AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

AI驱动软件研发
全面进入数字化时代

中国·北京 08.18-19

AI+
software
Development
Digital
summit

复杂安全攸关嵌入式系统的形式设计与实现

王淑灵 中国科学院软件研究所

科技生态圈峰会 + 深度研习 ——1000 + 技术团队的共同选择

K+峰会

2023K+
全球软件研发行业创新峰会
上海站
会议时间 | 06.09-10

2023K+
全球软件研发行业创新峰会
北京站
会议时间 | 07.21-22

2024K+
全球软件研发行业创新峰会
深圳站
会议时间 | 05.17-18

K+峰会详情

AiDD峰会

会议时间 | 08.18-19
AiDD AI+软件研发数字峰会
北京站

会议时间 | 11.17-18
AiDD AI+软件研发数字峰会
深圳站

AiDD峰会详情

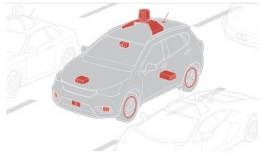
目录
CONTENTS

PART 01
背景和问题

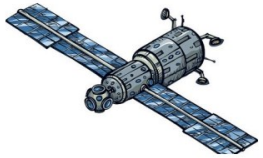
研究背景

安全攸关嵌入式系统是国民经济的重要基础和国家重大任务的核心支撑。

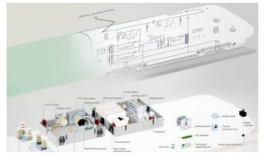
安全攸关嵌入式控制系统广泛应用于国防、交通、高端制造等关键领域，其失效可能导致人员重大伤亡、财产重大损失、环境重大破坏等灾难性后果。



automobiles



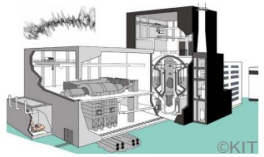
spacecrafts



high-speed rail

nuclear reactors

robot surgeon



robust control

近年，安全攸关嵌入式控制系统缺陷导致的重大事故频发，其可信性设计亟待突破

2011年“甬温”列车相撞事故

2018年狮航波音737 Max空难

安全攸关嵌入式系统的设计

如何设计安全可靠的嵌入式系统是计算机科学和控制理论的一个巨大挑战

"…the challenge of designing embedded systems offers a unique opportunity for reinvigorating computer science. The challenge, and thus the opportunity, spans the spectrum from theoretical foundations to engineering practice. To begin with, we need a mathematical basis for systems modeling and analysis which integrates both computation and physical constraints in a consistent, operative manner…."

