華中科技大學

程序设计

综合课程设计

(2018 级计算机科学与技术和物联网专业)

计算机科学与技术学院

程序设计综合课程设计课程组

2019 年 12 月

目录

1 程序设计综合课程设计课程概述

1.1 课程背景

对于计算机科学与技术、信息安全与物联网专业大二学生，在前三个学期已经学习了 C 语言程序设计、数据结构两门面向编程知识与技术的基础理论课，以及 C 语言程序设计实验、数据结构实验两门编程实践课程，不仅具有较为系统性的 C 语言、常用数据结构基本知识，而且具有初步的程序设计、数据抽象与建模、问题求解与算法设计的能力，奠定了进行复杂程序设计的知识基础。但两门实验课仍属于对基本编程模型与技术的验证性训练，而“程序设计”综合课程设计正是使大家从简单验证到综合应用，甚至在编程中实现智慧与风格升华的重要实践环节，为后续学习与进行计算机系统编程打下坚实的基础，让综合编程技能成为大家的固有能力与通向未来专业之门的钥匙。

1.2 课程目标

基于“程序设计”综合课程设计实践课程规划原则及其在计算机相关专业人才培养中的地位，其应该体现与达到如下目标：

⑴综合性训练目标：在该课程中涉及 C 语言的主要编程要素，如典型的数据类型与控制结构；覆盖多种典型的数据结构如线性结构、二叉树与树结构、图结构及查找表结构等。从先前实验课的单要素或单一结构训练向多要素，多结构综合应用训练转变。

⑵培养应用问题的求解能力：程序设计是为问题求解服务的，提高对应用问题进行分析，数据抽象与建模，及问题定义与功能划分等综合分析与表示能力。

⑶程序编写向程序设计转化：在实验课程中，老师基本描述了相关数据结构，程序框架及主要算法，基于此进行程序编写训练，其属于验证与复现性编程实践。综合程序设计要求同学们基于对应用问题的分析，建立求解模型，设计数据结构与主要算法，从而进行程序设计，更多地体现“设计”的内涵与份量。

⑷进一步培养编程规范性与工程化素养：通过“程序设计”综合课程设计实践进一步培养良好的规范性编程习惯，以及一定的程序设计与软件开发的工程化

素养，按照问题定义、必要的需求分析、系统设计、编程实现、程序测试分析及编制程序设计综合课程设计报告的流程组织本实践课程的开展与进行，形成初步的工程化程序设计素养。

1.3 课程任务

在选择与确定了“程序设计”综合课程设计问题之后，按工程化的基本流程分别完成如下任务：

⑴阅读“程序设计”综合课程设计任务书，熟悉问题，查阅文献，了解问题背景及相关知识。

⑵对设计问题进行需求分析，分析问题中所涉及的数据对象，划分功能，人机交互需求与数据文件读写等，并对问题进行形式化表示。

⑶基于上述需求分析，进行系统设计，明确程序的模块结构；设计数据结构（逻辑结构及其物理结构），参考并设计主要子问题的求解算法。

⑷程序实现，基于系统设计，制定相应的实现方案，编写各程序模块，完成程序编写与调试任务。

⑸程序测试，设计测试用例对程序进行功能测试，性能测量及理论分析。

⑹程序优化，对设计方案中的结构，算法进行一定优化，测试与分析性能改善结果。在设计报告中明确说明你的优化策略与方案。

⑺设计总结，按规范化要求撰写“程序设计”综合课程设计报告。

⑻成果提交：将程序源代码/工程文件、可独立运行的可执行程序、简要操作手册及“程序设计”综合课程设计报告电子版打包，文件夹名称格式为“专业班级-学号姓名”,如：CS1802-U201714999 李某某。并将设计报告打印为纸质版（A4 双面打印），然后以班为单位在指定时间（一般在设计课结束后两周内）集体提交到指导老师。

后面将对一个或多个候选设计问题进行问题与要求描述及设计指导，每个同学选择其中一题作为自己的程序综合设计课程设计问题。

2 设计问题一：基于 SAT 的二进制数独游戏求解程序

2.1 问题概述

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。SAT问题也是程序设计与竞赛的经典问题。

对于任一布尔变元x，x与其非“$\neg x$”称为文字(literal)。对于多个布尔变元，若干个文字的或运算$l_1 \vee l_2 \vee \ldots \vee l_k$称为子句(clause)。只含一个文字的子句称为单子句。不含任何文字的子句称为空子句，常用符号□表示。子句所含文字越多，越易满足，空子句不可满足。

SAT问题一般可描述为：给定布尔变元集合$\{x_1, x_2, ..., x_n\}$以及相应的子句集合$\{c_1, c_2, ..., c_m\}$，对于合取范式（CNF范式）：$F = c_1 \wedge c_2 \wedge \ldots \wedge c_m$，判定是否存在对每个布尔变元的一组真值赋值使$F$为真，当为真时（问题是可满足的，SAT），输出对应的变元赋值（一组解）结果。

一个CNF公式也可以表示成子句集合的形式：$S=\{c_1, c_2, \ldots, c_m\}$.

