贴心云衣柜

——基于机器学习和社交网络的服装搭配推荐

曹佳涵1) 金亦凡1) 李鑫烨1) 李光耀1)

1)(南京大学计算机科学与技术系 南京 210093)

**摘 要** 随着物质生活水平的不断提高，服装搭配已经成为一个出门前必须进行的活动。它既是展现自我个性与风采的恰当途径，又有着十分重要的社交意义。在这一背景下，本项目开发了一款基于机器学习和社交网络的服装搭配推荐APP——贴心云衣柜（SweetWard）。与市面上现有APP的差异之处在于，该APP不仅可以根据天气、场合等，从用户已有的衣物集当中为用户生成合理的穿衣搭配，还可以通过用户的穿衣历史信息和衣物搭配收藏，运用一种新兴的个性化推荐算法——协同过滤算法，向用户推荐社交网络中与该用户风格相近的用户以及他们的穿衣搭配收藏。文中对本APP的开发背景、工具及算法、实现过程、成品展示等进行了详细的阐述。

**关键词** Android；服装搭配推荐；机器学习；协同过滤；社交网络

SweetWard

—Dressing recommendations based on machine learning and social networking

Jiahan Cao 1) Yifan Jin 1) Xinye Li 1)  Guangyao Li 1)

1)(Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

**Abstract** With the continuous improvement of material standard of living, dressing has become a necessary activity before going out. It is not only an appropriate way to show one’s personality and style, but also of social significance. SweetWard, a dressing recommendation APP based on machine learning and social network, is developed in this paper. Different from existing applications in the APP store, this APP can offer satisfactory clothing collocation for users from their existing clothing collection according to the weather conditions, wearing occasions and so on. Besides, it also recommends users with similar styles and their clothing collocation collection in social networks to the certain user based on his dressing history information and clothing collocation collection, applying a personalized recommendation algorithm, collaborative filtering algorithm. In this paper, the development background, tools and algorithms, implementation process and finished product display of the APP are described in detail.

**Key words** Android；dress recommendation；machine learning；collaborative filtering；social networking

# 引言

## 项目开发背景

在物质生活水平不断提高的现代社会，人们对于服装搭配的要求也在一天天地提高：不仅要做到舒适得体，还要用不同的穿搭应对不同的场合。可以说，服装搭配已经成为了现代人出门之前必须要解决的一个问题。然而与此同时，随着生活节奏的进一步加快，思考如何搭配着装和选择购买新衣物的时间也愈加缺乏。在这种情况下，我们不难发现，一个智能的、个性化的，能应对上述诸多问题的穿衣搭配推荐APP，必然能够吸引一个很大的用户群体，在APP市场中占有一席之地。

## 需求分析

在项目开发的前期准备时，开发小组访问了APP商店，发现市场中已经存在很多有关穿衣搭配的APP，但是这些APP的推荐功能较为基础，它们只是简单地处理了用户现有的衣物的搭配问题，并没有推荐系统的处理。事实上，用户更加青睐于个性化的APP，期望APP能够符合自己的品味。更进一层的需求是满足大多数人都具有的融入集体的心理，在社交网络中分享自己的风格，并找到同道中人。

以网易云音乐为例，该款APP的成功之处并不在于其与其他音乐APP相同或相似的音乐播放功能或庞大的音乐库，而是在于它特有的个性化音乐推荐功能，以及完善的社交功能，包括音视频分享功能和评论功能等。用户可以通过强大的推荐功能找到更多符合自己听歌风格的歌曲，丰富个人歌单；同时用户还能够评论、分享自己喜欢的歌曲，参与到社区互动，也可以找到听歌风格相近的用户集群。这种用户体验，是一个单纯的音乐播放器所无法提供的。

因此，在分析大多数用户的需求、并与市面上的同类型或同性质的APP进行横纵向比较之后，开发小组明确了开发方向，即开发一款这样的APP：它既有普通穿衣搭配APP的功能（包括衣物管理系统、服装搭配收藏系统以及基于用户现有衣物和天气等客观信息的搭配推荐系统），又有类似于网易云音乐的个性化推荐系统和社交系统。用户能够收到APP为其提供的符合该用户个性的一套或多套穿衣搭配（内含的衣物单品可能不在用户现有衣物中），也能够参加社交活动，查看好友的衣物搭配，同时在社交网络中寻找到与其风格类似的其他用户。

