第十五讲

自动机及应用

自动机及应用

- 时间自动机
- Büchi自动机
- 软件的形式化验证
- 模型检测方法
- M^3C模型检测系统

自动机及应用

- 时间自动机
- Büchi自动机
- 软件的形式化验证
- 模型检测方法
- M^3C模型检测系统

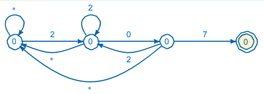
自动机

- 系统模型：常表示为变迁模式
- 自动机:
  - 有限自动机
  - 下推自动机
  - 图灵机
  - 时间自动机
  - Büchi自动机
  - Petri 网
  - …

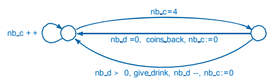
自动机应用

例：

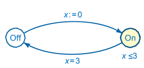
- 数码门禁锁



- 自动售货机

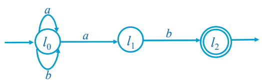


- 时间转换器

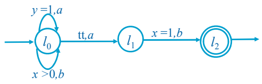


时间自动机

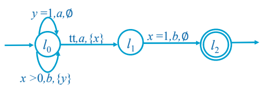
- 有限自动机：输入为时间



- 带警戒（条件）的变迁：$X=\{x,y\}$



- 带警戒和时钟复位的转移



时间自动机: 文法

定义：时间自动机为六元组

$$A = (L, L_0, L_{acc}, \Sigma, X, E)$$

有限位置集合

初始位置

接受位置集

有限字母表

有限时钟集

转移边集

$L_0 \subseteq L$

$L_{acc} \subseteq L$

$E \subseteq L \times G \times \Sigma \times 2^X \times L$

$G = \{\wedge x \lozenge c \mid x \in X, c \in N\}$ 为警戒集合，$\lozenge \in \{<, \leq, =, \geq, >\}$

时间自动机: 文法

例：

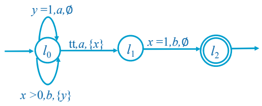
$L = \{l_0, l_1, l_2\}$

$L_0 = \{l_0\}$

$L_{acc} = \{l_2\}$

$\Sigma = \{a, b\}$

$X = \{x, y\}$



$$l_0 \xrightarrow{x>0, b, \{y\}} l_0$$

时间自动机: 语义

- 赋值: 对每一时钟赋值 $v \in R^x$
- 状态: $(l, v) \in L \times R^x$ 由位置和赋值构成.
- 自动机 A 状态间迁移:

  若 $\exists\, (l, g, a, Y, l') \in E$ 且 $v \models g$ 和

$$\begin{cases} v'(x) = 0, & \text{若 } x \in Y \\ v'(x) = v(x), & \text{其它} \end{cases}$$

  - 延时迁移: $(l, v) \rightarrow (l, v + \tau)$
  - 离散迁移: $(l, v) \rightarrow (l', v')$

时间自动机: 语义

- 自动机的链：

$$(l_0, v_0) \rightarrow (l_0, v_0+\tau_1) \rightarrow (l_1, v_1) \rightarrow (l_1, v_1+\tau_2) \rightarrow \cdots \rightarrow (l_k, v_k)$$

简记为:

$$(l_0, v_0) \xrightarrow{\tau_1, a_1} (l_1, v_1) \xrightarrow{\tau_2, a_2} (l_2, v_2) \xrightarrow{\tau_3, a_3} \cdots \xrightarrow{\tau_k, a_k} (l_k, v_k)$$

  - 延时迁移: $(l, v) \rightarrow (l, v + \tau)$
  - 离散迁移: $(l, v) \rightarrow (l', v')$

时间自动机: 语义

- 时间序列:

$$t = (t_i)_{0 \leq i \leq k}, \quad R_+ \text{中有限非降序列.}$$

- 时间串:

$$w = (\sigma, t) = (a_i, t_i)_{0 \leq i \leq k}$$
$$= (a_0, t_0)(a_1, t_1) \cdots (a_k, t_k)$$

  其中 $a_i \in \Sigma$， t 时间序列。

时间自动机: 语义

- 接受时间串：

$$w = (a_0, t_0)(a_1, t_1) \cdots (a_k, t_k)$$

  被时间自动机 A 接受, 如果存在链：

$$\rho = (l_0, v_0) \xrightarrow{\tau_0, a_0} (l_1, v_1) \xrightarrow{\tau_1, a_1} \cdots \xrightarrow{\tau_k, a_k} (l_{k+1}, v_{k+1})$$

  且 $l_0 \in L_0$, $l_{k+1} \in L_{acc}$, $t_i \in \Sigma_{j<i}\tau_j$.