"The Embedded Systems Design Challenge"
Invited talk at FM 2006

Joseph Sifakis
2007年图灵奖获得者
法兰西科学院荣誉研究主任

Tom Henzinger
AAAS、ACM、IEEE Fellow
欧洲科学院、德国科学院、奥地利科学院院士
奥地利科学院院长

形式建模、分析与验证是安全攸关嵌入式系统可信保障的核心技术，已经成为各种行业和国际标准的共识

基于模型的嵌入式系统形式设计（MBD）

主要挑战： 复杂性 可靠性

MBD的复杂结构

- **横向：组合/分解**
- **纵向：抽象/精化**

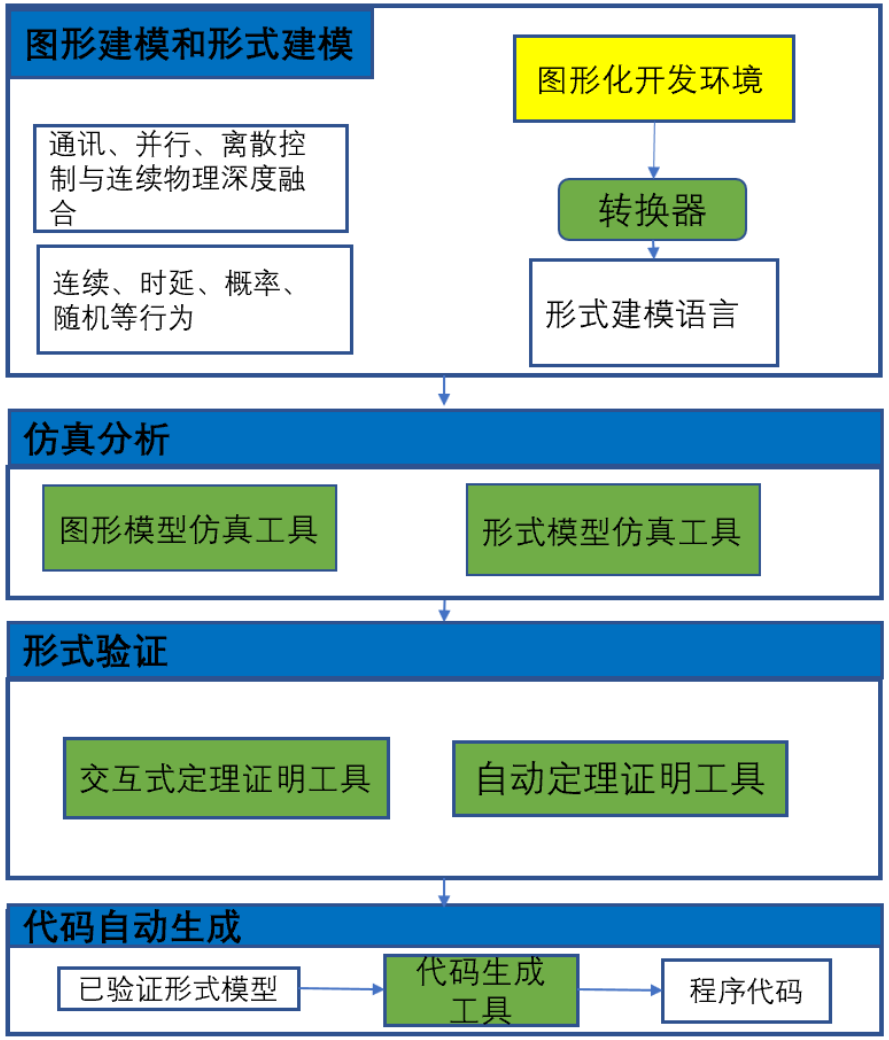
MBD的设计方法

- **基于图形建模与仿真的设计**

易使用（工程师）高效，但不完备

- **基于形式建模与验证的设计**

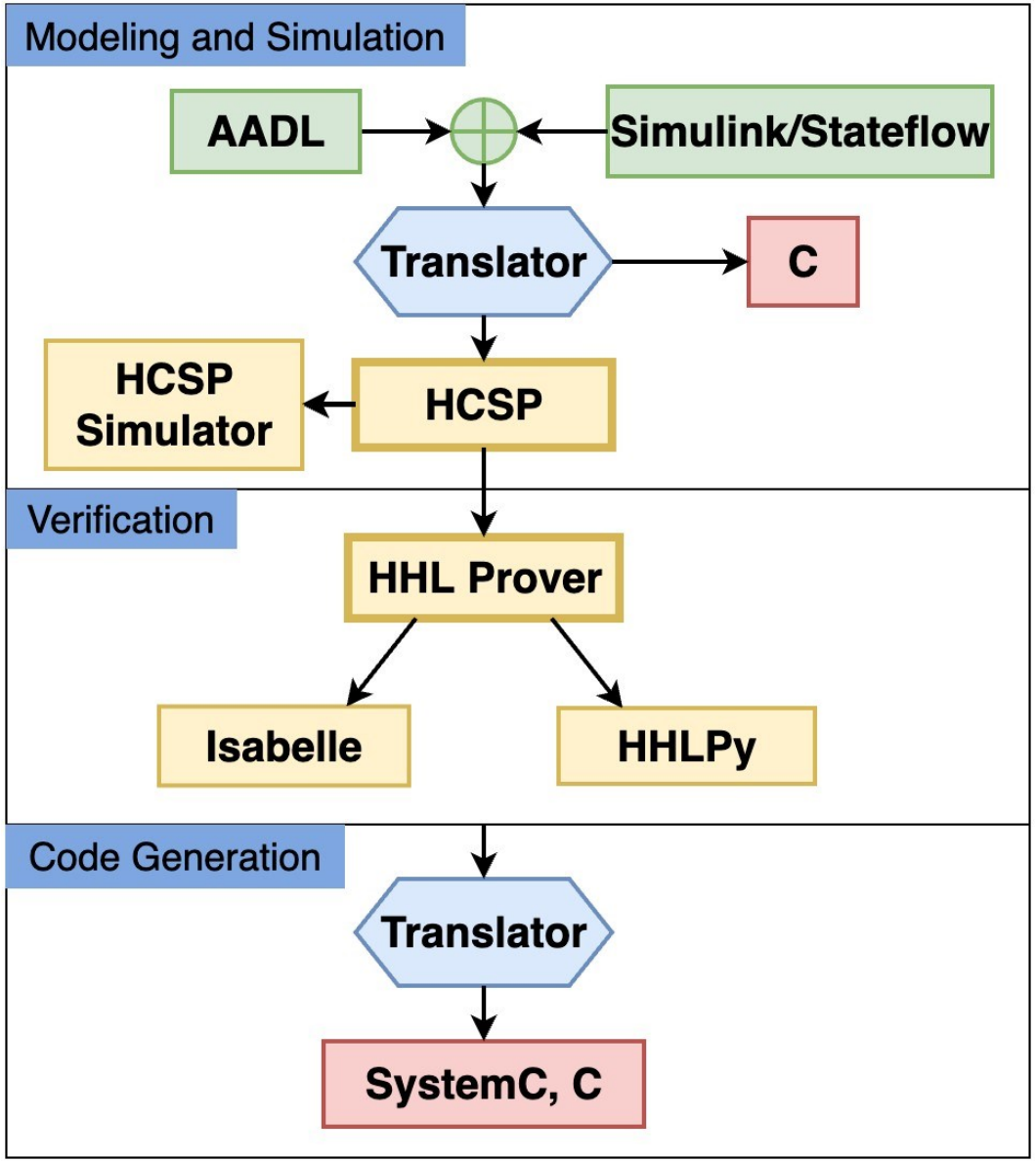
可靠，难掌握（理论专家）、代价高



我们的方法

MARS：Modelling, Analysis and verification of Hybrid Systems

研究团队：周巢尘，詹乃军，王淑灵，詹博华，徐雄等



PART 02

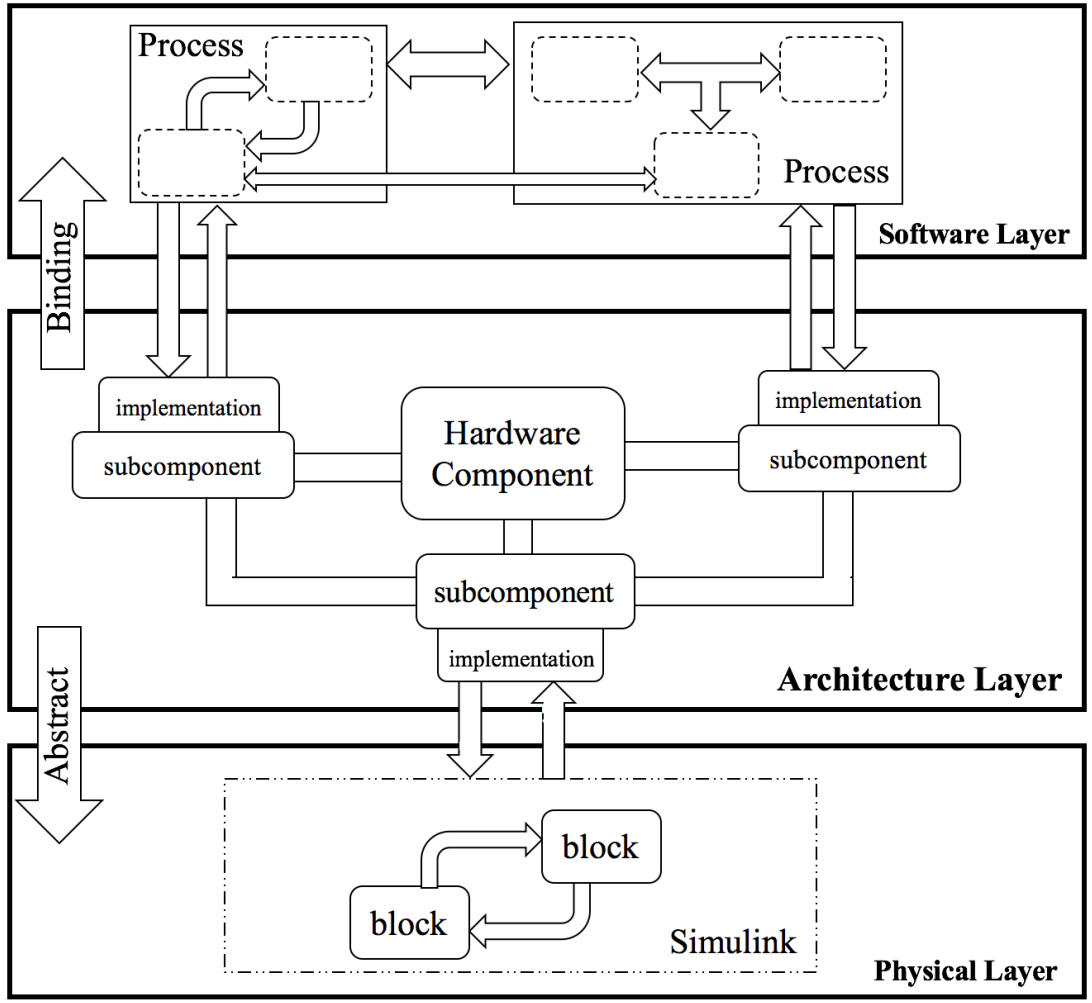
基于Simulink/Stateflow和AADL的协同设计

AADL + Simulink/Stateflow 图形化协同设计

- **嵌入式系统设计**
  - Functionality (F)：软件行为
  - Physicality (P)：物理环境
  - Architecture (A)：体系结构和硬件平台

- **AADL (F-, P-, A+)**
  - 基于模型驱动的系统架构设计语言，被广泛应用于航空航天等工程设计领域。

- **Simulink/Stateflow (F+, P+, A-)**
  - 工业界开发嵌入式系统的实际标准。
  - 建模嵌入式系统中复杂的离散和连续行为，支持高效仿真和代码生成。



AADL + Simulink/Stateflow协同建模和仿真

将嵌入式系统的软件、硬件以及物理环境在一个统一的框架下进行建模和仿真。

