**西安电子科技大学**

**研究生课程考试试题**

**（答案必须写在答题纸上或在答题卡上填涂）**

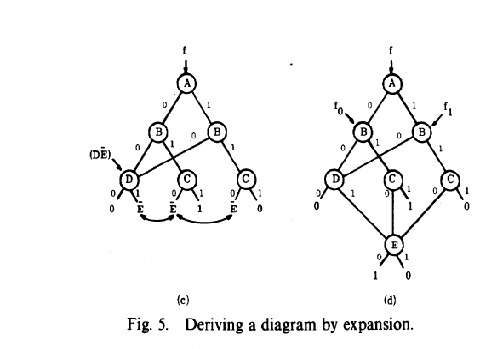
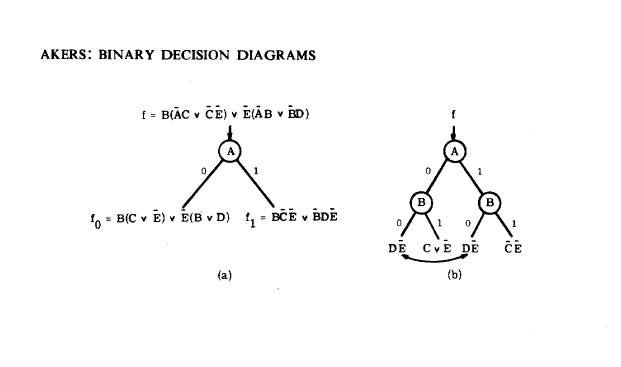
考试科目： 课程编号：

考试日期： 2016 年 1 月 22 日 考试时间： 120 分

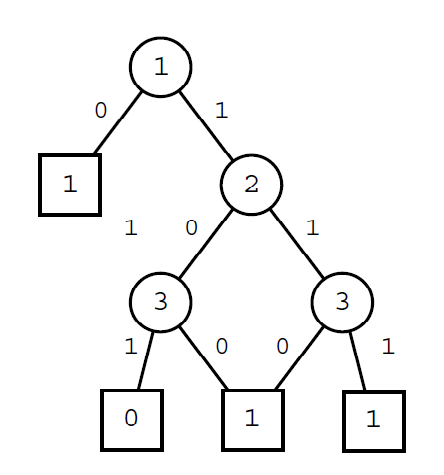
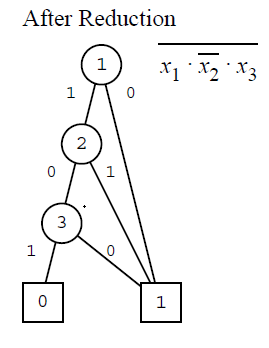
考试方式：(闭卷) 任课教师： 班号

学生姓名： 学 号：

1. (20分) Give the diagram deriving from the Shannon expansion formula of function(using Binary Decision Diagrams).



1. (20分) Transform the following arbitrary function graph into a reduced graph denoting the same function.

1. (15分) Judge if there are conflicting clauses in the following CNFs. If not, write all feasible assignments.

**不行就画个真值表。。。**



假设x=0,=>y=0+=>z=0

x=0,y=0,z=0=>=0

假设x=1,=>z=1+=>y=1

x=1,y=1,z=1,=>0

所以说，所有的值都使得这个语句结果为0，根据定义，是冲突语句



=>

=>

所以，x、z一个为1

1. x=1,z=0,=>y=0
2. x=0,z=1,随便

存在x、y、z的值，使得这个语句结果不为0，所以不是冲突语句。

=>

=>

=>

x=0,=>z=1,y=1

x=1,=>y=1,=>z=1

存在x、y、z的值，使得这个语句结果不为0，所以不是冲突语句。

1. (10分) Prove the following equation.

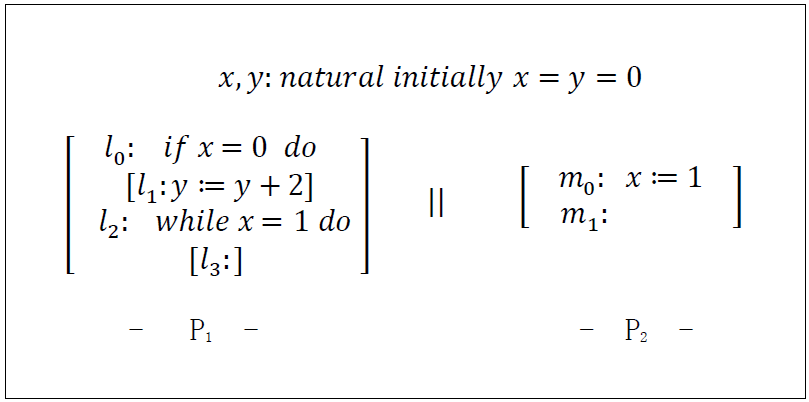
画个真值表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x | y | z | x+y | y’+z | 左 | x+z | 右 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

由真值表可知，

1. (15分) Translate the following program into logical formula.

Program:



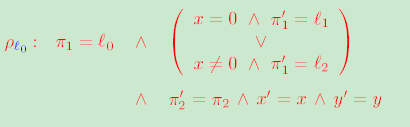
State variables:

Initial condition:

Transition relation:

Write the disjuncts and

(For example:)



1. (20分) Clause data store mechanism in Chaff solver.
   1. Chaff stores clause data in a large array. List the advantages of using array compared with pointer data structure.（5分）
2. 在存储器上利用非常有效
3. 占用连续内存空间，访问局部速度加快
4. 数组数据结构与链表相比，就高速缓存缺失（cache misses）而言，有一个很大优势，它在求解过程中转化为可观的加速。
   1. Design a data structure based on arrays to store clause data.（15分）

It is invariant that in any state where a clause can become newly implied, both watched literals are not assigned to 0. A key benefit of the two literal watching scheme is that at the time of backtracking, there is no need to modify the watched literals in the clause database. Therefore, unassigning a variable can be done in constant time. Further, reassigning a variable that has been recently assigned and unassigned will tend to be faster than the first time it was assigned. This is true because the variable may only be watched in a small subset of the clauses in which was previously watched. This significantly reduces the total number of memory accesses, which, exacerbated by the high data cache miss rate is the main bottleneck for most SAT implementations. Figure 1 illustrates this technique. It shows how the watched literals for a single clause change under a series of assignments and unassignments. Note that the initial choice of watched literals is arbitrary, and that for the purposes of this example, the exact details of how the sequence of assignments and unassignments is being generated is irrelevant.

1. 