1. (20分) 图1表示¬(*x*1·*x*3) 与*x*2·*x*3进行“或（or）”操作的过程，请给出该运算的**过程图（Evaluation Graph）与最终化简图，并给出化简后的表达式。**

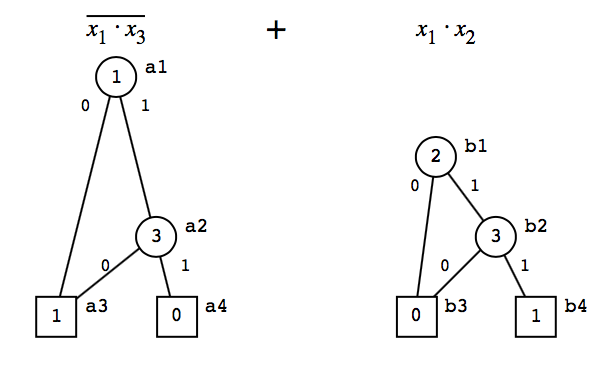


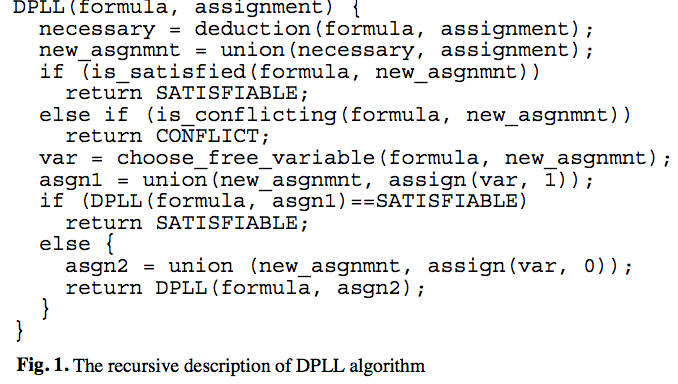
图1

1. (20分)子句（Clause）数据结构设计

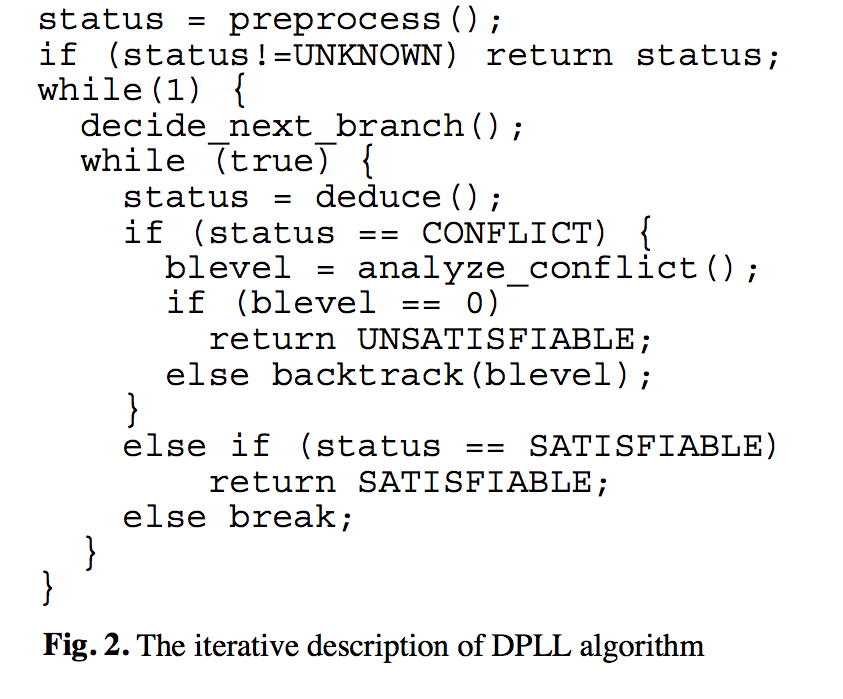
2.1 设计一个基于数组的数据结构存储子句数据（clause data）中的文字、子句与公式。（15分）

2.2 列举出相比于指针（pointer-heavy）数据结构，用数组存储子句的优点。（5分）

3.（10分）简述为什么在DPLL算法的实现中往往采用迭代（图3）实现而不是递归实现（图2）。



**图2 DPLL 算法的递归实现**

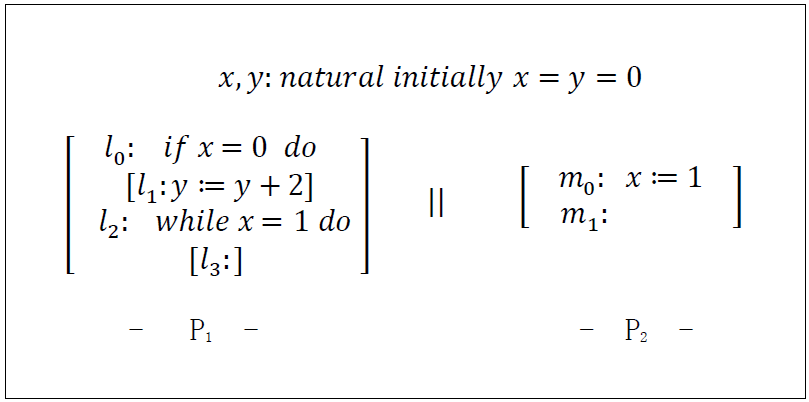


**图3 DPLL 算法的迭代实现**

4. （15分）基于计数器的BCP算法是一种容易理解与实现的 BCP 算法。假设每个子句（clause）拥有两个计数器（counter），一个用于子句中的值1字面量（literal）的计数，一个用于子句中的值0字面量的计数。每个变量（variable）都有两个列表，其中包含所有子句，其中该变量分别显示为正值和负值。当为变量分配一个值时，包含此字面量的所有子句将更新其计数器。设实例具有*m*个子句（clause）和*n*个变量（variable），并且平均每个子句具有*l*个字面值（literal）。**那么每当给一个变量赋值时，有多少个计数器****（counter）需要更新。简要概述分析过程。**

5. (15分) 将下列程序转换成逻辑公式。

Program:



State variables:

Initial condition:

Transition relation:

Write the disjoints and

(For example:)

6. (20分) “天花板与地板”模型（the Alloy Analyzer）

sig Platform {}

*there are “Platform” things*

sig Man {ceiling, floor: Platform}

*each Man has a ceiling and a floor Platform*

pred Above(m, n: Man) {m.floor = n.ceiling }

*Man m is “above” Man n if m's floor is n's ceiling*

fact {all m: Man | some n: Man | Above (n,m)}

*"One Man's Ceiling Is Another Man's Floor"*

**assert BelowToo**

**{all m: Man | some n: Man | Above (m,n)}**

*"One Man's Floor Is Another Man's Ceiling"?*

现在假设有2个Platform实例，2个 Man 实例，找出至少一个不满足断言“BelowToo”的反例，并用图形表示。