例如，由三个布尔变元a,b,c所形成的一个CNF公式（¬a∨b）∧（¬b∨c），可用集合表示为{¬a∨b,¬b∨c}，该公式是满足的，a=0, b=0,c=1是其一组解。

一个CNF SAT公式或算例的具体信息通常存储在一个.cnf文件中，下图2.1是算例problem1.cnf文件前若干行的截图。

```
c
c SOURCE: Kazuo Iwama, Eiji Miyano,and Yuichi Asahiro
c
c DESCRIPTION: Artifical instances from generator by source.
c
c
p cnf 200 320
46 72 115 0
-46 72 130 0
-46 72 -130 0
50 -72 115 0
-50 -72 115 0
92 -95 -115 0
```

图2.1 cnf文件格式

在每个cnf文件的开始，由‘c’开头的是若干注释说明行；‘p’开头的行说明公式的总体信息，包括：范式为CNF；公式有200个布尔变元，由1到200的整数表示；320个子句。之后每行对应一个子句，0为结束标记。46表示第46号变元，且为正文字；-46则是对应的负文字，文字之间以空格分隔。

DPLL算法是经典的SAT完备型求解算法，对给定的一个SAT问题实例，理论上可判定其是否满足，满足时可给出对应的一组解。**本设计要求实现基于DPLL的算法与程序框架，包括程序的改进也必须在此算法的基础上进行**。

2.2 DPLL 算法思想

DPLL 算法是基于树/二叉树的回溯搜索算法，主要使用两种基本处理策略：

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。

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.

交错使用上述两种策略可不断地对公式化简，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 2.2 所示。

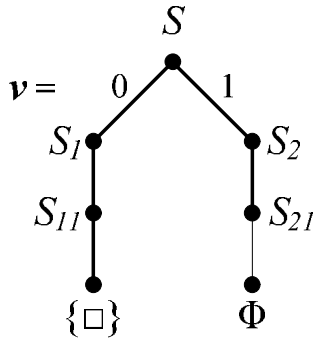

图 2.2 DPLL 算法搜索树

基于单子句传播与分裂策略的 DPLL 算法可以描述为一个如后所示的递归过程 DPLL(S),为了优化执行效率，可用非递归实现。

```
DPLL( S) :
/* S 为公式对应的子句集。若其满足，返回 TURE；否则返回 FALSE. */
{
  while(S 中存在单子句) {//单子句传播
     在 S 中选一个单子句 L；
     依据单子句规则，利用 L 化简 S；
     if S = Φ return(TRUE);
     else if (S 中有空子句 ) return（FALSE)；
  }//while
  基于某种策略选取变元 v；                    //策略对 DPLL 性能影响很大
  if DPLL（S ∪v ）return(TURE);          //在第一分支中搜索
  return DPLL(S ∪¬v); //回溯到对 v 执行分支策略的初态进入另一分支
}
```

对于公式{¬1∨2, ¬2，¬3∨4, 3∨¬5, 3∨4, 3∨5，¬2∨¬5∨6} ,大家可以利用 DPLL 算法进行手动推理其搜索处理及回溯过程，获得求解结果。

2.3 功能要求

本设计要求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，基于 DPLL 过程实现一个高效 SAT 求解器，对于给定的中小规模算例进行求解，输出求解结果，统计求解时间。要求具有如下功能：

⑴ **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%)

⑵ **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%)

⑶ **DPLL 过程：**基于 DPLL 算法框架，实现 SAT 算例的求解。(35%)

⑷ **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%)

⑸ **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，提供明确的性能优化率结果。优化率的计算公式为：[(t-t_o)/t]*100%,其中 t 为未对 DPLL 优化时求解**基准算例**的执行时间，t_o则为优化 DPLL 实现时求解同一算例的执行时间。(15%)

功能（1）至（5）为基础功能，占功能分值的 85%。

⑹ SAT 应用：将二进制数独游戏[5,6]问题转化为 SAT 问题[6]，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[6-9]。（15%）

2.4 实现与测试说明

二进制数独游戏归约为 SAT

与普通数独游戏（Sudoku）相似，n 阶（偶数 n=2m≧4）二进制数独游戏（Binary Puzzle）要求在 n×n 的网格中每个单元（cell）填入一个数字 1 或 0，必须满足约束：（1）在每一行、每一列中不允许有连续的 3 个 1 或 3 个 0 出现；（2）在每一行、每一列中 1 与 0 的个数相同；（3）不存在重复的行与重复的列。

