

Event-B Tutorial

赵永望

zhaoyw@buaa.edu.cn

北京航空航天大学 计算机学院

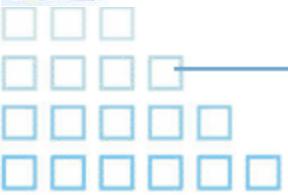
2013年12月

提 纲

- 1. 形式化方法(Formal Method)
 - 形式化方法
 - B方法
 - Event-B基本思想
- 2. 基本理论
 - 集合论(Set Theory)
 - 命题逻辑(Propositional Logic)与谓词逻辑(Predicate Logic)
 - 演绎推理(Deductive Reasoning)
- 3. Event-B详解
 - 抽象机(Machine)与上下文(Context)
 - 精化(Refinement)
 - 语义(Semantics)
 - 正确性证明(Proof)
- 4. Event-B实践
 - 开发、证明



1. 形式化方法

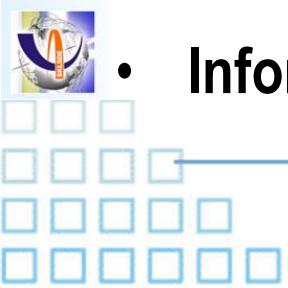


什么叫形式化方法 Formal Method

- Formal Method vs. Informal Method
- Formal
 - 形式主义? (NO)
 - 逻辑的、数学的、正规的 (YES)
- In computer science, specifically software engineering and hardware engineering, formal methods are a particular kind of mathematically based techniques for the specification, development and verification of software and hardware systems.

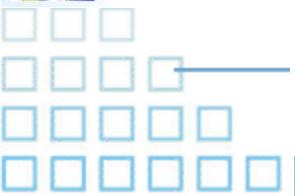
R. W. Butler (2001-08-06), “What is Formal Methods?”,
<http://shemesh.larc.nasa.gov/fm/fm-what.html>(NASA Langley Formal Methods)

- Informal method: 用word写需求, 用visio画设计图,



软件工程：问题和需要

- **最重要的问题：**
 - 软件的功能和质量的保证
 - 开发成本
 - 开发效率
- **有关计算机软件的研究都围绕着：**
 - 提高软件系统的质量和可靠性
 - 提高开发的效率
 - 降低开发的成本
 - 就是说：想让马儿快快壮，还不想让马儿多吃草



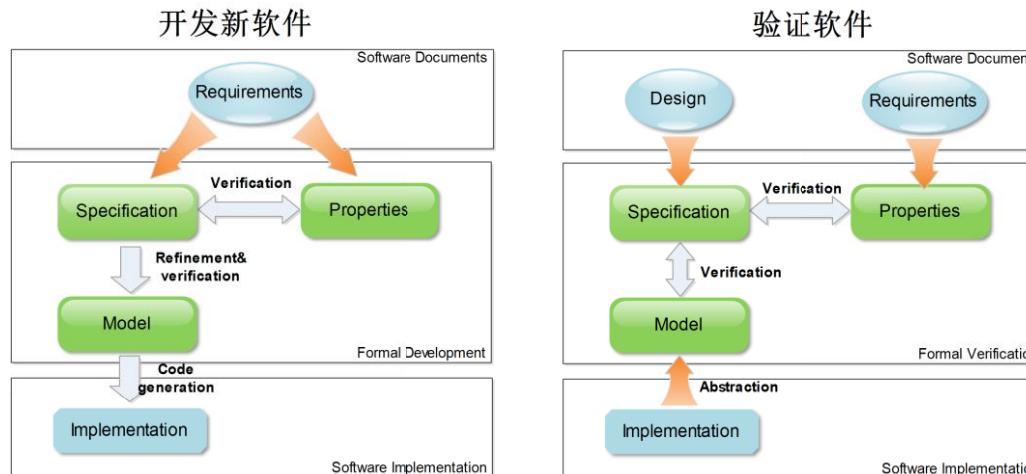
软件工程：实际情况

- 软件开发的实际情况并不乐观：
 - 项目经常延误，预算经常超支
 - 开发的后续阶段常发现许多前期设计错误，更正的代价高昂
 - 发布运行的软件中常常存在着许多错误，时常崩溃
 - 软件维护和更新工作的代价很高
- 一个情况：没有人愿意给软件做保险
 - 软件的一个小错误，可能导致整个系统的崩溃
 - 难以估计软件出问题的可能性和造成的损失
 - 机械工程：在一间**明亮**的房间里面寻找一只黑猫
 - 化学工程：在一间**黑暗**的房间里面寻找一只黑猫
 - 软件工程：在一间**黑暗**的房间里面寻找一只黑猫，而实际上房间里并没有猫
 - 系统工程：在一间**黑暗**的房间里面寻找一只黑猫，而实际上房间里并没有猫，但某人还大呼：我找到了
 - —— 摘自形式化方法的幽默故事：<http://shemesh.larc.nasa.gov/fm/fm-humor.html>



形式化方法

- 形式化方法不能确保系统的可靠性，但其可以通过揭示系统的不一致性、歧义性和不完备性来增加我们对系统的理解程度，从而提高我们对系统可靠性的可信度
- 形式化方法使用的方式(针对软件)



- 形式化方法使用的3个层面
 - Level 0: formal specification
 - Level 1: formal development, formal verification: proofs of properties or refinement from the specification to a program
 - Level 2: theorem prover: fully formal machine-checked proofs
- 形式化方法包括：
 - 规约 (specifying)
 - 建模 (Modeling)
 - Abstraction、Refinement
 - 验证 (verification)
 - 模型检测(model checking)、演绎推理/定理证明(Deductive reasoning, theorem proving)

提高软件质量的途径

- 分析：程序和产品的分析
 - 静态分析：通过对程序代码进行扫描分析而获得程序相关性质的技术
 - 动态分析：通过对程序的执行来获得程序的相关性质，一般与程序切片、调试和测试技术相结合
- 测试
 - 以运行系统为主要手段发现系统错误
 - 只能发现系统的错误，不能证明系统没有错，无法回答系统一定没有错误这样一类问题
 - 工业界常用的途径，成本高，代价大
 - 仿真（Simulation）：以运行系统模型为主要手段发现系统错误
- 验证
 - 建立系统模型，确认系统模型是否存在错误
 - 可以从某个角度回答系统一定没有错误这样一类问题，从而进一步提高我们对系统可靠性的可信度

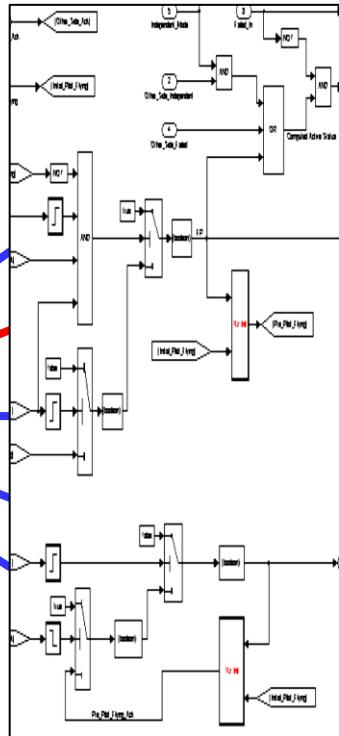


软件测试 vs 形式验证

测试只检测给定的情况

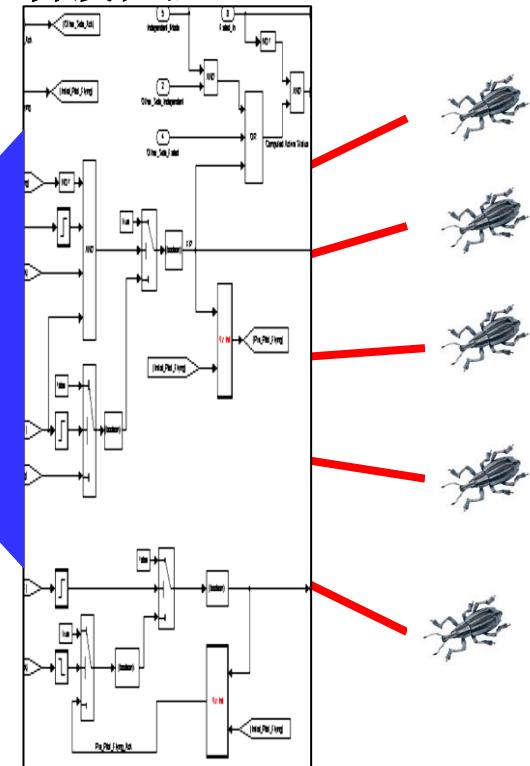
即使小规模系统也可能存在上万种待测情况

优势：简单、直观、工业级工具较多

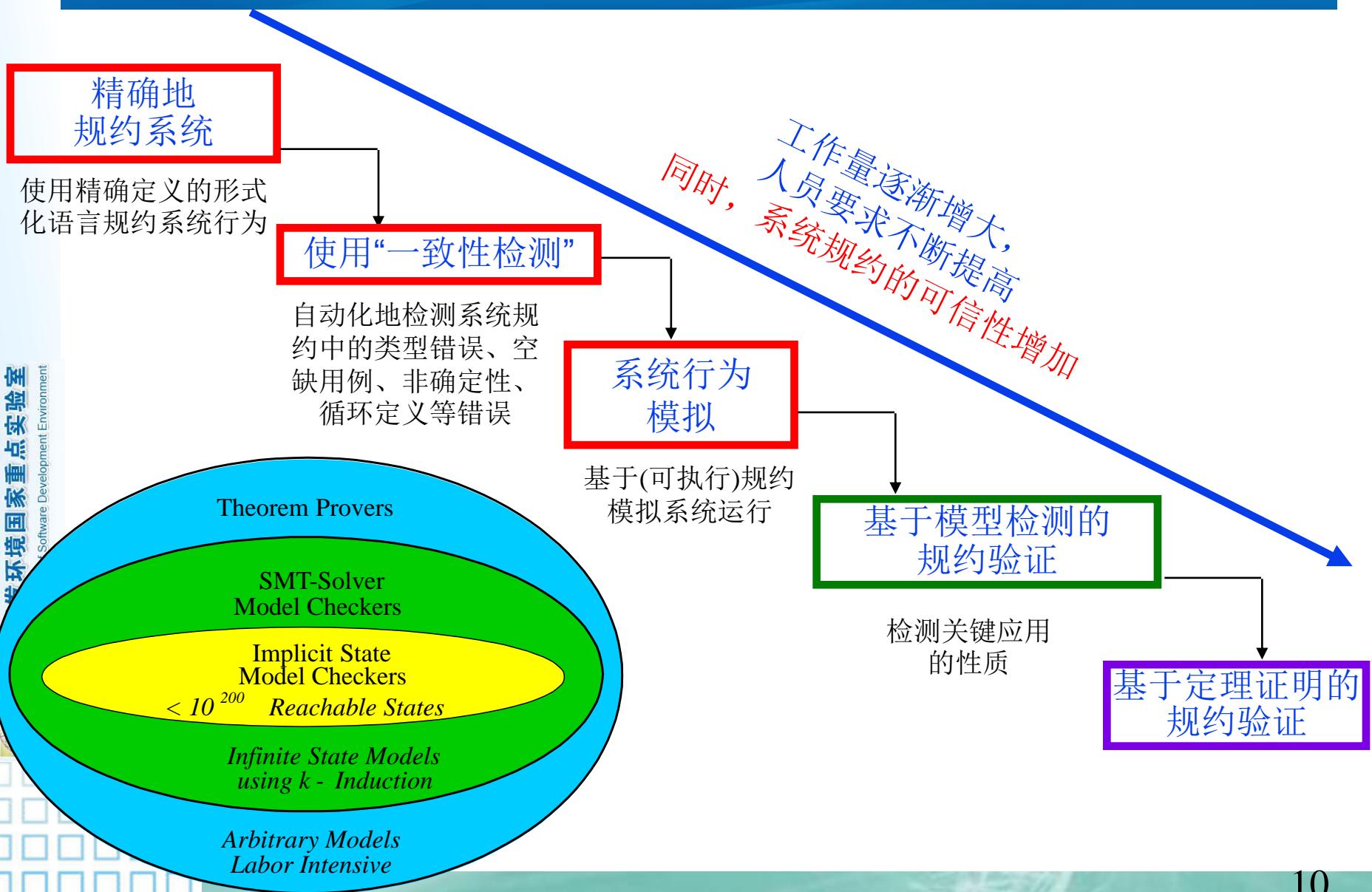


形式验证(模型检测)自动检测所有的情况

优势：可有效发现测试难以发现的错误、并发性验证



选择何种形式化方法?

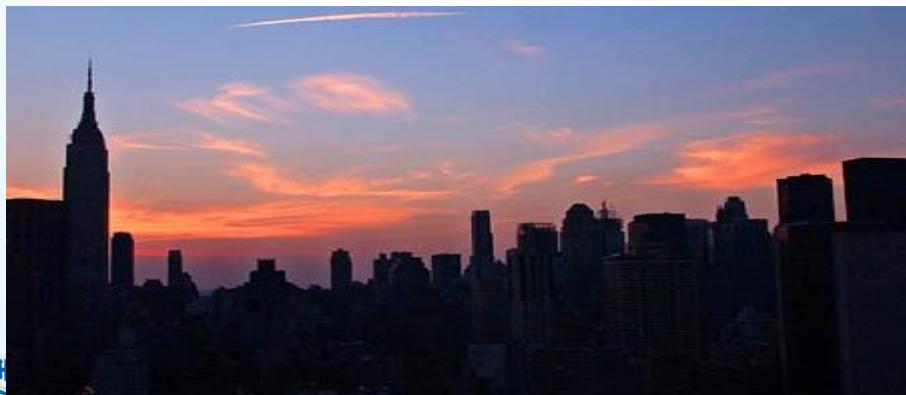


欧洲阿丽亚娜5型火箭



- On June 4, 1996, an unmanned Ariane 5 rocket launched by the European Space Agency exploded just 40 seconds after its lift-off.
 - A conversion from 64-bit floating point to 16 bit integer with a value larger than possible with Ariane 4. The overflow caused a hardware trap.
- Value of rocket and cargo: \$500 million

美国电力监测与控制管理系统 Blackout



August 14, 2003

A programming error has been identified as the cause of the Northeast power blackout. The failure occurred when **multiple computer systems** trying to access **the same information at once** got the equivalent of busy signals.

[Associated Press]

Price tag: \$ 6-10 billion

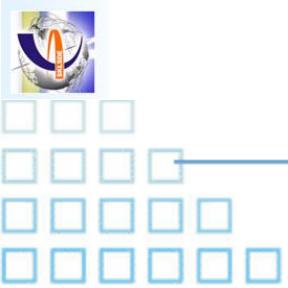
I S A T G e o S t a r 4 5
2 3 : 1 5 E S T 1 4 A u g . 2 0 0 3

- In August, 2003, the largest blackout in our country's history occurred.
 - The 2003 North America blackout was triggered by a local outage that went undetected.
 - A race condition in General Electric's monitoring software prevented an alarm.
- Estimated cost to New York City alone: \$1.1 billion.
- 多计算机系统试图同时访问同一资源引起的软件失效



美国F-22猛禽战斗机

- 2004年12月20日，美空军第422测试评估大队的一架F-22战斗机因软件问题在起飞过程中失控坠毁
- 2007年2月9日同样因软件问题延迟在日本部署



安全关键系统必须采用形式化方法

- 航空无线电技术委员会RTCA的DO-178B《机载系统和设备认证中的软件要求》标准用于规范机载系统和设备的软件开发过程，并指导软件认证到相应的安全级别。于2012年颁布DO-178C，增加了形式化方法的附件，强调采用形式化方法开展基于模型的开发和验证
- 美国国家宇航局NASA于1995年和1997年先后发布《Formal Methods Specification and Verification Guidebook》强调采用验证和形式化方法来保证系统可靠性
- 英国国防部在90年代发行了两种与软件开发生命周期中使用形式方法有关的安全至上标准Def Stan00-55和Def Stan 00-56标准，要求在安全至上的软件开发中必须采用形式方法



机载软件的安全性要求

- DO-178B/C对机载软件的安全性要求
- DO-178C 增加了DO-333 “Formal method supplement”

安全需求	严重性说明	接受频率 (发生的风险)	系统开发安全等级
高 	灾难	Extremely Improbable ($X < 10^{-9}$)	Level A
	危险	Extremely remote ($10^{-9} < X < 10^{-7}$)	Level B
	重要	remote ($10^{-7} < X < 10^{-5}$)	Level C
	次要	Probable ($X > 10^{-5}$)	Level D
	无	All	Level E

($X < 10^{-7}$ 表示100架客机按平均每天飞行8小时，33年内不发生错误)



DO-178B A级认证操作系统

- WindRiver VxWorks 653
- GreenHill INTEGRITY-178B
- LynxWorks LynxOS-178
- SYSGO PikeOS
- DDC-I Deos

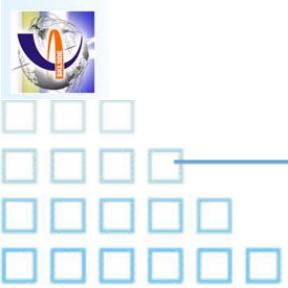
WIND RIVER



Open. Reliable. Safe. Secure.



Safety Critical Software Solutions
for Mission Critical Systems



安全关键系统必须采用形式化方法

- CC对安全关键软件的要求

国际信息技术
安全评估通用标准
Common Criteria
ISO/IEC 15408

EAL1—功能测试
EAL2—结构测试
EAL3—系统地测试和检查

EAL4—系统地设计、测试和复查
EAL5—部分形式化设计和测试
EAL6—部分形式化验证的设计和测试
EAL7—形式化验证的设计和测试

CC级别	需 求	功能规约	高层设计	低层设计	代码实现
EAL1	非形式化	非形式化	非形式化	非形式化	非形式化
EAL2	非形式化	非形式化	非形式化	非形式化	非形式化
EAL3	非形式化	非形式化	非形式化	非形式化	非形式化
EAL4	非形式化	非形式化	非形式化	非形式化	非形式化
EAL5	形式化	部分形式化	部分形式化	非形式化	非形式化
EAL6	形式化	部分形式化	部分形式化	部分形式化	非形式化
EAL7	形式化	形式化	形式化	部分形式化	非形式化
完全验证	形式化	形式化	形式化	形式化	形式化



CC认证的操作系统

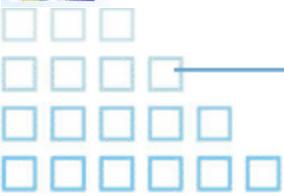
通过认证的操作系统	认证等级	认证时间
VMware® ESXi Server 3.5 and VirtualCenter 2.5	EAL4+	2010年2月
Microsoft Windows Mobile 6.5	EAL4+	2010年2月
Apple Mac OS X 10.6	EAL3+	2010年1月
Red Hat Enterprise Linux Ver. 5.3 on Dell 11G Family Servers	EAL4+	2009年12月
Windows Vista Enterprise; Windows Server 2008 Standard Edition; Windows Server 2008 Enterprise Edition; Windows Server 2008 Datacenter Edition	EAL4+ ALC_FLR.3	2009年8月
Oracle Enterprise Linux Version 5 Update 1	EAL4+ ALC_FLR.3	2008年10月
Green Hills Software INTEGRITY-178B Separation Kernel, comprising: INTEGRITY-178B Real Time Operating System (RTOS),	EAL6+	2008年9月

形式验证的效果

- 提高可靠性、降低安全认证成本
 - “Crash-Proof Code”
 - one of the 10 breakthrough technologies in 2011 by “MIT Technology Review”



- 澳大利亚国家信息通信技术研究中心
- 对seL4内核进行形式验证
 - seL4内核：构造OS的基本机制，包括线程、消息传递、中断、虚拟内存、访问控制等
 - 内核包含8700行C代码（其中1200行为内核启动代码），600多行ARM汇编代码
- 20万行手写Isabelle/HOL证明代码，形式验证工作量25人年
- 共发现460多个Bug(C代码中160多个，设计中150个，规约中150个)，而在测试中只发现16个Bug
- 完全形式化验证成本：600万美元
 - 走CC EAL6级认证，10000美元/行代码成本计算，需8700万美元
 - 而且CC EAL6认证比完全形式化验证的软件可靠性确保度低

