DPLL的基本的思想

基本概念

* CNF：一个逻辑或一个或多个子句。
* Clause子句：逻辑或一个或多个文字
* Literals文字：变量的正向或负向出现
* Conflicting clause 冲突子句：将所有文字都分配为值0的子句
* status = preprocess(); //预操作，对应 3.5 节内容
* if (status!=UNKNOWN) return status;
* while(1) {  
  decide\_next\_branch(); //变量决策环节，对应3.1节内容 while (true) {
* status=deduce(); //推理环节(BCP)，对应3.2节内容 if (status == CONFLICT) {
* blevel=analyze\_conflict(); //冲突分析，对应3.3节内容 if (blevel == 0)  
  return UNSATISFIABLE;
* } else backtrack(blevel); //智能回溯，对应 3.3 节内容
* else if (status == SATISFIABLE) return SATISFIABLE;
* else break; }
* }

简单分析一下迭代的框架，首先执行的是 preprocess()这个预处理操作，其实就是对 CNF 实例进行各种化简减轻后续的求解工作量。如果预处理不能直接 得出结果的话，就进行后面的 decide\_next\_branch()操作，这就是变量决策操作，它会分析从哪个变量开始赋值是最合适的。对变量赋值以后，会执行一个 deduce() 操作，可以管它叫推理操作，目的是识别该赋值所导致的必要赋值，这些会在 3.2 节中具体介绍。 当然不正确的赋值可能会产生错误，这就会产生冲突，我们需要 analyze\_conflict()分析这个冲 突得到冲突发生的根源位置，然后通过 backtrack()回溯到这个位置，为这个变量赋另外一个值， 继续往下搜索，如此循环，直到找到满足 SAT 的一组真值指派。如果回溯到了最顶层还没有解 决问题的话，那就表示这个 CNF 实例是不可满足的。 CNF就是合取范式。

DPLL使用迭代而不使用递归的原因

* 递归是老版本的DPLL框架 迭代是新版本的DPLL框架 因为递归速度慢并且容易发生溢出，但是迭代的速度就会比较快并且不容易发生溢出，并且这个迭代有很多自身的优势。就是非时间顺序（智能回溯）的优势，基于冲突分析和学习的非时序回溯算法就可以智能的分析冲突产生的原因并回跳多个决策层，把导致冲突的字句加到子句集，也就是找到冲突产生的原因（分析的过程）并且告诉SAT处理这些搜索空间会产生冲突，之后就不要再踩这些坑（这就是学习的过程）
* 递归解决冲突的方法就是回到上一层，将变量取值翻转，继续往下进行搜索，这也叫时序回 溯。但是这样做的结果很可能是冲突依旧存在，因为上一层的赋值也许并不是冲突产生的根本原 因，从而白白浪费一次计算的时间，效率非常低。

DPLL中使用数组存储Clause字句而不使用指针的原因

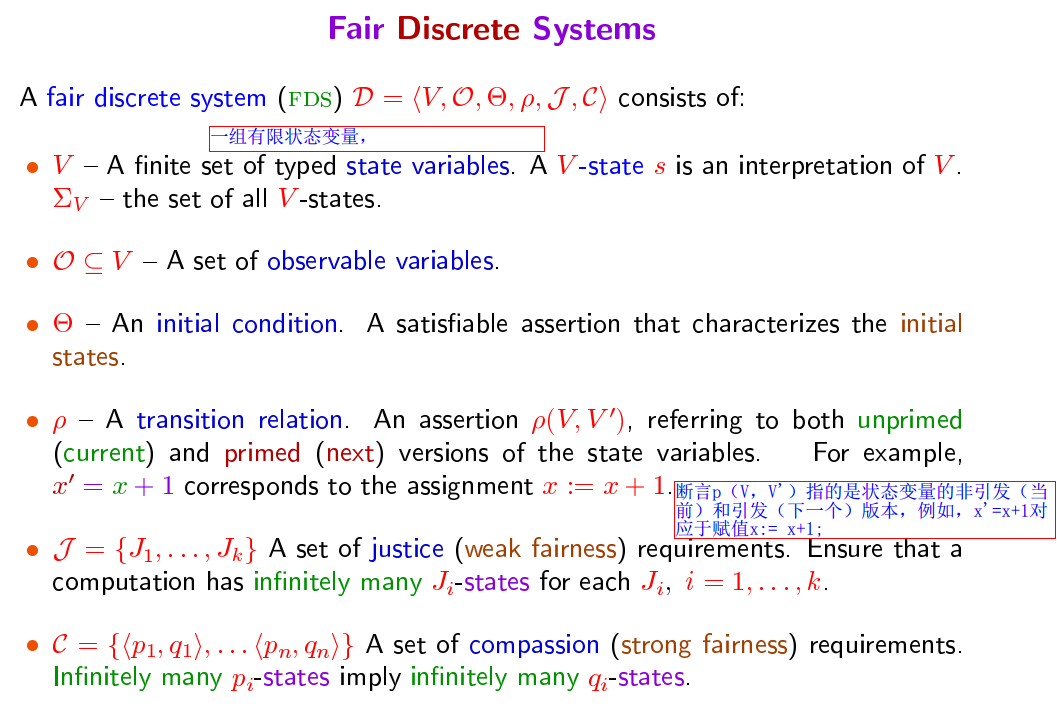
1. 在存储器上利用非常有效
2. 占用连续内存空间，访问局部速度加快
3. 数组数据结构与链表相比，就高速缓存缺失（cache misses）（也就是缓冲未命中）而言，有一个很大优势，它在求解过程大大加快求解的速度。
4. 链表存储方式，尽管比较方便对子句进行操作，例如增加和删除子句，但是存储效率很低，因为在处理的过程中因为缺少 access locality(局部访问)的能力，往往会造成很多的 cache misses(缓存丢失)。而采用的数组的方式，尽管没有链表灵活，但是存储效率大大提高，access locality(局部访问)的能力也有大幅提升。