如何将Simulink/Stateflow模型与AADL模型进行集成?

- 将S/S模型定义为AADL的抽象类型，声明输入/输出端口。
- 定义AADL和S/S之间的端口对应关系，进行数据之间的传输。

```
abstract phy
  features
    a: in data port;
    b: out data port;
end phy
```

如何进行协同仿真?

- 实现 AADL2C转换器，将AADL模型转换为C代码进行功能和性能的仿真。
- 应用Matlab的实时工具箱RTW，将S/S模型自动生成C代码。
- 将分布式构件之间对应的端口进行连接，进行系统级别的仿真。

$$(s_1, \sigma) \xrightarrow{cn_b!\sigma(b)} (s_1, \sigma)$$

$$(s_1, \sigma) \xrightarrow{cn_a?c} (s_1, \sigma[a \mapsto c])$$

PART 03

HCSP形式建模和验证

混成通信顺序进程（HCSP）

HCSP，在通信顺序进程（Hoare，CSP）的基础上，引入用于描述连续行为的微分方程，以及用于描述连续行为和离散行为交互的中断机制。

$$
\begin{array}{lll}
P & ::= & \mathsf{skip} \mid x := e \mid \mathsf{wait}\ d \mid ch?x \mid ch!e \mid P; P \mid B \to P \mid P \sqcup P \mid P^* \mid \\
 & & \langle F(\dot{\boldsymbol{x}}, \boldsymbol{x}) = 0 \& B \rangle \mid \langle F(\dot{\boldsymbol{x}}, \boldsymbol{x}) = 0 \& B \rangle \trianglerighteq_d P \mid \\
 & & \langle F(\dot{\boldsymbol{x}}, \boldsymbol{x}) = 0 \& B \rangle \trianglerighteq [\!]_{i \in I} (io_i \to P_i) \\
S & ::= & P \mid S \| S
\end{array}
$$