一个二进制数独游戏初始时已经提供了一些提示数，如图 2.3 左图，要求在剩下的空格中填满数字，图 2.3 右图给出了其解，即完整填充。

	0						
			1		1		0
		0					
	1						
				1			
	0				1		
	0			0			
				0		0	

1	0	1	0	1	0	1	0
0	1	0	1	0	1	1	0
1	0	0	1	1	0	0	1
0	1	1	0	0	1	1	0
0	1	0	1	1	0	0	1
1	0	1	0	1	1	0	0
1	0	0	1	0	0	1	1
0	1	1	0	0	1	0	1

图 2.3 一个 Binary Puzzle 格局与其解

下面以 8 阶二进制数独游戏为例简要说明将其转化为 SAT 问题的方法。该游戏共有 64 个单元，每个单元对应一个布尔变元，并用该单元的行号与列号两位数进行表示。如 24 表示第 2 行第 4 列单元对应的布尔变元，取真值时表示该单元填数字 1；取假值时表示该单元填数字 0。图 2.3 左图中该单元已预填 0，转换为 SAT 公式时产生单子句 24。同理，第 1 行 2 列单元预填 0，则产生单子句¬12。

因此，图 2.3 左图中的游戏可生成如下单子句集：

¬12 24 26 ¬28 ¬33 42 55 ¬62 66 ¬72 ¬75 ¬85 ¬87

除了这些单子句，还需将游戏规则的 3 条约束表示为对应的子句。

对于约束（1），以第二行第 4，5 与 6 三个连续的单元为例，满足约束（1）

必须：（24∨25∨26）∧（¬24∨¬25∨¬26）为真，即产生如下两个子句：

24∨25∨26，¬24∨¬25∨¬26

对于约束（2），以第 3 列为例，意味着本列中任选 5 个单元，则必须至少填一个 1 与 0，不能全填 1 或全填 0。假如选择第 3 列的第 1，3，4，6，7 五个单元，则必须：（31∨33∨34∨36∨37）∧（¬31∨¬33∨¬34∨¬36∨¬37）为真，即产生如下两个子句：

31∨33∨34∨36∨37，¬31∨¬33∨¬34∨¬36∨¬37

对于约束（3），以第 5 行与第 7 行为例，不能有完全相同的填充，则须满足：

¬{[（51∧71）∨（¬51∧¬71）]∧[（52∧72）∨（¬52∧¬72）] ∧…∧[（58∧78）∨（¬58∧¬78）]}

但上式不符合 CNF 范式，可参考文献【6】利用 Tseytin 变换将其转换成 CNF 范式子句集。下面以上述布尔表达式为例，通过引入附加变元进行转换。在这里，附加变元也可以用多位整数表示，且每位数字有相应含义，示例如下（仅供参考）：

15711= 51∧71；15710= ¬51∧¬71；1571= 15711∨15710；

15721= 52∧72；15720= ¬52∧¬72；1572= 15721∨15720；

…

15781= 58∧78；15780= ¬58∧¬78；1578= 15781∨15780；

157= ¬[1571∧1572∧…∧1578] .

其中，最高位数字 1 为行标志（2 则表示列）；次高位 5 及之后的一位 7 表示对应的第 5 行与第 7 行；第 4 位数字 1，2，…，8 分别表示行中的第 1 个单元，第 2 个单元，…，第 8 个单元；第 5 位取 1 或 0，含义自明。

因此，表示第 5 行与第 7 行不同的约束（3），需要引入附加变元 25 个；表示任意两行与两列不同的约束（3）共引入 1400 个附加变元；将会产生 4536 个子句。举例说明如下：