- 接受时间语言:

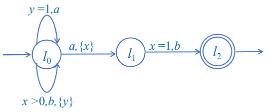
$$L(A) = \{w \mid w \text{ 被} A \text{接受}\}$$

时间自动机: 语义

注:

时间自动机中，常省略

- 警戒条件 *tt*,
- 时钟集为空集.



$w = (b, 0.1)(b, 0.3)(a, 1.3)(b, 1.5)(a, 1.5)(b, 2.5)$

为一个接受时间串. w 的接受链为:

$$(l_0,0,0) \xrightarrow{0.1,b} (l_0,0.1,0) \xrightarrow{0.2,b} (l_0,0.3,0) \xrightarrow{1,a} (l_0,1.3,1)$$

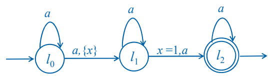
$$\xrightarrow{0.2,b} (l_0,1.5,0) \xrightarrow{0,a} (l_1,1.5,0) \xrightarrow{1,b} (l_2,2.5,1)$$

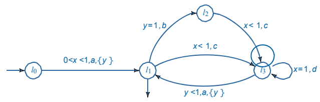
$(l_i,t_i,v(x))$

时间自动机: 语义

例：

$L(A) = \{(a, t_1) \cdots (a, t_k)$
$\mid \exists i < j,\ t_j - t_i = 1\}$





是否存在包含字符 b 的接受时间串?

自动机及应用

- 时间自动机
- Büchi自动机
- 软件的形式化验证
- 模型检测方法
- M³C模型检测系统

Büchi 自动机

- Büchi 自动机为六元组：

  ***A*= (*S*, Σ, δ, *L*, *I*, *F*)**

  有限状态集 — S
  原子命题集 — Σ
  变迁关系 — δ
  标记映射 — L
  初始状态集 — I
  接受状态集 — F

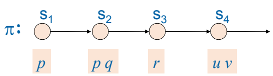
  $\Sigma = AP$
  $\delta \subseteq S \times S$
  $L: S \rightarrow 2^{AP}$
  $I \subseteq S$
  $F \subseteq S$

- 接受无限路径的自动机

Büchi 自动机

$L: S \rightarrow 2^{AP}$

$L(s_1) = \{p\}$,
$L(s_2) = \{p, q\}$,
$L(s_3) = \{r\}$,
$L(s_4) = \{u, v\}$,
……



Büchi 自动机

- 自动机 ***A*** 的状态链：

  $\pi = s_0 s_1 s_2 \cdots$..

  被接受当且仅当其中含有无限个接受状态。

- 自动机 ***A*** 的 ω-语言：

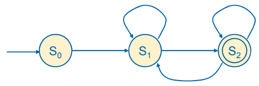
  $L(A) = \{ L(\pi) \mid \pi$ 是 A 接受的链 $\}$.

- ω-接受串

  $w \in \Sigma^{\omega}$ 被A接受，若 $w \in L(A)$.

Büchi 自动机

例：



$\sigma_1=S_0S_1S_2S_2S_2S_2\ldots$（**$S_2$** 保持自环） 接受

$\sigma_2=S_0S_1S_2S_1S_2S_1\ldots$（**$S_1$、$S_2$**保持交替） 接受

$\sigma_3=S_0S_1S_2S_1S_1S_1\ldots$（**$S_1$** 保持自环） 拒接

自动机及应用

- 时间自动机
- Büchi自动机
- 软件的形式化验证
- 模型检测方法
- M^3C模型检测系统

软件验证的必要性

软件应用广泛，可靠性备受关注

软件验证的必要性

- 软件充满Bugs

  在NASA高安全软件错误统计中：
  - 每100K代码中，可能存在0.4 错误；
  - 1Mb代码中，可能存在6.3个错误；
  - 每10Mb代码中，可能含100个以上的关键错误。

软件验证的必要性

- Bug代价高昂
  - Y2K bug：估计损失 >$5000亿
  - 2003年美国东北大停电估计损失: $60-$100亿
- Bug十分危险
  - 1996年6月阿里亚娜五号（Ariane V）火箭，由于控制软件的一个小小的逻辑错误导致发射失败，造成直接经济损失为五亿英镑。

安全攸关软件的危机

- 1996年6月阿里亚娜五号（Ariane V）火箭由于控制软件的一个小小的逻辑错误导致发射失败，造成直接经济损失为五亿英镑。

- 2011年日本福岛核电站事故既是天灾，
- 也是人祸，对周边地区乃至全世界造
- 成的灾难，其损失难以估算。

波音737 MAX

被疑酿成波音 737MAX 空难 飞控软件上天前要经历大考

2019 年 09 月 11 日 08:20 来源：科技日报

人民网 http://yn.people.com.cn/GB/n2/2019/0911/c378440-33343955.html

原标题：被疑酿成波音 737MAX 空难 飞控软件上天前要经历大考

被疑酿成波音 737MAX 空难

上天前，飞控软件要经历这些大考

短时间内波音 737MAX 复飞无望了。

近日，美国联邦航空管理局新局长斯蒂芬·迪克森在宣誓就职时表示，暂时没有 737MAX 复飞的计划，并重申安全是第一要务。美国航空公司也于最近宣布延长波音 737MAX 机型的停飞时间，停飞时间将延长至 12 月 3 日。