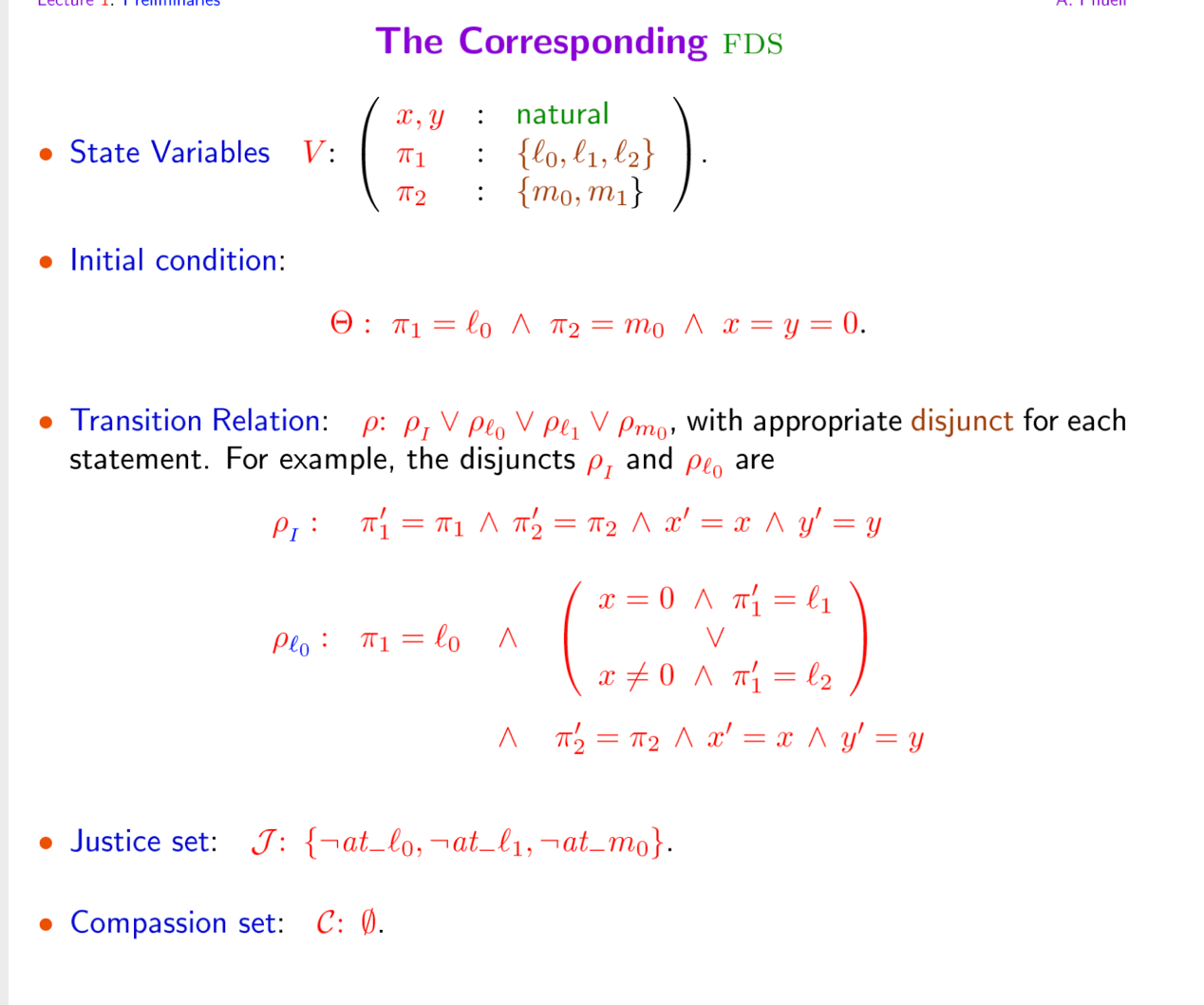
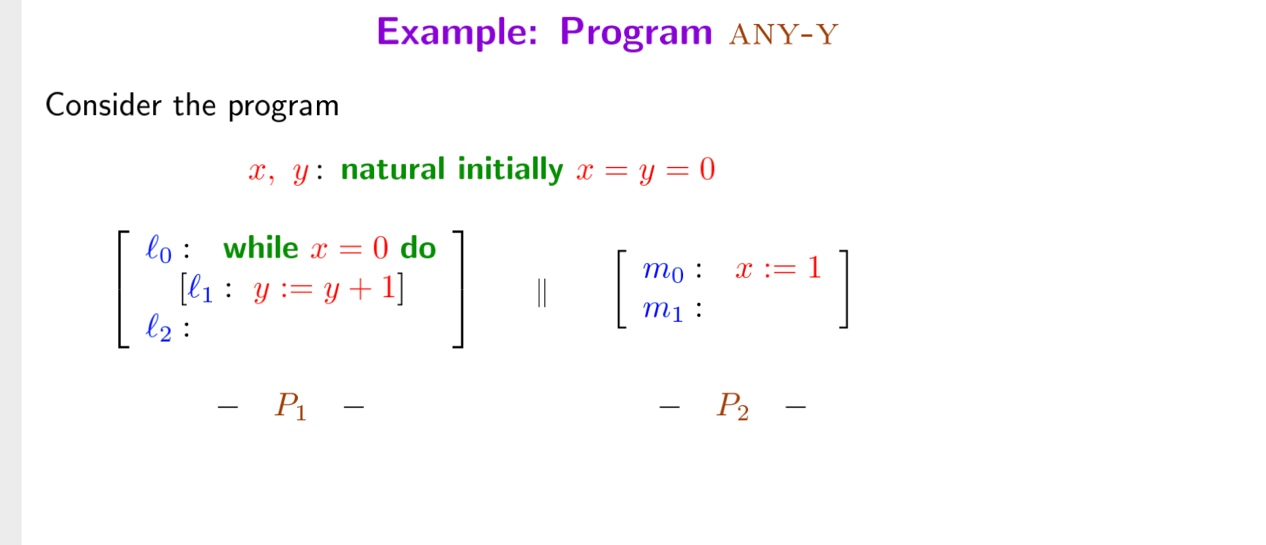
设计一个基于数组的数据结构存储子句数据（clause data）中的文字、子句与公式。