15711= 51∧71 转化为 CNF 时为

（51∨ ¬15711）∧（71∨ ¬15711）∧（¬51∨ ¬71∨15711）

即生成 3 个子句：51∨¬15711；71∨¬15711；¬51∨¬71∨15711。

15720= ¬52∧¬72 转化为 CNF 时为

（¬52∨ ¬15720）∧（¬72∨ ¬15720）∧（52∨72∨15720）。

1578= 15781∨15780 转化为 CNF 时为

（¬15781∨1578）∧（¬15780∨1578）∧（15781∨15780∨¬1578）。

157= ¬[1571∧1572∧…∧1578] 转化为 CNF 时为

（¬157∨¬1571∨¬1572∨…∨¬1578）∧（1571∨157）∧（1572∨157）…（1578∨157），即可产生 9 个子句。

将三个约束的所有具体要求分别转换成 CNF 子句集，连同预填提示数对应的单子句，便得到二进制数独游戏所生成的完整 CNF 公式。

在实现 DPLL 求解算法时，布尔变元一般用连续的自然数表示，因此，上述对二进制数独游戏的布尔变元编码可进行如下转换（对第 i 行 j 列单元）：

$$ij \rightarrow (i-1)\times 8+j \qquad (2.1\text{ 式})$$

这样，8 阶二进制数独游戏每个单元对应的布尔变元自然数表示如下表：

表 2.1 变元编码表

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

当利用 DPLL 算法求得对应 CNF 公式的解后，需通过（2.1 式）对应的逆变换对解的含义进行解析，获得游戏的填充方案。对于引入的附加变元也可以从 65 起进行连续表示，同学们自行设计对应的转换函数。

游戏转化为 CNF 时规模较大，游戏测试可以 6 阶甚至 4 阶算例为主。

二进制数独游戏格局生成

如何生成一个有效的二进制数独游戏格局？一种方案可以从互联网（http://www.binarypuzzle.com/）读取不少于 30 个不同的初始合法格局，用串表示每个格局，并用文件存储（此生成设计计分评定为良）；另一种方案是设计一种算法自动随机生成（此生成设计计分评定为优），一般可采用从完整合法填充开始，基于挖洞法生成[10,11]。生成的游戏格局一般要求有唯一解，但允许有多个解。如果生成算法保证只有唯一解，则生成设计计分评定为特优。

程序主控流程

根据设计问题的功能要求，图 2.4 提供了一个程序处理流程图，红色部分为基于 DPLL 的 SAT 求解相关功能模块（**课程设计首先必须完成的功能**），蓝色部分是二进制数独游戏生成、转化、求解等处理模块（**求解必须调用 DPLL SAT 求解过程**）。此流程图仅供参考，不限定同学们的设计，可以以此为参照自由发挥。

