

**课程设计报告**

**题目： 基于SAT的数独游戏求解程序**

**课程名称： 程序设计综合课程设计**

**专业班级： CS1704**

**学 号： U201714674**

**姓 名： 黄明锋**

**指导教师： 袁凌**

**报告日期：**

**计算机科学与技术学院**

**任□务□书**

* **设计内容**

SAT问题即命题逻辑公式的可满足性问题（satisfiability problem），是计算机科学与人工智能基本问题，是一个典型的NP完全问题，可广泛应用于许多实际问题如硬件设计、安全协议验证等，具有重要理论意义与应用价值。本设计要求基于DPLL算法实现一个完备SAT求解器，对输入的CNF范式算例文件，解析并建立其内部表示；精心设计问题中变元、文字、子句、公式等有效的物理存储结构以及一定的分支变元处理策略，使求解器具有优化的执行性能；对一定规模的算例能有效求解，输出与文件保存求解结果，统计求解时间。

* **设计要求**

要求具有如下功能：

1. **输入输出功能：**包括程序执行参数的输入，SAT算例cnf文件的读取，执行结果的输出与文件保存等。(15%)
2. **公式解析与验证：**读取cnf算例文件，解析文件，基于一定的物理结构，建立公式的内部表示；并实现对解析正确性的验证功能，即遍历内部结构逐行输出与显示每个子句，与输入算例对比可人工判断解析功能的正确性。数据结构的设计可参考文献[1-3]。(15%)
3. **DPLL过程：**基于DPLL算法框架，实现SAT算例的求解。(35%)
4. **时间性能的测量：**基于相应的时间处理函数（参考time.h），记录DPLL过程执行时间（以毫秒为单位），并作为输出信息的一部分。(5%)
5. **程序优化：**对基本DPLL的实现进行存储结构、分支变元选取策略[1-3]等某一方面进行优化设计与实现，提供较明确的性能优化率结果。优化率的计算公式为：[(t-to)/t]\*100%,其中t 为未对DPLL优化时求解基准算例的执行时间，to则为优化DPLL实现时求解同一算例的执行时间。(15%)
6. **SAT应用：**将数独游戏[5]问题转化为SAT问题[6-8]，并集成到上面的求解器进行问题求解，游戏可玩，具有一定的/简单的交互性。应用问题归约为SAT问题的具体方法可参考文献[3]与[6-8]。(15%)

* **参考文献**

[1] 张健著. 逻辑公式的可满足性判定—方法、工具及应用. 科学出版社，2000

[2]Tanbir Ahmed. An Implementation of the DPLL Algorithm. Master thesis, Concordia University,Canada,2009

[3] 陈稳. 基于DPLL的SAT算法的研究与应用.硕士学位论文，电子科技大学，2011

[4]Carsten Sinz.Visualizing SAT Instances and Runs of the DPLL Algorithm.J Autom Reasoning (2007) 39:219–243

[5] 360百科：数独游戏<https://baike.so.com/doc/3390505-3569059.html>

[6] Tjark Weber. A sat-based sudoku solver. In 12th International Conference on Logic forProgramming, Artificial Intelligence and Reasoning, LPAR 2005, pages 11–15, 2005.

[7]Ins Lynce and Jol Ouaknine. Sudoku as a sat problem.In Proceedings of the 9th InternationalSymposium on Artificial Intelligence and Mathematics, AIMATH 2006, Fort Lauderdale.Springer,2006.