众所周知，去年 10 月和今年 3 月，印尼狮航和埃塞航空先后发生波音 737MAX 客机失事事件，该机型随即在全球范围内遭到停飞或禁飞。后调查发现，两起空难皆与客机的自动防失速软件(MCAS 系统，即机动特性增强系统，以下简称 MCAS)被错误激活有关。

随着自动化程度不断加强，越来越多类似 MCAS 这样的飞控软件"登"上飞机，那么它们是如何被研制出来的？又如何确保其绝对安全？

标准充当安全研发指挥棒

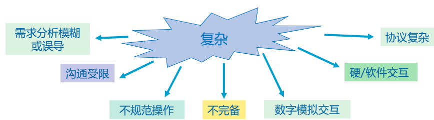
作为一种自动安全软件，MCAS 的设计原理其实并不复杂。简言之，它是通过对飞机迎角传感器信号的判断来驱动飞机控制系统。其可自动将飞机机头向下推，以防止升力损失，从而实现飞机自动安全保护。

存在设计缺陷？细数波音737客机的坠毁事故，已造成上千人丧生

https://www.163.com/dy/article/H37OKUT9051593PM.html

软件验证的必要性

为什么有如此多的错误？



- 软件复杂：复杂的算法、多组件交互、多编程API
- 软件增长快速，代码每3 - 4年翻一番。
- 由于复杂性的增加，错误越发细微。

软件验证的必要性

软件测试特点:

- 优点 易于实现，广泛使用
- 缺点 与用例相关，难以产生有效的测试序列；不完备
- Dijkstra 测试仅查找已知的bugs, 但不能查找未知的
- NIST 测试可以减少软件中三分之一的 bugs，但难以消去所有的错误

软件验证的必要性

Software Engineering and Theory

- A major goal of software engineering ????
  - is to enable developers to construct systems that operate reliably despite this complexity
- Tools?
  - Software Engineers need better tools to deal with this complexity!

软件验证的必要性

What Software Engineers Need Are

- Tools that give better confidence than testing:
  - fully automatic
  - (reasonably) easy to use
  - provide (measurable) guarantees
  - come with guidelines and methodologies to apply effectively
  - apply to real software systems

软件验证的必要性

- 随着计算机软硬件系统日益复杂，如何保证其正确性和可靠性成为日益紧迫的问题．对于并发系统，由于其内在的非确定性，这个问题难度更大．
- 在过去二十多年间，各国研究人员为解决这个问题付出了巨大的努力，取得了重要的进展．在为此提出的诸多理论和方法中，形式化方法和模型检测(**model checking**)以其简洁明了和自动化程度高而引人注目．

软件的形式化验证

- 随着计算机软硬件系统日益复杂：
  - 如何保证其正确性和可靠性成为日益紧迫的问题；
  - 并发系统，由于内在的非确定性，其难度更大．
- 在过去二十多年间，各国研究人员为解决这个问题付出了巨大的努力，取得了重要的进展．
  - 为此提出的诸多理论和方法中，形式化方法以其简洁明了和自动化程度高而引人注目.

软件的形式化验证

- 形式化方法
  - 是基于数学和逻辑的方法和工具，解释和验证对象软件系统.
- 形式化验证方法类型
  - 定理证明
  - 模型检测

形式化验证的优点

- 形式化方法以逻辑理论为基础，可以揭示系统如下问题：
  - 不一致，
  - 歧义
  - 不完备性
- 增强对系统的理解

自动机及应用

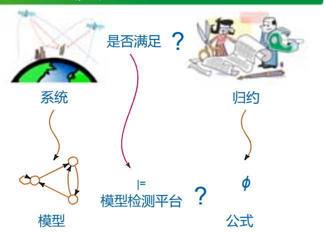
- 时间自动机
- Büchi自动机
- 软件的形式化验证
- 模型检测方法
- M^3C模型检测系统

模型检测方法

- 模型检测
  - 基本思想：用自动机 (S) 表示系统，用模态 / 时序逻辑公式 (F) 描述系统的性质．检测 “系统是否具有所期望的性质”
  - 转化为数学问题：“自动机 S 是否是公式 F 的一个模型?” 表示为检测 $S \models F$．
- 对有穷状态系统，这个问题是可判定的，即可以用计算机程序在有限时间内自动执行．