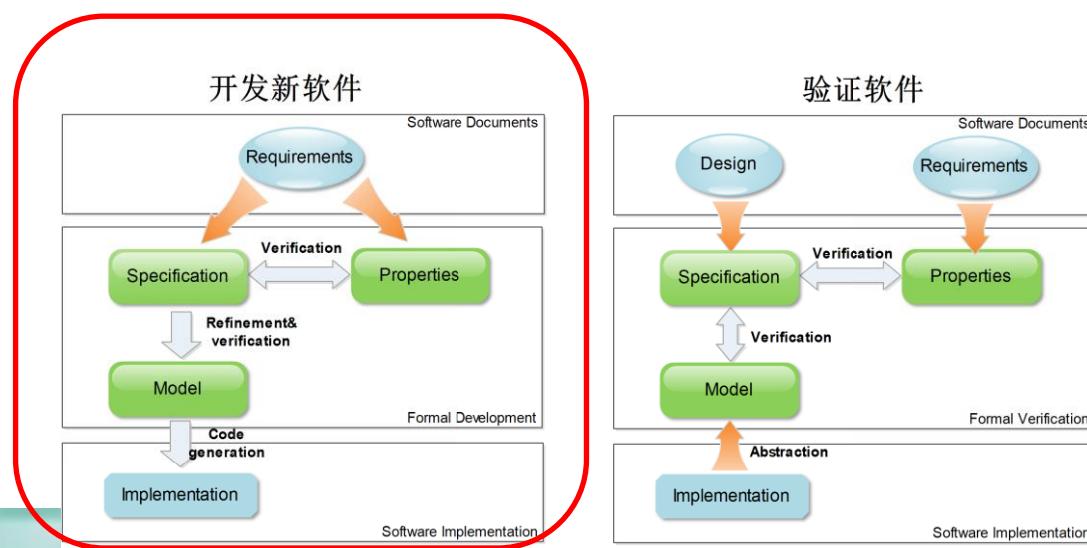


B方法

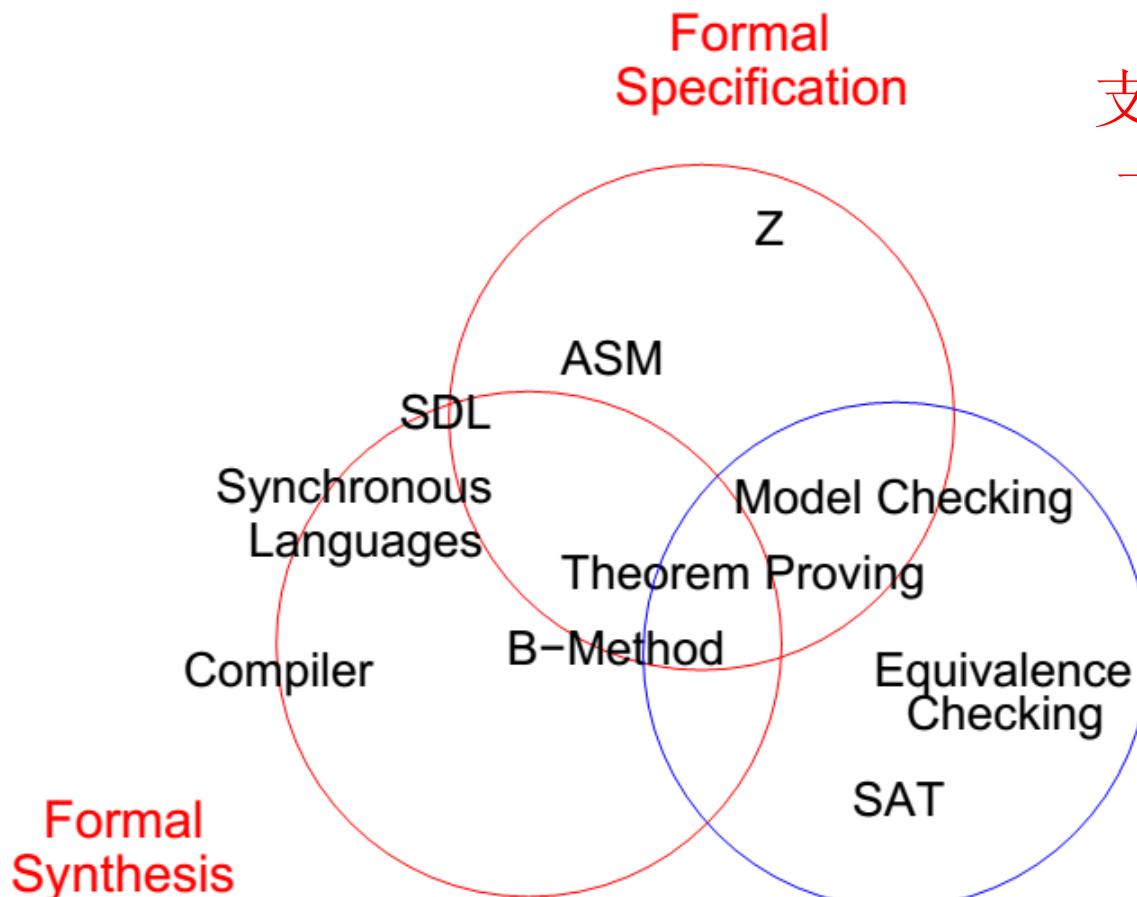
- 由Jean-Raymond Abrial提出
 - 1980s, 提出Z语言
 - 1990s, 提出B语言
 - 太复杂
 - 2000s, 提出Event-B
 - 替代经典B
- 基于集合论、基本算术、一阶逻辑



B方法
属于这一类



B方法



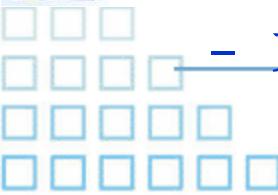
支持Refinement，而一般的定理证明器中没有直接的Refinement概念

**Formal
Verification**



B方法

- B方法是在开发巴黎地铁信号系统的过程中发展起来的，后来又用于开发巴黎的若干地铁/机场自动车的驾驶系统
- J-R Abrial 在2006 年国际软件工程大会上的特邀报告“工业开发中的形式化方法，成就、问题和未来”中介绍了B方法在工业界的广泛应用
- 特征
 - B 语言和B 方法基于一阶逻辑和集合论，有严格的数学理论基础，写出的规范简明、精确，无歧义性
 - 面向软件(classical B)、面向系统(event-B)
 - 支持事件机制：可用于表达分区内核的中断、健康监控等
 - 支持Refinement
 - 自动证明程度高，明确的上下层可追溯性
 - 一般的定理证明器，不直接支持Refinement
 - 擅长描述规范
 - 支持代码生成



B方法

- B方法有成熟的工业应用
- 西门子开发巴黎地铁14号线自动驾驶系统
 - 107,000 lines of B; 29,000 proofs;
 - 1998投入运行
- ClearSy公司开发巴黎Roissy机场自动穿梭车
 - 28,000+155,000 lines of B; 43,000 proofs;
 - 2006年投入运行
- EADS公司开发阿丽亚娜火箭的任务调度模型
- 正在采用类似的方式进行开发的系统：纽约城市地铁，巴塞罗那地铁，布拉格地铁，巴黎地铁1号线等



B方法开发地铁的历史

KVB
6000 trains
France



METEOR L14
Paris



Delhi



San Juan

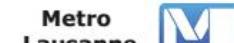


2000



Beijing

Metro L10
Beijing



Metro
Lausanne

2010



L1 L2
Malaga



L5
Milano



NY
Flushing



SHUTTLE
ROISSY AIRPORT
Paris

L1
Algiers



L2 L3
Sao Paulo



NY
Flushing

L3
Paris



Lyon



Instanbul

L9
Seoul



L9
Barcelona



Toronto

L2
Budapest



Beijing Metro Line 16



Circle Line
Singapore



Instanbul



Delhi



NY
Canarsie



Algeria



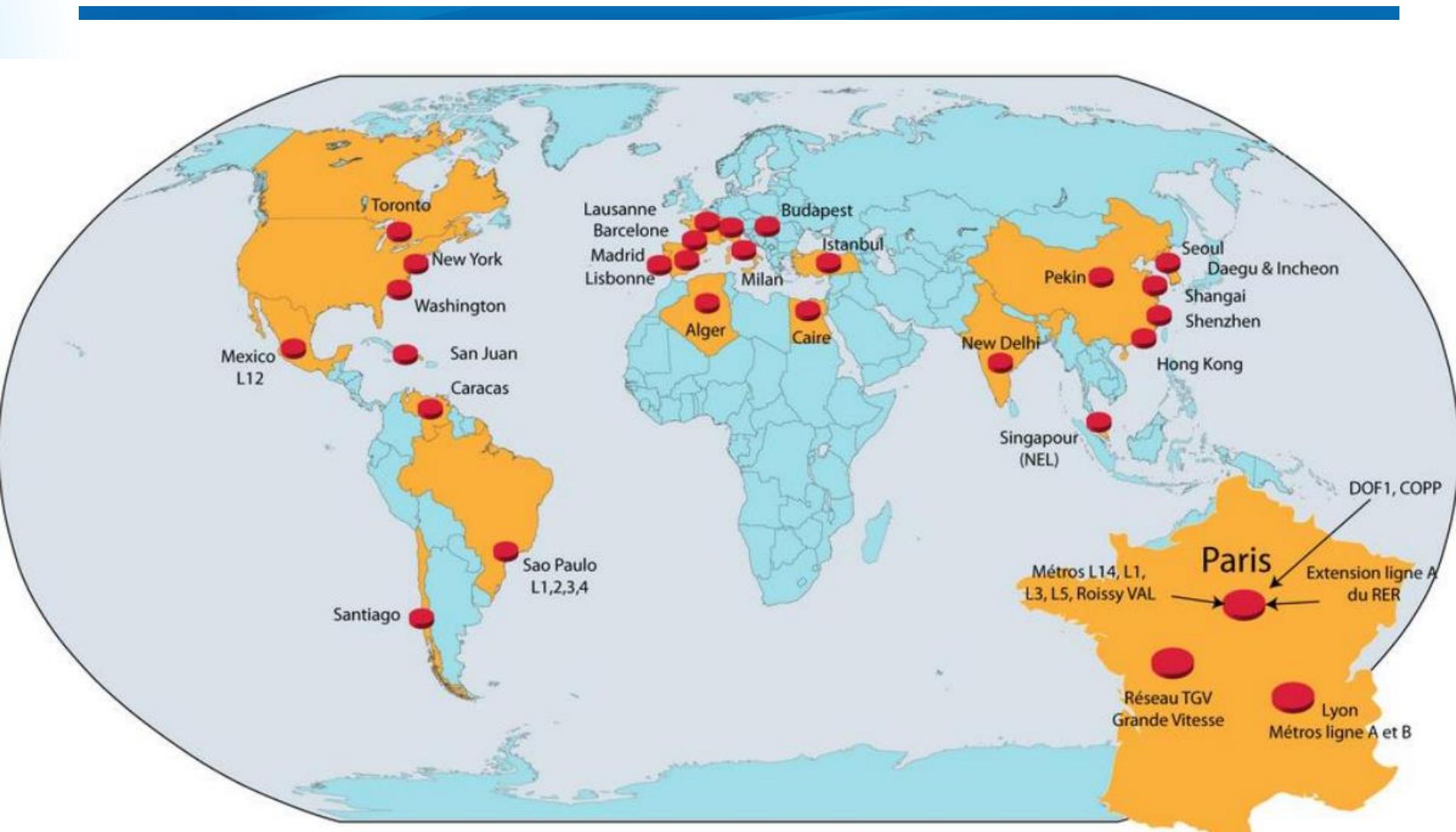
Sao Paulo



NY
Flushing

2010

全球采用B方法开发的地铁项目



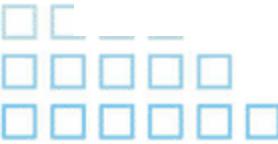
B方法应用状况

- Space
 - Space Systems Finland
 - Bepi Colombo (ESAs first mission to Mercury)
 - On-Board Satellite Software: Attitude and Orbit Control System (AOCS)
 - SSF
- Transportation
 - Siemens, Systerel: Communication-based Train Control (CBTC)
 - Battelle and ClearSy: automation of the Flushing and Culver metro lines in New York
 - Alstom: Dynamic Trusted Railway Interlocking
- Automotive
 - Bosch: Cruise-control system, Engine start-stop system
- 其他
 - SAP: Service choreography modelling
 - Critical Software Technologies: Integrated Secondary Flight Display, used on-board commercial and military aircraft; Smart Energy Grids
 - XMOS: XCore Microprocessor 指令集验证
 - The CCF Display and Information System, air traffic controllers based at the London Terminal Control Centre
 - STMicroelectronics: modelling and generation of a VHDL code for a smartcard-based microcircuit
 - Dependable Systems Forum (DSF) project in Japan (NTT-Data, Fujitsu, Hitachi, NEC, Toshiba, and SCSK)

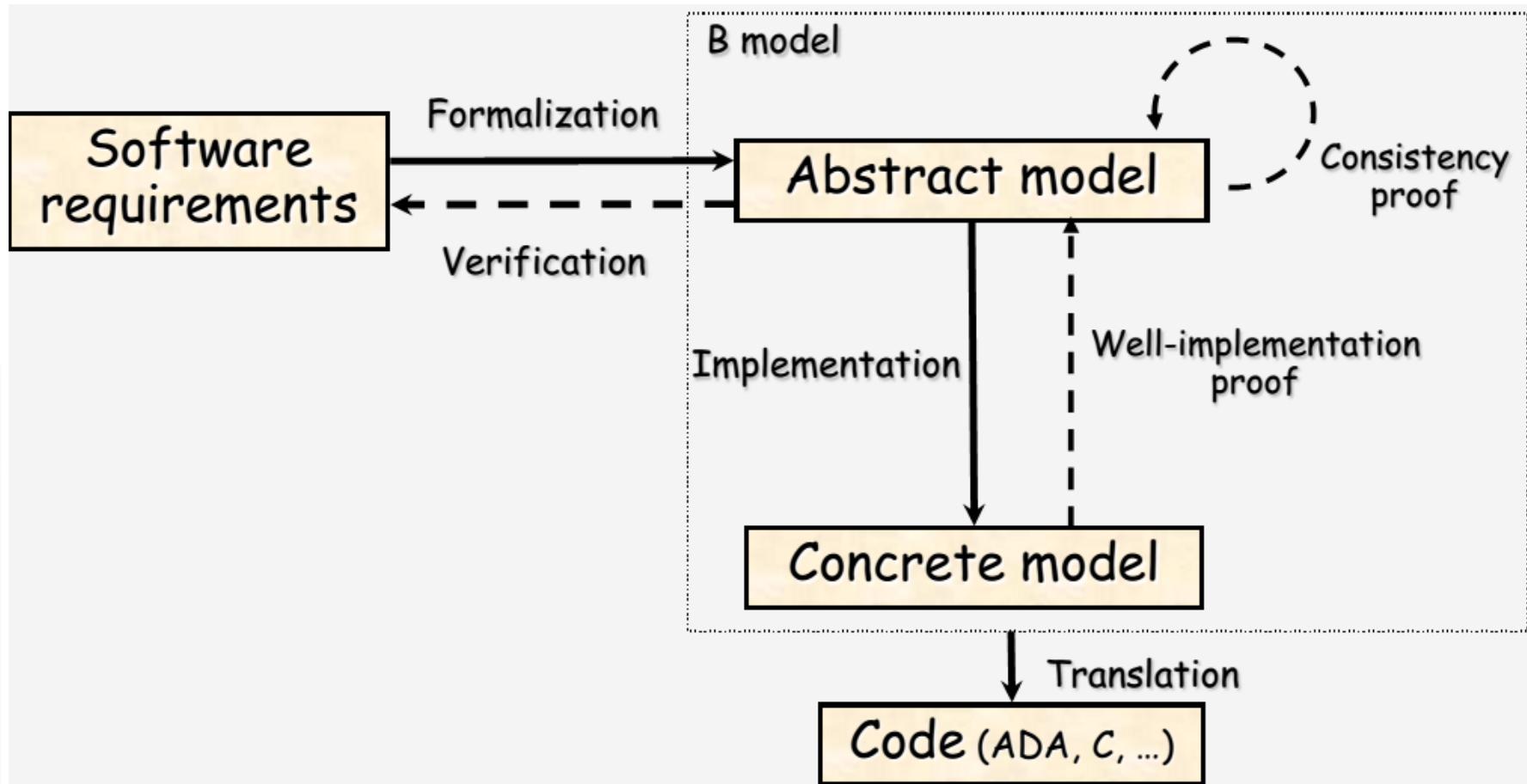


B方法应用状况

- Automotive:
 - Diagnosis (Peugeot)
 - Contactless keycard (Renault)
- Banking:
 - Reconciliation (Société Générale)
- Space:
 - Ariane 5 flight software (EADS)
- Microelectronics
 - Smartcard (STMicroelectronics)
- Nuclear
 - Control System Design (EDF)
- Industry
 - Pneumatic Press (CNAM)



用B方法开发软件系统的过程



用B方法开发软件系统的过程

- 第一阶段：根据软件需求文档构造系统的抽象模型
 - 基于需求文档，构造初步的抽象模型
 - 向抽象模型中加入必要的不变式，基于工具提取并证明不变式定理（证明义务）
 - 一些定理可以自动证明，有些需要人工交互证明
 - 扩充已有模型，加入更多细节，证明不变式定理
 - 不断重复上面几步，直至得到完整的抽象模型并证明所有定理可能发现需求文档中不完全/不一致，需要交流/调整文档和模型
- 手工/自动任务
 - 建立和扩充模型，加入不变式等完全是人工的工作
 - 不变式定理（证明义务）的抽取将自动完成
 - 工具可以自动证明许多定理，会有一些定理需要交互式证明
- 工作结果：待开发软件系统的一个完整的抽象模型



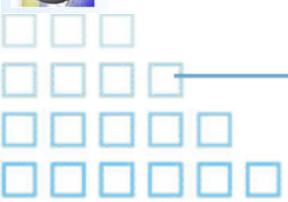
用B方法开发软件系统的过程

- **第二阶段：开发软件系统的具体模型**

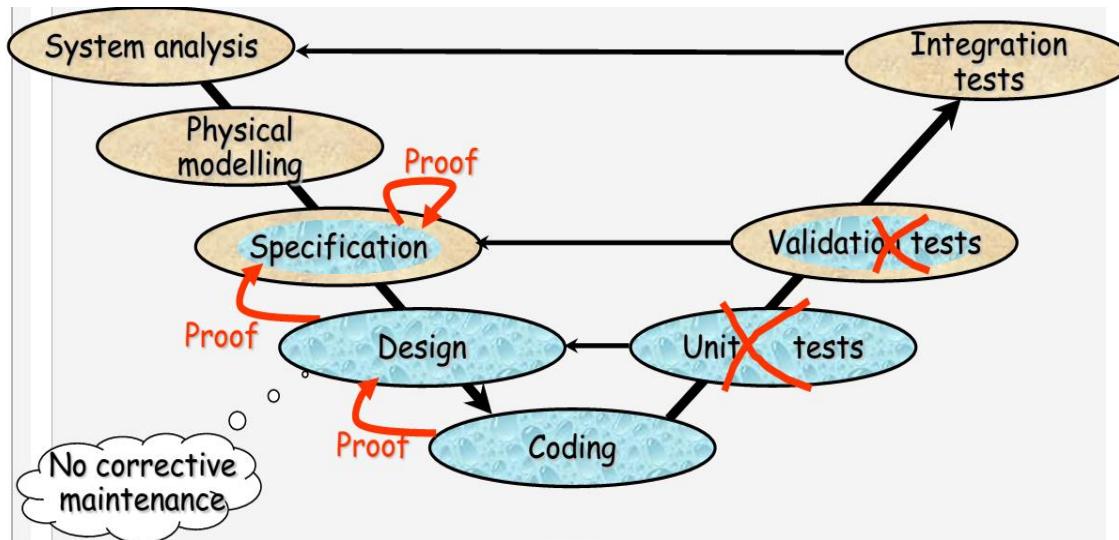
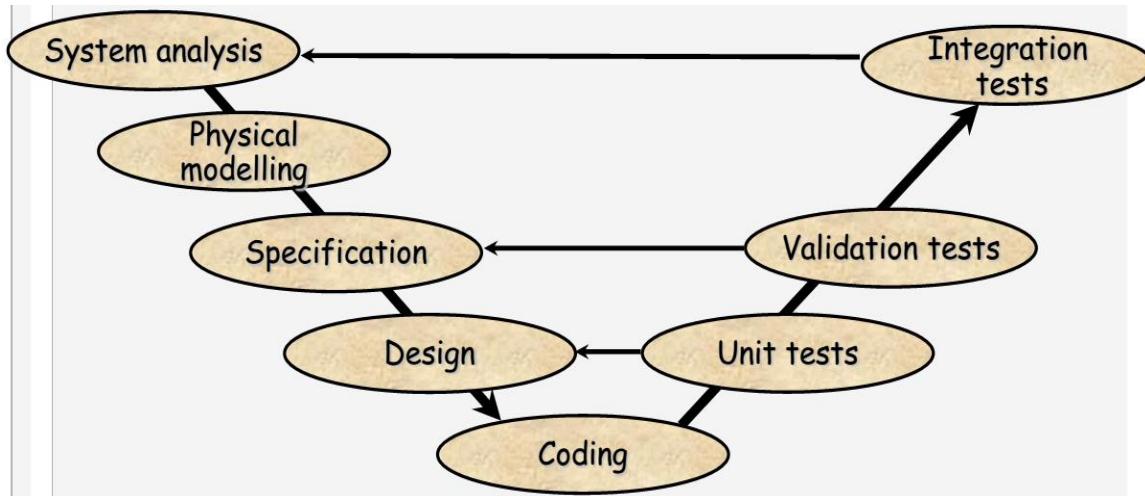
- 在已有模型中加入更多细节，使更多的操作具体化，用更接近计算机实现的数据结构代替抽象结构，得到一个更为“具体”的模式
- 证明新的具体模型是原有模型的“精化”
 - 使用工具自动生成一组“精化关系定理”，并证明这些定理
- 不断重复上面两步，最终做出一个可以直接对应到实现的模型
- 精化工作主要由人做，人们也开发了做精化的辅助工具
- 精化定理由工具自动生成，定理证明自动或交互式完成