## 开发意义

从用户体验的角度分析，该APP能够提供当下市面上各种穿衣搭配APP所不能提供的一些用户体验。当用户想要在自己的衣柜中增添新衣物时，个性化的推荐功能可以更快地给用户指明方向；同时，在用户社群中找到风格类似的同道中人也能为用户带来极好的体验感，用户之间可以互相借鉴穿衣搭配，交流某种风格的穿搭心得等。

项目主要阶段以实现功能为主，在开发完成之后如果对整个项目进行优化，包括界面优化、性能优化等，那么这款APP成品必定能够占据一定市场，吸引一定量的用户集群。

# 使用工具与算法

## 工具介绍

# Android Studio

开发小组使用Android Studio进行APP的客户端开发。Android Studio是Google公司推出的一个集成开发工具，基于IntelliJ IDEA，功能十分强大

# 服务器

## 算法介绍

本文采用的是基于模型检验的方法对数字系统的设计进行验证，采用的模型为有穷状态机（Finite State Machine，FSM）。有穷状态机，是表示有限个状态以及这些状态之间的转移和动作等行为的数学模型。状态存储着关于过去的信息，它反映系统从开始到现在时刻的输入变化。转移引导状态的变更，并且用必须满足确使转移发生的条件来描述它。所有状态中，可以从初始状态迁移到达的都是可达状态，对模型的检验就是检验这些可达状态是否符合给定的规约。

在数字电路系统中，有穷状态机是一种十分重要的时序逻辑电路模块。它对数字系统的设计和验证都具有十分重要的作用。

## NuSMV介绍

模型检验是一种自动验证有穷状态机是否满足规范的形式化方法。目前已有较多的模型检验工具，如SPIN[7]、NuSMV。NuSMV擅长同步系统的验证[8]，更适合于硬件描述语言的验证，本项目模型检验功能基于NuSMV。

NuSMV含义是New Symbolic Model Verifier，是对第一代SMV的重新实现，加入了很多新的模型检测算法，采用CUDD（Colorado University Decision Diagram）包提供高效的BDD（Binary decision diagram，二叉决策图）操作，缓解了模型检验中状态爆炸的问题。

模型检验中通常使用线性时态逻辑（Linear-time Temporal Logic, LTL）和计算树逻辑（Computation Tree Logic, CTL）。NuSMV对两种逻辑均支持。一个典型的用smv语言描述的有穷状态机的结构如下：

MODULE main

VAR …

ASSIGN …

MODULE submodule

…

SPEC …

NuSMV必须有一个main模块，main模块可以对其他模块进行实例化，SPEC是对该状态机的规约，即要验证的属性。变量有两大类，一类是State Variables（用VAR表示，状态的值表示某个状态），另一类是Input Variables（通过关系表示状态）。对状态机的描述有ASSIGN、TRANS、INIT等等语句。

# 基于决策树的本地衣物搭配推荐

## 提取过程

语法树的提取是一个编译原理范畴内的工作，这是整个项目的第一步，也是将具体的Verilog代码抽象化，化为一个有层次数据结构进行存储的一步。一般来说语法树的提取需要进行词法分析，语法分析[9]。词法分析是将程序分割成一个个词素流，语法分析则是在词素的基础上将他们按照规定的产生式从终结符号生成非终结符号并组织成树。考虑到Verilog语法的复杂性所以我们使用了PyVerilog[10]来辅助我们生成关于Verilog的语法树。PyVerilog是一个专门用来将Verilog代码的语法层次化结构化输出的工具，非常便于进行Verilog代码的语法分析，有了这个工具，我们可以直接跳过词法分析的步骤，直接生成语法树。生成语法树的过程就是通过读入PyVerilog产生的语法结构然后根据不同的语法模块生成对应的语法节点，在特定的节点中写入必要的信息，然后将整个节点组织成一棵树。