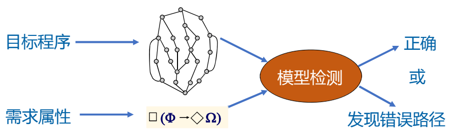
模型检测方法

- 自动检测
- 符合属性的完整检测
- 保证正确无误
- 无需得到期望输出。若检测出错误，则生成错误路径



模型检测方法

- 描述需求属性，并建立系统模型；
- 从系统产生自动机模型，检测模型是否满足需求属性



模型检测技术

- 归约语言
  - 描述系统的行为
  - 需求逻辑
    例： LTL, CTL, TCTL, PCTL, CSL
- 语义模型
  - Kripke 结构
  - Markov 链
- 模型检测：构建与Kripke 结构关联的状态图

模型检测发展

- 模型检测的研究始于八十年代初，由Clarke等人提出检测有穷状态系统是否满足给定时序公式的方法，并实现了一个检测的原型系统．
- 模型检测可应用于
  - 计算机硬件、通信协议、控制系统、安全协议认证等验证；
  - 从学术界辐射到了产业界：
    许多大公司，如Intel、HP、Phillips 等成立了专门的小组，准备将模型检测技术应用于生产过程中。

模型检测方法

E. M. Clarke

E. A. Emerson

J. Sifakis

Clarke，Emerson 和 Sifakis 因模型检测的创始性工作获得了2007年Turing Award.

模型检测发展

- 美国：
  - （1991至今）Bell实验室研发的SPIN检测工具；
  - （2001至今）CMU提出SMV及NuSMV检测工具；
  - （2009至今）微软研发MODIST系统，用于分布式系统。
- 英国：
  - （2009至今）牛津大学计算实验室提出PRISM系统。
- 欧盟：
  - （2000至今）法国Verimag实验室提出的BIP框架，应用于嵌入式系统；
  - （2000至今）瑞典和丹麦研发的UPPAAL，用于实时系统。

All MCs need…

- specification language – to describe behaviour of system
  - logic for requirements
  - e.g. LTL, CTL, TCTL, PCTL, CSL
- semantic model (usually a Kripke structure or Markov chain)
- MC builds a graph (state space) relating to the KS, which must be searched somehow. Need

Model expressed as a boolean formula and represented as BDD

模型检测工具

模型检测软件:

SPIN (Simple Promela Interpreter) Holzmann

SMV (Symbolic Model Verifier) McMillan

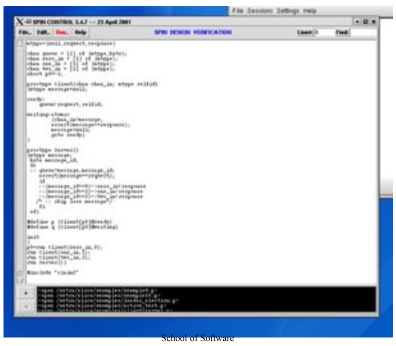
Uppaal (UPPsala and AALborg)

PRISM (Probabilistic Symbolic Model checker)

SPIN

- Specification language: Promela
  - Logic: LTL
  - Search strategy: DFS
  - Explicit state
- Processes communicate via synchronous/asynchronous channels.
- Asynchronous, interleaving semantics.
- Has been used to verify:

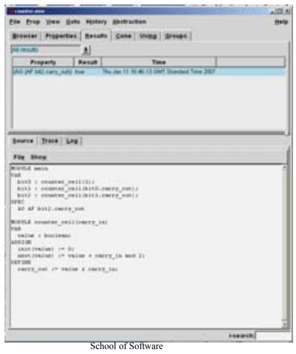
•Operating systems, railway signalling systems, feature interaction analysis, flight software, Mars Exploration Rovers (MER)



SMV

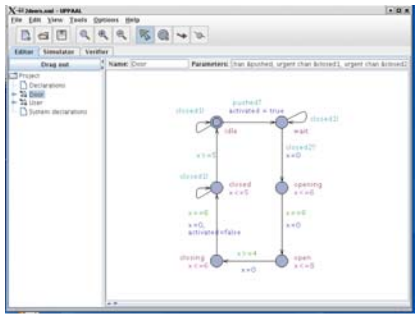
- Specification language: SMV language
  - Logic: CTL
  - Search strategy: BFS
  - Symbolic
- Communication via shared events. Synchronous behaviour (via MODULES)
- Asynchronous, interleaving behaviour (via PROCESSES)
- Preferred for hardware verification. Has been used to verify:
  Avionics triple sensor voter, Gigamax CCP, T9000 virtual channel processor

NuSMV: Cimatti et al, Added GUI and LTL model checking.