其中，ch?x – 输入，ch!e – 输出， <F(x′, x) = 0& B> – 连续行为。

下面是用HCSP描述的一个连续运行的列车以及它的控制系统。

$$
\begin{array}{l}
(\langle x' = v, v' = a \& \mathbf{True} \rangle \trianglerighteq [\!](\mathrm{sensor}!\{x, v\} \to \mathrm{actuator}?a))^* \\
\quad (\mathbf{wait}\ d; \mathrm{sensor}?s; \mathrm{actuator}!\mathrm{Ctrl}(s))^*
\end{array}
$$

HCSP自动仿真器

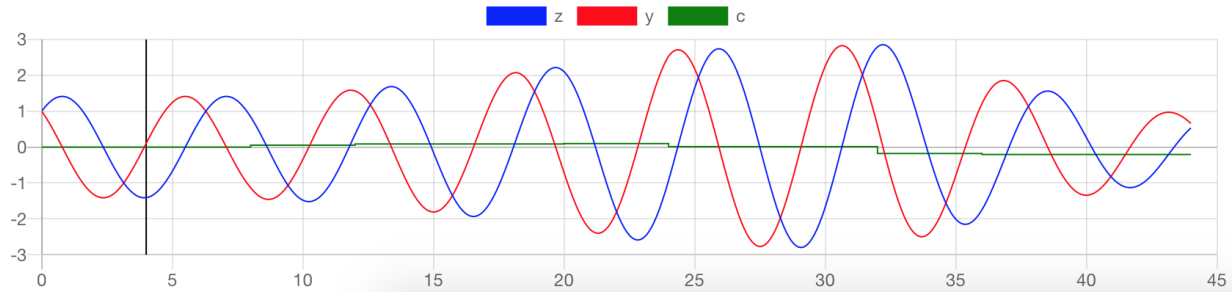
HCSP Simulator | Read HCSP File | van_der_pol.txt | Number of steps: 200 | Showing 200 | starting from 0

199+ events.

```
)**
  t: 4 z: -1.41 b: 0 c: 0  Show graph

Process: PC0

z := 1;
y := 1;
(
  <z_dot = y, y_dot = y*c-z & true> |> [] (
    ch_c_0?c -->
      skip
    ch_z!z -->
      skip
  )
)**
  z: -1.41 y: 0.103 c: 0  Hide graph
```



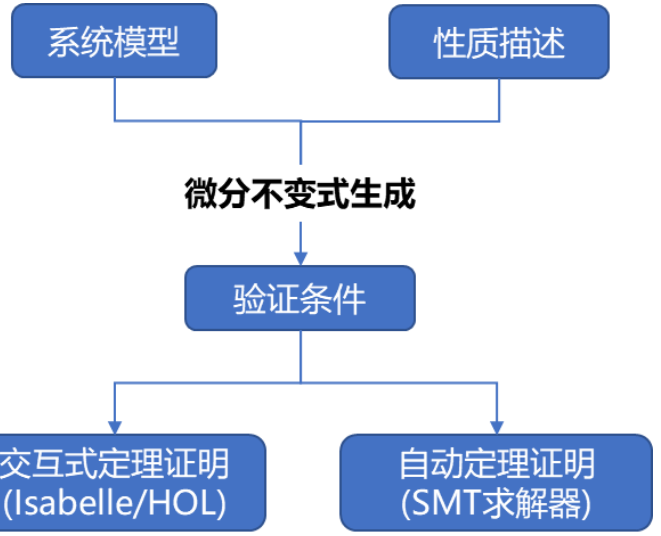

```
start
step
step
step
IO ch_z 1
step
step
step
step
step
IO ch_c_0 -0.0
step
delay 2
step
IO ch_z 0.493
step
step
step
IO ch_c_0 -0.0
step
delay 2
step
IO ch_z -1.41
step
step
```

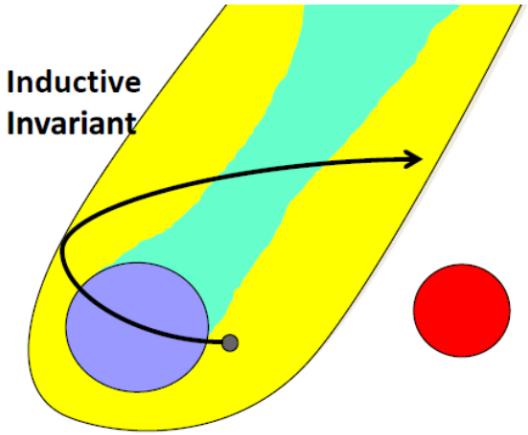
- 图形用户界面使用网页界面的形式实现。
- 展示仿真的结果，包括变量的变化曲线、事件序列等。

我们的一个目标是开发自主可控的工具软件，支持建模、自动仿真、验证、代码生成。

HCSP形式验证

- 扩充经典Hoare逻辑到混成系统，在Isabelle中实现了定理证明工具。
  - 规约{Pre} P {Post}，Pre和Post分别为P的前条件和后条件，描述通讯事件、连续行为。
  - P||Q 对应P和Q中通讯和时间的同步。
  - 微分不变式用于证明微分方程表示的连续行为，提出不变式生成方法。