## 语法树结构

在程序中所采取的树的结构是子女-父母-兄弟链表树。即，每个节点有一根直接指向第一个子女节点的指针，有一根直接指向自己下一个兄弟节点的指针，（除根节点之外）有一根直接指向自己父节点的指针。之所以要有指向父节点，子节点和兄弟节点的指针，是因为考虑到在进行后续的语义分析，转换工作时，经常需要联合多个语法模块的信息来综合考虑。结合编译原理的知识，可以解释为：分析继承属性时需要父节点和兄弟节点的信息，分析综合属性时需要子节点的信息。但是因为每个节点可能的子女数并不确定，所以我们很难却硬性规定每个节点该有多少子女指针。解决的方法就是只配备一根仅仅指向第一位子节点的指针，如果从父节点想要访问其中的某个子节点，仅仅需要先根据子节点指针找到第一子节点，然后根据子节点的兄弟指针寻找之后的子女，直到找到目标子节点。这个过程也可以包装成为一个API方便使用。这种设计方式虽然在寻找靠后的子节点时效率较低，但是极大的简化了数据结构，便于后续的程序设计，考虑到实际中的Verilog语法树每个节点的子节点数目并不会太多，所以时间的开销也不会过大。

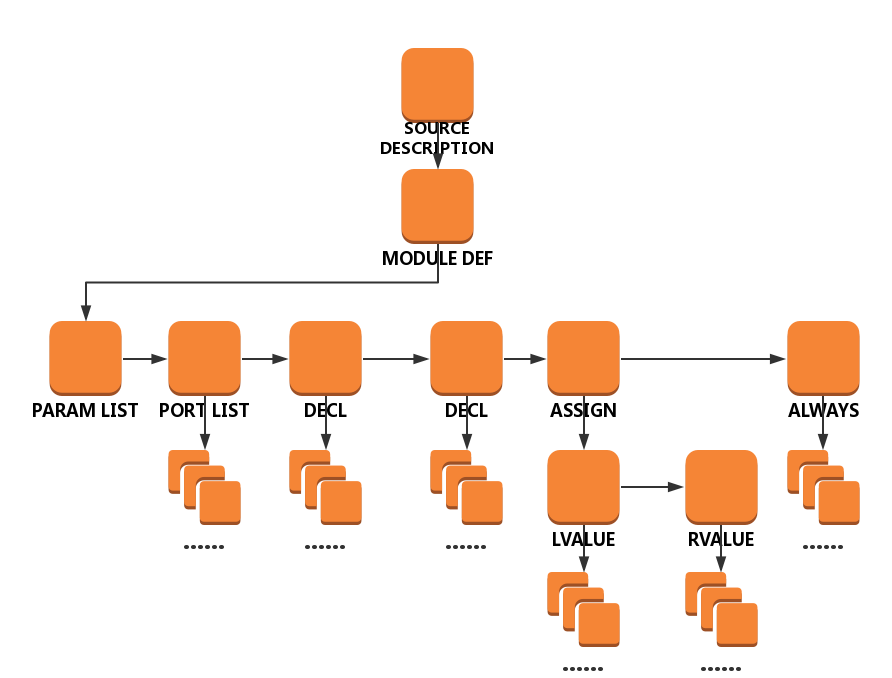


Figure 1 语法树结构示意图（省略了父指针）

上图是语法树的结构示意图（省略了父指针），可以看到根节点SOURCE DESCRIPTION节点直接指向自己的子女节点MODULE DEF节点，该节点又指向了第一个子女节点PARAM LIST节点，PARAM LIST节点则指向了自己的后继兄弟节点。

## 节点的设计

考虑到不同的语法模块所需的信息类型也不同，所以语法树节点SyntaxNode的设计必须满足多种类型的复用。采用的方法是将语法树节点进行分级，所有种类的节点都必须的信息直接存储在一级节点中，如果遇到特殊节点需要特殊信息，例如模块名，变量名，敏感列表类型等等，则将他们存放在二级节点中，一级节点有指针指向二级节点，这样就可以存储所有类型的语法信息了。

生成所有对应的语法节点并且将他们全部组织成语法树之后，为了方便之后的程序设计，这一部分又设计了一些常用的API，例如获取一个表达式的操作数等。语法分析结束后可以选择打印语法树进行检查，语法树打印结果如下。