Uppaal – Larson et al.

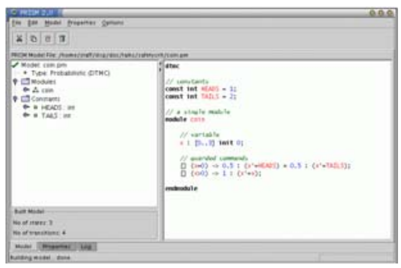
- Real time model checker
  - Specification language: Timed automata
  - Logic: Uppaal logic (subset of TCTL)
  - Search strategy: BFS of symbolic state space
  - Combination of explicit state/symbolic
- Finite control structure+ real valued clocks
- Synchronous communication via input/output actions
- Asynchronous communication via shared variables.
- Has been used to verify:
  •Audio/video protocol, QoS in personal area network, power controller, gearbox controller, TDMA protocol start up mechanism

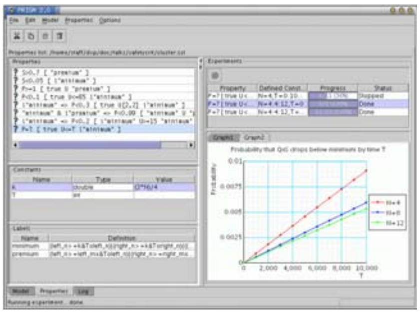


PRISM

- Specification language: PRISM language (based on reactive modules language, like SMV)
  - Logic: PCTL or CSL
  - Symbolic
  - Communication via shared events
- Supports 3 types of models
  - Discrete time Markov chains (DTMCs)
  - Continuous time Markov chains (CTMCs)
  - Markov decision processes (MDPs)
- Synchronous execution (apart from MDPs)
- Quantitative analysis using costs/rewards
- Can run experiments

  - Has been used to verify: signalling pathways in systems biology, PIN block
  - attacks, communications protocols (e.g. bluetooth, SMAC), aviation security
  - procedures

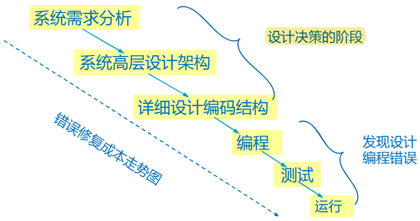




模型检测的优势

- 早期查找并更正错误
  - 模型检测开始在软件编码之前，与系统设计并行；
  - 错误发现越早修复错误成本更容易且更低。
- 理解系统需求
  - 属性是否为系统需求的精确表述？
  - 需求属性表示是否二义性？
  - 属性描述是否完备？
- 减少模型错误，增强模型的正确性
- 自动检测，测试周期缩短。

模型检测的优势



问题

- 状态空间爆炸问题
  - 模型检验方法难以解决无穷状态系统或状态数目超出现有计算机处理能力的复杂的有限状态系统.
- 白箱检测、形式化方法、软件安全保密问题
  - 模型检测是白箱检测；
  - 形式化方法基于软件逻辑理论，学习和掌握有一定难度；
  - 只有高安全软件才能做模型检测，但高安全软件保密性要求也很高。
- 建模、转换算法问题
  - 建模和转换算法复杂，本身算法存在一些不可靠性。

自动机及应用

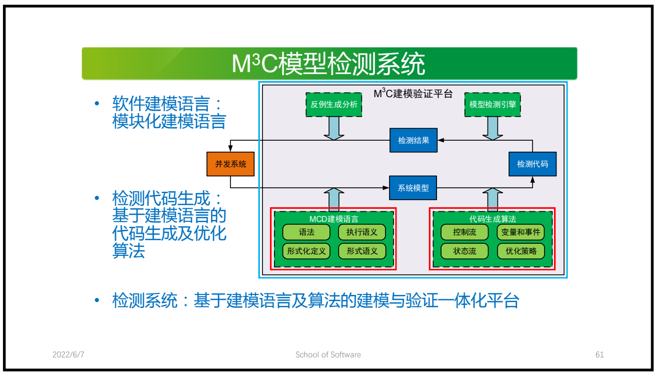
- 时间自动机
- Büchi自动机
- 软件的形式化验证
- 模型检测方法
- M^3C模型检测系统