[8] Uwe Pfeiffer, Tomas Karnagel and Guido Scheffler. A Sudoku-Solver for Large Puzzles using SAT. LPAR-17-short (EPiC Series, vol. 13), 52–57

[9] Sudoku Puzzles Generating: from Easy to Evil.

http://zhangroup.aporc.org/images/files/Paper\_3485.pdf

[10] Robert Ganian and Stefan Szeider. Community Structure Inspired Algorithms for SAT and #SAT. International Conference on Theory and Applications of Satisfiability Testing(SAT 2015),223-237360

**目□□录**

**任务书** I

**1□□□引言** 1

1.1□□×××××× 1

1.2□□×××××× 3

1.2.1□□×××××× 7

1.3□□×××××× 10

**……**

**……**

**……**

3**□□□××××××** 20

3.1□□×××××× 20

3.2□□×××××× 23

3.2.1□□×××××× 25

3.3□□×××××× 30

**……**

**……**

**4□□□** 40

**参考文献** 44

**附录□××××××** 45

（章为宋体小4号加粗，其余宋体小4号，字母、阿拉伯数字为Time New Roman小4号）

**1□引言**

**1.1□课题背景与意义**

从1960年至今，SAT问题一直备受人们的关注，世界各国的研究人员在这方面都做了大量的工作，提出了许多的求解算法，取得了一批相当重要的理论和实践成果。尽管命题逻辑的可满足性问题理论研究已趋于成熟，但在SAT求解器被越来越多地应用到各种实际问题领域的今天，探寻解决SAT问题的高效算法仍然是一个吸引人并且极具挑战性的研究方向。

**1.1.1□××××××**（黑体小4号加粗,字母、阿拉伯数字为Time New Roman小4号加粗）

××××××××××××××××××××××××

**1.1.2□××××××**

××××××××××××××××××××××××

**1.2□国内外研究现状**

近十多年来，可满足性问题研究逐渐升温，已成为了国际国内的研究热点。每年可满足性理论和应用方面的国际会议都会组织一次SAT竞赛以求找到一组最快的SAT求解器，而且会详细展示一系列的高效求解器的性能。2003年的SAT竞赛中，就有30多种解决方案针对从成千上万的基准问题中挑选出的一些SAT问题实例同台竞争。国内也经常会组织一些SAT竞赛及研讨会，这些都促进了SAT算法的飞速发展。当前的SAT问题的算法主要包含两类：局部搜索算法和回溯搜索算法。其中，回溯搜索算法大都是基于回溯搜索的，局部搜索算法是基于局部随机搜索的。回溯搜索算法基于穷举法思想，它的优点是能保证找到对应 SAT问题的解或证明公式不可满足，但是效率极低，它的平均时间复杂度虽是多项式级的，但是最坏情况下的时间复杂度却是指数级的。一些回溯搜索算法采用了精巧的技术来减小搜索空间和问题规模，提高了算法的时间效率。局部搜索算法相对于回溯搜索算法而言，由于采用了启发式策略来指导搜索，使得求解速度相对较快，但是在某些实例上可能得不到解，它不保证一定能够找到对应 SAT问题的解，即它不能证明SAT问题的不可满足性。近十多年来，国际上已提出了各种不同的局部搜索算法和回溯搜索算法，使得SAT求解器解决不同领域中的SAT问题的能力不断增强，能解决的问题的规模不断增大。当前已提出了一大批采用回溯搜索算法的高效的SAT问题求解器，其中绝大多数提出来的回溯搜索算法是对原始的DPLL回溯搜索算法的改进。这些改进措施包括：新的变量决策策略，新的搜索空间剪除技术，新的推理和回溯技术以及新的更快的算法思想方案和数据结构等。当前水平的SAT问题求解器已经取得了举足轻重的改进，但是仍有一些问题没有得到高校的解决，已经解决的问题可能还存在更好的求解算法，因此研究并实现高效率的求解算法仍是当前要解决的中心问题之一。

**1.3□课程设计的主要研究工作**

××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××

--------章与章之间插入分页符----------

**2□系统需求分析与总体设计**

**2.1□系统需求分析**

设计一个能求解SAT问题和生成数独并求解的演示系统，在求解SAT问题的时，应当设计能够实现读取cnf算例文件，解析文件，根据cnf算例文件设计一定的物理结构，建立公式的内部表示，并实现对SAT问题的求解，打印和保存求解的结果等功能；在生成数独时，应当设计能够实现生成数独棋盘，并且将其转化为cnf范例，然后对其求解，打印和保存求解结果等功能。