这里就是Chaff采用的数组方式

1. （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。**

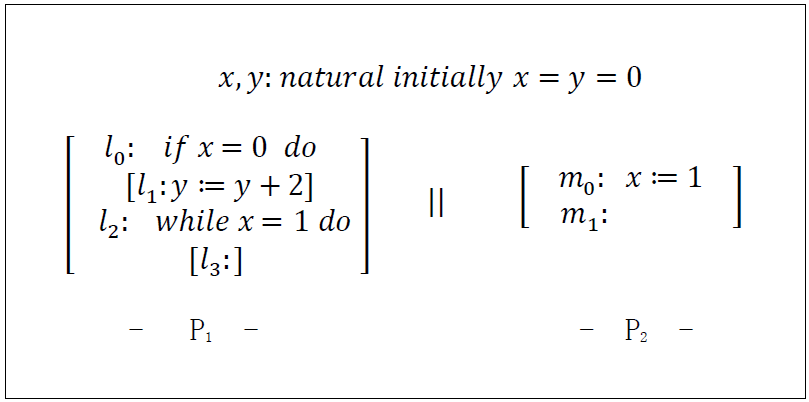
有一个变量被赋值的时候，会有平均 ml/n 个计数器需要更新，在回溯的时候，每取消 一个变量赋值，也会平均有 ml/n 个计数器的更新

