基于 SMT 的混成系统仿真与验证

方徽星

华东师范大学
国家可信嵌入式软件工程技术研究中心

2017 年 3 月 5 日

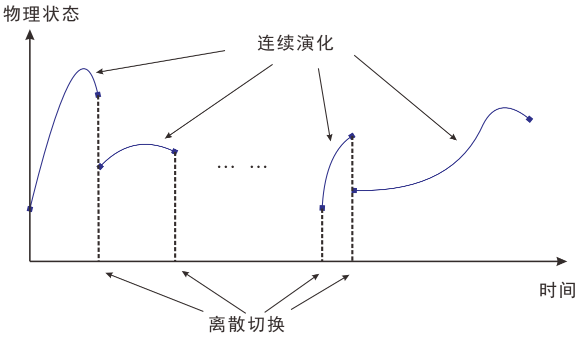
大纲

大纲

1. 建模语言
2. 模型转换
3. 原型工具
4. 案例分析
5. 总结

研究背景

- 混成系统是一种动态系统，其同时具有连续和离散的动态行为特征
- 混成系统可以作为嵌入式系统设计的基本数学模型，可应用于：汽车电子，高速列车，轨道交通，航空航天以及工业生产等领域



HML 语言

- 混成系统建模语言 HML:

$$
\begin{aligned}
AP &::= \texttt{skip} \mid v = e \mid\ !s \mid \texttt{wait}(e) \\
EQ &::= R(v, \dot{v}) \mid R(v, \dot{v})\ \texttt{init}\ v_0 \mid EQ \,||\, EQ \\
P &::= AP \mid P; P \mid \{P \,||\, P\} \mid (P \langle\ b\ \rangle\ P) \mid EQ\ \texttt{until}\ g \\
&\quad \mid \texttt{when}\{\ G\ \} \mid \texttt{while}\ (b)\{\ P\ \} \mid Id(e_s) \\
g &::= @(s) \mid b \mid g\ \langle\texttt{and}\rangle\ g \mid g\ \langle\texttt{or}\rangle\ g \\
b &::= \texttt{true} \mid \texttt{false} \mid v \circ c \mid \sim b \mid b\ \texttt{and}\ b \mid b\ \texttt{or}\ b \\
G &::= (g\ \texttt{then}\ P) \mid G, G
\end{aligned}
$$

其中，$\circ \in \{>=, >, ==, <, <=\}$

HML 语言

- 在 HML 语言中引入四种数据类型：

类型

Type ::= `Signal` | `boolean` | `int` | `float`

- 类型 `Signal` 用于声明信号变量
- 类型 `boolean` 用于布尔变量
- 类型 `int` 和 `float` 分别用于整数和实数变量

HML 语言

- 在 HML 语言中引入四种数据类型：

类型

$Type ::= \texttt{Signal} \mid \texttt{boolean} \mid \texttt{int} \mid \texttt{float}$

- 类型 `Signal` 用于声明信号变量
- 类型 `boolean` 用于布尔变量
- 类型 `int` 和 `float` 分别用于整数和实数变量

- 新增语法结构：

语法结构

为了在模型中表示变量的有界约束以及实现代码重用等功能，定义如下语法结构：

$$Constraint ::= v \ \texttt{in} \ [l, \ h] \quad (1)$$

$$Template ::= \texttt{Template} \ Id(f_s)\{ \ P \ \} \quad (2)$$

$$Main ::= \texttt{Main} \ \{ \ P \ \} \quad (3)$$

大纲

模型转换

- SMT 公式是以前缀的形式表示的，为了公式的可读性和表述的方便，以中缀的方式表示，并对某些特殊的公式以抽象的形式给出

符号约定

m ：模型转换（展开）的深度，τ ：标记变量新值，ρ ：标记变量初值

- 对于赋值语句 $x = e(\vec{v})$，可将其转换为如下公式：

$$x_{m,\tau} = e(\vec{v}_{m,\rho})$$

其中，$e(\vec{v}_{m,\rho})$ 为表达式 $e(\vec{v})$ 通过将 $\vec{v}$ 替换为初值 $\vec{v}_{m,\rho}$ 而得到的新的表达式

模型转换

- SMT 公式是以前缀的形式表示的，为了公式的可读性和表述的方便，以中缀的方式表示，并对某些特殊的公式以抽象的形式给出

符号约定

m ：模型转换（展开）的深度，τ ：标记变量新值，ρ ：标记变量初值

- 对于赋值语句 $x = e(\vec{v})$，可将其转换为如下公式：

$$x_{m,\tau} = e(\vec{v}_{m,\rho})$$

其中，$e(\vec{v}_{m,\rho})$ 为表达式 $e(\vec{v})$ 通过将 $\vec{v}$ 替换为初值 $\vec{v}_{m,\rho}$ 而得到的新的表达式