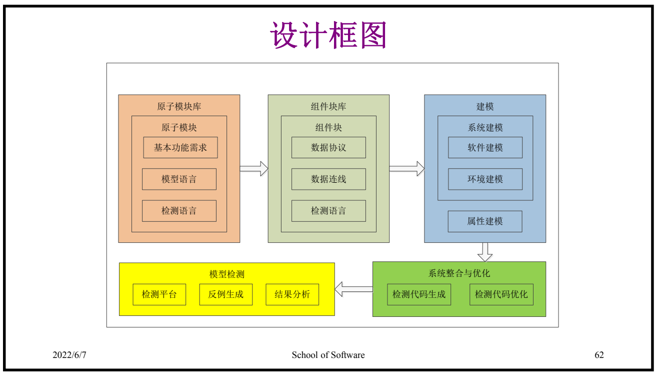
M^3C模型检测系统

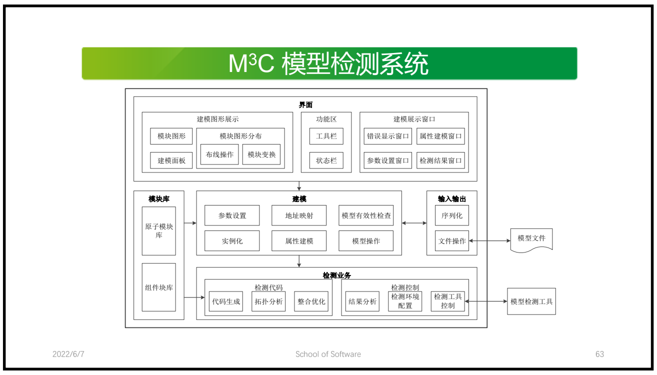
M^3C 开发有效的模型检测工具是模型检测所面临的任务和目标

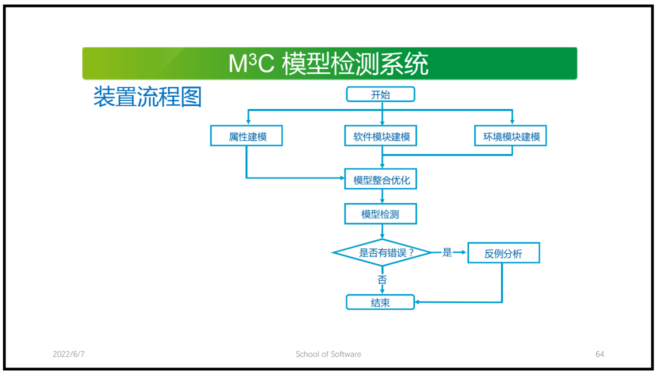
模型检测难以应用

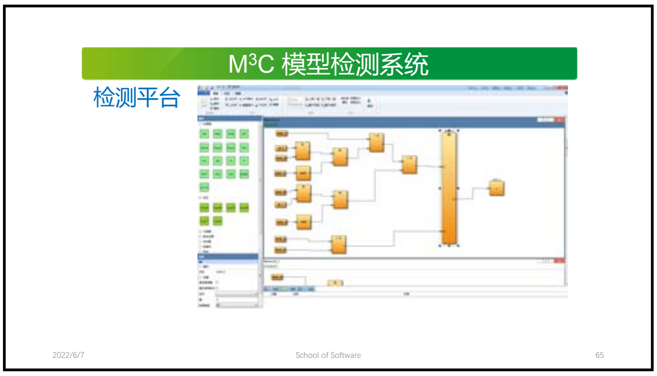
- 模型检测为白箱检测或灰箱检测。软件建模与检测将对软件及工程安全保密性造成危险
- 模型检测需要专门的形式化理论、时序逻辑、计算机知识。要求领域专家掌握有难度
- 软件系统和检测属性都是表示为自动机。状态爆炸是目前模型检测最大的问题

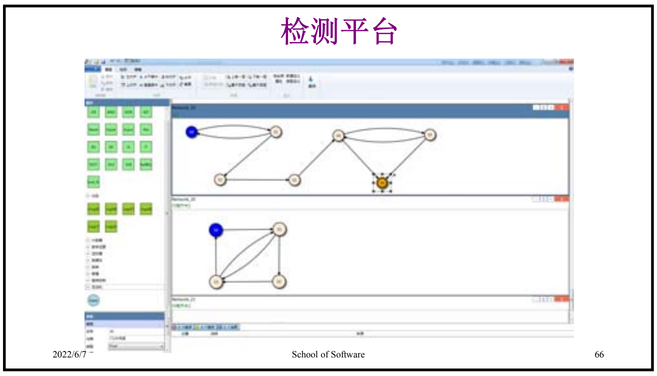

M³C模型检测系统
软件建模语言：模块化建模语言
检测代码生成：基于建模语言的代码生成及优化算法
检测系统：基于建模语言及算法的建模与验证一体化平台
M³C建模验证平台
反例生成分析
模型检测引擎
检测结果
并发系统
检测代码
系统模型
MCD建模语言
语法
执行语义
形式化定义
形式语义
代码生成算法
控制流
变量和事件
状态流
优化策略
2022/6/7
School of Software
61