**2.2□系统总体设计**

这部分可根据用户需求，设计和规划一个系统，说明清楚系统应该有哪些功能模块，每个模块做什么。最后给出完整的系统模块结构图。

系统应当具有display、cnfparser 、solver、sudoku四个模块。

Display模块：此即主控(main)、交互与显示模块，在程序启动时用来选择执行的功能：SAT或Sudoku，针对不同的功能给出不同的输出提示。在执行SAT功能时，输出要用户输入读取的cnf算例文件路径，并且输出针对读取的算例文件进行的各项操作（如是否解析公式，是否打印cnf范例，求解结束后可满足则输出是否验证结果，是否打印变元值，最后输出是否保存执行结果），用户则根据输出的各项信息，输入对应的值，进行对应的操作。

Cnfparser模块：此即cnf算例解析模块。用来实现对读取成功的cnf范例进行解析，建立公式的内部结构，并且在此基础上，增加对建立好的cnf结构进行的各项操作（包括增加变元、增加子句、复制子句、公式解析、打印cnf范例、保存运行结果等）。

Solver模块：此即执行DPLL算法模块。用来实现对读取成功的cnf算例进行求解。

Sudoku模块：此即数独模块。用来随机初始化生成数独，并且转化为cnf范例，并且调用Solver模块中的DPLL函数对转化好的cnf范例求解，并将结果转化为数独并且输出。

程序流程图如图3-1所示。

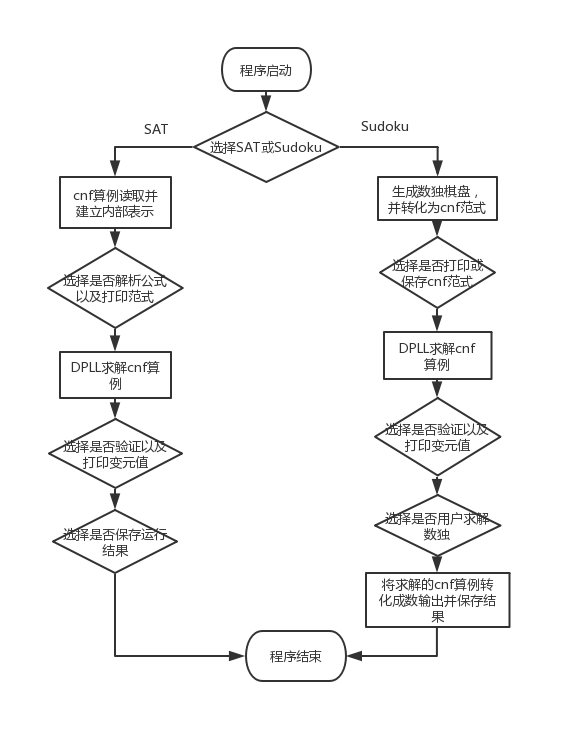


图3-1□程序流程图

--------章与章之间插入分页符----------

**3□系统详细设计**

**3.1□有关数据结构的定义**

这部分要写的：（1）首先描述系统中要处理那些数据，每种类型的数据包括哪些数据项，每个数据项的数据类型，最后可用一个表格表示出来；（2) 描述这多种数据在系统中如何关联，可通过图直观的说明这多种数据间的关联。

系统中主要处理

表3-1□系统处理数据表

|  |  |  |
| --- | --- | --- |
| 结构名(用typedef定义) | 数据项与数据类型 | 数据项与数据类型 |
| Line（存储当前行子句信息） | Clause \* firstvariable(行子句的第一个变元) | Line \* nextline(指向下一行子句) |
| Clause（存储变元值） | int name(记录变元值，有正负之分) | Clause \*next(指向一个变元) |
| Sudoku(数独结构) | int s[9][9](存储数独棋盘各个位置的值) | Sudoku \*next(指向下一个数独) |