例（赋值）

赋值语句 $x = x + y + 1$ 可以转换为公式 $x_{m,\tau} = x_{m,\rho} + y_{m,\rho} + 1$，若用 SMT 公式的形式表示，则为：$(= \ x_{m,\tau} \ (+ \ x_{m,\rho} \ (+ \ y_{m,\rho} \ 1)))$

模型转换

- 信号发送

  信号发送程序 $!s$ 可以当做为一个离散赋值操作，可以将信号的发送转换为如下简单的公式：

$$s_m = global_{m,\rho}$$

其中，$global_{m,\rho}$ 代表信号 s 产生的时刻点。信号的变量名中没有下标 τ

模型转换

- 信号发送

  信号发送程序 $!s$ 可以当做为一个离散赋值操作，可以将信号的发送转换为如下简单的公式：

$$s_m = global_{m,\rho}$$

  其中，$global_{m,\rho}$ 代表信号 s 产生的时刻点。信号的变量名中没有下标 τ

- 串行组合

  对于 $P_1; P_2$，若 P_1 转换后的公式为 $\mathcal{F}_1(\vec{x}^1_{m,\tau}, \vec{x}^1_{m,\rho}) \wedge (\vec{x}^1_{m,\tau} = \vec{e}_1(\vec{x}^1_{m,\rho}))$，$P_2$ 转换后的公式为 P_2 is $\mathcal{F}_2(\vec{x}^2_{m,\tau}, \vec{x}^2_{m,\rho}) \wedge (\vec{x}^2_{m,\tau} = \vec{e}_2(\vec{x}^2_{m,\rho}))$，则有：

$$\mathcal{F}_1(\vec{e}_1(\vec{x}_{m,\rho}), \vec{x}_{m,\rho}) \wedge \mathcal{F}_2(\vec{x}_{m,\tau}, \vec{e}_1(\vec{x}_{m,\rho})) \wedge (\vec{x}_{m,\tau} = \vec{e}_2(\vec{e}_1(\vec{x}_{m,\rho})))$$

  其中，$\vec{x}_{m,\rho}$ 代表 $\vec{x}$ 的初值，公式中的上标（1 和 2）只是为了区分 P_1 和 P_2 的公式用的。$\vec{x}^1_{m,\rho} = \vec{x}_{m,\rho}$，$\vec{x}^2_{m,\rho} = \vec{x}^1_{m,\tau}$，$\vec{x}^2_{m,\tau} = \vec{x}_{m,\tau}$

模型转换

- 条件选择

条件选择语句 $(P_1 \langle\ b\ \rangle P_2)$ 可以表示成如下公式：

$$[b(\vec{x}_{m,\rho}) \wedge \vec{x}_{m,\tau} = \vec{e}_1(\vec{x}_{m,\rho})] \vee [\neg b(\vec{x}_{m,\rho}) \wedge \vec{x}_{m,\tau} = \vec{e}_2(\vec{x}_{m,\rho})]$$

$b(\vec{x}_{m,\rho})$ 代表转换后的布尔表达式公式。表达式 $\vec{e}_1$ 和 $\vec{e}_2$ 分别用于程序 P_1 和 P_2 转换后的公式。一般地，条件选择对应多个可能的执行路径

模型转换

- **条件选择**

  条件选择语句 $(P_1 \langle\ b\ \rangle\ P_2)$ 可以表示成如下公式：

$$[b(\vec{x}_{m,\rho}) \wedge \vec{x}_{m,\tau} = \vec{e}_1(\vec{x}_{m,\rho})] \vee [\neg b(\vec{x}_{m,\rho}) \wedge \vec{x}_{m,\tau} = \vec{e}_2(\vec{x}_{m,\rho})]$$

  $b(\vec{x}_{m,\rho})$ 代表转换后的布尔表达式公式。表达式 $\vec{e}_1$ 和 $\vec{e}_2$ 分别用于程序 P_1 和 P_2 转换后的公式。一般地，条件选择对应多个可能的执行路径