设计框图
原子模块库
原子模块
基本功能需求
模型语言
检测语言
组件块库
组件块
数据协议
数据连线
检测语言
建模
系统建模
软件建模
环境建模
属性建模
模型检测
检测平台
反例生成
结果分析
系统整合与优化
检测代码生成
检测代码优化
2022/6/7
School of Software
62

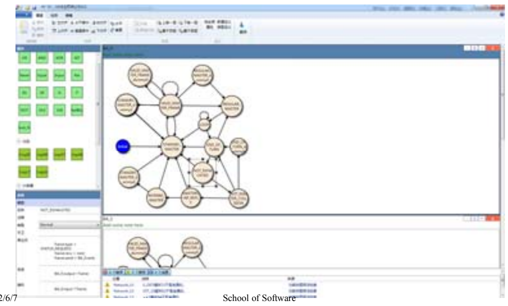

M³C 模型检测系统
界面
建模图形展示
功能区
建模展示窗口
模块库
建模
输入输出
检测业务
模型文件
模型检测工具
2022/6/7
School of Software
63


M³C 模型检测系统
装置流程图
开始
属性建模
软件模块建模
环境模块建模
模型整合优化
模型检测
是否有错误？
是
反例分析
否
结束
2022/6/7
School of Software
64


M³C 模型检测系统
检测平台
2022/6/7
School of Software
65


检测平台
2022/6/7
School of Software
66

检测平台

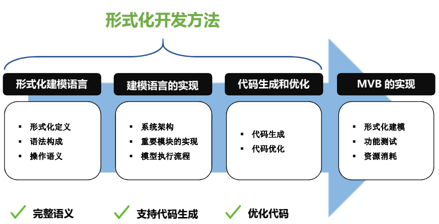


M^{3}C 模型检测系统

优点：

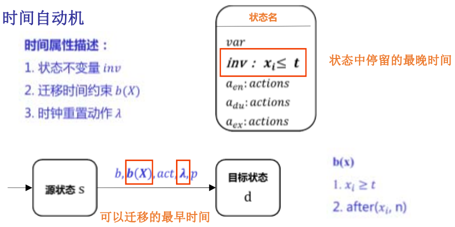
- 建模检测一体化，能分层检测，局部检测，边建模边检测。可减小状态爆炸问题；
- 模块建模，不需要形式化和自动机的专门知识，只需根据对象需求，通过模块搭建系统模型。可解决白箱检测和安全保密问题；
- 不需要编程与代码转换，检测一键即可。可解决软件形式化建模、代码转换出错问题。