- **第三阶段：生成系统的可执行代码**

- 用工具检查模型是否已能翻译到编程语言代码
- 用工具把最终模型翻译到编程语言，用编译器生成可执行程序
- 这一阶段完全是自动的

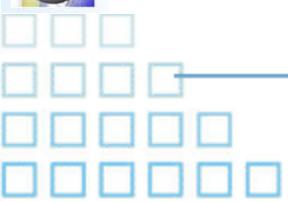


与传统开发过程的比较



B方法支撑工具

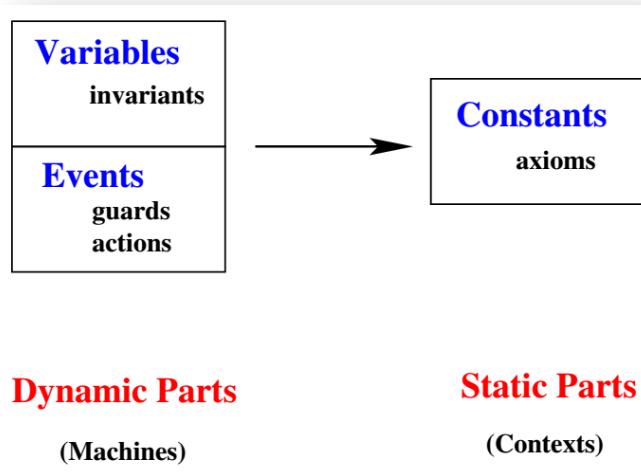
- **Atelier B (ClearSy)**
 - created to develop industrial B-Software projects
 - a set of tools integrated into a project manager tool
 - static checkers
 - automatic proof obligation generator
 - automatic provers and interactive prover
 - code translators: Ada, C, ...
 - it is also used for B-System
- **Rodin platform (September 2007)**
 - a new open platform dedicated to B-System (in construction)
- **B4free (www.b4free.com)**
 - free but restricted to academic users and owners of Atelier B
 - the core tools of Atelier B + a new x-emacs interface



Event-B基本思想

- Event-B模型

- 离散迁移系统：离散状态及迁移(事件触发)
- 状态：带公理(axiom)的常量(constant)、带不变式(invariant)的变量(variable)
- 事件：包括卫式(guard)和动作(action)，卫式表明事件触发条件，动作表明状态被修改的方式
- 常量、变量、卫式和动作均采用集合论表达式来书写



Context、Machine和Event

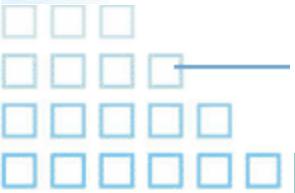
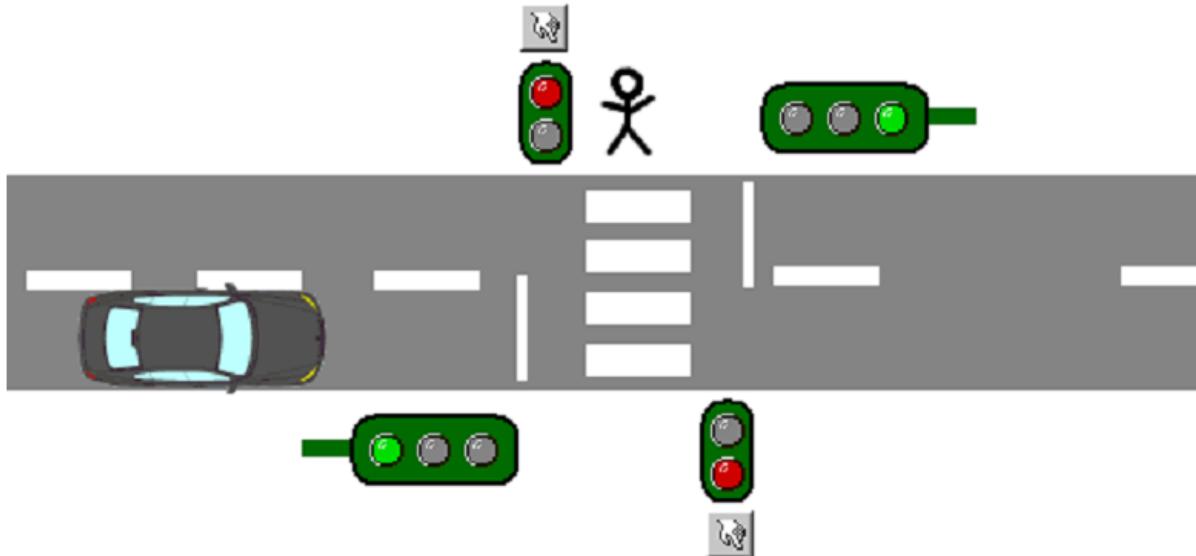
```
context
  < context_identifier >
extends *
  < context_identifier >
...
sets *
  < set_identifier >
...
constants *
  < constant_identifier >
...
axioms *
  < label >: < predicate >
...
theorems *
  < label >: < predicate >
...
end
```

```
machine
  < machine_identifier >
refines *
  < machine_identifier >
sees *
  < context_identifier >
...
variables
  < variable_identifier >
...
invariants
  < label >: < predicate >
...
theorems *
  < label >: < predicate >
...
events
  < event >
...
variant *
  < variant >
end
```

```
< event_identifier > ≡
status
  {normal, convergent, anticipated}
refines *
  < event_identifier >
...
any *
  < parameter_identifier >
...
where *
  < label >: < predicate >
...
with *
  < label >: < witness >
...
then *
  < label >: < action >
...
end
```

一个简单的例子(1)

- 交通信号灯控制器



一个简单的例子(1)

MACHINE mac

VARIABLES

cars_go
peds_go

INVARIANTS

inv1 : cars_go ∈ BOOL
inv2 : peds_go ∈ BOOL
inv3 : $\neg(cars_go = \text{TRUE} \wedge peds_go = \text{TRUE})$

EVENTS

Initialisation

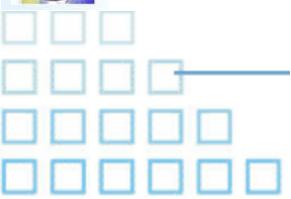
begin
 act1 : cars_go := FALSE
 act2 : peds_go := FALSE

end

Event set_peds_go ≡

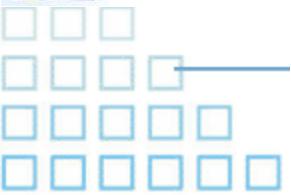
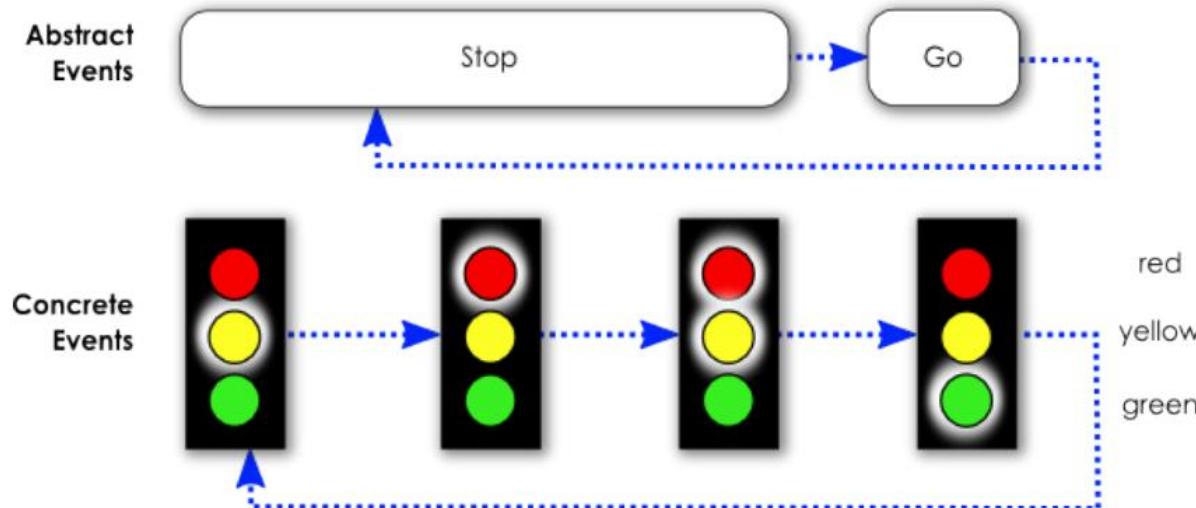
when
 grd1 : cars_go = FALSE
then
 act1 : peds_go := TRUE
end

Event set_peds_stop ≡
begin
 act1 : peds_go := FALSE
end
Event set_cars ≡
any
 new_value
where
 grd1 : new_value ∈ BOOL
 grd2 : new_value = TRUE ⇒ peds_go = FALSE
then
 act1 : cars_go := new_value
end
END



一个简单的例子(2)

- 交通信号灯控制器——求精



一个简单的例子(2)

- 交通信号灯控制器——求精
 - 引入Context

CONTEXT agatha

CONSTANTS

red

yellow

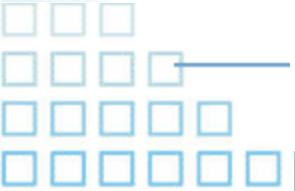
green

SETS

COLOURS

AXIOMS

type : *partition(COLOURS, {red}, {yellow}, {green})*

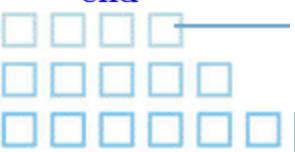


一个简单的例子(2)

• 交通信号灯控制器——求精

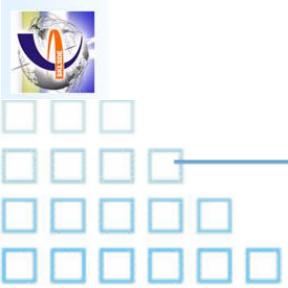
```
MACHINE mac1
REFINES mac
SEES ctx1
VARIABLES
    peds_colour
    cars_colours
INVARIANTS
    inv4 : peds_colour ∈ {red, green}
    inv5 : cars_colours ⊆ COLOURS
    gluing_peds : peds_go = TRUE ⇔ peds_colour = green
    gluing_cars : cars_go = TRUE ⇔ green ∈ cars_colours
EVENTS
Initialisation
begin
    init4 : peds_colour := red
    init5 : cars_colours := {red}
end
Event set_peds_green ≡
refines set_peds_go
when
    grd1 : green ∉ cars_colours
then
    act2 : peds_colour := green
end
```

```
Event set_peds_red ≡
refines set_peds_stop
begin
    act1 : peds_colour := red
end
Event set_cars_colours ≡
refines set_cars
any
    new_value_colours
where
    grd1 : new_value_colours ⊆ COLOURS
    grd2 : green ∈ new_value_colours ⇒ peds_colour = red
    grd_y_r : cars_colours = {yellow} ⇒ new_value_colours = {red}
    grd_r_ry : cars_colours = {red} ⇒ new_value_colours = {red, yellow}
    grd_ry_g : cars_colours = {red, yellow} ⇒ new_value_colours = {green}
    grd_g_y : cars_colours = {green} ⇒ new_value_colours = {yellow}
with
    new_value : new_value = TRUE ⇔ green ∈ new_value_colours
then
    act1 : cars_colours := new_value_colours
end
END
```

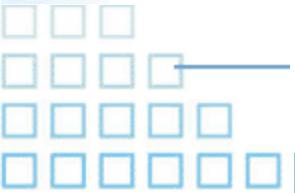


最终的目的是什么

- 对上述模型进行证明，确保正确性
- 根据上述模型，生成Ada/C/Java代码，得到软件实体
- 怎么做？后面再讲

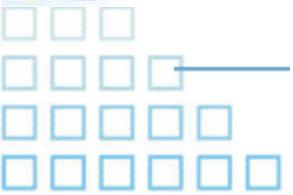


2. 基本理论



本部分内容

- 命题逻辑
- 集合论
- 谓词逻辑



命题逻辑

- 逻辑常量： TRUE(1), FALSE(0)
- 命题符号： P, Q, S, ...
 - 命题，如：我国于2013年11月划定了东海防空识别区
- 逻辑连接词

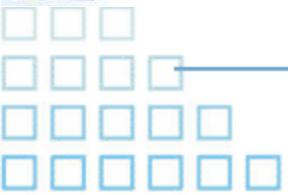
\wedge ...与

\vee ...或

\rightarrow ...蕴涵

\leftrightarrow ..当且仅当

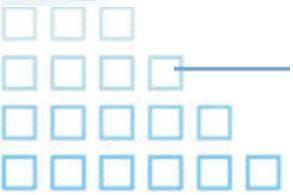
\neg ...非



命题公式的解释

- 真值表

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1



等价公式

- \Leftrightarrow 公式的等价关系， \Leftrightarrow 是逻辑连接词

双重否定律	$\neg\neg A \Leftrightarrow A$
结合律	$(A \wedge B) \wedge C \Leftrightarrow A \wedge (B \wedge C)$ $(A \vee B) \vee C \Leftrightarrow A \vee (B \vee C)$ $(A \Leftrightarrow B) \Leftrightarrow C \Leftrightarrow A \Leftrightarrow (B \Leftrightarrow C)$
交换律	$A \vee B \Leftrightarrow B \vee A$ $A \wedge B \Leftrightarrow B \wedge A$
分配律	$A \wedge (B \vee C) \Leftrightarrow (A \wedge B) \vee (A \wedge C)$ $A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C)$
吸收律	$A \vee (A \wedge B) \Leftrightarrow A$ $A \wedge (A \vee B) \Leftrightarrow A$
德·摩根律	$\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$ $\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$
同一律	$A \vee F \Leftrightarrow A$ $A \wedge T \Leftrightarrow A$
零律	$A \vee T \Leftrightarrow T$ $A \wedge F \Leftrightarrow F$
否定律	$P \vee \neg P \Leftrightarrow T$ $P \wedge \neg P \Leftrightarrow F$
条件转化律	$A \rightarrow B \Leftrightarrow \neg A \vee B$ $A \rightarrow B \Leftrightarrow \neg B \rightarrow \neg A$



命题逻辑的推理

- 设 A 和 B 是两个命题公式，当且仅当 $A \rightarrow B$ 为一永真式（或重言式），即 $A \Rightarrow B$ ，称 B 是 A 的有效结论，或 B 可由 A 逻辑推出
- Logical inference is used to create new sentences that logically follow from a given set of predicate calculus sentences
- An inference rule is sound
- An inference rule is complete



推理规则

- 一些推理的正确性规则

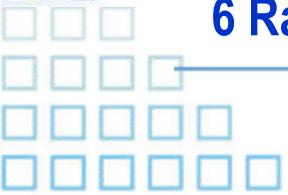
<u>RULE</u>	<u>PREMISE</u>	<u>CONCLUSION</u>
Modus Ponens	$A, A \rightarrow B$	B
And Introduction	A, B	$A \wedge B$
And Elimination	$A \wedge B$	A
Double Negation	$\neg\neg A$	A
Unit Resolution	$A \vee B, \neg B$	A
Resolution	$A \vee B, \neg B \vee C$	$A \vee C$



证明过程

- A **proof** is a sequence of sentences, where each sentence is either a **premise** or a **sentence** derived from earlier sentences in the proof by one of the rules of inference.
- The last sentence is the **theorem** (also called goal or query) that we want to prove.
- Example for the “weather problem” given above.

1 Humid	Premise	“It is humid”
2 Humid→Hot	Premise	“If it is humid, it is hot”
3 Hot	Modus Ponens(1,2)	“It is hot”
4 (Hot∧Humid)→Rain	Premise	“If it’s hot & humid, it’s raining”
5 Hot∧Humid	And Introduction(1,2)	“It is hot and humid”
6 Rain	Modus Ponens(4,5)	“It is raining”



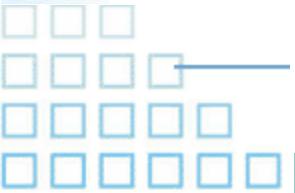
集合论

- **集合：**是由确定的对象(客体)构成的集体。
 - “确定”是指：论域内任何客体，要么属于这个集合，要么不属于这个集合，是唯一确定的。
- **元素：**集合中的对象，称之为元素
 - \in ：表示元素与集合的属于关系。
- **有限集合与无限集合**
- **集合的表示**
 - 列元素法 $A = \{1, 2, 3, 4, 5\}$
 - 谓词表示法 $B = \{x \mid x \in N \wedge 1 \leq x \leq 5\}$
- **集合的特征**
 - 互异性 $\{1, 2, 3, 2, 4\} = \{1, 2, 3, 4\}$
 - 无序性 $\{4, 2, 1, 3\} = \{1, 2, 3, 4\}$
- **空集**
 - 不含任何元素的集合叫做空集
 - 空集是唯一的



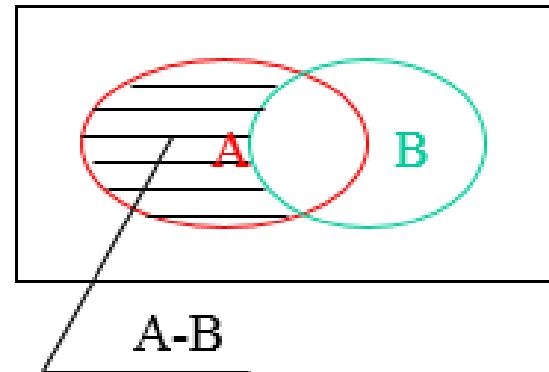
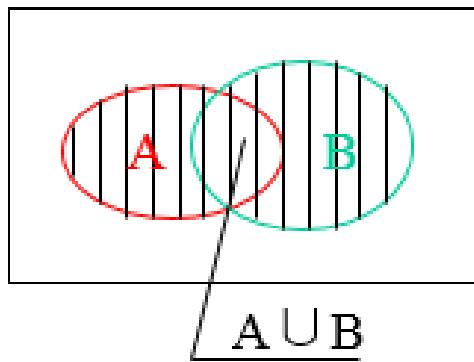
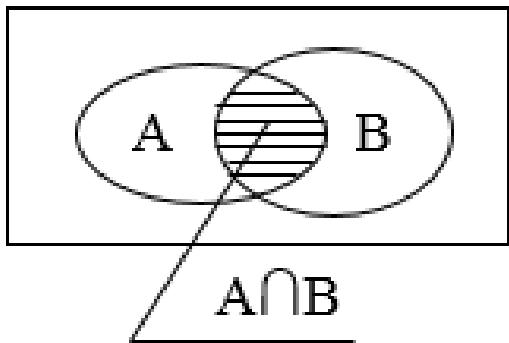
集合之间的关系

- 子集
 - 如果B中的每个元素都是A中的元素，则称B是A的子集合，简称子集
 - $B \subseteq A \Leftrightarrow \forall x(x \in B \rightarrow x \in A)$
- 相等
 - 设A, B为集合，如果 $A \subseteq B$ 且 $B \subseteq A$ ，则称A与B相等，记作 $A=B$
- 真子集
 - $B \subseteq A$ 且 $B \neq A$ ，则称B是A的真子集