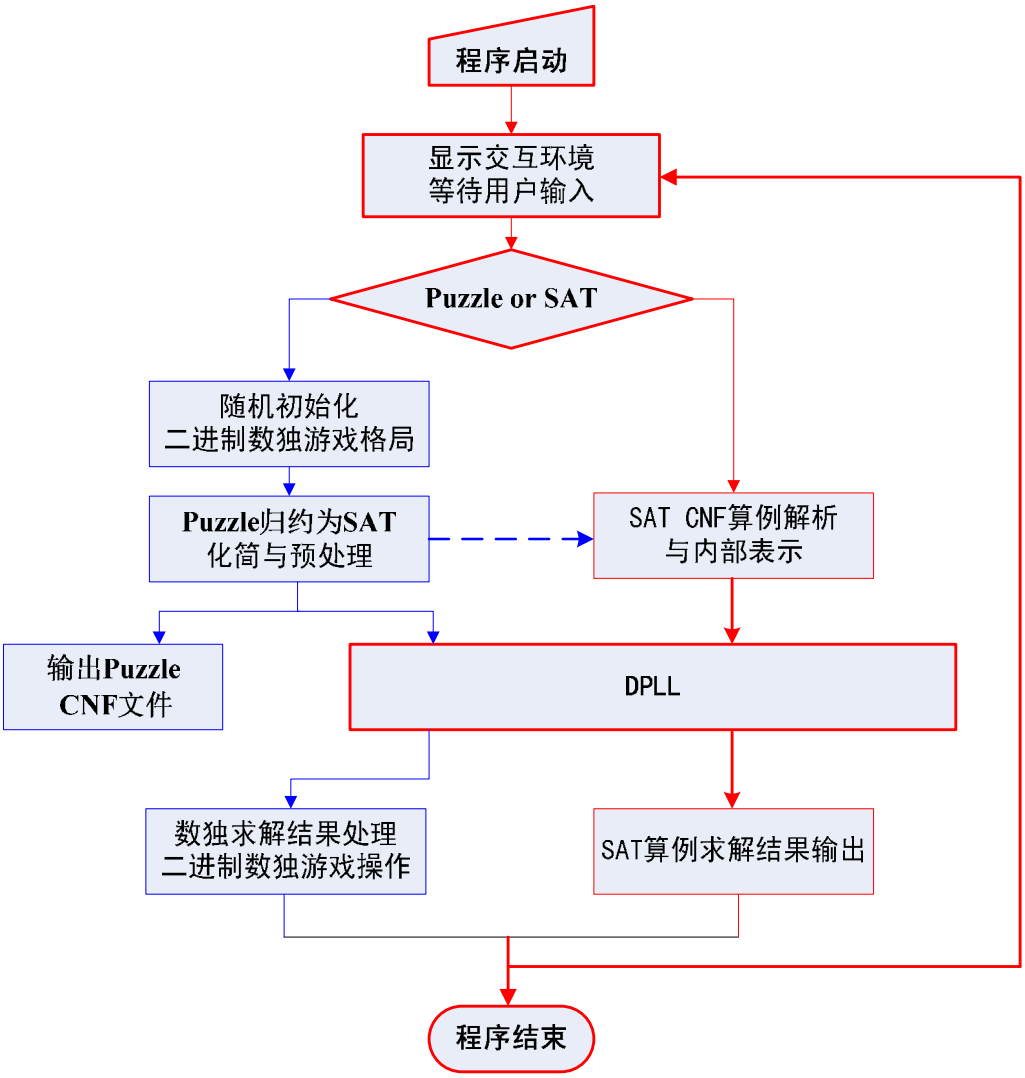

图 2.4 参考程序流程图

程序模块化

设计程序要求模块化，程序源代码进行模块化组织。主要模块包括如下：

主控、交互与显示模块（display）

CNF **解析模块**（cnfparser）

核心 DPLL **模块**(solver)

二进制数独模块,**包括游戏格局生成、归约、求解**(BinaryPuzzle)

CNF 公式的内部存储结构

本应用处理的主要数据对象有变元或文字、子句、公式等。同学们可以分析这些数据的逻辑关系及其施加的基本运算而建立相应的抽象数据类型，设计其物理存储结构。如子句有创建 createClause、销毁 destroyClause、增加 addClause、删除 removeClause、判断是否为单子句 isUnitClause、评估子句的真假状态 evaluateClause 等运算。由于每个 CNF 公式变元与子句数可能不同，同一个实例中子句长度也可能不等，一种基本的处理方式是将子句表示为由文字构成的链表；整个公式则是由子句构成的链表，如图 2.5 所示，这里仅供参考（也许并非最优结构），同学们可自行设计相应的物理存储结构并进行优化，**有效支持回溯**。

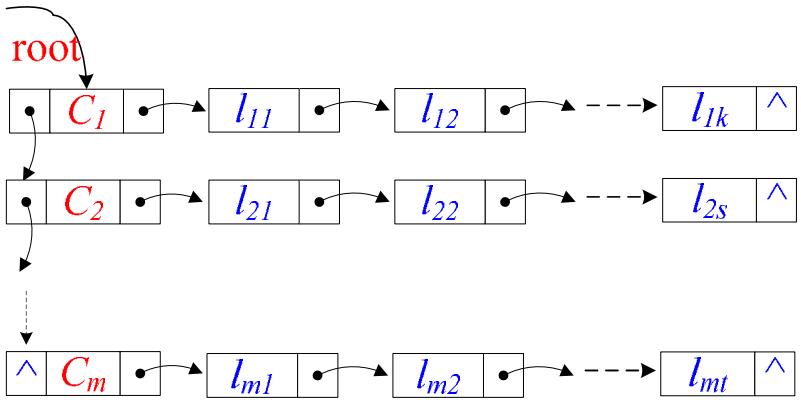

图 2.5 参考 cnf 公式存储结构图

测试算例要求(建议在内存≥8G 的计算机上执行测试)

不少于 18 个 SAT 算例，其中可满足的算例不少于 15 个，不满足的算例不少于 3 个，大中小算例各占三分之一。鉴于大家实现的可能只是初级求解器，对算例规模的要求为：小型算例变元数为 100 个左右；中型算例变元数介于 200-500 个； 大型算例变元数 600 个以上。本设计提供部分 cnf 算例集，同学们可寻找与选择、扩充测试算例。在设计报告的测试分析部分列表给出每个测试算例下列信息：算例名、算例变元数、子句数与变元数比值、满足还是不满足或不确定、DPLL 求解时间(t 与 t_0)以及优化率等信息。课堂检查时，主要对**基准算例**进行测试。

输出文件规范

对每个算例的求解结果要求输出到一个与算例同名的文件（文件扩展名为.res），文件内容与格式要求如下：

s 求解结果//1 表示满足，0 表示不满足，-1 未定

v -1 2 -3 … //满足时，每个变元的赋值序列，-1 表示第一个变元 1 取假，2 表示第二个变元取真，用空格分开，此处为示例。

t 17 //以毫秒为单位的 DPLL 执行时间，可增加分支规则执行次数信息

2.5 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] TanbirAhmed. An Implementation of the DPLL Algorithm. Masterthesis, 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] CarstenSinz. Visualizing SAT Instances and Runs of the DPLL Algorithm. J Autom Reasoning (2007) 39:219–243

[5]Binary Puzzle：http://www.binarypuzzle.com/

[6] Putranto H. Utomo and Rusydi H. Makarim. Solving a Binary Puzzle. Mathematics in Computer Science,(2017) 11:515–526

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

[8] 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.