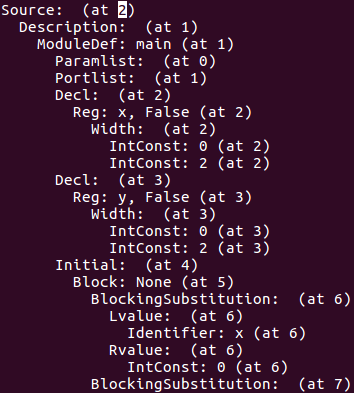


Figure 2 语法树打印结果

# 基于协同过滤的多用户推荐

## 以用户为基础(User-based)的协同过滤

协同过滤(Collaborative Filtering)，简单来说是利用某兴趣相投、拥有共同经验之群体的喜好来推荐用户感兴趣的信息，个人透过合作的机制给予信息相当程度的回应（如评分）并记录下来以达到过滤的目的进而帮助别人筛选信息，回应不一定局限于特别感兴趣的，特别不感兴趣信息的纪录也相当重要。协同过滤又可分为[评比](https://zh.wikipedia.org/w/index.php?title=%E8%A9%95%E6%AF%94&action=edit&redlink=1)(rating)或者[群体过滤](https://zh.wikipedia.org/w/index.php?title=%E7%BE%A4%E9%AB%94%E9%81%8E%E6%BF%BE&action=edit&redlink=1)(social filtering)。其后成为电子商务当中很重要的一环，即根据某顾客以往的购买行为以及从具有相似购买行为的顾客群的购买行为去推荐这个顾客其“可能喜欢的品项”，也就是借由社群的喜好提供个人化的信息、商品等的推荐服务。

在2001年，Sarwar提出了基于项目的协同过滤推荐算法([Item-based Collaborative Filtering Algorithms](https://zh.wikipedia.org/w/index.php?title=Item-based_Collaborative_Filtering_Algorithms&action=edit&redlink=1))。以项目为基础的协同过滤方法有一个基本的假设“能够引起用户兴趣的项目，必定与其之前评分高的项目相似”，透过计算项目之间的相似性来代替用户之间的相似性。 方法步骤：

(1) 收集用户信息：收集可以代表用户兴趣的信息。一般的网站系统使用评分的方式或是给予评价，这种方式被称为“主动评分”。另外一种是“被动评分”，是根据用户的行为模式由系统代替用户完成评价，不需要用户直接打分或输入评价数据。电子商务网站在被动评分的数据获取上有其优势，用户购买的商品记录是相当有用的数据。

(2) 针对项目的最近邻搜索：先计算已评价项目和待预测项目的相似度，并以相似度作为权重，加权各已评价项目的分数，得到待预测项目的预测值。例如：要对项目 A 和项目 B 进行相似性计算，要先找出同时对 A 和 B 打过分的组合，对这些组合进行相似度计算。

(3) 对于项目来讲，它们之间的相似性要稳定很多，因此可以脱机完成工作量最大的相似性计算步骤，从而降低了在线计算量，提高推荐效率，尤其是在用户多于项目的情形下尤为显著。

## 隐语义模型(Latent Factor Model)

在我们的推荐环境设定下，每个用户的数据由他对各类衣物的偏好程度组成，这个偏好程度由他的历史穿衣数据以及浏览记录数据计算得到。但我们真正要从数据中提取的，是每个用户对于不同穿衣风格的偏好，由于不同种类的衣物可能会属于同一种风格，因此衣物种类之间并不是完全独立的。在理想情况下，我们希望通过一种相互独立的属性，能将不同风格的用户划分开来，而这种属性并不是显式地体现在数据中的。另一方面，由于每个用户并不是对于每种衣物都有数据，因此用户-衣物矩阵是稀疏的，很多信息因此缺失，我们需要重新计算并预测新的用户-衣物矩阵来进行用户相似度计算，因此我们采用隐语义模型(Latent Factor Model)来求解问题。

# 基本思想

核心思想: 通过隐含特征(latent factor)联系用户兴趣和物品。具体来说，就是对于某个用户，首先得到他的兴趣分类，然后从分类中挑选他可能喜欢的物品。基于兴趣分类的方法需要解决3个问题：

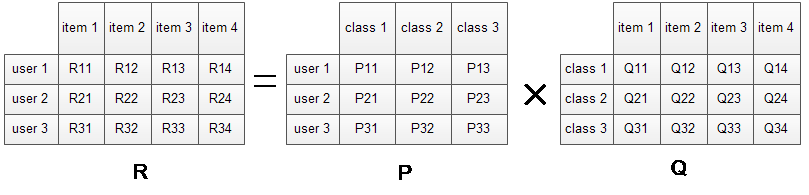
1. 如何对物品进行分类？
2. 如何确定物品对哪些类的物品感兴趣，以及感兴趣的程度？
3. 对于一个给定的类，选择哪些属于这个类的物品推荐给用户，以及如何确定这些物品在一个类中的权重？