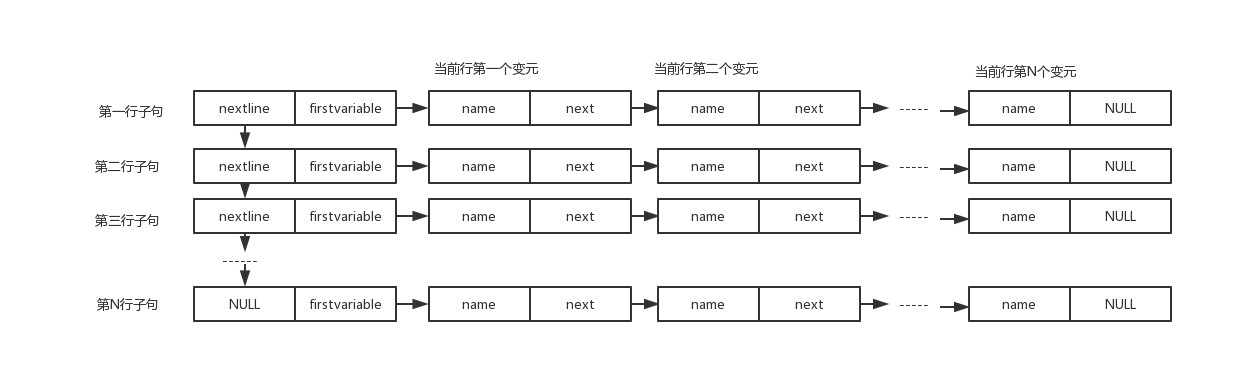


图3-2□邻接表结构图

**3.2□主要算法设计**

这部分主要描述系统中的模块实现的流程，可采用文字配合流程图的方式表示各模块的算法思想及流程。

cnfparser模块(即cnf解压模块)：

Line \* creatClause(char \* cnfpath)：此函数用来根据字符指针cnfpath读取对应路径的cnf范例文件并创建对应的数据结构。首先声明一个文件指针用来打开指定文件，如果打开文件失败则返回一个空指针并提示读取文件错误请重新输出。打开文件成功后，用malloc声明Line \*类型的cnf指针，用来作为cnf范例的头节点，指向cnf范例的第一行，并将其内容初始化为空。接着用fetc读取文本文件中的内容，用while循环跳过开头为c的注释部分。到达标明变元数目，子句数目一行的时候，先用for循环读取当前变元数目子句数目前的字符，之后用fscanf读取接下来的2个数字，并存入全局变量variablenumbers 和Clausenumbers中。接下来为依次用calloc声明int \*的空间，并赋给value\_(用来存储变元值)，occurtimes(记录变元出现次数)，positivetimes(记录变元出现正状态的次数)，negativetimes(记录变元出现负状态的次数)，backtrack(用来追踪变元是否被引用)，multiplytimes(记录变元正状态\*负状态的值),maxtimes(用来对multiplttimes排序，其值为变元正负状态相乘的值按降序排列)。声明完空间后先对value\_数组进行初始化，将所有值赋为-1，意味着当前变元没有赋值，1表示对当前变元赋真值，0表示对当前变元赋假值。前期准备工作做完之后就可以开始读取cnf范例中的内容并创建相应的数据结构了。用2层for循环来读取变元，在最内层循环中，用malloc声明Clause \*型的指针并赋给p，用fscanf读取文件指针指向的一个整形数组，倘若为0意味着当前行子句已经读取完毕，可以跳出内层循环；若不为0，则将p->name设置为所读取的内容，p->next设置为空。同时根据读取的数据正负，对应的positivetimes数组或negativetimes数组对应的值加一。判断当前行是否是第一行，所读取的数据是否当前行的第一个数据，并依据此将读取的数据链接到正确的位置。读取完当前行内容后，用malloc声明Line \*型指针并赋给指针q，对指针q的内容赋空，并通过循环将创建好的新一行子句q链接到cnf范式结构的最后一行，但是若当前已读取完所有行的内容，那么这一行空子句就是多余的，需要释放掉。执行完以上流程之后，就是按照正负变元次数对multiplytimes数组赋值，并且调用sorttimes函数对multiplytimes数组进行从大到小的排序。最后释放掉无用数组(occurtimes,multiplytimes,positivetimes,negativetimes)占用的内存,关闭文件指针，返回cnf指针就完成了对cnf范例结构的创建。