嵌入式系统形式化设计



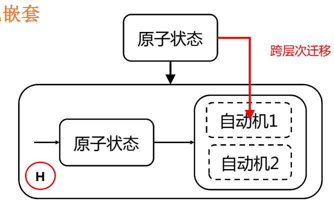
嵌入式系统形式化设计

时间自动机



嵌入式系统形式化设计

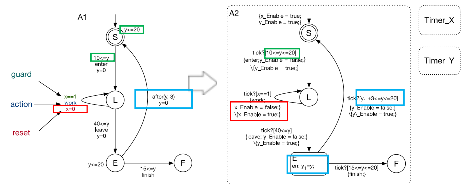
自动机嵌套

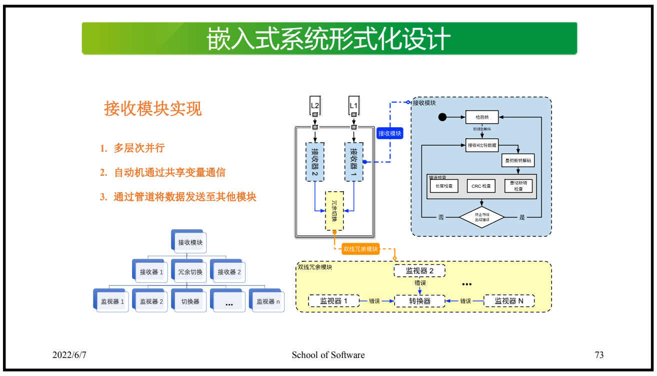


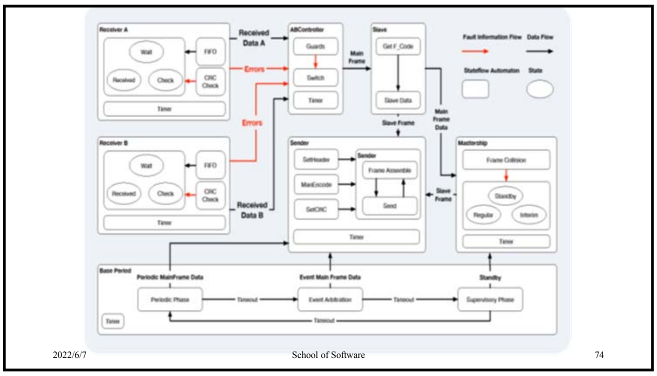
历史结点 $H: (H_{state}, H_{var})$

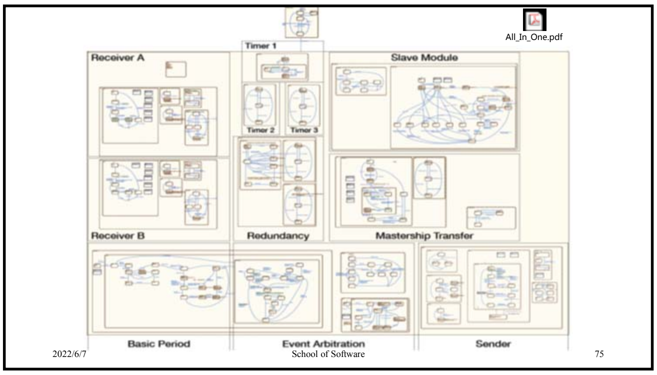
嵌入式系统形式化设计

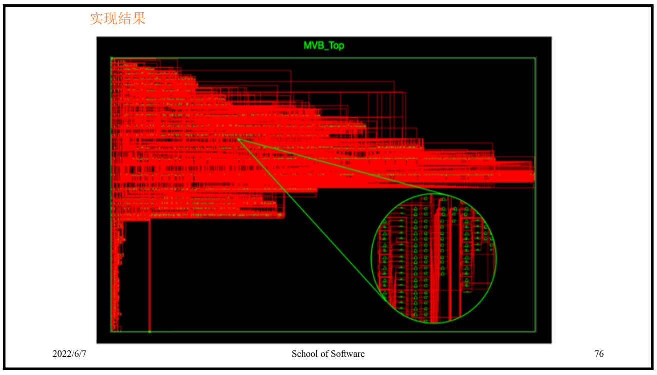
时间自动机的实现



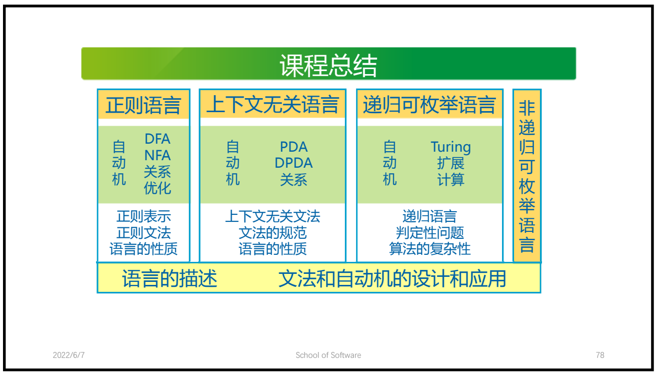

嵌入式系统形式化设计
接收模块实现
1. 多层次并行
2. 自动机通过共享变量通信
3. 通过管道将数据发送至其他模块
接收模块
接收器 1
冗余切换
接收器 2
监视器 1
监视器 2
切换器
...
监视器 n
双线冗余模块
监视器 2
监视器 1
转换器
监视器 N
错误
2022/6/7
School of Software
73


Receiver A
Received Data A
Errors
Received Data B
Receiver B
ABController
Guards
Switch
Timer
Slave
Get F_Code
Main Frame
Slave Data
Slave Frame
Main Frame Data
Sender
SetHeader
MainEncode
SetCRC
Frame Assemble
Send
Slave Frame
Mastership
Frame Collision
Standby
Regular
Interim
Fault Information Flow
Data Flow
Stateflow Automaton
State
Wait
Received
Check
FIFO
CRC Check
Base Period
Periodic MainFrame Data
Periodic Phase
Timeout
Event Main Frame Data
Event Arbitration
Supervisory Phase
2022/6/7
School of Software
74


All_In_One.pdf
Receiver A
Timer 1
Slave Module
Timer 2
Timer 3
Receiver B
Redundancy
Mastership Transfer
Basic Period
Event Arbitration
Sender
2022/6/7
School of Software
75


实现结果
MVB_Top
2022/6/7
School of Software
76


Formal Codesign and Implementation for Multifunction Vehicle Bus Circuits
08693971_1906.pdf
2022/6/7
77


课程总结
正则语言
自动机
DFA
NFA
关系
优化
正则表示
正则文法
语言的性质
上下文无关语言
自动机
PDA
DPDA
关系
上下文无关文法
文法的规范
语言的性质
递归可枚举语言
自动机
Turing
扩展
计算
递归语言
判定性问题
算法的复杂性
非递归可枚举语言
语言的描述
文法和自动机的设计和应用
2022/6/7
School of Software
78

Thank you