

**课程设计报告**

**题目： 基于SAT的数独游戏求解程序**

**课程名称： 综合程序设计课程设计**

**专业班级： CS1703**

**学 号： U201714607**

**姓 名： 钟子琛**

**指导教师： 袁凌**

**报告日期： 2019.3.27**

**计算机科学与技术学**

**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 **课题背景与意义** 1

1.2 **国内外研究现状** 1

1.3 **课程设计的主要研究工作** 1

**2 系统需求分析与总体设计** 2

2.1 **系统需求分析** 2

2.2 **系统总体设计** 2

**3 系统详细设计** 3

3.1 **有关数据结构的定义** 3

3.2 **主要算法设计** 5

**4 系统实现与测试** 7

4.1 **系统实现** 7

4.2 **系统测试** 3

**5 总结与展望** 1

5.1 **全文总结** 1

5.2 **工作展望** 3

**6 体会** 1

**参考文献** 44

**附录**  45

**1 引言**

**1.1 课题背景与意义**

可满足性问题（Satisfiability Problem）即 SAT 问题，是对一个以合取范式 （Conjunctive Normal Form，常简称 CNF）的形式给出的命题逻辑公式进行判断， 以找出是否存在一个真值指派，使得该命题逻辑公式的值为真。SAT 问题看似简 单，但它却是计算机领域和人工智能领域所要研究的中心问题，被称为理论计算 机科学和数理逻辑中的第一问题，在硬件验证、人工智能、电子设计自动化、自 动化推理、组合等式检测等领域具有非常重要的理论和实践意义。

**1.2 国内外研究现状**

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，提出了许多求解算法。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。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。

**1.3 课程设计的主要研究工作**

(1) 对 SAT 问题的研究背景、意义及现状进行了简要总结，学习了命题逻辑可 满足性问题的基本理论知识。

(2) 基于DPLL算法，研究了关于SAT问题的求解相关的知识，并自编写了一个SAT问题的求解器。

(3) 将SAT问题应用到实际问题中，本文中选择的是数独问题，并编写了运用SAT求解器来解决数独问题的程序。

**2 系统需求分析与总体设计**

**2.1 系统需求分析**

能够读取已有的SAT算例cnf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。

在此基础上，还可以使用该系统随机生成数独问题，并将数独问题转换为SAT问题，再由SAT求解器解出最后的结果。

**2.2 系统总体设计**

该系统主要功能是，读取cnf文件，输出cnf文件的解，能够求解数独，因此包含了一下几个模块。

1. 允许用户使用系统读取cnf文件。
2. 运用基于DPLL算法的SAT求解器，得到该SAT问题的解。
3. 以创建res文件的方式，输出该cnf文件的解。
4. 能够随机生成一个有解的数独。首先创建一个完整的数独，然后在该数独的 基础上挖空，从而形成一个新的数度问题。
5. 将该数独问题转化为SAT问题。
6. 运用板块1中的SAT求解器，输出创建的SAT问题的解，再将解转化为数独的解。

完整系统模块结构图如图2-1所示。

图2-1 系统模块结构图

**3 系统详细设计**

**3.1 有关数据结构的定义**

每种数据结构的具体定义如下：

Formula：

sta：标识公式是够可满足，0为不满足，1为满足，-1为待定

num\_st：表示子句数

num\_v：变元数

Statement\* root：指向第一个子句

Statement：

num\_lit：文字数

elem：指向子句中的第一个文字

next：指向下一个子句

Literal：

pos：表示文字状态 1代表正文字 0代表负文字

num：表示文字是第几号变元

next：指向下一个文字

Stack\_F：

V：储存公式中确定的变元

F：栈中储存的公式

next：下一个节点

hole：

x：行

y：列

dig\_num：可能的数字数

elem：指向第一个可能的数字

digtal：

num：数字

v\_num：数字所对应的变元编号

next：下一个数字

总结如表3-1所示：

|  |  |
| --- | --- |
| 数据结构 | 说明 |
| Formula | 公式，表示CNF文件中保存的CNF公式 |
| Statement | 子句，表示公式中的子句 |
| Literal | 表示每一个子句中的变元文字 |
| Stack\_F | 用来储存公式的栈 |
| hole | 数独中挖出来的洞 |
| digtal | 数独中每一个格子内存放的数字 |

