Generating Range Fixes for Software Configuration

MF1233055 周昊一

许多的软件都有复杂的配置选项,用以应对不同的用户以及目标平台的不同需求.比如Linux内核可以被配置成在不同的硬件系统上运行,用户通常需要选择CPU架构(x86,ARM等),文件类型(ext4,JFS等),以及显卡驱动(ATI,NVIDIA等).

这些配置选项,或者说特性,通常使用不同的模型语言来描述的.比如Linux内核使用的是Kconfig,一个叫做eCos的嵌入式操作系统使用的是CDL.

配置器将这些模型变成交互式的控制界面,并协助用户进行正确和完整的配置.为了达到这一目的,配置器需要检测出可能存在的配置错误.配置错误是指与一些约束条件相冲突的选择.

像Kconfig这样的配置工具,会在某个约束条件被违反时使相关的选项自动无效化来避免配置错误.而CDL则是允许违反一些约束,但是会为每一条被违反的约束提供可能的修复操作.

现在主要面临的问题是:1. 当需要激活一个被无效化的特性时,通常需要花去很多的时间才能搞清楚如何激活它;2. 配置器所提供的修复建议很可能是没有用的.这时就需要人工修复不完整性问题.一个可能的解决方案就是生成修复建议时要包含完整的修复方案集合,并且还要为被无效化的特性提供激活方案.

本文提出了一种新的生产修复方案的方法.主要的贡献有三点:1. 提出了一种新的概念range fixes,即不是告诉用户具体的改动,而是告诉用户可以被接受的改动范围.2. 设计了一个修复方案生成算法;3. 在eCos系统上对这种方法进行了评估.

range fix需要满足4个条件:

1. 正确性,即修改后的参数e’要满足约束c
2. 范围最大,即不存在一个修复方案f’,使f’是正确的,并且f’覆盖的范围比f要大.
3. 变量最少,即不存在一个修复方案f’,使f’是正确的,并且f’所包含的参个数数比f要少.
4. 完备性,对于一个约束的违反(V,e,c),所产生的修复集合L要满足:1) L中的任何两项在语义上都是不同的;2) L中的任一项都满足1,2,3; 3) 任意一个满足1,2,3的修复都在语义上等同于L中的某一项.

在生成range fix时,首先从约束得到最小的参数集合,再从参数集合生成对应的修复方案.