把程序翻译为逻辑公式



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

Program:



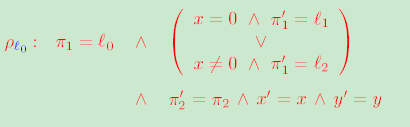
State variables:

Initial condition:

Transition relation:

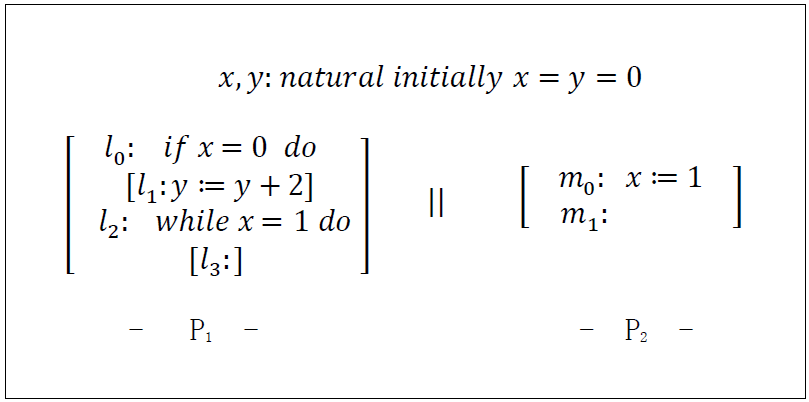
Write the disjuncts and

(For example:)



5. (15分) 将下列程序转换成逻辑公式。

Program:



State variables:

Initial condition:

Transition relation:

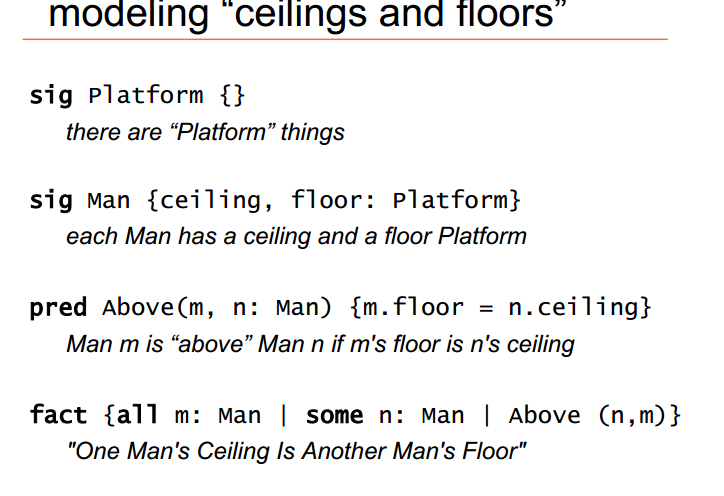
Write the disjoints and

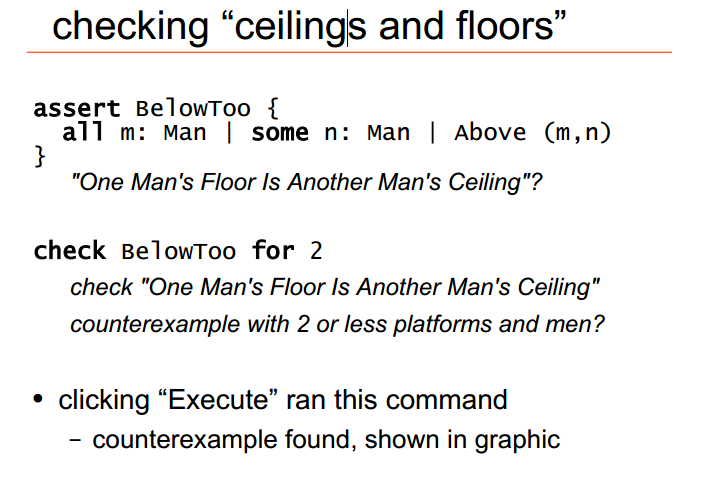
(For example:)

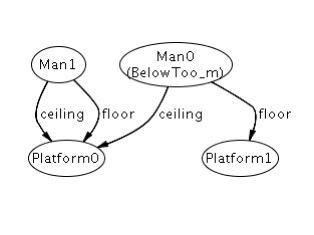
# 子句冲突规则

# 恒等式（天花板与地板模型） （the alloy analyzer）

# Ceiling and floors







pred p(x1: e1, ..., xn: en) { F } named formula with declaration parameters用声明参数命名公式

assert : constraint intended to follow from facts of the model 约束意在跟随模型的事实