隐含语义分析技术(latent variable analysis)采取基于用户行为统计的自动聚类，可以较好解决上面提出的问题。

1. 代表用户意见分类来自对用户行为的统计，和 ItemCF 在物品分类方面的思想类似，如果两个物品同时被多个用户喜好，那么这两个物品可能属于同一个类
2. 控制分类粒度自定义分类个数
3. 一个物品多分类计算出物品属于某个类的权重，因此每个物品都不是硬性地被分到某一个类中
4. 多维度分类基于用户的共同兴趣计算出来的，如果用户的共同兴趣是某一个维度，那么 LFM 给出的类也是相同维度
5. 物品在分类下的权重统计用户行为决定物品在某一个分类中的权重，如果某个类的用户都会喜欢某个物品，那么这个物品在这个类中的权重可能比较高

# 算法

对于一个的用户行为数据集(数据集是一个的矩阵，为用户数量，为衣物种类数量，表示第个用户对于第种衣物的偏好值)，使用LFM对其建模后，我们可以得到如下图所示的模型：(假设数据集中有3个用户, 4种衣物, LFM建模的分类数为4)



对于一个用户来说，当计算出他对所有item的偏好后，就可以进行排序并作出推荐。LFM算法设定K个类(Class)，将R矩阵表示为P矩阵和Q矩阵相乘。其中P矩阵是User-Class矩阵，矩阵值表示的是对的偏好值；Q矩阵式class-item矩阵，矩阵值表示的是在中的权重，权重越高越能作为该类的代表。所以LFM根据如下公式来计算用户u对衣物种类i的兴趣度：

在计算矩阵P和矩阵Q中参数值时，我们使用最小化损失函数的方法来求参数。损失函数如下所示：

上式中最后两项是用来防止过拟合的正则化项，和为超参数，根据实验调参得到。损失函数的优化使用随机梯度下降算法：

1. 通过求参数和的偏导确定最快的下降方向；
2. 迭代计算不断优化参数，直到参数收敛。

其中，是学习速率，越大，迭代下降的越快。经过实验和调参，我们最终确定的值。

# NuSMV代码生成

从FSM到NuSMV的转换涉及到许多细节处理。程序的整个处理流程可以类比于从源代码编译到目标代码，本环节将FSM中间层表示转换为NuSMV代码类比于汇编代码转换为机器码。

## 基本表达式

NuSMV支持绝大多数常见的操作符。操作数和操作符结合使用可产生基本表达式，基本表达式与次态表达式结合可产生复杂表达式。

处理基本表达式的转换首先要解决的问题是操作数的问题。Verilog与NuSMV有两种截然不同的类型系统。Verilog有wire、reg、input、output等，而NuSMV的VAR和IVAR两种变量声明类型可以表示bool、array、word、integer。[11] 可见两者关联性很低。

NuSMV对类型约束很严格。任何不同类型之间不允许隐式转换。integer的值不能超出限定范围。我们尝试过reg变量用bool表示0（False）和1（True）的状态，一维数组用integer来表示范围。然而这两种类型不兼容，使用时必须使用类型转换符。且integer类型变量一旦超出范围不会自动进行模运算，这与寄存器级的逻辑表达不相符，且验证结果会因为死锁而总是为真。我们最终采用全部使用word来表达一个变量，而处理input、output、reg时采用方法有所不同。word的麻烦之处在于两个操作数的类型和宽度必须匹配，若右边是整形常量，它就不符合类型系统，必须进行类型转换。

另一个转换是操作符的转换。Verilog与C语言很相似，NuSMV表示方式也很相似。但仍然存在不同之处。比如NuSMV没有逻辑与、逻辑或，只有按位与、按位或。这使得在分支语句中必须进行额外的检查。

完成相应工作后，将基本表达式语句的转换封装为一个函数，供多处使用。

## 状态转移描述

状态的描述有初态和次态。初态使用init来约束，次态使用next来约束。如果不对变量的初态进行约束，则NuSMV验证时在第一状态直接给出规约的反例，即初始化一个不满足规约的状态集合，使得检测停止。如果不对次态进行约束，则次态可以在该变量的表示范围内任意变化，这个特性可以用来描述Verilog模块调用的input类型变量的参数，比如产生时钟。

