-459.

[34] 钱锋. 计算语言学引论. 学林出版社, 1990.

[35] 马庆株. 结合语义表达的语法研究.汉语学习, 2000, 2:1-6.

[36] 李明琴，李涓子，王作英. 语义分析和结构化语言模型. 软件学报, 2005, 16(9): 1523-1532.

[37] 宋昭，李芬. 一种基于数学表达式的二义文法分析算法. 计算机工程与应用, 2004, 34:88-90.

攻读硕士学位期间发表论文情况

[1] X. D. Tian and S.Wang. Structural analysis of printed mathematical expression with heuristic rules. International Conference on Computational Intelligence Security, Harbin, China, December 2007, 1030-1034. (EI 检索)

致 谢

值此论文完成之际，向我的导师田学东教授表示深深的谢意。本文从选题、撰写到完成都凝聚了导师大量的心血和汗水。在此衷心感谢田老师的悉心指导和教诲！没有导师的帮助，就没有我今天的成果，他严谨治学的作风和平易近人的态度将是我终生学习的榜样。

感谢我的同窗：王菲、左丽娜、郝楠、张艳，她们在我的论文写作方面提出了宝贵的建议和热心的帮助，在此向他们表示衷心的感谢。实验室良好、和谐、融洽、积极向上的学习氛围为我论文的顺利完成提供了保证。另外我还要感谢智能图文信息处理实验室的所有老师！感谢您们的帮助！

同时还要感谢我的父母，有了他们的支持我才完成了今天的学业。

最后，对所有在攻读硕士学位期间给予我帮助的每个人表示感谢。

感谢对论文进行评审并提出宝贵意见的各位老师和专家！