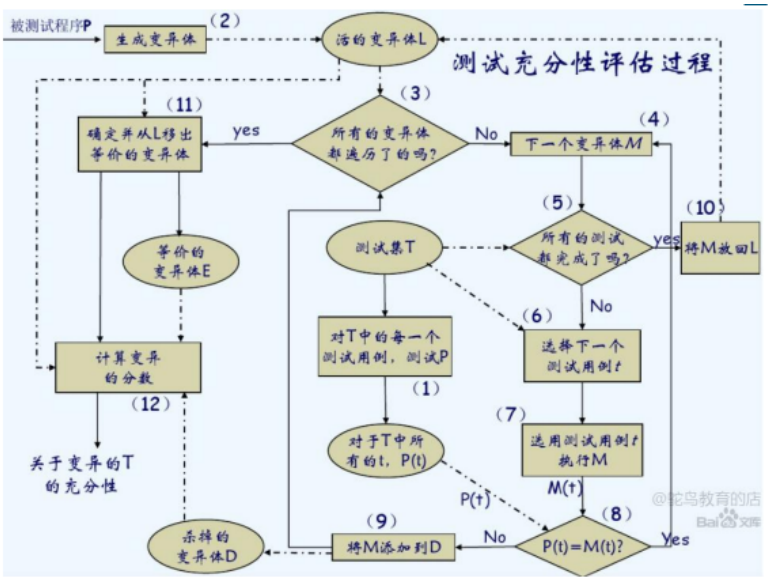
5.5 变异测试

- 基本思想
  - 给定一个程序 P 和一个测试数据集 T，通过**变异算子**为 P 产生一组**变异体** Mi，对 P 和 M 都使用 T 进行测试
    - 如果某 Mi 在某个测试输入 t 上与 P 产生不同的结果，则该 Mi 被**杀死**
    - 如果某 Mi 在所有测试数据集上都与 P 产生相同的结果，则称其为 **活的变异体**
    - 接下来对活的变异体进行分析，检查其是否**等价于**P
      - 对不等价于 P 的变异体 M 进行进一步的测试，知道**充分性度量**达到满意的程度
- 程序变异概念
  - 如果T中所有的测试t使得P(t)=P'(t)，称T**不能区别**P和P'。那么称在测试过程中P'是活的
  - 如果在程序P的输入域中不存在任何测试用例t使得P与P'区别，则称P'等价于P
  - 如果P'不等价于P，且T中没有测试能够将P'与P区别，则认为T是不充分的。
  - 不等价而且是活的变异体为测试人员提供了一个生成新测试用例的机会，进而增强测试T
- 应用变异评估测试的充分性
  - 产生了 k 个突变体，其中 k1 个突变体被杀死，并且 k - k1 个突变体是活的
  - k - k1 = 0，测试样例 T 对于变异是充分的
  - k - k1 > 0，则变异得分： M = k1 / (k - e)，e 是等价变异的数量，e ≤ (k - k1)
- 评估过程



- 变异差错
- 变异算子

Mutant operator	In P	In mutant
Variable replacement	z=x*y+1;	x=x*y+1; z=x*x+1;
Relational operator replacement	if (x<y)	if(x>y) if(x<=y)
Off-by-1	z=x*y+1;	z=x*(y+1)+1; z=(x+1)*y+1;
Replacement by 0	z=x*y+1;	z=0*y+1; z=0;
Arithmetic operator replacement	z=x*y+1;	z=x*y-1; z=x+y-1;

- 一阶和高阶变异体
  - 通过一次改变获得的变异体成为一阶突变
  - 通过两次改变活的的突变体为二阶突变体
  - 实践中仅采用一级突变体
    - 降低测试成本
    - 大多数高阶突变体通过与一阶突变体相关的测试被杀死
- 理论基础
  - 程序员能力假设：被测试程序是由足够程序设计能力的程序员书写，所产生的的程序是接近正确的
  - 组合效应假设：假设简单的程序设计错误和复杂的程序设计错误之间具有组合效应，一个测试数据如果能够发现简单的错误，也能够发现复杂的错误
  - 以上两个假设来自**软件开发实践**，确立了变异测试**基本特征**：通过变异算子对程序做一个较小的语法上的变动来产生一个变异体
- 变异测试工具
  - 提供特征
    - 变异算子可选调色板
    - 对测试集 T 的管理
    - 在测试集 T 上执行被测程序病保存输出，与在变异体上执行被测程序的结果进行比较
    - 变异体的生成
    - 使用用户识别的等效变异体执行变异和计算变异分数
    - 增量变异测试：即允许将变异算子子集应用于被测程序的一部分
- 优缺点
  - 优点
    - 排错能力强
    - 自动化程度高
    - 灵活性高
    - 变异体与被测试程序的差别信息可以较容易地发现软件的错误
    - 可以完成语句覆盖和分支覆盖

- 缺点
  - 需要大量的计算机资源完成充分性分析
    - n 行产生 n^2 变异体
    - 存储变异体的开销大
    - 变异体与被测程序的等价判断需要人工判定