对于初态的约束，若在Verilog源代码没有给出的明确的初始化值时，则初始化为0，这与Verilog的硬件综合是相一致的。

对于次态的约束，依赖于FSM。通过FSM结构取得相应的状态转移条件和转移执行动作。读这些条件和动作做相应分析，生成一个case语句，约束这个变量的次态变化。

## 时序逻辑转换

时序逻辑在Verilog代码中主要表现在具有上升沿或下降沿的always语句块中。

上升沿和下降沿属于硬件特性，Verilog是专门为硬件系统设计的描述语言，直接支持该特性。而NuSMV变量中没有一种是能描述这种性质的。这个问题在项目初期显得很难解决，主要原因在于我们使用NuSMV的TRANS语句进行转换。TRANS是描述状态转移的语句之一，通过研究和测试，其对组合逻辑支持较好。而对于时序逻辑，如上升沿，很难加入到TRANS中。

我们设计了一种解决方式，将一个具有边沿触发属性的Verilog变量表示为NuSMV的两个变量，这两个变量的组合描述Verilog中该变量状态。举例说明，Verilog的变量a被表示为NuSMV的当前状态变量a和上一状态变量a#，a=0且a#=1则可以表示下降沿。

为此，我们全面转移到ASSIGN语句描述的状态机，修改了前期的大量代码。但这个过程是值得的，ASSGIN语句使得另外一些难以解决的问题有了解决方案。

## 模块实例化

模块的概念在Verilog和NuSMV中很相似，均有实例名和模块名的调用，使得转换上更加自然。

NuSMV有顶层模块的概念，相当于一个程序的入口。而Verilog描述的硬件，上电工作，代码可能存在一个顶层模块（只调用其他模块且没有被其他模块调用的属于顶层模块）。为了使得“程序入口”更加清晰，我们需要用户在规约中提供顶层模块名。这个顶层模块被一个生成的main模块实例化，main模块中还会做其他的初始化工作。

另一个问题是Verilog参数可以乱序，而NuSMV不允许这样做，解决方式是实例化时对参数重新排序。

# 工具的实现与检验

## 整体框架

整个项目的整体框架如Figure 3所示，程序的输入包括需要检查的Verilog源代码以及用户提供的规约SPEC。Verilog源代码经过PyVerilog程序处理得到相应的AST,然后经过NuSMV Generator 程序生成NuSMV合法的状态机。与输入文件SPEC一起作为NuSMV的输入进行模型检测，得到最终的输出结果，并且产生反馈，对应到Verilog源代码上，提供可能出现错误区域的提示。

在此基础上，我们编写的图形化界面能够智能的将这些功能集成在一起。图形化界面支持用户从界面内写入修改或者从本地文件打开。通过这种方式获取项目的输入，然后利用系统调用的方法依次进行上述的处理过程，得到输出的结果与反馈。更新在图形化的界面上，使得用户能够更加清晰直观的得到模型检测的结果。

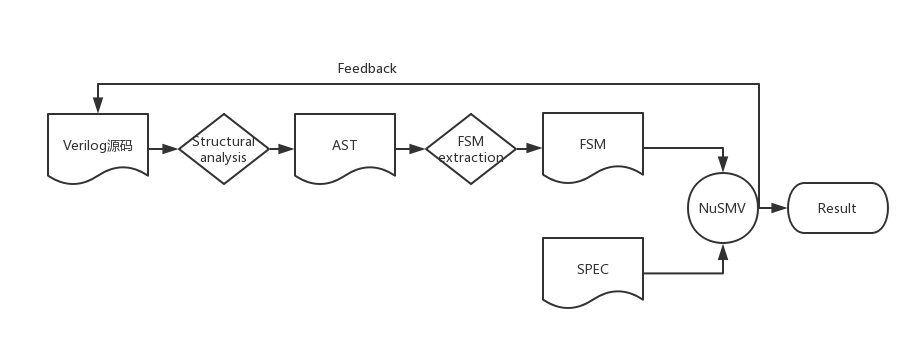


Figure 3 整体框架

## 工具实现

上文对本项目的理论、实现技术大体框架等进行了详细说明。本节对我们实现的工具的具体效果进行展示。

代码的实现更加需要在细节上加以注意，如Verilog输入后应该有预处理的过程，使得语法分析过程不需要考虑如宏定义头文件包含等等问题。程序运行时应该检测运行时环境，避免多个进程间通讯时共享资源分配问题。