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

[10] Sudoku Puzzles Generating: from Easy to Evil.
http://zhangroup.aporc.org/images/files/Paper_3485.pdf

[11] Baochen SUN, Xiwei SUN, Yue WU,etc. A New Algorithm for Generating Unique-solution Sudoku. IEEE Fourth International Conference on Natural Computation,2008

3 设计问题二：基于高级语言源程序格式处理工具

3.1 问题概述

在计算机科学中，抽象语法树（abstract syntax tree 或者缩写为 AST），是将源代码的语法结构的用树的形式表示，树上的每个结点都表示源程序代码中的一种语法成分。之所以说是“抽象”，是因为在抽象语法树中，忽略了源程序中语法成分的一些细节，突出了其主要语法特征。

抽象语法树(Abstract Syntax Tree ,AST)作为程序的一种中间表示形式，在程序分析等诸多领域有广泛的应用.利用抽象语法树可以方便地实现多种源程序处理工具,比如源程序浏览器、智能编辑器、语言翻译器等。

在《高级语言源程序格式处理工具》这个题目中，首先需要采用形式化的方式，使用巴克斯（BNF）范式定义高级语言的词法规则（字符组成单词的规则）、语法规则（单词组成语句、程序等的规则）。再利用形式语言自动机的的原理，对源程序的文件进行词法分析，识别出所有单词；使用编译技术中的递归下降语法分析法，分析源程序的语法结构，并生成抽象语法树,最后可由抽象语法树生成格式化的源程序。

3.2 实现的基本原理和方法

由源程序到抽象语法树的过程，逻辑上包含 2 个重要的阶段，一是词法分析，识别出所有按词法规则定义的单词；二是语法分析，根据定义的语法规则，分析单词序列是否满足语法规则，同时生成抽象语法树。

词法分析的过程，就是在读取源程序的文本文件的过程中，识别出一个个的单词。在词法分析前，需要先给每一类单词定义一个类别码（用枚举常量形式），识别出一个单词后，即可得到该单词的类别码和单词自身值（对应的符号串）。实现词法分析器的相关技术是采用有穷自动机的原理，例如：用 EBNF 表示的标识符：

<标识符>::=字母{字母|数字}

<整数>::=数字 {数字}

对应的确定有穷自动机 DFA 如图 3.1 所示。其中带圆圈的数字表示状态，双箭头指向的状态 0 为开始状态，环形的数字表示结束状态。在 DFA 中，从开始状态开始进行单词的识别，一个状态识别到一个符号后转移到下一个状态，一旦到达结束状态，成功的识别出一个单词。