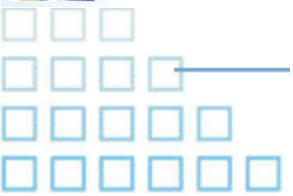


集合的基本运算

- 交、并、差



- 各种运算的性质



集合的基本运算

- 笛卡尔积
 - $S \times T = \{x \mapsto y | x \in S \wedge y \in T\}$
 - $x \mapsto y$ 是偶对
- 幂集 Powerset
 - $\mathbb{P}(S) = \{s | s \subseteq S\}$
- 集合的势
 - $card(S)$



集合上的谓词

- $s \in S$ (是 $\in (s, S)$ 的习惯写法, \in 是谓词)
- $s \notin S$
- $S \subset T$
- $S \not\subset T$
- $S \subseteq T$
- $S \not\subseteq T$
- 是否有限集 $finite(S)$
- 是否集合的分割 $partition(S, x, y)$
 - $S = x \cup y \wedge x \cap y = \emptyset$



集合上的关系

- 关系是偶对的集合，是多对多的映射
- 集合S和T之间的关系集合： $S \leftrightarrow T = \mathbb{P}(S \times T)$
- S和T之间的某个关系： $r \in S \leftrightarrow T$
- Domain: $\text{dom}(r)$
 - $\forall r. r \in S \leftrightarrow T \Rightarrow \text{dom}(r) = \{x | (\exists y. x \mapsto y \in r)\}$
- Range: $\text{ran}(r)$
 - $\forall r. r \in S \leftrightarrow T \Rightarrow \text{ran}(r) = \{y | (\exists x. x \mapsto y \in r)\}$
- Total relation: $S \leftrightarrow\rightarrow T$
 - $r \in S \leftrightarrow\rightarrow T \Rightarrow \text{dom}(r) = S$
- Subjective relation: $S \leftrightarrow\rightarrow T$
 - $r \in S \leftrightarrow\rightarrow T \Rightarrow \text{ran}(r) = T$
- Total subjective relation: $S \leftrightarrow\rightarrow T$
 - $r \in S \leftrightarrow\rightarrow T \Rightarrow \text{dom}(r) = S \wedge \text{ran}(r) = T$



关系运算

- Forward composition: $p; q$
 - $\forall p, q. p \in S \times T \wedge q \in T \times U \Rightarrow p; q = \{x \mapsto y | (\exists z. x \mapsto z \in p \wedge z \mapsto y \in q)\}$
- Backward composition:
 - $p \circ q = q; p$
- Identity
 - $S \triangleleft id = \{x \mapsto x | x \in S\}$
- Domain restriction
 - $S \triangleleft r = \{x \mapsto y | x \mapsto y \in r \wedge x \in S\}$
- Domain subtraction
 - $S \triangleleft r = \{x \mapsto y | x \mapsto y \in r \wedge x \notin S\}$
- Range restriction
 - $r \triangleright T = \{x \mapsto y | x \mapsto y \in r \wedge y \in T\}$
- Range subtraction
 - $r \triangleright T = \{x \mapsto y | x \mapsto y \in r \wedge y \notin T\}$
- Inverse
 - $r^{-1} = \{y \mapsto x | x \mapsto y \in r\}$



关系运算

- Relational image
 - $r[S] = \{y | \exists x. x \in S \wedge x \mapsto y \in r\}$
- Overriding
 - $r_1 \trianglelefteq r_2 = r_2 \cup (\text{dom}(r_2) \triangleleft r_1)$
- Direct product
 - $p \otimes q = \{x \mapsto (y \mapsto z) | x \mapsto y \in p \wedge x \mapsto z \in q\}$
- Parallel product
 - $p \parallel q = \{(x \mapsto y) \mapsto (m \mapsto n) | x \mapsto m \in p \wedge y \mapsto n \in q\}$
- Projection: prj1
 - $(S \times T) \triangleleft \text{prj1} = \{(x \mapsto y) \mapsto x | x \mapsto y \in S \times T\}$
- Projection: prj2
 - $(S \times T) \triangleleft \text{prj2} = \{(x \mapsto y) \mapsto y | x \mapsto y \in S \times T\}$



函数

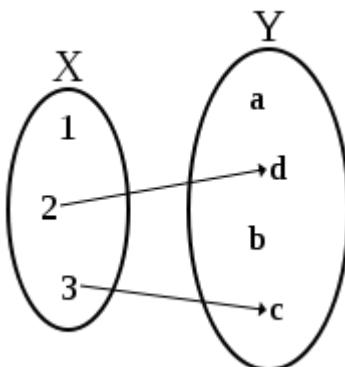
- 函数：一种特殊的关系，多对一的关系
- Partial functions
 - $S \rightarrow T = \{r. r \in S \leftrightarrow T \wedge r^{-1}; r \subseteq T \lhd id\}$
- Total functions：定义域是S的partial function
 - $S \rightarrow T = \{f | f \in S \rightarrow T \wedge \text{dom}(f) = S\}$
- Partial injections: One-to-one relations
 - $S \nrightarrow T = \{f. f \in S \rightarrow T \wedge f^{-1} \in T \rightarrow S\}$
- Total injections：定义域是S的partial injection
 - $S \rightarrow T = S \nrightarrow T \cap S \rightarrow T$
- Partial surjections：Onto relations，值域是T的partial function
 - $S \nrightarrow T = \{f. f \in S \rightarrow T \wedge \text{ran}(f) = T\}$
- Total surjections：定义域是S的partial surjection
 - $S \rightarrow T = S \nrightarrow T \cap S \rightarrow T$
- Bijections：定义域是S，值域是T的partial injection
 - $S \nrightarrow T = S \rightarrow T \cap S \rightarrow T$



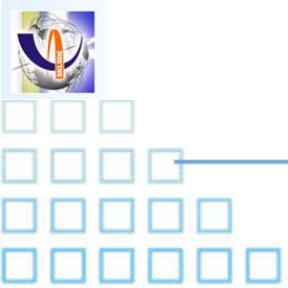
概念澄清

- 以Partial functions为例

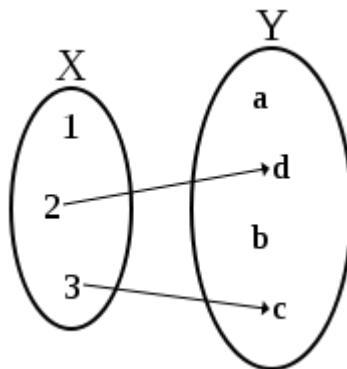
- 定义: $S \rightarrow T = \{r. r \in S \leftrightarrow T \wedge r^{-1}; r \subseteq T \lhd id\}$
- $S \rightarrow T$ 是S到T的所有partial function的集合, $S \rightarrow T$ 不是一个function
- 若 $r \in S \rightarrow T$, 则r是一个partial function, 函数r的定义域不一定是S、值域不一定是T
- 如:



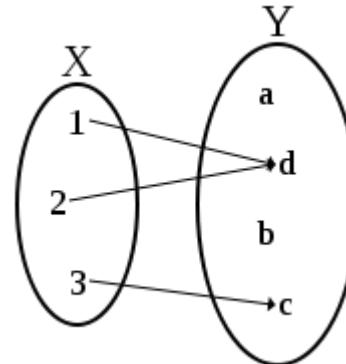
- $r \in X \rightarrow Y, r = \{2 \mapsto d, 3 \mapsto c\}, dom(r) = \{2,3\} \neq X, ran(r) = \{c,d\} \neq Y$



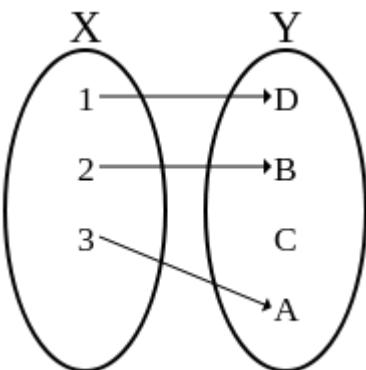
函数图示



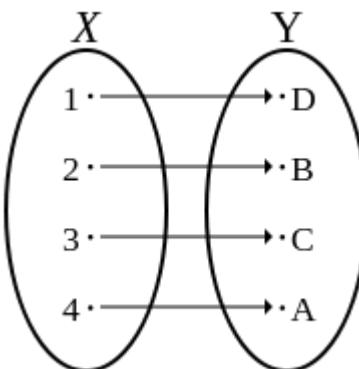
partial function, partial injection



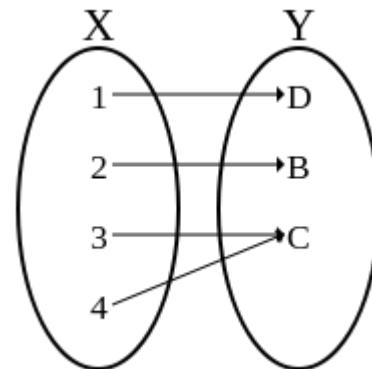
total function, but not injection)



Total injection, but not surjection



bijection

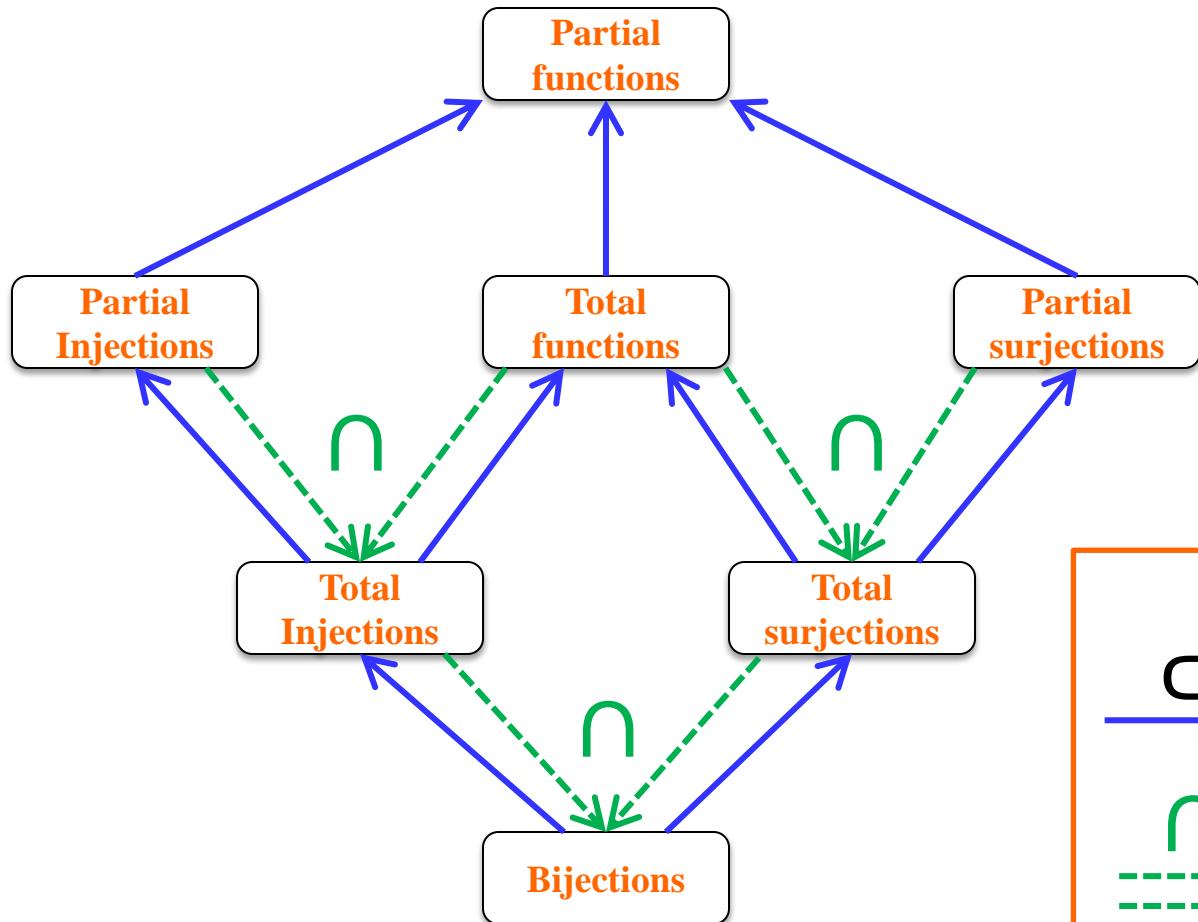


Total surjections



不同函数间的关系

- 集合S到T的不同函数



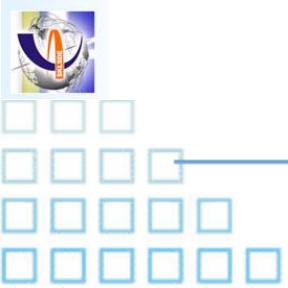
谓词逻辑

- 命题逻辑
 - 与、或、非、命题符合
 - 没有量词
 - “**All** poodles are dogs.”
 - “There is **at least one** black swan.”
 - “**Only** supervisors are allowed to fill in the form.”
 - “**No** person can solve this problem.”
 - 著名的苏格拉底三段论:
 - “人都是要死的，苏格拉底是人，所以苏格拉底是要死的。”
 - $\forall x(\text{person}(x) \rightarrow \text{dead}(x)), \text{person}(s)$
 - $\text{person}(s) \rightarrow \text{dead}(s), \text{person}(s)$
 - $\text{dead}(s)$
- 谓词
 - 谓词是一个bool值的函数: $f \in S \rightarrow \{\text{true}, \text{false}\}$ 或写成 $f: S \rightarrow \{\text{true}, \text{false}\}$

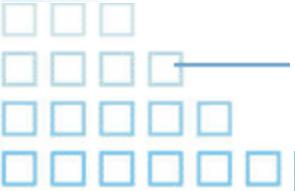


谓词逻辑

- 变量: x, y, \dots
- 谓词: $\text{female}(x), \text{sister}(x, y), \dots$
- 量词: $\forall x, \exists x$
- 逻辑连接词: 与命题逻辑一样

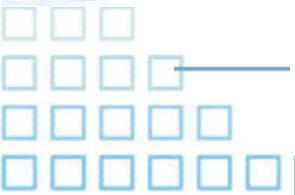


3. Event-B详解

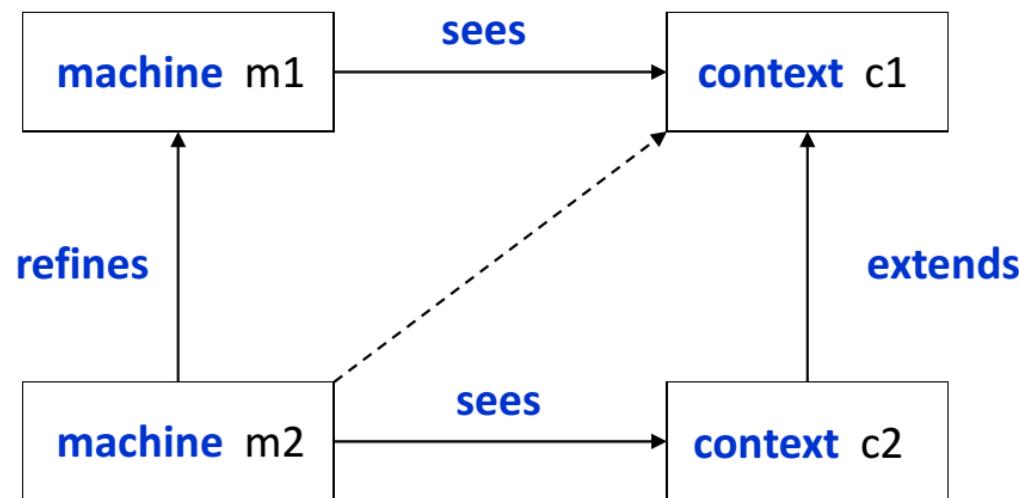
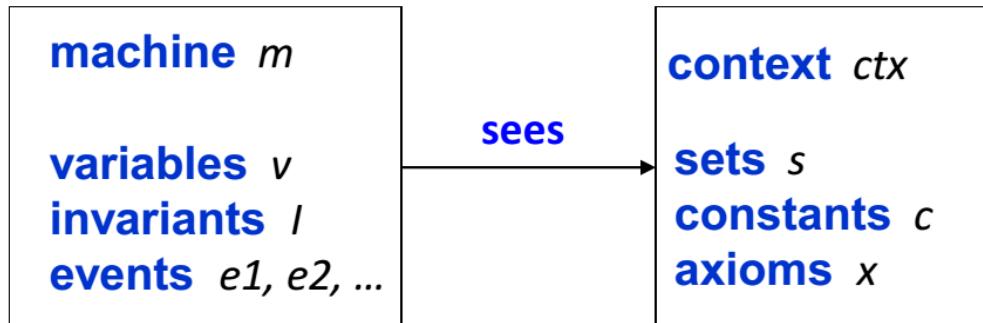


本部分内容

- 抽象机与上下文
- 求精
- 正确性证明



Event-B的基本Component



Context

- **SETS**: 声明用户自定义的、规约中使用的抽象类型
- **CONSTANTS**: 不可修改的逻辑变量，即常量，常量的类型须在**AXIOMS**中声明
- **AXIOMS**: 逻辑谓词，**context**中元素的规则。在开发模型时这些谓词是真的，并作为证明的前提
 - 某些**axiom**可以标记为**theorem**，表明这些**theorem**可以通过之前的**axiom**来证明，一旦被证明，这些**theorem**在后续的证明中就跟**axiom**一样

CONTEXT agatha

CONSTANTS

red

yellow

green

SETS

COLOURS

AXIOMS

`type : partition(COLOURS, {red}, {yellow}, {green})`



Machine

MACHINE mac

VARIABLES

cars_go
peds_go

INVARIANTS

inv1 : cars_go ∈ BOOL
inv2 : peds_go ∈ BOOL
inv3 : $\neg(cars_go = \text{TRUE} \wedge peds_go = \text{TRUE})$

EVENTS

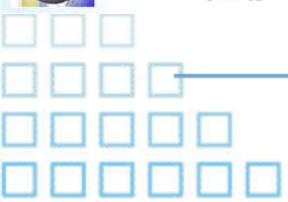
Initialisation

begin
 act1 : cars_go := FALSE
 act2 : peds_go := FALSE
end

Event set_peds_go ≡

when
 grd1 : cars_go = FALSE
then
 act1 : peds_go := TRUE
end

Event set_peds_stop ≡
begin
 act1 : peds_go := FALSE
end
Event set_cars ≡
any
 new_value
where
 grd1 : new_value ∈ BOOL
 grd2 : new_value = TRUE ⇒ peds_go = FALSE
then
 act1 : cars_go := new_value
end
END



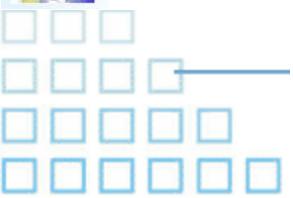
Machine

- 一个machine通过变量以及修改变量值的事件，描述一个模型的动态行为
- Refines: 后续再讨论
- Sees
 - 引用context，可以表明该machine中可以使用context的sets和constants
 - 并且在machine的每个证明中，axioms都是前提假设(hypotheses)
- Variables
 - 状态变量
 - 变量的值由initialisation事件确定，并可被其他event修改
 - 变量表明了machine的状态(state)
 - 每个变量的类型，必须在INRATIANTS中声明
- Invariants
 - 声明machine的每个可达状态下必须为真的谓词
- Events
 - 可为变量赋新值
 - 事件的guards子句表明了该事件可执行的条件
 - 每个machine唯一的initialisation是一个特殊的事件



Event

- **Any**
 - 事件的参数列表
- **Where(when)**
 - 该事件可执行的条件
- **With**
 - **Witness:** 在refinement中介绍
- **Then**
 - 事件执行的动作



事件内的动作

- skip 空动作
- 赋值动作
 - $z := E$
- 通过集合赋值
 - $x: \in S$
 - “becomes in”: 从集合S内任取一个元素赋给x
- 通过谓词赋值
 - $z: |P$
 - “becomes such that”: 为z内的变量赋任意、满足谓词P的值
- 函数赋值
 - $f(x) := E$
 - 将函数f中x的对应值赋为E
 - $f(x) := E = f := f \trianglelefteq \{x \mapsto E\}$