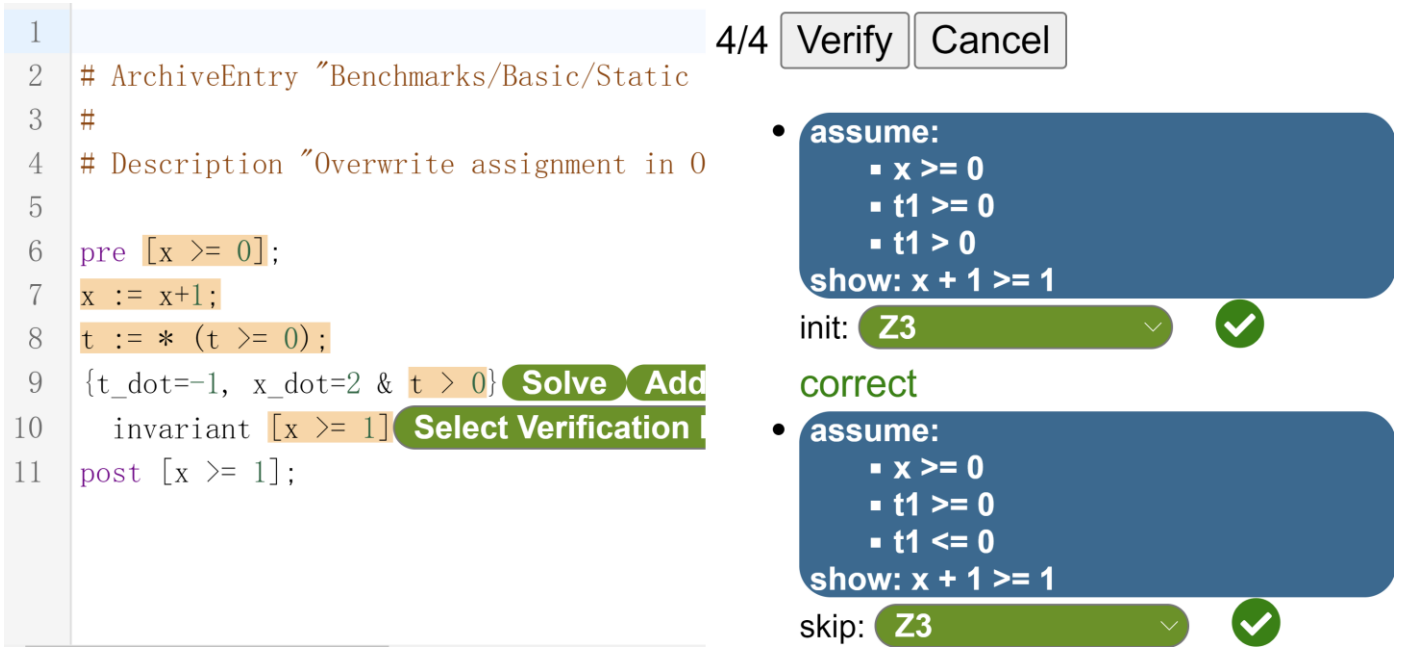




1. $Init \Rightarrow I;$
2. $I \Rightarrow [\frac{dx}{dt} = f(x)]I;$
3. $I \Rightarrow Safe$

HCSP形式验证

- 针对HCSP的顺序子集，主要是连续行为部分，实现了自动证明工具HHLPy。
- 在HCSP标注微分不变式的基础上，自动生成验证条件（一阶逻辑公式）。
- 支持外部SMT求解器Z3和Wolfram Engine的调用，自动证明逻辑公式。



AADL+Simulink/Stateflow到HCSP的转换

- 实现了从AADL+Simulink/Stateflow图形模型到HCSP形式模型的转换工具。
- 将关键模块转换到HCSP中进行形式验证。
- UTP（Unifying Theories of Programming）是Hoare和何积丰院士提出，其目标是统一不同程序的设计风格。我们提出了能够处理通讯、连续行为的高阶统一程序理论（HUTP），证明了模型转换的正确性。

给定任意一个 AADL⊕Simulink/Stateflow 图形模型 $\mathcal{D}$，那么

$$[\![\mathcal{D}]\!]_{\mathrm{HUTP}} \equiv [\![[\![\mathcal{D}]\!]_{\mathrm{HCSP}}]\!]_{\mathrm{HUTP}}$$