表3-1

各个数据结构之间的关系：

Formula保存了所有的Statement，每一个Statement由若干Literal组成。Stack\_F是在关于Formula的函数中起到储存Formula的作用。hole是数独的基本单位，每一个hole中存在若干个可能填在该空格内的数字。

数据结构之间的关系如图3-1所示。



图3-1

**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。

程序主要分为两大模块：

1. SAT

该模块能够读取cnf文件生成cnf范式并求解cnf范式。

在求解cnf范式的过程中主要运用的是DPLL算法：

DPLL算法是一种基于树的回溯算法，主要使用两种基本处理策略：

a.单子句规则。如果子句集*S*中有一个单子句*L*,那么*L*一定取真值，于是可以从*S*中删除所有包含*L*的子句（包括单子句本身），得到子句集*S*1，如果它是空集，则*S*可满足。否则对*S*1中的每个子句，如果它包含文字*¬L*,则从该子句中去掉这个文字，这样可得到子句集合*S*2。*S*可满足当且仅当*S*2可满足。单子句传播策略就是反复利用单子句规则化简*S*的过程。

b.分裂策略。按某种策略选取一个文字*L*.如果*L*取真值，则根据单子句传播策略，可将*S*化成*S*2；若*L*取假值（即*¬L*成立）时，*S*可化成*S*1.

根据上述规则可不断对公式化简，并最终达到终止状态，其执行过程可表示为一棵二叉搜索树,如下图3-2所示。



图3-2

考虑到执行效率的问题，使用循环的方法而非递归。

运用这个算法最终将cnf范式解出来。

1. Sudoku

该模块能够创建一个数独问题。

创建数独的算法：首先使用一个二维数组来储存这个数组，然后使用random函数来生成一个随机的已解数独，再通过随机挖洞的方式来创建这个数独游戏。

该模块还能够将这个数独问题转化为cnf范式，转化为cnf文件，这样我们就把数独问题转化成了SAT问题，由此我们能够使用模块一通过DPLL算法得到该SAT问题的解，最终得到这个数独的解。

大致流程图如图3-3所示：



图3-3

**4 系统实现与测试**

**4.1 系统实现**

1、main()

首先创建变量time\_start和time\_end，用来记录程序运行的时间。然后让用户选择是进行cnf求解还是生成数独并求解。

若选择cnf求解，则调用SAT函数。若选择生成数独并求解，则调用Sudoku函数。

2、SAT()

该函数能够读取cnf文件生成cnf范式并求解cnf范式。

1. 让用户输入文件路径，按照路径打开对应的cnf文件，打开文件成功后调用CNF\_Reader函数，读取对应的cnf文件并由此生成cnf范式。
2. 读取文件结束后，首先对应文件中的每个变元创建一个数组Varies[i]，用来储存它们的解。之后开始计时并调用SAT\_Solver中的Solver函数，该方法能够通过传递的cnf范式将对应的解解出来并储存到Varies[i]数组中。
3. 最后，经过Check函数的检测，依据Varies[i]数组中储存的内容，生成.res文件。

3、void CNF\_Reader(FILE\* fp, Formula\* formula)

通过该函数能够读取cnf文件来生成cnf范式。

1. 若每一行开头为c，则判断该行为注释，忽略该行。
2. 若某一行开头为cnf，则说明之后为cnf公式，将之后的变量数和子句数保存到formula->num\_v以及formula->num\_st中。
3. 创建储存子句的首节点，赋值到传入参数的formula的root中，之后每读取完一个字

这部分应该写的是用户需求，明确你做的系统要实现的目标，能处理一些什么样的事务、事务处理流程等。

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**

[1]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

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

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

**附录**

×××××××××××××××××××××××××××

( 宋体小4号)