Event-B模型的语义

- 事件的语义
 - 系统动态行为由卫式命令（事件）表示

WHEN *guard* THEN *action* END

- Machine的语义
 - 系统总体行为是系统事件的无限循环

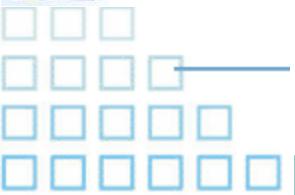
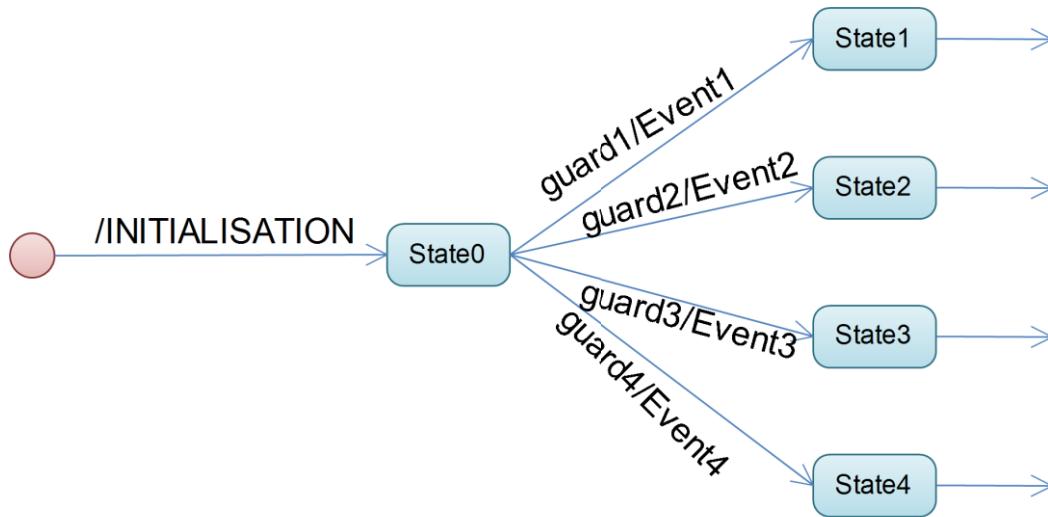
```
forever do  
    Event1 or  
    Event2 or  
    Event3 or ...  
end
```

- 模型不变式定义了一个允许状态的集合，每个事件都必须保持不变式



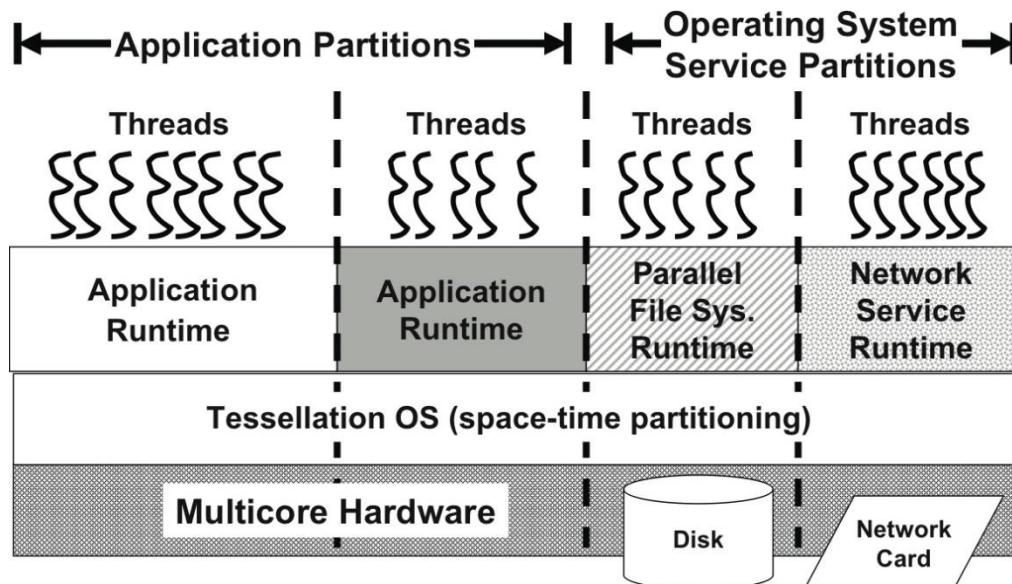
Event-B模型的语义

- 状态机语义



基于集合的建模

- Event-B建模方式：基于集合
- 分区操作系统内核（Partitioning Kernel）



分区内核的集合

context Kernel_ctx 分区的集合

sets Partitions 所有线程的集合

AllThreads // *the thread including used and unused threads*

PartitionModes 分区运行模式
类型的集合

ThreadStates 线程状态类
类型的集合

CriticalityLevels 分区关键等级
类型的集合

SystemState // *system state of HM*

DeadlineType 线程deadline
类型的集合

.....



分区内核的集合

- constants
 - PM_IDLE PM_NORMAL PM_WARM_START PM_COLD_START** // partition operating mode, according to ARINC653
分区状态的
类型常量
 - TS_Dormant TS_Ready TS_Waiting TS_Suspend TS_WaitandSuspend TS_Running** // thread state
线程状态的
类型常量
 - CRITILITY_LEVEL_A CRITILITY_LEVEL_B CRITILITY_LEVEL_C CRITILITY_LEVEL_D CRITILITY_LEVEL_E** // denotes criticality level of partition, Each partition is assigned a DO-178B criticality level from level A to E
关键等级的
类型常量
 - Critical_Level** /* denotes criticality level of partition, Each partition is assigned a DO-178B criticality level from level A to E*/
Period_of_Partition /* defines the activation period of the partition, and is used to determine the partition's runtime placement within the core module's overall time frame. */
每个分区的关
键等级(函数)
 - Duration_of_Partition** // the amount of processor time given to the partition every period of the partition.
 - System_Partition** // denotes the system partition running the kernel
 - System_HM_Table Module_HM_Table Partition_HM_Table**
 - DEADLINE_HARD DEADLINE_SOFT**

分区内核的集合

axioms

@axm_finite_partitions finite(**Partitions**)

分区集合是
有限集

分区模式类型
集合的分割

@axm_partition_partitionmodes partition(**PartitionModes**, {**PM_IDLE**}, {**PM_NORMAL**},
{**PM_WARM_START**}, {**PM_COLD_START**})

@axm_partition_threadstates partition(**ThreadStates**, {**TS_Dormant**}, {**TS_Ready**}, {**TS_Waiting**},
{**TS_Suspend**}, {**TS_WaitandSuspend**}, {**TS_Running**})

@axm_partition_nums card(**Partitions**) ≥ 2 // at least two partition, the kernel partition and a
application partition

分区集合的势

@axm_periodofpart **Period_of_Partition** \in **Partitions** $\rightarrow \mathbb{N}$

都是Total
function

@axm_durationofpart **Duration_of_Partition** \in **Partitions** $\rightarrow \mathbb{N}$

@axm_criticallevel **Critical_Level** \in **Partitions** \rightarrow **CriticalityLevels**

是分区集合的
一个元素

@axm_syspart **System_Partition** \in **Partitions**

@axm_partition_criticallevel partition(**CriticalityLevels**, {**CRITILITY_LEVEL_A**}, {**CRITILITY_LEVEL_B**},
{**CRITILITY_LEVEL_C**}, {**CRITILITY_LEVEL_D**}, {**CRITILITY_LEVEL_E**})

@axm_syshtable **System_HM_Table** \in **ErrorCode** \times **SystemState** \rightarrow **ErrorLevel**

@axm_modulehtable **Module_HM_Table** \in **ErrorCode** \times **SystemState** \rightarrow **ModuleActionType**

@axm_parhtable **Partition_HM_Table** \in **Partitions** \rightarrow (**ErrorCode** \times **SystemState** \rightarrow
Partition ActionType)

@axm_partition_deadlinetype partition(**DeadlineType**, {**DEADLINE_HARD**}, {**DEADLINE_SOFT**})



分区内核的健康监控

- System HM table

 - 根据检测到的错误及系统状态，定义一个错误的等级

System States

→ State Symbolic
→ State Codes

ML = Module Level Error
PL = Partition Level Error

System HM Table Mappings

Detected Error		module init	Module Function	partition switching	partition init	partition process management	partition error handler	Process Execution	OS Execution	...
Symbolic Name	Error ID	state 1	state 2	state 3	state 4	state 5	state 6	state 7	state 8	...
configuration error	1	ML								
module config error	2	ML								
partition config error	3				PL					
partition init error	4				PL					
segmentation error	5	ML	ML	ML	PL	PL	PL	MEMORY_VIOLATION	PL	
time duration exceeded	6	ML	ML	ML	PL	PL	PL	DEADLINE_MISSED	PL	
invalid OS call	7	ML	ML	ML	PL	PL	PL	ILLEGAL_REQUEST		
divide by 0	8	ML	ML	ML	PL	PL	PL	NUMERIC_ERROR	PL	
overflow	9	ML	ML	ML	PL	PL	PL	NUMERIC_ERROR	PL	
floating point error	10	ML	ML	ML	PL	PL	PL	NUMERIC_ERROR	PL	
application reports error	11							APPLICATION_ERROR		
stack overflow	12	ML	ML	ML	PL	PL	PL	STACK_OVERFLOW	PL	
parity error	13	ML	ML	ML	PL	PL	PL	HARDWARE_FAULT	PL	
io access error	14	ML	ML	ML	PL	PL	PL	HARDWARE_FAULT	PL	
supervisor priv. violation	15				ML	PL	PL	MEMORY_VIOLATION	PL	
power interrupt	16	ML	ML	ML	PL	PL	PL	POWER_FAILURE	PL	
power failure	17	ML	ML	ML	ML	ML	ML		ML	
...	...									

Partition HM Tables define the actions

Module HM Table defines the actions

分区内核的健康监控

- Module HM Table
 - 定义模块级错误发生时的动作

System States		Module HM Table Mappings								
→	State Symbolic					partition process management				
→	State Codes									
↓ Detected Error ↓		module init	Module Function	partition switching	partition init	partition process management	partition error handler	Process Execution	OS Execution	
Symbolic Name	Error ID	state 1	state 2	state 3	state 4	state 5	state 6	state 7	state 8	...
configuration error	1	SH								
module config error	2	SH								
partition config error	3									
partition init error	4									
segmentation error	5	SH	SH	SH						
time duration exceeded	6	IG	IG	IG						
invalid OS call	7	IG	IG	IG						
divide by 0	8	IG	IG	IG						
overflow	9	IG	IG	IG						
floating point error	10	IG	IG	IG						
application reports error	11									
stack overflow	12	RS	RS	RS						
parity error	13	RS	RS	RS						
io access error	14	IG	IG	IG						
supervisor priv. violation	15				RS					
power interrupt	16	IG	IG	IG	IG					
power failure	17	RS	RS	RS	RS	RS	RS	RS	RS	
...	...									
Module HM Callback	callback name / addr									

Module Actions

RS = RESET

SH = SHUTDOWN

IG = IGNORE



分区内核的健康监控

- Partition HM Table
 - 定义分层级错误发生时的动作

System States		Partition 1 HM Table Mappings							
→	State Symbolic								
→	State Codes								
↓ Detected Error ↓		module init	system function execution	partition switching	partition init	partition process management	partition error handler	Process Execution	OS Execution
Symbolic Name	Error ID	state 1	state 2	state 3	state 4	state 5	state 6	state 7	state 8
configuration error	1								
module config error	2								
partition config error	3					ID			
partition init error	4					CS			
segmentation error	5					CS	ID	ID	
time duration exceeded	6					IG	IG	IG	WS
invalid OS call	7					IG	WS	ID	IG
divide by 0	8					IG	IG	WS	ID
overflow	9					IG	IG	WS	ID
floating point error	10					IG	IG	WS	ID
application reports error	11							IG	
stack overflow	12					CS	ID	ID	ID
parity error	13					CS	IG	IG	ID
io access error	14					CS	IG	IG	ID
supervisor priv. violation	15						ID	ID	ID
power interrupt	16						WS	WS	ID
power failure	17								
...	...								
Partition 1 HM Callback	callback name								



分区内核的健康监控定义

@axm_syshtable **System_HM_Table** ∈ $\text{ErrorCode} \times \text{SystemState} \rightarrow \text{ErrorLevel}$

@axm_modulehtable **Module_HM_Table** ∈ $\text{ErrorCode} \times \text{SystemState} \rightarrow \text{Module ActionType}$

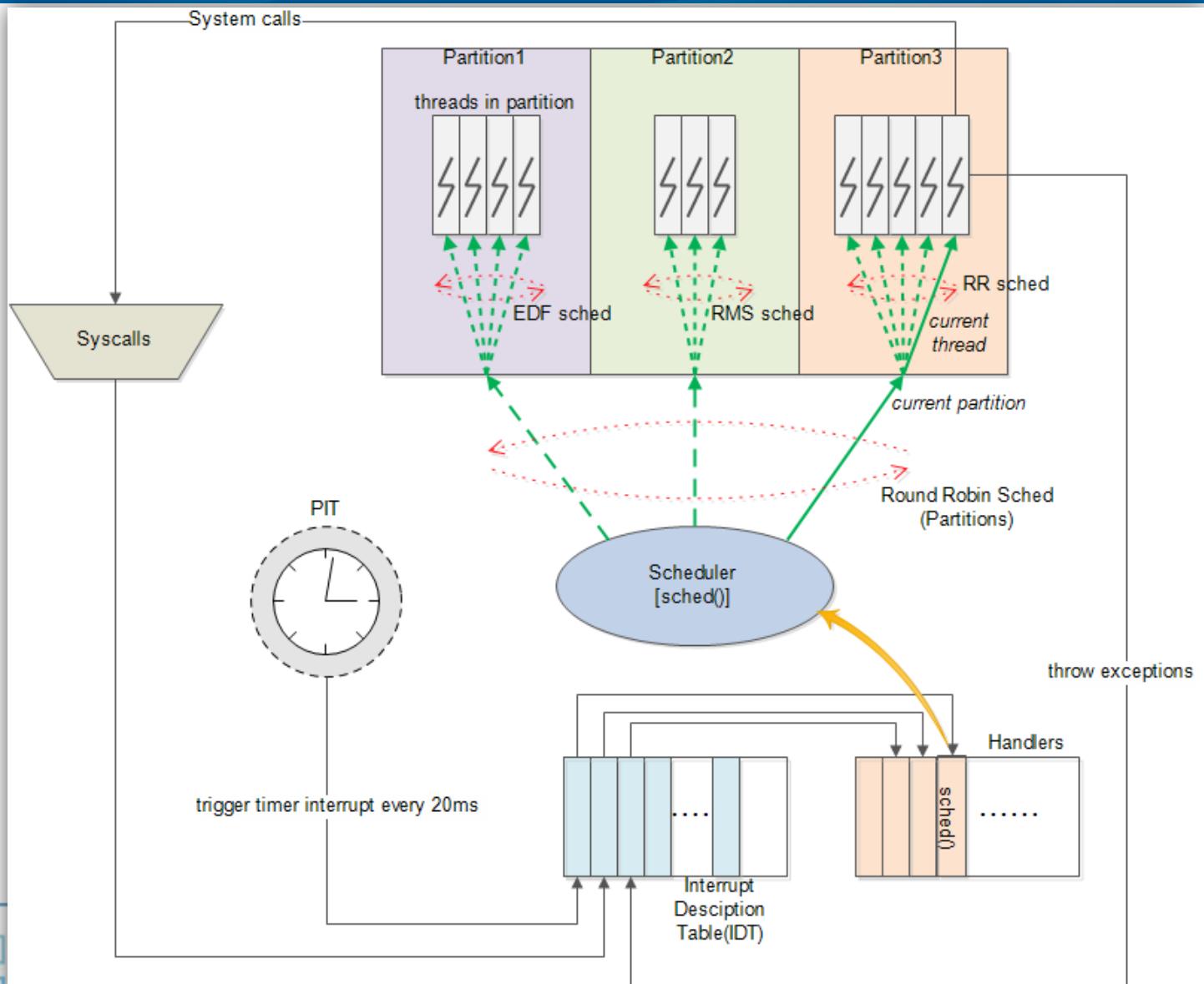
@axm_partitiontable **Partition_HM_Table** ∈ $\text{Partitions} \rightarrow (\text{ErrorCode} \times \text{SystemState} \rightarrow \text{Partition ActionType})$

两个集合笛卡尔积
到ErrorLeve集合
的Partial function

每个分区都有一个
健康监控表，是
Total function

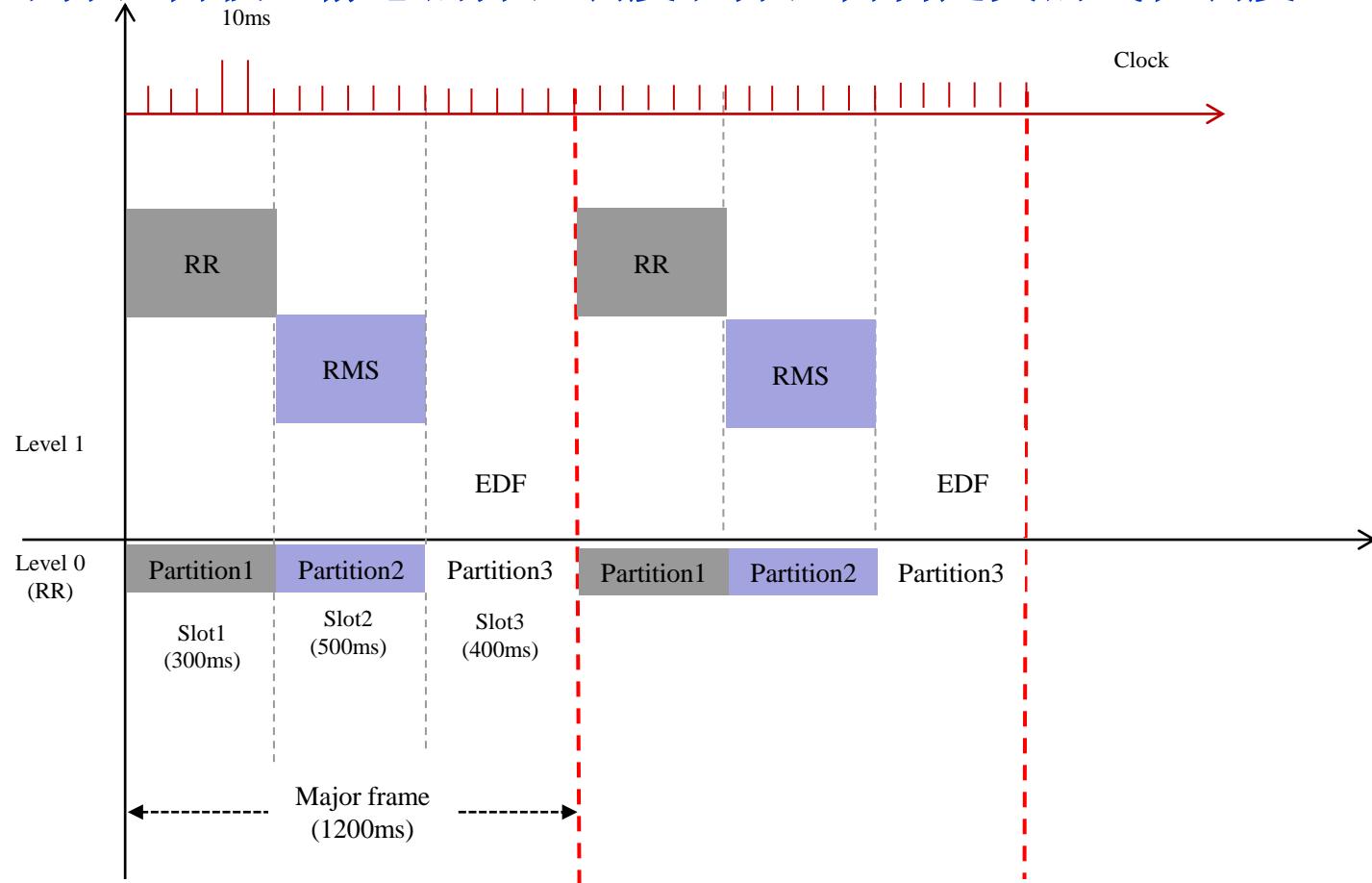


分区内核的动态行为



分区内核的动态行为

- 分区管理和分区调度
 - 分区管理包括分区内线程，调度是两级调度
 - 对于分区内核，静态的分区调度和分区内自定义的线程调度



分区内核的动态行为

- 分区调度

context Kernel_ctx

Constants

schedulingFrameofPartition // the scheduling time frame of each partition

schedulingFrames // the scheduling time frames

majorFrame // the total time of all partitions

axioms

@axm_scheduleframes **schedulingFrames** $\in \mathbb{P1}(\mathbb{N} \times \mathbb{N})$ // each $\langle x \rightarrow y \rangle : schedulingFrames$,
x is the start time of the frame and y is the end time

@axm_scheduleframes2 $\forall x \cdot (x \in \text{dom}(\text{schedulingFrames}) \Rightarrow x \leq \text{schedulingFrames}(x))$ //
the end time should be larger than the start time of each frame

@axm_scheduleframes3 $\forall x, y \cdot (x \in \text{dom}(\text{schedulingFrames}) \wedge y \in \text{dom}(\text{schedulingFrames}))$
 $\Rightarrow (x \geq \text{schedulingFrames}(y) \vee y \geq \text{schedulingFrames}(x))$ // scheduling frames are disjoint

@axm_scheduleframeofpar **schedulingFrameofPartition** $\in \text{schedulingFrames} \twoheadrightarrow \text{Partitions}$
// in a major frame, each partition should have one or more than one scheduling frame

@axm_majorframe **majorFrame** $\in \mathbb{N1}$ // majorFrame should be larger than sum of all
schedulingframe. there may be spare time space between scheduling frames



分区内核的动态行为

machine Kernel_Machine sees Kernel_ctx

variables

Threads

threadsOfPartition // threads of each partition

partitionstate

threadstate

current_partition

current_thread

系统运行态的线程

集合

分区内的线程集合

(函数)

lock_level // denotes the current lock level of the partition

start_condition // denotes the reason the partition is started

error_thread // for health monitoring, a error handler process could be created in a partition

basepriority_threads // Denotes the capability of the process to manipulate other processes.

period_threads // Identifies the period of activation for a periodic process. A distinct and unique value should be specified to designate the process as aperiodic

timecapacity_threads // Defines the elapsed time within which the process should complete its execution.

deadline_threads // Specifies the type of deadline relating to the process, and may be "hard" or "soft".

currentpriority_threads // Defines the priority with which the process may access and receive resources. It is set to base priority at initialization time and is dynamic at runtime.

deadlinetime_threads // The deadline time is periodically evaluated by the operating system to determine whether the process is satisfactorily completing its processing within the allotted time.