PART 04
已验证模型到代码的生成

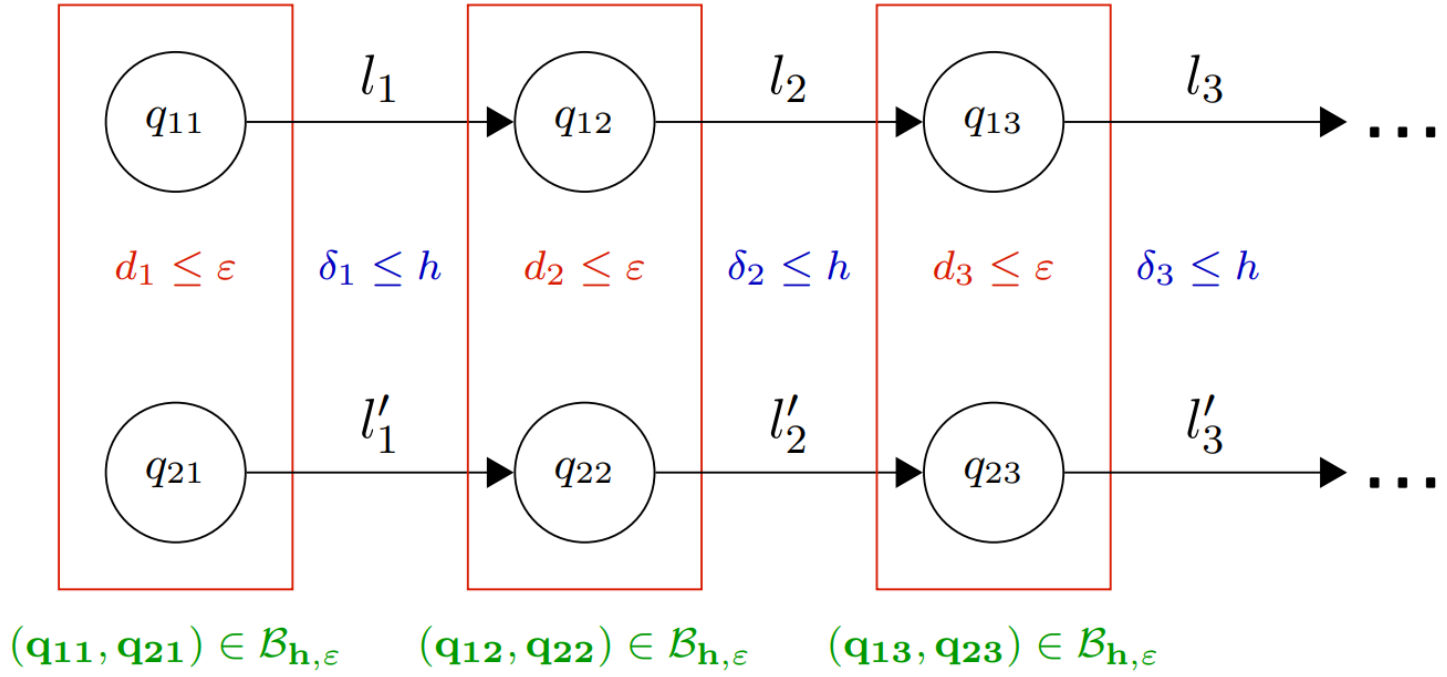
HCSP模型到代码的转换

- HCSP含有连续微分方程，通讯，以及离散和连续之间的交互。
- 程序代码仅含有离散行为。
- HCSP模型的正确性已由形式验证保证，如何确保正确性在生成的代码中依旧得到保持?

  - HCSP的离散化；
  - 建立HCSP模型和代码之间的等价关系，从而保证代码的正确性。

近似互模拟关系

允许连续控制模型和代码之间存在一定的误差，建立它们之间的近似互模拟关系。



$$d_i = \mathbf{d}(H_1(q_{1i}), H_2(q_{2i})), \text{ and } \delta_i = \mathbf{d}(l_i, l'_i)$$

连续行为的离散化

连续行为在时间上限T内的离散化

$$\frac{\langle \dot{\boldsymbol{x}} = f(\boldsymbol{x}) \& B \rangle}{\begin{array}{c}(N(B,\epsilon) \wedge N'(B,\epsilon) \to (\text{wait } h; \boldsymbol{x} := \boldsymbol{x} + h\Phi(\boldsymbol{x},h)))^{\lfloor \frac{T}{h} \rfloor}; \\ (N(B,\epsilon) \wedge N'(B,\epsilon) \to (\text{wait } h'; \boldsymbol{x} := \boldsymbol{x} + h'\Phi(\boldsymbol{x},h')); \\ N(B,\epsilon) \wedge N'(B,\epsilon) \to \text{stop}\end{array}}$$

- 使用Runge-Kutta方法，将ODE转换为一系列离散赋值，误差控制精度高。
- HCSP进程相对于误差满足健壮安全条件。
- 微分方程满足局部Lipschitz条件。

C代码的自动生成

通讯ch?x的实现

```
lock t mutex;
channelInput[chi.channelNo] = t;
int i = channelOutput[chi.channelNo];
if (i != -1 & (threadState[i] == STATE_AVAILABLE || threadState[i] == STATE_WAITING_AVAILABLE
    ))
{
  threadState[i] = chi.channelNo;
  copyFromChannel(chi);
  signal i cond[i];
  cwait t cond[t] mutex;
  threadState[t] = STATE_UNAVAILABLE;
  channelInput[chi.channelNo] = -1;
}
else
{
  threadState[t] = STATE_AVAILABLE;
  threadPermWait[t] = 1;
  cwait t cond[t] mutex;
  if (localTime[t] < localTime[channelOutput[ch.channelNo]])
  { localTime[t] = localTime[channelOutput[ch.channelNo]; }
  updateCurrentTime(t);
  copyFromChannel(ch);
  threadState[t] = STATE_UNAVAILABLE;
  threadPermWait[t] = 0;
  channelInput[ch.channelNo] = -1;
  signal channelOutput[ch.channelNo] cond[channelOutput[ch.channelNo]];
}
unlock t mutex;
```

- ✓使用lock协调多线程对同一个通道的读和写
- ✓使用signal实现多线程的值同步和时钟同步

- 实现了从HCSP各语法结构到C代码的自动转换。
- 证明了HCSP模型和所产生C代码的近似互模拟关系。

PART 05
应用

MARS形式设计

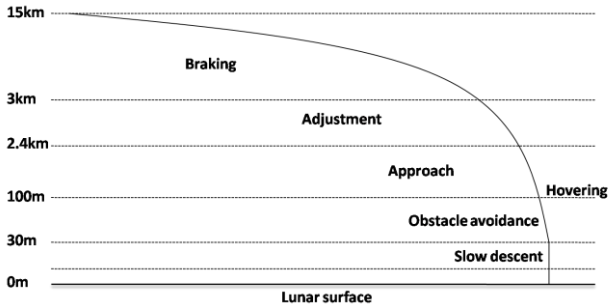
MARS形式设计整体思路：AADL+Simulink/Stateflow建模 ➡ 仿真 ➡ HCSP形式模型 ➡ （仿真）形式验证 ➡ 代码生成