Figure 4 是验证一个Verilog代码是否满足规约的图形化界面。最左边是输入Verilog代码的区域，中上方是输入用户规约的地方，中下方是验证规程，如果执行成功为绿色，否则为红色，出错的原因可能是Verilog语法错误、规约语法错误等等。最右边的框显示了验证的属性是否成立，成立为绿色，不成立及其反例路径为红色。



Figure 4 图形化界面

利用FSM，我们可以对Verilog代码生成其状态转移图。Figure 5是一个状态转移图的例子。每个变量都具有一个触发器，触发它进行状态迁移，状态迁移的条件为一条边，若边上没有信息则表明触发器信号使得该变量无条件执行相应动作。第三层及以下的每个节点表示执行动作。

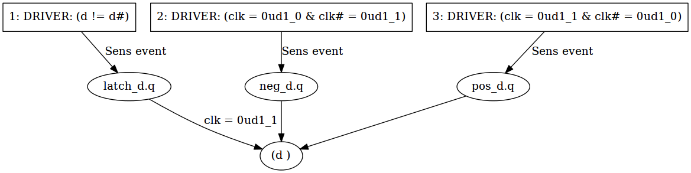


Figure 5

Figure 6是反向路径追踪，根据反例在FSM中找出不满足条件的状态，定位在Verilog代码中，为用户给出提示，表示可能不满足规约的语句范围。

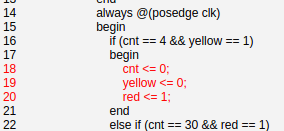


Figure 6

## 基于案例的工作流程具体分析

本案例的Verilog代码见附录A，规约见附录B。

这段Verilog代码描述了一个交通灯系统，有红、绿、黄三种颜色交替变化。本处验证其交替变化的几种条件是否满足。代码第16行存在一个错误，黄灯的转移条件应为计数器为3的状态。

首先，对Verilog代码进行预处理，本文件没有宏定义和头文件包含。预处理输出结果。

对文件进行语法分析产生语法树并输出结果，如下：

Source: (at 1)

Description: (at 1)

ModuleDef: traffic\_lights (at 1)

Paramlist: (at 0)

Portlist: (at 1)

Decl: (at 2)

Input: clk, False (at 2)

……

分析代码中的符号，产生符号表，以下为NuSMV的表示形式：

clk : unsigned word[1];

cnt : unsigned word[5];

green : unsigned word[1];

……

根据语法树生成状态机，下面为初始节点和一个状态：

traffic\_lights.cnt: DRIVER: (clk = 0ud1\_1 & clk# = 0ud1\_0)

COND: (((cnt = (0ud5\_4 ))& (yellow = (0ud1\_1 )))), BEHAVIOUR: (((0ud5\_0 )))

对规约文件进行预处理，提取相关信息等。

将FSM转换为Graphviz的输入文件，使用dot工具生成状态转移图。

根据NuSMV语法规则，将FSM生成NuSMV输入文件。合成输入文件和规约。

使用NuSMV进行验证，并得到验证结果。

分析验证结果，给出相应提示。

根据提示可以很容易找到相应错误，修改之后再次验证，此时规约全部通过。

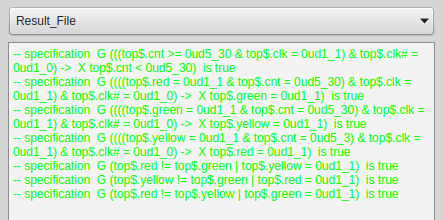


Figure 7 修改后规约全部通过

# 总结

在本次项目实践中，我们对于模型化检验以及Verilog程序语言都有了更加深刻的了解，同时设计了一个能够输入为Verilog程序和对应规约，输出为以程序对应的状态机，NuSMV代码，对规约检查的结果以及在原代码上给出的修改建议。转换程序实现了对本科“数字电路设计实验”课程涵盖的Verilog语法的大部分内容，并且以实验课程中学生自己编写的代码作为测试样例进行测试，在给出错误检查结果同时也能给出修改意见，极大的简化了在编写Verilog程序时调试错误的成本和代价，节约了大量的调试时间。在之后的后续推广过程中，我们会进一步增加转换程序所能够涵盖的Verilog程序的语法范围，进一步提高错误检查的效率以及错误修改建议的精度，并且在项目进一步成熟之后在本科生之中进行推广，希望能够最终在“数字电路设计实验”课程中发挥作用，作为辅助教学的工具，实质性的为本科教学创新和进步贡献力量。