int addLiteral(Line \*L,int literal):此函数用来在传入的行子句指针增加一个变元，变元值为传入的整形数。首先用malloc声明Clause \*指针并赋给p，若p为空，返回-2，p->name设置为literal，p->next设置为空。倘若当前行子句L没有变元，则L->firstvariable设置为p，否则将p插在当前行子句的最前面，即将p的下一个变元设置为L的第一个变元，再将L的第一个变元设置为p。此即完成了添加变元的操作。

Line \* addClause(Line \*L,int n,int \*clause):此函数功能用来对当前创建好的cnf邻接表结构增加一行行子句，增加的行子句变元数目为传入值n，变元的值通过传入的整形数组clause确定。

**4□系统实现与测试**

**4.1□系统实现**（黑体4号加粗,字母、阿拉伯数字为Time New Roman4号加粗）

这部分可首先叙述一下你的系统实现的软硬件环境；

根据3.1的设计，用C语言定义各种数据类型；

程序代码部分在这里不需要给出来，只需要叙述清楚在系统中包括哪些函数，各函数的说明，如何利用这些函数实现系统各模块的功能，以及函数间的调用关系（可用图表示出来）。程序详见附录。

**4.2□系统测试**（黑体4号加粗,字母、阿拉伯数字为Time New Roman4号加粗）

首先叙述一下常用的软件测试方法，在选择几个主要的功能模块（自行掌握数量，关键要体现你的水平的一些模块）描述测试过程，（1）先明确模块的功能、设计目标等。（2）分析、叙述如何选取测试数据，要求有完整的测试大纲。（3）运行结果（这时可用截图）。（4）分析运行结果、确认程序满足该模块的设计目标。

**5□总结与展望**

（黑体小2加粗居中,字母、阿拉伯数字为Time New Roman小2号加粗）

**5.1□全文总结**（黑体4号加粗,字母、阿拉伯数字为Time New Roman4号加粗）

对自己的工作做个总结，主要工作如下：

（1）对。

（2）。

（3）

。

**5.1□工作展望**（黑体4号加粗,字母、阿拉伯数字为Time New Roman4号加粗）

在今后的研究中，围绕着如下几个方面开展工作。。。。。。。。

（1）。

（2）

# 6□体会(黑体小2号加粗居中)

这部分就自由发挥了。

**参考文献**(黑体小2号加粗居中)

[1]□王静康,张凤宝,夏淑倩等.论化工本科专业国际认证与国内认证的“实质性”.高等工程教育研究,2014,5:1-4

[2]□Stone J A, Howard L P. A simple technique for observing periodic nonlinearities in Michelson interferometers. Precision Engineering,1998,22(4):220-232

[3]□朱印红,袁衍明.Dreamweaver完美网页设计——技术入门篇.(第一版).北京:中国电力出版社,2006:19～20

[4]□Lewis S L. Physics and chemistry of the solar system.北京:北京大学出版社,2014.1～2

[5]□陈剑.上博简《民之父母》“而得既塞於四海矣”句解释[EB/OL］.简帛研究网站，http://www.bamboosilk.org/Wssf/2003/chenjian03.htm．2003-01-18

( 宋体小4号)

……

……

……

**附录** (黑体小2号加粗居中)

×××××××××××××××××××××××××××

( 宋体小4号)

……

……

……