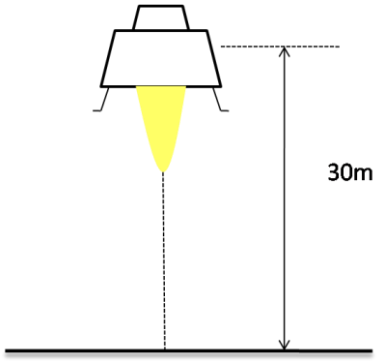
案例应用

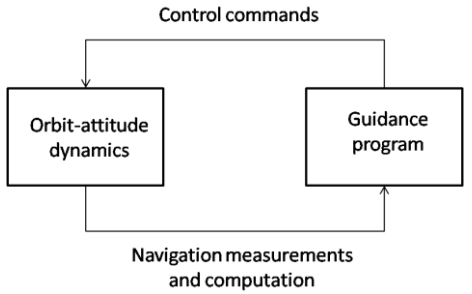
- **高速列车控制系统CTCS-3安全性验证**
- **嫦娥-3着陆器控制程序**
- **火星探测任务“天问一号”软着陆控制程序**
- **高速飞行列车运行控制系统（HCSP建模、仿真与验证）**
- **……**

嫦娥-3着陆器控制程序

任务描述







系统动力模型

$$\begin{cases} \dot{r} &= v \\ \dot{v} &= \frac{F_c}{m} - gM \\ \dot{m} &= -\frac{F_c}{Isp_1} \\ \dot{F}_c &= 0 \\ F_c &\in [1500, 3000] \end{cases}$$

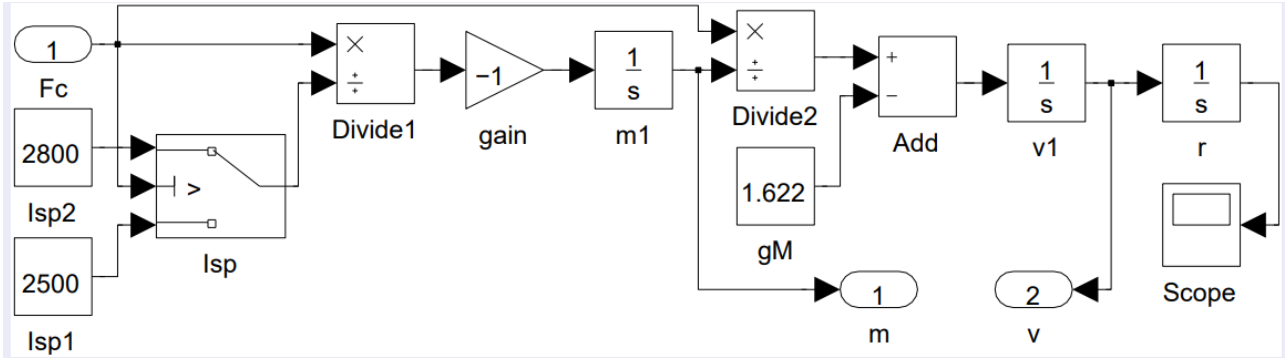
$$\begin{cases} \dot{r} &= v \\ \dot{v} &= \frac{F_c}{m} - gM \\ \dot{m} &= -\frac{F_c}{Isp_2} \\ \dot{F}_c &= 0 \\ F_c &\in (3000, 5000] \end{cases}$$