致 谢 特别感谢项目指导老师——卜磊副教授自始至终对我们所给予的支持和鼓励！

参 考 文 献

[1] 赵丽芳. 基于 UPPAAL 和 UML 的实时系统形式化分析与应用[D]. 苏州大学,2008

[2] NuSMV: A New Symbolic Model Verifier A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri <http://nusmv.fbk.eu/>

[3] Automatic verification of knowledge and time with NuSMV A Lomuscio, C Pecheur, F Raimondi

[4] Model checking time Petri nets using NuSMV, A Bobbio, A Horvath

[5] Modeling freshness concept to overcome replay attack in kerberos protocol using nusmv S Adyanthaya, S Rukmangada

[6] 龙菲, 赵一帆. 基于FPGA的逻辑分析仪[J]. 硅谷, 2014(8):197-198.

[7] Spin - Formal Verification: <http://spinroot.com/spin/whatispin.html>

[8] 张军林. NuSMV模型验证器实现分析. 中山大学硕士学位论文. 2010-06-02

[9] Aho, Sethi, Ullman, Compilers: Principles, Techniques, and Tools, Addison-Wesley, 1986. ISBN 0-201-10088-6

[10] Shinya Takamaeda-Yamazaki: Pyverilog: A Python-based Hardware Design Processing Toolkit for Verilog HDL, 11th International Symposium on Applied Reconfigurable Computing (ARC 2015) (Poster), Lecture Notes in Computer Science, Vol.9040/2015, pp.451-460, April 2015.

[11] Roberto Cavada, Alessandro Cimatti,. Charles Arthur Jochim, NuSMV 2.6 User Manual. <https://nusmv.fbk.eu/NuSMV/userman/v26/nusmv.pdf>

附录A

1 module traffic\_lights();

2 input clk;

3 reg red;

4 reg green;

5 reg yellow;

6 reg [0:4]cnt;

7 initial

8 begin

9 red = 1;

10 green = 0;

11 yellow = 0;

12 cnt = 0;

13 end

14 always @(posedge clk)

15 begin

16 if (cnt == 4 && yellow == 1)

17 begin

18 cnt <= 0;

19 yellow <= 0;

20 red <= 1;

21 end

22 else if (cnt == 30 && red == 1)

23 begin

24 cnt <= 0;

25 green <= 1;

26 red <= 0;

27 end

28 else if (cnt == 30 && green == 1)

29 begin

30 cnt <= 0;

31 green <= 0;

32 yellow <= 1;

33 end

34 else

35 begin

36 cnt <= cnt + 1;

37 end

38 end

39 endmodule

附录B

--TOP traffic\_lights

LTLSPEC

G((top$.cnt >= 0ud5\_30 & top$.clk = 0ud1\_1 & top$.clk# = 0ud1\_0) -> X(top$.cnt < 0ud5\_30))

LTLSPEC

G((top$.red = 0ud1\_1 & top$.cnt = 0ud5\_30 & top$.clk = 0ud1\_1 & top$.clk# = 0ud1\_0) -> X(top$.green = 0ud1\_1))

LTLSPEC

G((top$.green = 0ud1\_1 & top$.cnt = 0ud5\_30 & top$.clk = 0ud1\_1 & top$.clk# = 0ud1\_0) -> X(top$.yellow = 0ud1\_1))

LTLSPEC

G((top$.yellow = 0ud1\_1 & top$.cnt = 0ud5\_3 & top$.clk = 0ud1\_1 & top$.clk# = 0ud1\_0) -> X(top$.red = 0ud1\_1))

LTLSPEC

G(top$.red != top$.green | top$.yellow = 0ud1\_1)

LTLSPEC

G(top$.yellow != top$.green | top$.red = 0ud1\_1)

LTLSPEC

G(top$.red != top$.yellow | top$.green = 0ud1\_1)