分区内核的动态行为

invariants

- @inv_threadstype **Threads** ∈ AllThreads → BOOL // true: the thread has been created,
false: the thread has not been used
- @inv_partstate **partitionstate** ∈ Partitions → PartitionModes
- @inv_threadofpart **threadsOfPartition** ∈ AllThreads ↼ Partitions // partial surjection,
each used thread is belonging to a partition, each partition has one thread at least
- @inv_threadstate **threadstate** ∈ AllThreads → ThreadStates
- @inv_currpart **current_partition** ∈ Partitions
- @inv_currthread **current_thread** ∈ AllThreads ∧ Threads(**current_thread**) = TRUE ∧
threadsOfPartition(**current_thread**) = **current_partition**
- @inv_atleastoneonrunningthread card(**threadstate** ▷ {TS_Running}) ≤ 1
- @inv_locklevel **lock_level** ∈ Partitions → N
- @inv_start_condition **start_condition** ∈ Partitions → PartitionStartCondition
- @inv_error_thread **error_thread** ∈ Partitions ↼ AllThreads // partial injection. at most
one error thread for each partition
- @inv_basepriority_threads **basepriority_threads** ∈ AllThreads → N
- @inv_period_threads **period_threads** ∈ AllThreads → N

分区内的线程集合, partial surjection, 已创建的线程肯定

属于某个分区, 每个分区肯定有线程

对于单处理器系统, 只有一个线程处于执行态, Range restriction

每个分区的错误处理线程,
partial injection, 每个分区最多一个处理线程, 而且不同分区不能是同一个线程



分区内核的动态行为

- 集合的操作

$\forall t \cdot t \in \text{AllThreads} \Rightarrow \text{card}(\text{Threads}'[\{t\}]) = 1$

$\text{threadsOfPartition}' \subseteq \text{AllThreads} \times \text{Partitions}$

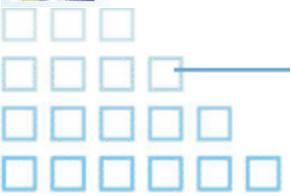
$\text{dom}(\text{threadsOfPartition}') = \text{Threads}'^{\sim}[\{\text{TRUE}\}]$

$\text{ran}(\text{threadsOfPartition}') = \text{Partitions}$

$\forall t \cdot t \in \text{Threads}'^{\sim}[\{\text{TRUE}\}] \Rightarrow \text{card}(\{t\} \triangleleft \text{threadsOfPartition}') = 1$

$\text{threadstate} = ((\text{AllThreads} \setminus \{\text{current_thread}\}) \times \{\text{TS_Dormant}\}) \cup \{\text{current_thread} \rightarrow \text{TS_Running}\}$

$\text{deadline_threads} = \text{AllThreads} \times \{\text{DEADLINE_HARD}\}$



分区内核的动态行为

- 事件

event schedule

any $p t$

where

@grd1 $p \in \text{Partitions}$

@grd11 $t \in \text{AllThreads} \wedge \text{Threads}(t) = \text{TRUE}$

@grd2 $\text{threadsOfPartition}(t) = p$

@grd4 $\text{threadstate}(t) = \text{TS_Ready}$

@grd3 $\text{partitionstate}(p) = \text{PM_NORMAL} // @grd5 t \neq \text{current_thread}$

@grd6 $\exists x, y, n \cdot (x \in \mathbb{N} \wedge y \in \mathbb{N} \wedge n \in \mathbb{N} \wedge x \rightarrow y \in \text{schedulingFrames}$

$\wedge \text{schedulingFrameofPartition}(x \rightarrow y) = p$

$\wedge (x + n * \text{majorFrame}) < \text{clock_tick} \wedge \text{clock_tick} < (y + n * \text{majorFrame})) // \text{current}$

tick is in the scheduling frame of the partition

@grd0 $\text{need_reschedule} = \text{TRUE}$

then

@act1 $\text{threadstate} := (\text{threadstate} \trianglelefteq (\text{threadstate} \sim [\{\text{TS_Running}\}] \times \{\text{TS_Ready}\})) \trianglelefteq \{t \mapsto \text{TS_Running}\}$

@act5 $\text{current_partition} = p$

@act6 $\text{current_thread} = t$

@act7 $\text{need_reschedule} = \text{FALSE}$

end

关系的override

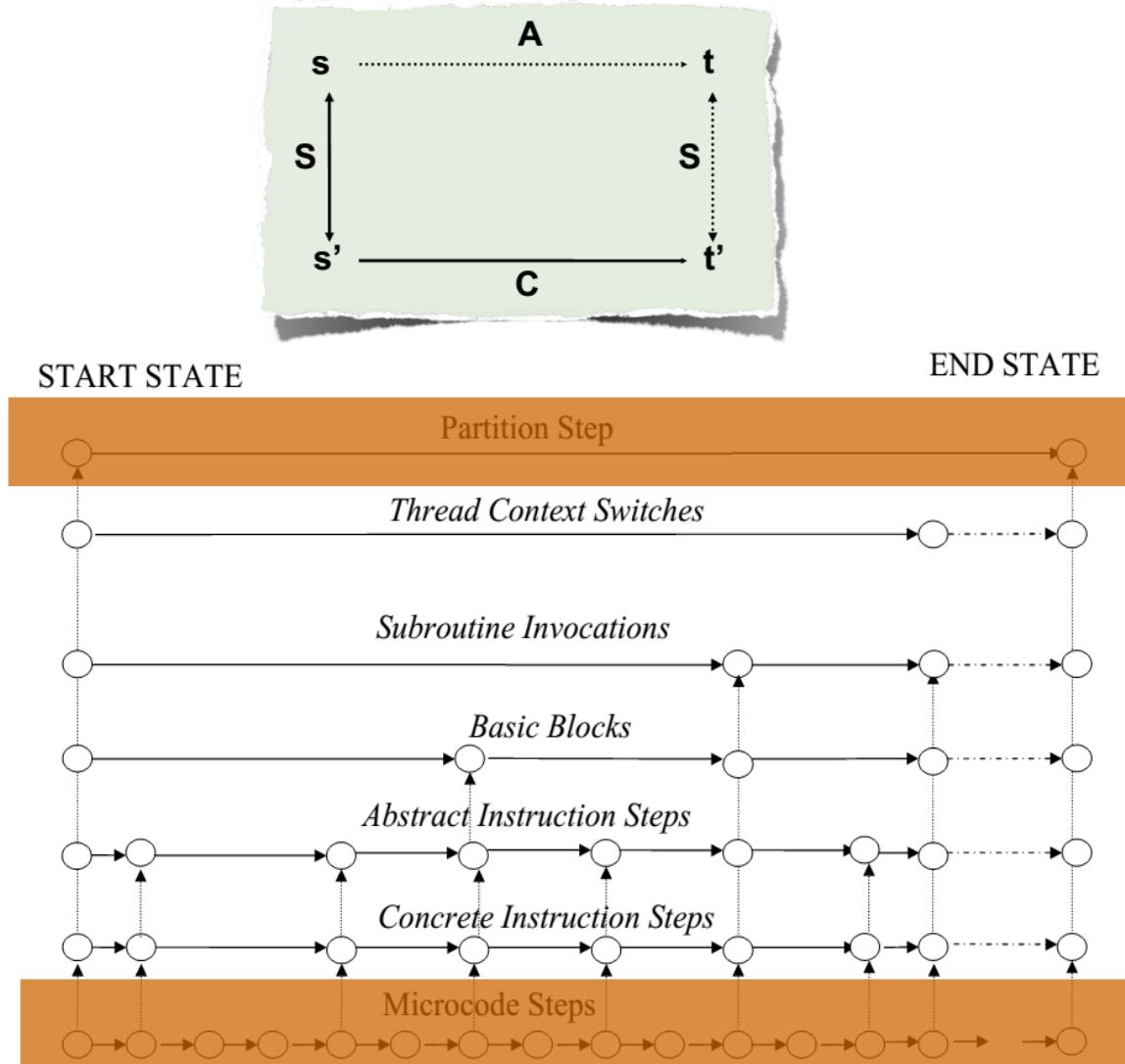
将当前执行线程的状态置为ready

将被调度线程t的状态置为running



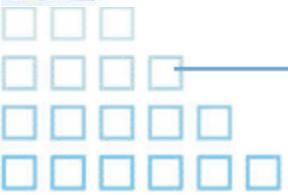
求精Refinement

- C是对A的求精，若C的所有行为均包含在A中，或被A所允许
- 举例：



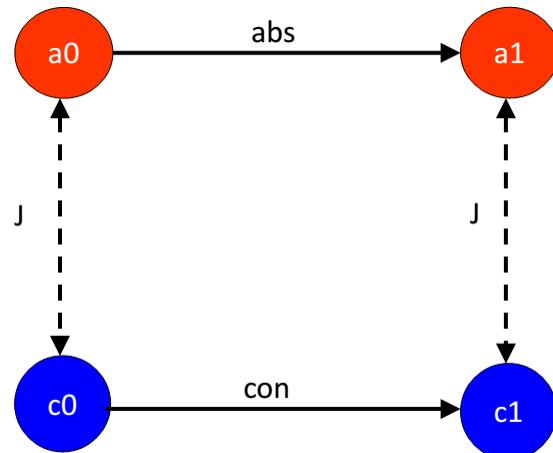
Event-B中模型求精的类型

- 1. 扩展:
 - 增加新变量，及其不变式
 - 扩展已有事件，使其作用于新增的变量
 - 增加新事件，使其作用于新增的变量
- 2. 带卫式条件修改的扩展:
 - 与基本扩展一样，不同在于修改了已有事件的卫式条件
- 3. 变量替换/数据具体化:
 - 用新的变量替换某些变量，比如用具体变量替换抽象变量
 - 修改已有事件，增加新事件
- 4. 变量删除:
 - 先前引入新变量后，某些已有变量变得冗余了，删除这些冗余变量

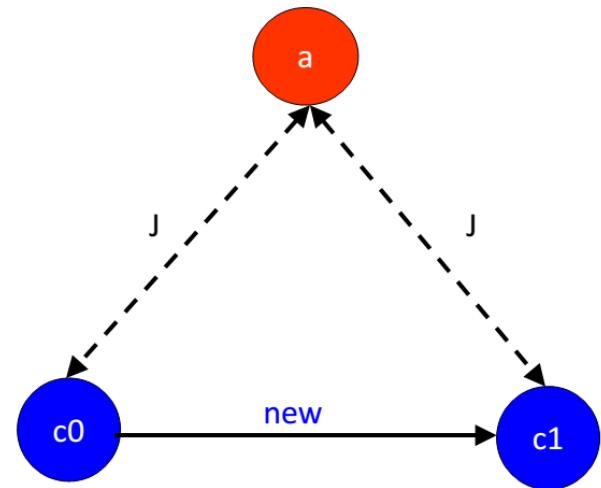


事件求精的类型

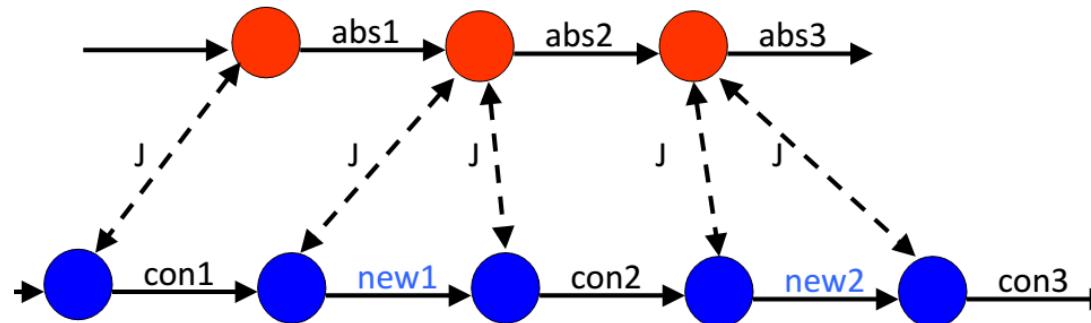
Simulation: maintaining a gluing relation



New concrete events refine skip (stuttering step)

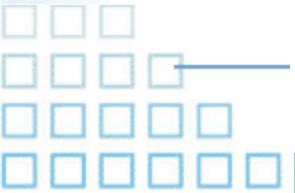
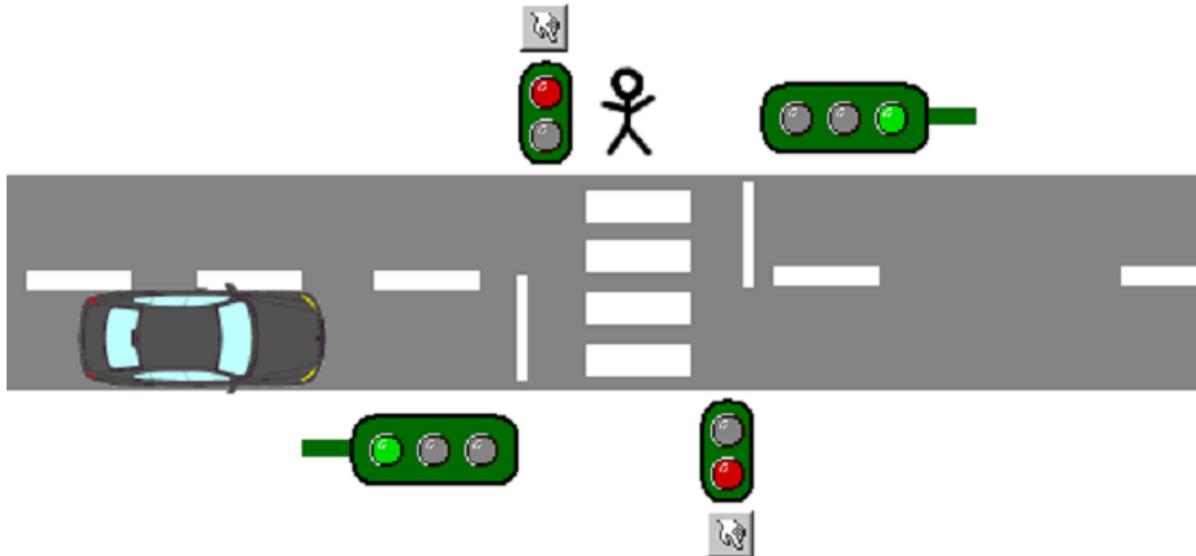


Refining traces



一个简单的例子(1)

- 交通信号灯控制器



一个简单的例子(1)

MACHINE mac

VARIABLES

cars_go
peds_go

INVARIANTS

inv1 : cars_go ∈ BOOL
inv2 : peds_go ∈ BOOL
inv3 : $\neg(cars_go = \text{TRUE} \wedge peds_go = \text{TRUE})$

EVENTS

Initialisation

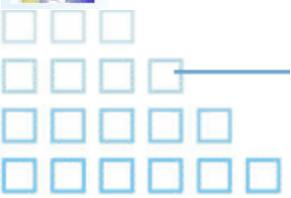
begin
 act1 : cars_go := FALSE
 act2 : peds_go := FALSE

end

Event set_peds_go ≡

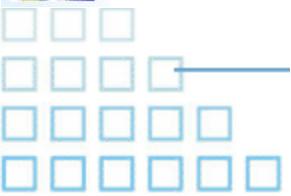
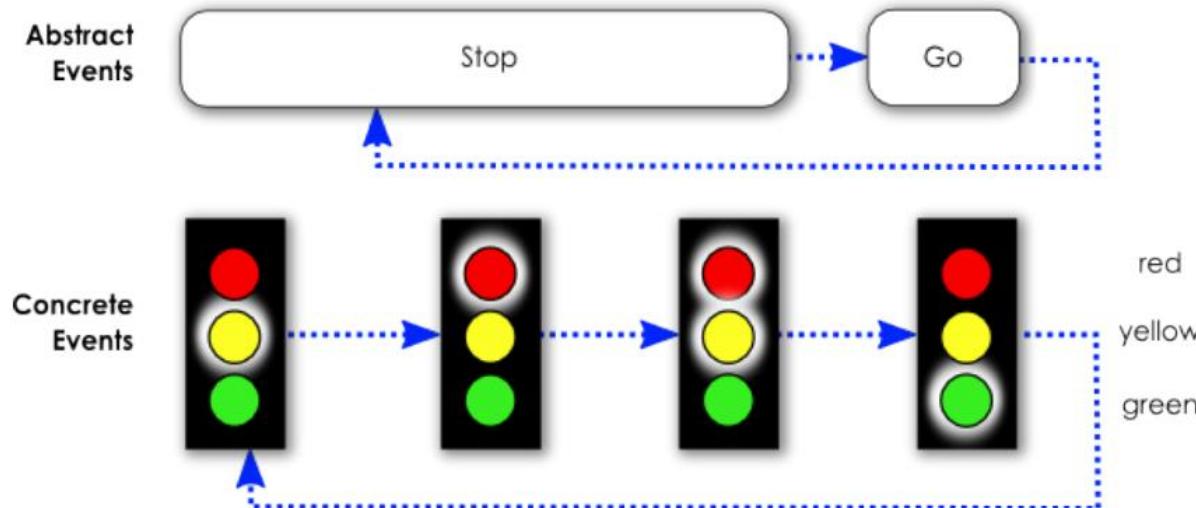
when
 grd1 : cars_go = FALSE
then
 act1 : peds_go := TRUE
end

Event set_peds_stop ≡
begin
 act1 : peds_go := FALSE
end
Event set_cars ≡
any
 new_value
where
 grd1 : new_value ∈ BOOL
 grd2 : new_value = TRUE ⇒ peds_go = FALSE
then
 act1 : cars_go := new_value
end
END



一个简单的例子(2)

- 交通信号灯控制器——求精



一个简单的例子(2)

- 交通信号灯控制器——求精
 - 引入Context

CONTEXT agatha

CONSTANTS

red

yellow

green

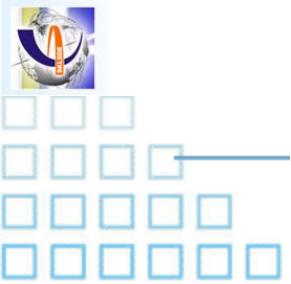
增加了常量和集合

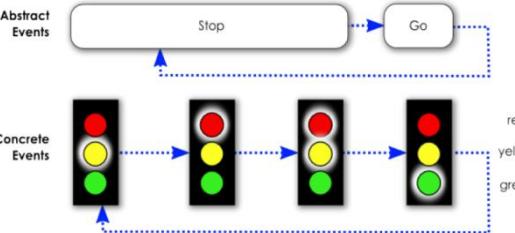
SETS

COLOURS

AXIOMS

`type : partition(COLOURS, {red}, {yellow}, {green})`





一个简单的例子(2)