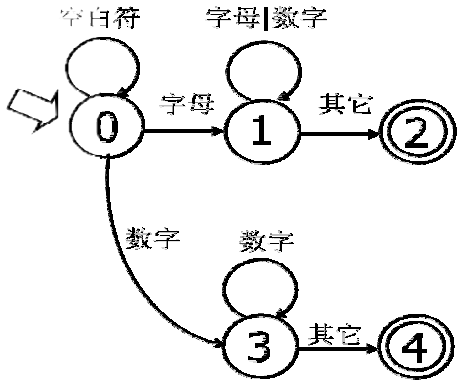

图 3.1 识别标识符和整数的 DFA

根据图 3.1 所示的 DFA，很容易得到识别标识符和整数的算法流程，如图 3.2 所示，识别出的单词类别码为 kind，自身值为 w。

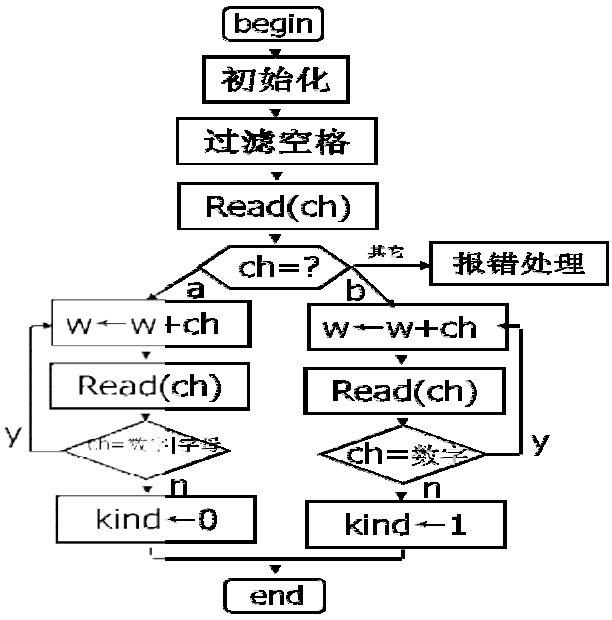

图 3.2 识别标识符和整数的算法流程图

如何来识别特定语言的所有单词，详见《综合程序设计课程设计指导-源程序格式处理》。

语法分析的过程，建议采用的实现方法是编译技术中的递归下降子程序法，递归下降子程序法是一种非常简洁的语法结构分析算法，基本上是每个语法成分对应一个子程序，每次根据识别出的前几个单词，明确对应的语法成分，调用相应子程序进行语法结构分析。例如在分析语句的语法结构时，当识别出单词 if 后，进行条件语句的处理，同时生成的子树根结点对应条件语句。处理时，首先调用表达式的子程序，得到表达式子树的根指针 T1；再递归调用语句处理部分，得到 if 子句的子树根指针 T2；再看随后的单词，如果不是 else，就表示是一个 if 语句，条件语句子树的根结点标记为“IF 语句”，有 2 棵子树，对应 T1 和 T2；如果随后的单词是 else，就再递归调用语句处理部分，得到 else 子句的子树根指针 T3；最后分析出的是一个 if-else 语句，条件语句子树的根结点标记为“IF_ELSE 语句”，有 3 棵子树，对应 T1、T2 和 T3。按此处理流程，对给定条件语句： if (a>b) m=a; else m=b；分析后生成的抽象语法树形式如图 3.3 所示。

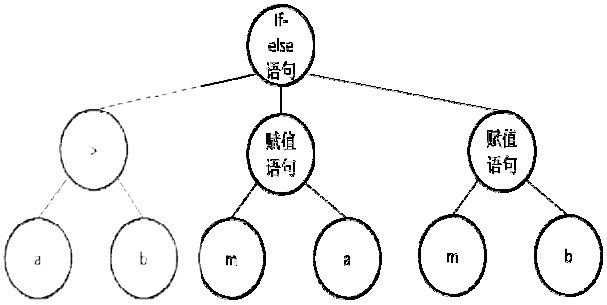

图 3.3if-else 语句的抽象语法树

在《综合程序设计课程设计指导-源程序格式处理》中，首先使用巴克斯（BNF）范式定义定义了一个简单的语言，给出了各个语法成分的处理流程框架作为实验的参考。

3.3 功能要求

1．语言定义

选定 C 语言的一个子集，要求包含：

（1）基本数据类型的变量、常量，以及数组。不包含指针、结构，枚举等。

（2) 双目算术运算符（+-*/%），关系运算符、逻辑与（&&）、逻辑或（||）、赋值运算符。不包含逗号运算符、位运算符、各种单目运算符等等。

（3）函数定义、声明与调用。

（4）表达式语句、复合语句、if 语句的 2 种形式、while 语句、for 语句，return 语句、break 语句、continue 语句、外部变量说明语句、局部变量说明语句。

（5）编译预处理（宏定义，文件包含）

（6）注释（块注释与行注释）

2．单词识别

设计 DFA 的状态转换图（参见实验指导），实验时给出 DFA，并解释如何在状态迁移中完成单词识别（每个单词都有一个种类编号和单词的字符串这 2 个特征值），最终生成单词识别（词法分析）子程序。

注：含后缀常量，以类型不同作为划分标准种类编码值，例如 123 类型为 int，123L 类型为 long，单词识别时，种类编码应该不同；但 0x123 和 123 类型都是 int，种类编码应该相同。

3．语法结构分析

（1）外部变量的声明；

（2）函数声明与定义；

（3）局部变量的声明；

（4）语句及表达式；

（5）生成（1)-(4)（包含编译预处理和注释）的抽象语法树并显示。

4．按缩进编排生成源程序文件。

3.4 实现与评测说明

实验部分按阶段进行检查评分，检查时，要求同学们自行准备好测试用例，可以不必考虑测试的源程序文件功能意义，以测试用例能覆盖全部任务要求为准。即:自行设计的测试用例务必反映系统的功能(含异常情况处理)，用例未能反映的功能，视同未实现情况评定。

（1）识别语言的全部单词。（50%）。

要求测试用例包含所有种类的单词，测试用例中没有出现的单词种类视作没有完成该类单词的识别。由于每类单词有一个种类编码（参见实验指导书用枚举常量定义），可以将识别出来的单词按种类编码进行排序显示，这样既能方便自