设计目标：触地前缓速下降过程保持|v + 2| ≤ 0.05m/s（安全性）

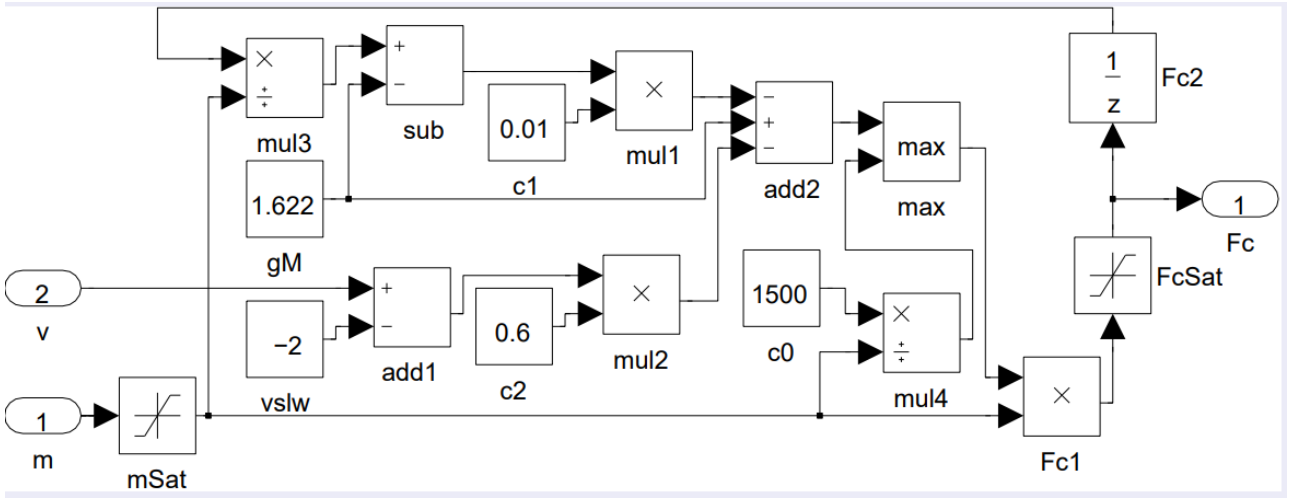
能够正确接收到信号，退出缓速下降模式（可达性）

嫦娥-3着陆器控制程序

系统动力模型

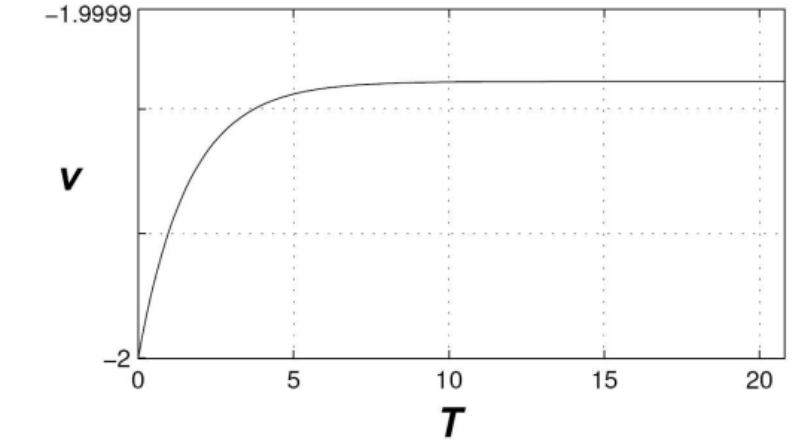


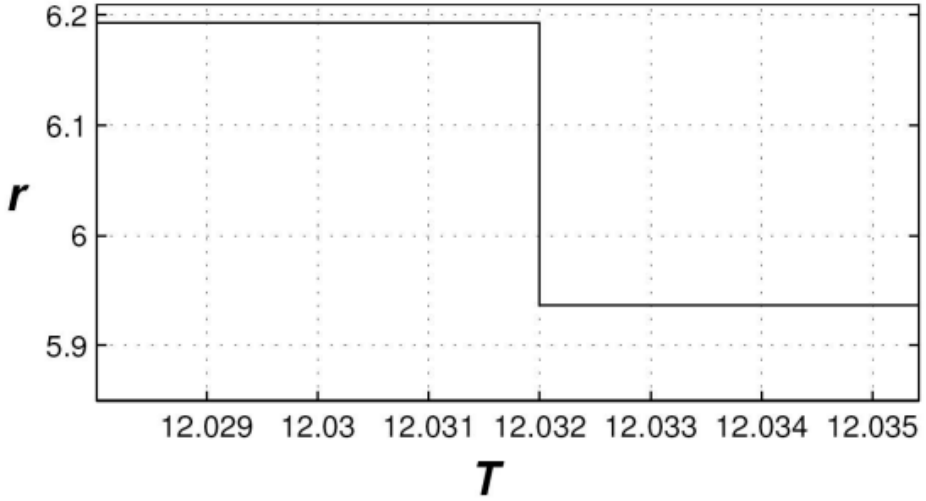
GNC程序



嫦娥-3着陆器控制程序

仿真结果





嫦娥-3着陆器控制程序

由S/S图形模型转换成HCSP形式模型

$$
\begin{array}{rcl}
P & \widehat{=} & PC \,\|\, PD \\
PC & \widehat{=} & v := -2;\ m := 1250;\ r := 30; \\
 & & (\ \langle Sys_1 \& f > 3000\rangle \trianglerighteq Comml; \\
 & & \langle Sys_2 \& f \leq 3000\rangle \trianglerighteq Comml\ )^* \\
PD & \widehat{=} & t := 0;\ g := 1.622;\ vslw := -2;\ f_1 = 2027.5; \\
 & & (\ ch_v?v_1;\ ch_m?m_1;\ f_1 := m_1 * aIC;\ ch_f!f_1; \\
 & & temp := t;\ \langle \dot{t} = 1 \& t < temp + 0.128\rangle\ )^* \\
aIC & \widehat{=} & g - 0.01 * (f_1/m_1 - g) - 0.6 * (v_1 - vslw) \\
Sys_1 & \widehat{=} & \dot{m} = -f/2548,\ \dot{v} = f/m - 1.622,\ \dot{r} = v \\
Sys_2 & \widehat{=} & \dot{m} = -f/2842,\ \dot{v} = f/m - 1.622,\ \dot{r} = v \\
Comml & \widehat{=} & ch_f?f \rightarrow \textbf{skip}\ \square\ ch_v!v \rightarrow \textbf{skip}\ \square\ ch_m!m \rightarrow \textbf{skip}
\end{array}
$$

使用MARS中的HHL验证工具，严格证明了控制程序的正确性。

正在展开的工作

建模和规约

- 实现AADL和Simulink/Stateflow的组合开发环境
- 扩展建模语言，处理信息安全和隐私、处理移动物联网等

验证

- 概率、随机、信息安全等的验证
- 不变式的高效生成，HCSP的高效验证

代码自动生成

- 生成代码的优化
- HCSP到Rust代码的生成

更多实际案例应用

AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

感 谢 聆 听