• 交通信号灯控制器——求精

MACHINE mac1

REFINES mac

SEES ctx1

VARIABLES

peds_colour

cars_colours

替换原有的peds_go和
cars_go变量

为新变量增加新的不变式

INVARIANTS

inv4 : peds_colour ∈ {red, green}

inv5 : cars_colours ⊆ COLOURS

gluing_peds : peds_go = TRUE ⇔ peds_colour = green

gluing_cars : cars_go = TRUE ⇔ green ∈ cars_colours

EVENTS

Initialisation

begin

init4 : peds_colour := red

init5 : cars_colours := {red}

end

为新旧变量之间的关联不变
式

引入新变量后,
修改已有事件

Event set_peds_green ≈
refines set_peds_qo

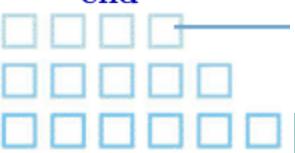
when

grd1 : green ∉ cars_colours

then

act2 : peds_colour := green

end



Event set_peds_red ≈

refines set_peds_stop

begin

act1 : peds_colour := red

end

引入新变量后,
修改已有事件

Event set_cars_colours ≈

refines set_cars

any

new_value_colours

where

grd1 : new_value_colours ⊆ COLOURS

grd2 : green ∈ new_value_colours ⇒ peds_colour = red

grd_y_r : cars_colours = {yellow} ⇒ new_value_colours = {red}

grd_r_yr : cars_colours = {red} ⇒ new_value_colours = {red, yellow}

grd_ry_g : cars_colours = {red, yellow} ⇒ new_value_colours = {green}

grd_g_y : cars_colours = {green} ⇒ new_value_colours = {yellow}

with

new_value : new_value = TRUE ⇔ green ∈ new_value_colours

then

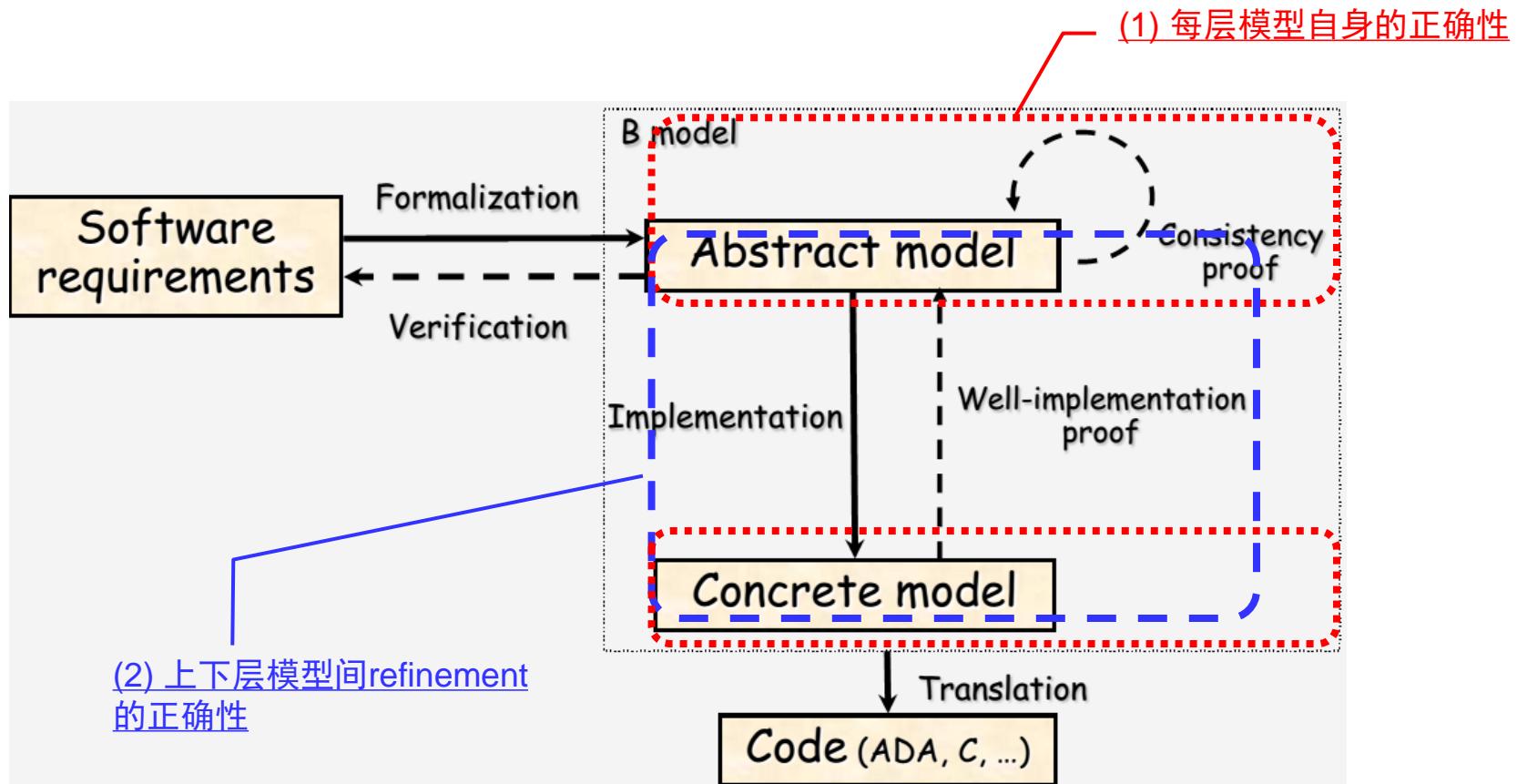
act1 : cars_colours := new_value_colours

end

END

Witness是新旧事
件参数间的关联
不变式

Event-B模型的正确性



正确性验证(1)——模拟simulation

ProB - Rodin Platform

File Edit Navigate Search Project Run ProB BMotion Studio Window Help

Events X Checks Event(s)

Event Parameter(s)

- ticktock
- schedule
- partition_cold_restart (x2) System_...
- partition_warm_restart
- partition_go (x2) System_...
- partition_shutdown (x2) System_...
- setPreemptive (x2) TRUE
- createThread
- setPriorityofThread (x4) 0, AllThre...
- thread_suspend_self
- thread_suspend

Event-B Explorer X Rodin Problems

- 93_HV_1
- 93_HV_2
- ARINC653_Kernel_v1
- DepSatSpec015Model000
- Heating_ControllerTutorial_Completed
- IntfRef-130106
- SSF_pilot
- VKernel
- VKernel2
- A_Project_Descript
- Kernel_c01
- Kernel_c03

State X Ltl Counter-Example

Name	Value	Previous value
schedulingFrameof	$\{((0 \mapsto 0) \mapsto \text{System...})\}$	$\{((0 \mapsto 0) \mapsto \text{System...})\}$
schedulingFrames	$\{(0 \mapsto 0), (0 \mapsto 1)\}$	$\{(0 \mapsto 0), (0 \mapsto 1)\}$
Kernel_Machine		
Threads	$\{(\text{AllThreads}1 \mapsto \dots)\}$	$\{(\text{AllThreads}1 \mapsto \dots)\}$
basepriority_thre	$\{(\text{AllThreads}1 \mapsto \dots)\}$	$\{(\text{AllThreads}1 \mapsto \dots)\}$
clock_tick	2	2
current_partition	System_Partition	System_Partition
current_thread	AllThreads1	AllThreads1
currentpriority	$\{(\text{AllThreads} \mapsto \dots)\}$	$\{(\text{AllThreads} \mapsto \dots)\}$
deadline_threads	$\{(\text{AllThreads}1 \mapsto \dots)\}$	$\{(\text{AllThreads}1 \mapsto \dots)\}$
deadlinetime_thre	$\{(\text{AllThreads}1 \mapsto \dots)\}$	$\{(\text{AllThreads}1 \mapsto \dots)\}$
error_thread	\emptyset	\emptyset
lock_level	$\{(\text{System_Partition} \mapsto \dots)\}$	$\{(\text{System_Partition} \mapsto \dots)\}$
need_reschedule	TRUE	TRUE
partitionstate	$\{(\text{System_Partition} \mapsto \dots)\}$	$\{(\text{System_Partition} \mapsto \dots)\}$
period_threads	$\{(\text{AllThreads}1 \mapsto \dots)\}$	$\{(\text{AllThreads}1 \mapsto \dots)\}$
preemptive	FALSE	FALSE
remain_timecapaci	$\{(\text{AllThreads}1 \mapsto \dots)\}$	$\{(\text{AllThreads}1 \mapsto \dots)\}$
start_condition	$\{(\text{System_Partition} \mapsto \dots)\}$	$\{(\text{System_Partition} \mapsto \dots)\}$
threadsOfPartition	$\{(\text{AllThreads}1 \mapsto \dots)\}$	$\{(\text{AllThreads}1 \mapsto \dots)\}$
threadstate	$\{(\text{AllThreads}1 \mapsto \dots)\}$	$\{(\text{AllThreads}1 \mapsto \dots)\}$
timecapacity_thre	$\{(\text{AllThreads}1 \mapsto \dots)\}$	$\{(\text{AllThreads}1 \mapsto \dots)\}$
wakeuptime_thread	$\{(\text{AllThreads}1 \mapsto \dots)\}$	$\{(\text{AllThreads}1 \mapsto \dots)\}$
Formulas		
► invariants	T	T
► axioms	T	T
► guards		

Invariant ok no event errors detected

History X Event Error View

Kernel_Machine

- setPriorityofThread
- ticktock
- setPriorityofThread
- partition_cold_restart
- setPriorityofThread
- partition_cold_restart
- setPriorityofThread
- INITIALISATION
- SETUP_CONTEXT
- (uninitialised state)

Pro B

英 月 ; 人 下 21:10
2013/11/8

正确性验证(2)——模型检测



Model Check: - □ ×

Model Check Stop

Find Deadlocks
 Find Invariant Violations
 Find B THEOREM Violations
 Find GOAL (from DEFINITIONS)
 Stop when All Operations Covered
 Search for New Errors

Checked States: 6411 (22 %)
Total Distinct States: 28619
Total Transitions: 91316
Checking...

LTL model checking

Enter an LTL formula
use {...} for B predicates,
G,F,X,U,W,R,true/false/not,&,or and => are part of the supported LTL syntax,
use e(op) to check if an operation op is enabled,
use sink to check if no operation is enabled that leads to another state,
use brackets to check what is the next operation, e.g. [reset] => X{db={}},
Past-LTL is supported: Y,H,O,S,T are the duals to X,G,F,U,R

Max no. of new states: 1000
 Start search in initialisation
 Start search in current state
 Start in initialisation, but check formula in current state

OK Cancel

CTL model checking

Enter a CTL formula
use {...} for B predicates,
ExUy,EXx,AXx,EFx,AGx are supported CTL syntax,
use e(op) to check if an operation op is enabled,
use EX[Op]x to check what is the next operation, e.g. EX[reset]{db={}}

Max no. of new states: 1000
 Start search in initialisation
 Start search in current state

OK Cancel



正确性验证(3)——演绎推理(Deductive reasoning)

- 正确性的必要条件
 - 一个模型是否正确，其必要条件由**证据义务(proof obligation)**给出
 - **证据义务**：就是一系列需证明的命题
 - 对于一个模型，**证据义务**可由Event-B工具自动生成
- 对于一个**Machine**
 - 事件的可行性**Feasibility**
 - 事件的不变式保持**Invariant preservation**
- 对于一个**Refined Machine**
 - 事件的可行性**Feasibility**
 - 事件的不变式保持**Invariant preservation**
 - **Machine**的无死锁**deadlockfree**
 - **事件的求精正确性Correct refinement**
 - 事件的卫式条件被加强或保持不变
 - 事件的动作模拟被求精事件的动作（求精后每个迁移都被抽象模型所允许）
 - 使用**关联不变式(gluing invariant)**来连接新旧模型的状态



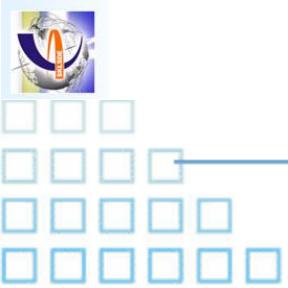
Event-B模型的Proof Obligation类型

generated in contexts		
well-definedness of an axiom axiom as theorem	label/WD label/THM	3.2.5 3.2.6
generated for machine consistency		
well-definedness of an invariant invariant as theorem	label/WD label/THM	3.2.5 3.2.6
well-definedness of a guard guard as theorem	event/guardlabel/WD event/guardlabel/THM	3.2.5 3.2.6
well-definedness of an action feasibility of a non-det. action	event/actionlabel/WD event/actionlabel/FIS	3.2.5 3.2.4
invariant preservation	event/invariantlabel/INV	3.2.4
generated for refinements		
guard strengthening action simulation	event/abstract_grd_label/GRD event/abstract_act_label/SIM	3.2.4 3.2.4
equality of a preserved variable guard strengthening (merge)	event/variable/EQL event/MRG	3.2.4 3.2.4
well definedness of a witness feasibility of a witness	event/identifier/WWD event/identifier/WFIS	3.2.5 3.2.4
generated for termination proofs		
well definedness of a variant finiteness for a set variant	VWD FIN	3.2.5 3.2.4
natural number for a numeric variant decreasing of variant	event/NAT event/VAR	3.2.4 3.2.4



Well-definedness PO (WD)

$\text{inter}(S)$	$S \neq \emptyset$
$\cap x \cdot x \in S \wedge P(x) \mid T(x)$	$\exists x \cdot x \in S \wedge P(x)$
$f(E)$	f is a partial function $E \in \text{dom}(f)$
E/F	$F \neq 0$
$E \bmod F$	$F \neq 0$
$\text{card}(S)$	$\text{finite}(S)$
$\min(S)$	$S \subseteq \mathbb{Z}$ $\exists x \cdot x \in \mathbb{Z} \wedge (\forall n \cdot n \in S \Rightarrow x \leq n)$
$\max(S)$	$S \subseteq \mathbb{Z}$ $\exists x \cdot x \in \mathbb{Z} \wedge (\forall n \cdot n \in S \Rightarrow x \geq n)$



Context Theorem PO (THM)

```
context
  ctx
extends
...
sets
  s
constants
  c
axioms
  A(s, c)
theorems
...
  thm : P(s, c)
...
end
```

s: sets
c: constants
 $A(s, c)$: axioms和被证明的theorems
 $P(s, c)$: 特殊的theorem

Axioms	\vdash	thm/THM
Theorem		

$A(s, c)$
 \vdash
 $P(s, c)$

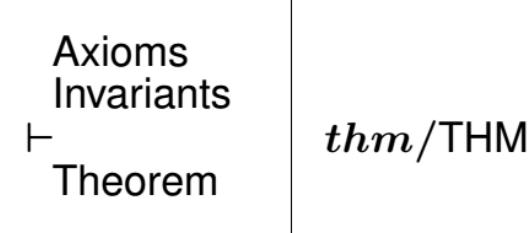


Machine Theorem PO (THM)

```
machine  
  m0  
refines  
  ...  
sees  
  ...  
variables  
  v  
invariants and thms.  
  I(s, c, v)  
theorems  
  ...  
thm : P(s, c, v)  
  ...  
events  
  ...  
end
```

s : sets
 c : constants
 v : variables
 $A(s, c)$: axioms 和 theorems
 $I(s, c, v)$: invariants 和 被证明的 theorems
 $P(s, c, v)$: 特殊的 theorem

$A(s, c)$
 $I(s, c, v)$
 \vdash
 $P(s, c, v)$



Invariant Preservation (INV)

```
evt
any x where
  G(x, s, c, v)
then
  v :| BAP(x, s, c, v, v')
end
```

s :	sets
c :	constants
v :	variables
$A(s, c)$:	axioms和theorems
$I(s, c, v)$:	invariants和theorems
x :	事件的参数
$G(x, s, c, v)$:	事件的正式条件
$BAP(x, s, c, v, v')$:	事件的前后谓词
$inv(s, c, v')$:	该事件内修改的特殊不变式

Axioms Invariants Guards of the event Before-after predicate of the event ⊤ Modified Specific Invariant	$evt/inv/INV$
----------------------------------------------------------------------------------------------------------------------------	---------------

$A(s, c)$
 $I(s, c, v)$
 $G(x, s, c, v)$
 $BAP(x, s, c, v, v')$
⊤
 $inv(s, c, v')$



Feasibility PO (FIS)

```
evt
any x where
  G(x, s, c, v)
then
  v :| BAP(x, s, c, v, v')
end
```

s :	sets
c :	constants
v :	variables
$A(s, c)$:	axioms和theorems
$I(s, c, v)$:	invariants和theorems
x :	事件的参数
$G(x, s, c, v)$:	事件的正式条件
$BAP(x, s, c, v, v')$:	事件的前后谓词

Axioms Invariants Guards of the event \vdash $\exists v'. \text{Before-after predicate}$	$evt/act/FIS$
--------------------------------------------------------------------------------------------------------	---------------

$$\begin{array}{l} A(s, c) \\ I(s, c, v) \\ G(x, s, c, v) \\ \vdash \\ \exists v'. BAP(x, s, c, v, v') \end{array}$$



Guard Strengthening PO (GRD)

```
evt0
any
x
where
g(x, s, c, v)
...
then
...
end
```

```
evt
refines
evt0
any
y
where
H(y, s, c, w)
with
x : W(x, y, s, c, w)
then
...
end
```

$s:$	sets
$c:$	constants
$v:$	抽象variables
$w:$	具体variables
$A(s, c):$	axioms和theorems
$I(s, c, v):$	抽象invariants和theorems
$J(s, c, v, w):$	具体invariants和theorems
$x:$	抽象事件的参数
$y:$	具体事件的参数
$g(x, s, c, v):$	抽象事件的卫式条件
$H(y, s, c, w):$	具体事件的卫式条件
$W(x, y, s, c, w):$	具体事件的witness谓词

Axioms
Abstract invariants and thms.
Concrete invariants and thms.
Concrete event guards
witness predicate

Abstract event specific guard

evt/grd/GRD

$A(s, c)$
 $I(s, c, v)$
 $J(s, c, v, w)$
 $H(y, s, c, w)$
 \vdash
 $g(x, s, c, v)$

Simulation PO (SIM)

evt0
any
 x
where
 \dots
then
 $v : | BA1(v, v', \dots)$
end

evt
refines
 evt0
any
 y
where
 $H(y, s, c, w)$
with
 $x : W1(x, y, s, c, w)$
 $v' : W2(y, v', s, c, w)$
then
 $w : | BA2(w, w', \dots)$
end

$s:$	sets
$c:$	constants
$v:$	抽象variables
$w:$	具体variables
$A(s, c):$	axioms和theorems
$I(s, c, v):$	抽象invariants和theorems
$J(s, c, v, w):$	具体invariants和theorems
$x:$	抽象事件的参数
$y:$	具体事件的参数
$BA1(v, v'):$	抽象事件的动作
$BA2(w, w'):$	具体事件的动作

$A(s, c)$
 $I(s, c, v)$
 $J(s, c, v, w)$
 $W1(x, y, s, c, w)$
 $W2(y, v', s, c, w)$
 $BA2(w, w')$
 \vdash
 $BA1(v, v')$

Axioms
Abstract invariants and thms.
Concrete invariants and thms.
Concrete event guards
witness predicate
witness predicate
Concrete before-after predicate
 \vdash
Abstract before-after predicate

evt/act/SIM

Grd. Strengthening PO (MRG)

when Merging Abs. Events

```
evt01  
any  
x  
where  
G1(x, s, c, v)  
then  
A  
end
```

```
evt02  
any  
x  
where  
G2(x, s, c, v)  
then  
A  
end
```

```
evt  
refines  
evt01  
evt02  
any  
x  
where  
H(x, s, c, v)  
then  
A  
end
```

s : sets
 c : constants
 v : variables
 $A(s, c)$: axioms和theorems
 $I(s, c, v)$: invariants和theorems
 x : 事件的参数
 $H(x, s, c, v)$: 具体事件的卫式条件
 $G1(x, s, c, v)$: 抽象事件1的卫式条件
 $G2(x, s, c, v)$: 抽象事件2的卫式条件
 A : 事件的动作