己的调试，也能方便检查。注意相同种类编码的多种形式，都应该包含在测试用例中，例如类型为 int 的常量，有三种形式 0123、123、0x123。

报错功能，指出不符合单词定义的符号位置。测试文件中不必包含错误符号，检查时由老师随机修改测试文件，设置错误，检查报错功能是否实现。

（2）语法结构分析与生成抽象语法树。（40%）。

要求测试用例包含函数声明，定义、表达式（各种运算符均在某个表达式中出现）、所有的语句，以及 if 语句的嵌套，循环语句的嵌套。测试用例中没有出现的语句和嵌套结构，视作没有完成该种语法结构的分析。

报错功能，指出不符合语法规则的错误位置。测试文件中不必包含错误语句等，检查时由老师随机修改测试文件，设置错误，检查报错功能是否实现。

显示抽象语法树，要求能由抽象语法树说明源程序的语法结构，这也是检查时验证语法结构分析正确性的依据。

（3）缩进编排重新生成源程序文件（10%）。对（2）的测试用例生成的抽象语法树进行先根遍历，按缩进编排的方式写到.c 文件中，查看文件验证是否满足任务要求。

实现语法结构分析时，不局限使用递归下降子程序法，但不能使用工具自动生成单词识别和语法结构分析的程序。

3.5 参考文献

[1] 王生原，董渊，张素琴，吕映芝等．编译原理（第 3 版）．北京：清华大学出版社．前 4 章

[2] 严蔚敏等.数据结构(C 语言版).北京：清华大学出版社

4 程序设计综合课程设计总体要求

4.1 坚守学术诚信

鼓励创新，进行有一定特色的设计。严禁对程序与报告的抄袭行为（包括对网络资源及其他同学的设计），一经发现，课程设计成绩计 0 分，以考试抄袭舞弊行为处理。

4.2 程序规范

程序遵从一般性规范：

⑴ 源码依据模块组织到不同.h 与.c 文件中，不要将全部程序放到一个源文件中。

⑵ 变量尽量基于描述性命名，看其名知其意。

⑶ 函数头有统一注释，说明功能，输入输出与条件等。

⑷函数内部关键处理步骤处加上注释予以说明。

4.3 报告规范，内容完善

按照计算机学院课程设计报告的要求及本课程设计报告的格式规范与内容要求撰写设计报告，避免出现错别字及形式的不规范现象。报告主要内容应至少涵盖如下方面(以下非报告目录)。

一、问题描述

二、需求与技术现状分析

三、程序总体设计(含模块结构图)

四、数据结构和算法详细设计

五、程序实现

(C 语言程序实现的简要说明，如开发环境、支持包、函数原型与功能及调用关系；全部源程序以电子版提供，报告中只能作为附录内容之一)

六、程序测试及结果分析

七、复杂度分析

八、总结、特色与不足

主要参考文献

附录一：源程序

附录二：程序使用说明

4.4 课堂与考勤要求

要求按时到实验室完成综合程序设计，根据完成与验收情况由指导老师批准方可在其它场所查阅资料，撰写报告。设计课坚持记录考勤。

4.5 检查与验收

在设计课内，全体同学需给指导老师或助教演示程序，解释程序，回答老师提问，验收或报告完成情况。

指导参考书目录

[1] 曹计昌，卢萍，李开. C 语言与程序设计. 电子工业出版社，2013

[2] 严蔚敏等.数据结构（C 语言版）.清华大学出版社，

[3] Larry Nyhoff. ADTs, Data Structures, and Problem Solving with C++. Second Edition,Calvin College,2005

[4] 殷立峰. Qt C++跨平台图形界面程序设计基础. 清华大学出版社, 2014:192～197

[5] 严蔚敏等. 数据结构题集（C 语言版）.清华大学出版社

附录 A 程序设计综合课程设计评价指标

评价指标	满分	评价标准
程序功能（40%）	100	题目一成绩=（1）至（6）项功能得分之和，其中，处理满足算例记 0.8，不满足算例记 0.2。
		题目二成绩=（2）至（4）项功能得分之和。其中（2）（3），处理正常用例记 0.9，报错用例记 0.1。
程序规范（10%）	100	程序规范：80，注释：80+，模块化且注释好：90+，不规范：80-。
设计特色（10%）	100	特色不明显：70，有一定特色：80+，特色突出或有创意：90+
报告内容（30%）	100	问题描述与分析：20，程序总体设计、数据结构、算法设计和理论分析：60，测试计划及测试分析：20。
报告规范（10%）	100	基本规范：80，规范：80+，不规范：80-。
逾期扣分	10	逾期提交：2/天。超过 5 天者本次实验记 0。
综合成绩＝设计成绩×92%＋实验考勤×8% 设计成绩＝(∑程序功能×40%＋设计特色×10%＋程序规范×10%＋报告内容×30%＋报告规范×10%－逾期扣分)		

注：实验考勤原则上仅记录签到情况，不考虑任何请假情形。