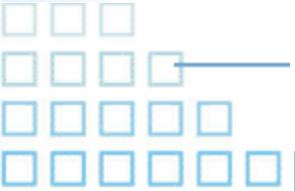
Axioms
Abstract invariants and thms.
Concrete event guards
 \vdash
Disjunction of abstract guards

evt/MRG

$A(s, c)$
 $I(s, c, v)$
 $H(x, s, c, v)$
 \vdash
 $G1(x, s, c, v) \vee G2(x, s, c, v)$

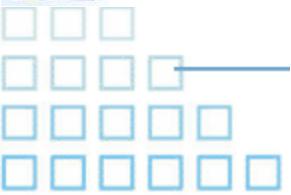


4. Event-B实践



RODIN

- RODIN
 - Rigorous Open Development Environment for Complex Systems
 - RODIN是一个建模开发环境
 - 基于Eclipse平台
 - 采用定理证明作为主要的验证技术
 - 以Event-B作为主要形式化语言
- 受欧盟项目的支持
 - RODIN project (2004-2007)
 - DEPLOY project (2008-2012)
 - ADVANCE project (2011-2014)



RODIN平台界面

The screenshot displays the Rodin Platform interface for editing Event-B models. The main window shows an Event-B specification with the following code:

```
theorem @inv5 card(partitionstate > {PM_NORMAL}) ≥ 1
@inv6 card(threadstate > {TS_Running}) ≤ 1

events
event INITIALISATION
then
@act1 partitionstate = Partitions × {PM_NORMAL}
@act3 current_thread, Threads, threadsOfPartition, threadstate :| Threads' ⊑ AllThreads × BOOL ∧ (vt:t ∈ current_thread' ⊑ AllThreads ∧ Threads(current_thread') = TRUE ∧ card(Threads'-{[TRUE]}) ≥ card(Partitions) ∧ threadsOfPartition' ⊑ AllThreads × Partitions ∧ dom(threadsOfPartition') = Threads'-{[TRUE]} ∧ ran(threadsOfPartition') = {PM_NORMAL} ∧ (vt:t ∈ Threads'-{[TRUE]}) = card({t} < threadsOfPartition') = 1) ∧
threadstate' = ((AllThreads \ {current_thread'}) × {TS_Dormant}) ∪ {current_thread' - TS_Running}
@act4 current_partition :∈ Partitions
end

event schedule
any p t
where
@axm1 p ∈ Partitions
@axm11 t ∈ AllThreads ∧ Threads(t) = TRUE
@axm2 threadsOfPartition(t) = p
@axm3 partitionstate(p) = PM_NORMAL
then
@act1 threadstate = (threadstate ∩ (threadstate - {[TS_Running]} × {TS_Ready})) ∪ {t - TS_Running}
@act5 current_partition = p
@act6 current_thread = t
/* Threads × {TSTATE_INACTIVE} // @act3 threadstate(currentthread) = TSTATE_INACTIVE
@act4 threadstate(t) = TSTATE_ACTIVE
```

The interface includes several toolbars and panes:

- Event-B Explorer**: Shows a tree view of the project structure, including files like 93_HV_1, ARINC653_Kernel_v1, and VKernel2.
- Toolbar**: Standard file operations (File, Edit, Navigate, Search), project management (Project, Run, ProB, BMotion Studio), and help.
- Central Editor Area**: Displays the Event-B code.
- Outline**: Shows a hierarchical outline of the model elements.
- Rodin Problems**: A table for managing problems.
- Sym**: A symbol palette.

RODIN编辑器

Event-B - VKernel2/Kernel_Machine.bum - Rodin Platform

File Edit Navigate Search Project Run Rename ProB BMotion Studio Window Help

Event-B Explorer

M Kernel_Machine

Kernel_Machine

SEES

- Kernel_ctxt

VARIABLES

- Threads Physical Unit: Inferred Physical Unit: private >
- threadsOfPartition Physical Unit: Inferred Physical Unit: private >
- partitionstate Physical Unit: Inferred Physical Unit: private >
- threadstate Physical Unit: Inferred Physical Unit: private >currentp
- current_partition Physical Unit: Inferred Physical Unit: private >
- current_thread Physical Unit: Inferred Physical Unit: private >
- clock_tick Physical Unit: Inferred Physical Unit: private >
- lock_level Physical Unit: Inferred Physical Unit: private >denotes
- start_condition Physical Unit: Inferred Physical Unit: private >deno
- error_thread Physical Unit: Inferred Physical Unit: private >for
- basepriority_threads Physical Unit: Inferred Physical Unit: priva
- period_threads Physical Unit: Inferred Physical Unit: private >Iden
- timecapacity_threads Physical Unit: Inferred Physical Unit: priva
- deadline_threads Physical Unit: Inferred Physical Unit: private >
- currentpriority_threads Physical Unit: Inferred Physical Unit: priva
- deadlinetime_threads Physical Unit: Inferred Physical Unit: priva
- remain_timecapacity_threads Physical Unit: Inferred Physical Unit: P
- wakeuptime_threads Physical Unit: Inferred Physical Unit: private >
- need_reschedule Physical Unit: Inferred Physical Unit: private >indi
- preemptive Physical Unit: Inferred Physical Unit: private >indicate

INVARIANTS

- inv_threadstype: Threads \in AllThreads \rightarrow BOOL not theorem >true: the
- inv_partstate: partitionstate \in Partitions \rightarrow PartitionModes not theor
- inv_threadofpart: threadsOfPartition \in AllThreads \leftrightarrow Partitions not
- inv_threadstate: threadstate \in AllThreads \rightarrow ThreadStates not theor
- inv_currepart: current_partition \in Partitions not theorem >
- inv_currthread: current_thread \in AllThreads \wedge Threads(current_thread) =

- inv_atleastoneunningthread: card(threadstate > {TS_Running}) \leq 1 no
- inv_clocktick: clock_tick \in N not theorem >
- inv_locklevel: lock_level \in Partitions \rightarrow N not theorem >
- inv_start_condition: start_condition \in Partitions \rightarrow PartitionStartC

Open With
Show In
[Child]-> Add Invariant
[Child]-> Add Refines Machine Relationship
[Child]-> Add Variable
[Child]-> Add Event
[Child]-> Add Variant
[Child]-> Add Sees Context Relationship
[Sibling]-> Add Invariant
[Sibling]-> Add Refines Machine Relationship
[Sibling]-> Add Variable
[Sibling]-> Add Event
[Sibling]-> Add Variant
[Sibling]-> Add Sees Context Relationship

Refresh F5
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Move Up Alt+Up
Move Down Alt+Down
Preferences...

Rodin Problems Properties Tasks Error Log

Sym Prog Rodi Stati

Building workspace

Processing PO for In...Mgr_Trans_Error/INV

Building workspace: (44%)

Camille编辑器(插件)

The screenshot shows the Rodin Platform interface with the Camille editor plugin installed. The main workspace displays an Event-B specification for a Kernel Machine. The left sidebar shows the Event-B Explorer with various projects and files, including 'Kernel_Machine' which is currently selected. The central area contains the event text:

```
machine Kernel_Machine sees Kernel_ctx

variables Threads
    threadsOfPartition // threads of each partition
    partitionstate
    threadstate // currentpartition currentthread
    current_partition // the partition in which a thread is now running. at each time, only one thread is running
    current_thread clock_tick
    lock_level // denotes the current lock level of the partition
    start_condition // denotes the reason the partition is started
    error_thread // for health monitoring, a error handler process could be created in a partition
    basepriority_threads // Denotes the capability of the process to manipulate other processes.
    period_threads // Identifies the period of activation for a periodic process. A distinct and unique value is assigned to each periodic process
    timercapacity_threads // Defines the elapsed time within which the process should complete its execution
    deadline_threads // Specifies the type of deadline relating to the process, and may be "hard" or "soft".
    currentpriority_threads // Defines the priority with which the process may access and receive resources.
    deadlinetime_threads // The deadline time is periodically evaluated by the operating system to determine if a process has met its deadline
    remain_timecapacity_threads
    wakeuptime_threads
    need_reschedule // indicate the flag to reschedule after some events, for example suspend a thread
    preemptive // indicate the system is preemptive or not

invariants
```

The right sidebar shows the Outline view, listing all the variables defined in the machine. The bottom right corner shows a progress bar for 'Building workspace'.

Proof Obligation自动生成

The screenshot shows the Rodin Platform interface for the Event-B methodology. The top menu bar includes File, Edit, Navigate, Search, Project, Run, ProB, BMotion Studio, Window, and Help. The title bar indicates "Event-B - VKernel2/Kernel_Machine.bum - Rodin Platform".

The left pane is the "Event-B Explorer" showing a tree structure of the model:

- Kernel_Machine
 - Variables
 - Invariants
 - Events
 - Proof Obligations
 - inv_currthread/WD
 - inv_atleastonecurrentthread/WD
 - inv_errthread2/WD
 - INITIALISATION/inv_threadtype/INV
 - INITIALISATION/inv_partstate/INV
 - INITIALISATION/inv_threadofpart/INV
 - INITIALISATION/inv_threadstate/INV
 - INITIALISATION/inv_currthread/INV
 - INITIALISATION/inv_atleastonecurrentthread/INV
 - INITIALISATION/inv_clocktick/INV
 - INITIALISATION/inv_locklevel/INV
 - INITIALISATION/inv_start_condition/INV
 - INITIALISATION/inv_error_thread/INV
 - INITIALISATION/inv_errthread2/INV
 - INITIALISATION/inv_basepriority_threads/INV
 - INITIALISATION/inv_period_threads/INV
 - INITIALISATION/inv_timecapacity_threads/INV
 - INITIALISATION/inv_deadline_threads/INV
 - INITIALISATION/inv_currentpriority_threads/INV
 - INITIALISATION/inv_deadlinetime_threads/INV
 - INITIALISATION/inv_remain_timecapacity_threads/INV
 - INITIALISATION/inv_wakeuptime_threads/INV
 - INITIALISATION/act3/WD
 - INITIALISATION/act3/FIS
 - ticktock/inv_clocktick/INV
 - schedule/grd11/WD
 - schedule/grd2/WD
 - schedule/grd4/WD
 - schedule/grd3/WD
 - schedule/grd6/WD
 - schedule/inv_threadstate/INV
 - schedule/inv_currthread/INV
 - schedule/inv_atleastonecurrentthread/INV

machine Kernel_Machine sees Kernel_ctx

variables Threads

threadsOfPartition // threads of each partition
partitionstate
threadstate // currentpartition currentthread
current_partition // the partition in which a thread is now running. at each time, only current_thread clock_tick
lock_level // denotes the current lock level of the partition
start_condition // denotes the reason the partition is started
error_thread // for health monitoring, a error handler process could be created in a partition
basepriority_threads // Denotes the capability of the process to manipulate other processes
period_threads // Identifies the period of activation for a periodic process. A distinct activation time
timecapacity_threads // Defines the elapsed time within which the process should complete its execution
deadline_threads // Specifies the type of deadline relating to the process, and may be absolute or relative
currentpriority_threads // Defines the priority with which the process may access and use system resources
deadlinetime_threads // The deadline time is periodically evaluated by the operating system
remain_timecapacity_threads wakeuptime_threads
need_reschedule // indicate the flag to reschedule after some events, for example suspend/resume
preemptive // indicate the system is preemptive or not

invariants

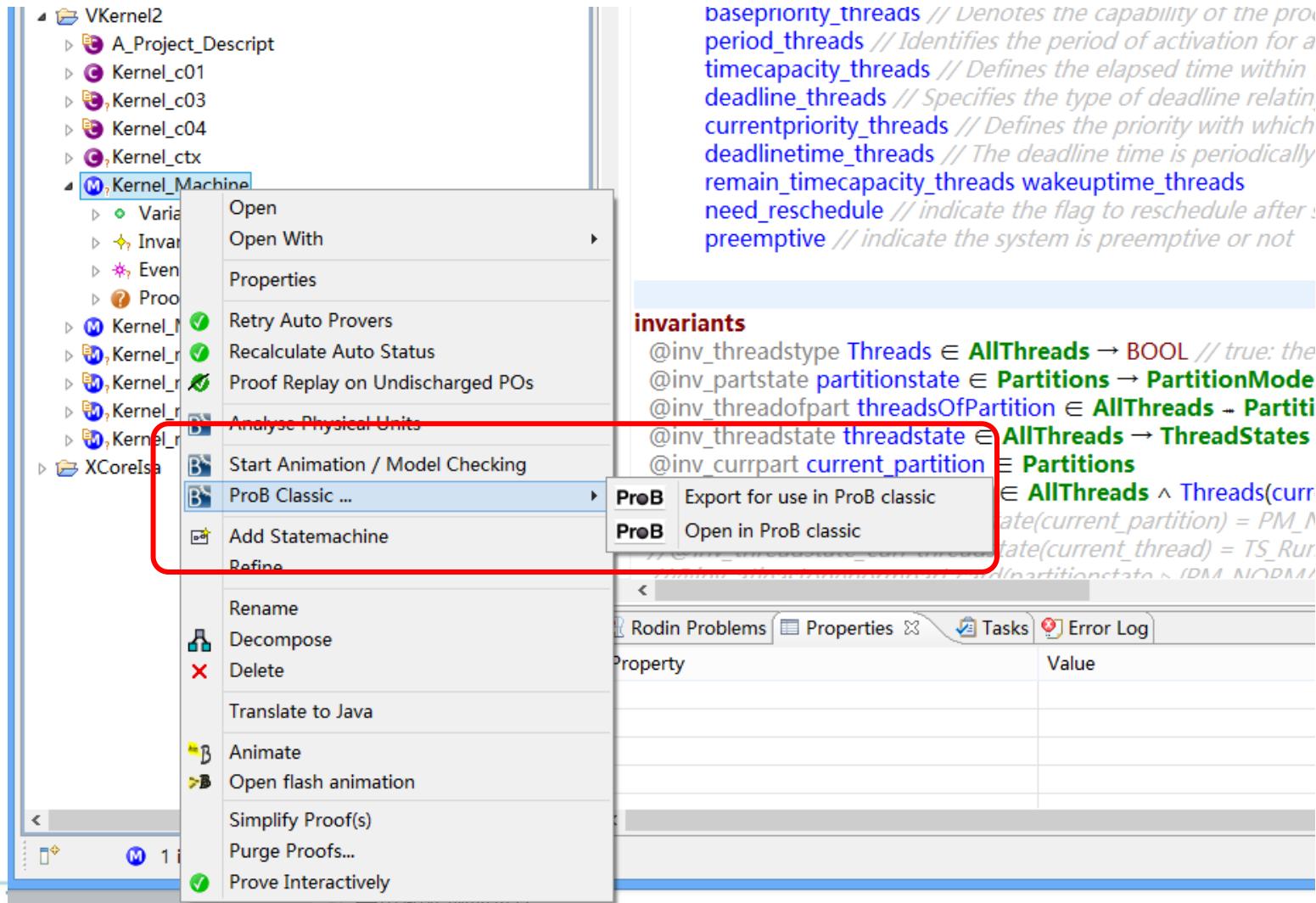
@inv_threadtype Threads ∈ AllThreads → BOOL // true: the thread has been created, false otherwise
@inv_partstate partitionstate ∈ Partitions → PartitionModes
@inv_threadofpart threadsOfPartition ∈ AllThreads -> Partitions // partial surjection, each thread belongs to exactly one partition
@inv_threadstate threadstate ∈ AllThreads → ThreadStates
@inv_currpart current_partition ∈ Partitions
@inv_currthread current_thread ∈ AllThreads ∧ Threads(current_thread) = TRUE ∧ threadsOfPartition(current_thread) = current_partition
//@inv_partstate_curr partitionstate(current_partition) = PM_NORMAL
//@inv_threadstate_curr threadstate(current_thread) = TS_Running
//@inv_atleastonecurrentpartition cardinality(partitionstate < IDMA_N / DMAX / 11 > 1

Rodin Problems Properties Tasks Error Log

Property Value



使用ProB工具



ProB:模拟和模型检测

ProB - Rodin Platform

File Edit Navigate Search Project Run ProB BMotion Studio Window Help

Events X Checks X State X Ltl Counter-Example

Model Checking LTL Model Checking Constraint Based Checking

tickclock schedulers partition_cold_restart (x2) partition_warm_restart (x2) partition_go partition_shutdown (x2) setPreemptive (x2) setPriorityOfThread (x2) thread_suspend_self thread_suspend thread_suspendonwaiting thread_resume thread_resumeonwaiting thread_stop thread_stonself

Kernel ctx Critical_Level PartitionTable _Table PartitionTable majorFrame schedulingFrameOfPa schedulingFrames

Deadlock Freedom Invariant Preservation Refinement Proof Obligations Context Theorems

Kernel Machine Threads basepriority_thre clock_tick current_partition current_thread currentpriority_t deadline_threads deadlineTime_thre error_thread lock_level need_reschedule partitionstate period_threads preemptive remain_timecapacity start_condition threadsOfPartition threadstate timecapacity_thre wakeupTime_thread

Variables Invariants Events Proof Obligations

Kernel_Machine1 Kernel_m01 Kernel_m02 Kernel_m03 Kernel_m04 XCoreIsa

invariant ok no event errors detected

History X Event Error View

Kernel_Machine INITIALISATION SETUP_CONTEXT (uninitialised state)

常量、变量、不变式等
模型检测
可执行、不可执行事件
执行的历史轨迹
错误跟踪区

Building workspace: (97%)

RODIN的Proving Perspective

证明树
已选择的hypotheses
证明目标
证明控制区
可供证明用的hypotheses

R

File Edit Navigate Search Project Run ProB BMotion Studio Window Help

Proof Tree G Kernel_Machine Kernel_Machine *Celebrity_1 Celebrity_1 remove_1/inv2/INV

Event-B Explorer

Variables Invariants Events Proof Obligations \wedge^A INITIALISATION/inv1/INV \wedge^A INITIALISATION/act1/FIS \wedge^A celebrity/inv1/INV

Celebrity_1 Variables Invariants Events Proof Obligations \wedge^A INITIALISATION/inv1/INV \wedge^A INITIALISATION/inv2/INV \wedge^A celebrity/act1/SIM \wedge^A remove_1/inv1/INV \wedge^A remove_1/inv2/INV \wedge^A remove_2/inv1/INV \wedge^A remove_2/inv2/INV

Celebrity_2 Celebrity_3 DepSatSpec015Model000 Doors doors_ctx1 doors ctx2

Proof Control Statistics Rodin Problems

Search hypotheses



一个简单的证明过程

应用程序工具

Proving - Celebrity/Celebrity_1.bps - Rodin Platform

Plate Screen Recorder

R

File Edit Navigate Search Project Run ProB BMotion Studio Window Help

Proof Tree G remove_1/inv2/INV

Kernel_Machine Kernel_Machine *Celebrity_1 Celebrity_1

Event-B Explorer

Variables Invariants Events Proof Obligations

- ✓^AINITIALISATION/inv1/INV
- ✓^AINITIALISATION/act1/FIS
- ✓^Acelebrity/inv1/INV

Celebrity_1 Variables Invariants Events Proof Obligations

- ✓^AINITIALISATION/inv1/INV
- ✓^AINITIALISATION/inv2/INV
- ✓^Acelebrity/act1/SIM
- ✓^Aremove_1/inv1/INV
- remove_1/inv2/INV
- ✓^Aremove_2/inv1/INV
- ✓^Aremove_2/inv2/INV

Celebrity_2

Celebrity_3

DepSatSpec015Model000

Doors

doors_cbx1 doors_cbx2

Proof Control Statistics Rodin Problems

Tactic applied successfully

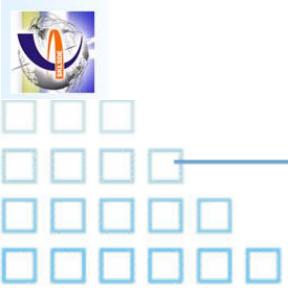
Sy Typ Sea Rule Pro

The screenshot shows the Rodin Platform interface for proving the Celebrity problem. The main window displays a proof tree for the hypothesis $c \in Q \setminus \{x\}$. The tree includes nodes for ct and $c \in Q$, $x \in Q$, $y \in Q$, and $x \mapsto y \in k$. A selected hypothesis is shown in the Goal panel: $\exists t \quad c \in Q \setminus \{x\}$. The Proof Control panel shows a red sad face icon, indicating an error or warning. The Event-B Explorer panel lists proof obligations and variables for the Celebrity_1 model, including initialisation and celebrity-related obligations. The bottom status bar indicates "Tactic applied successfully".

小结

- **本报告内容**

- 形式化方法概述
- 基本理论：命题逻辑、集合论、谓词逻辑等
- Event-B详解
- 实践



谢谢！

请批评指正！