- **循环**

  对于循环语句 `while` $(b)\{\ P\ \}$，若用自然数 i 表示循环的总层数，则当 $i = 0$ 时，程序转换为：

$$LP_0 = \neg b(\vec{x}_{m,\rho})$$

  一般地，有 $LP_i = [b(\vec{x}_{m,\rho}) \wedge \vec{x}_{m,\tau} = \vec{e}^{\,i}(\vec{x}_{m,\rho})] \wedge [\neg b(\vec{x}_{m,\tau})]$，其中 $\vec{e}^{\,i}(\vec{x}_{m,\rho}) = \vec{e}(\vec{e}^{\,i-1}(\vec{x}_{m,\rho}))$

模型转换

- 等待

对于 `wait`$(e(\vec{x}))$，可将其转换为微分方程的形式：

$$[\langle clock_{m,\tau}\rangle] = (\mathtt{integral}\ [0, time_m]\ clock_{m,\rho}\ \ flow)$$

$[\langle\cdot\rangle]$ 用于连续变量取值的表示，微分方程 $flow =_{def} (\frac{d[clock]}{d[t]} = 1)$，$clock$ 的初值为 $clock_{m,\rho}$，$[0, time_m]$ 为积分区间，变量 $time_m$ 为连续行为的时间上界，关键字 `integral` 表示积分运算。同时还需：

$$\forall t \in [0, time_m].\neg(clock_{m,\tau} \geq e(\vec{x}_{m,\rho}))$$

表示在程序终止之前，变量 $clock$ 的值始终小于表达式 e 的取值。对于程序执行结束的情况，也需要相应的公式来表示：

$$clock_{m,\tau} \geq e(\vec{x}_{m,\rho})$$

需要注意的是，该公式没有使用量词，因为终止行为只是发生在一个时刻点上的瞬时行为

模型转换

- 微分方程

  对于微分方程 $(\dot{v} = e(v)$ $\texttt{init}$ v_0 $\texttt{until}$ $g(\vec{x}))$，可以将其转换为：

$$[\langle v_{m,\tau}\rangle] = (\texttt{integral}\ [0, time_m]\ v_{m,\rho}\ \ flow)$$

  其中，$v_{m,\rho} = v_0$，方程 $flow =_{def} (\frac{d[v]}{d[t]} = e(v))$。同样地，还有约束：

$$\forall t \in [0, time_m].(\neg g(\vec{x})_{m,\tau})$$

  不同的卫兵条件 g，其对应的约束公式的具体形式是不同的：

  - 对于信号卫兵 $@(s)$，其约束公式的具体形式为 $\neg(s_m \geq global_{m,\tau})$。若在连续行为开始时信号刚刚发出，则公式 $(s_m \geq global_{m,\tau})$ 为真，连续行为将立刻终止
  - 对于布尔条件 b，有约束公式 $\neg b(\vec{x}_{m,\tau})$

模型转换

- 等待选择

对于等待选择程序 $\texttt{when}\{(g\ \texttt{then}\ P)\}$，可以转换如下：

$$[\langle clock_{m,\tau}\rangle] = (\texttt{integral}\ [0, time_m]\ clock_{m,\rho}\ flow)$$

其中，微分方程 $flow =_{def} (\frac{d[clock]}{d[t]} = 1)$，约束公式为

$$\forall t \in [0, time_m].(\neg g(\vec{x})_{m,\tau})$$

子程序 P 将用于下一深度（$m+1$）的模型展开，当有多个子句时，可用采用动态展开的方式处理

模型转换

- 离散- 连续

  对于离散动作和连续行为的并发，有如下规则：

$$\frac{D \parallel C}{D\ ;\ C}$$

  在模型转换时，先转换离散程序然后处理连续程序的转换

- 离散- 离散

  对于两个离散动作（D_1 和 D_2）的并发，有如下规则：

$$\frac{D_1 \parallel D_2}{(D_1\ ;\ D_2) \vee (D_2\ ;\ D_1)}$$

  由于两个离散动作执行的先后是非确定的，在转换时可以使用逻辑或操作符表示

模型转换

- 连续- 连续

  对于两个连续行为（ C_1 和 C_2 ）的并发，有如下基本规则：

$$\frac{C_1 \parallel C_2}{[(\sim_{1,2})\ \texttt{until}\ g_{1,2}];[(g_1 \wedge g_2 \wedge \texttt{skip}) \vee (g_1 \wedge \neg g_2 \wedge C_2) \vee (\neg g_1 \wedge g_2 \wedge C_1)]}$$

  其中，$(\sim_{1,2})$ 为从 C_1 和 C_2 中抽取出来的方程组成的联立方程组，g_1 和 g_2 分别是 C_1 和 C_2 的卫兵条件，$g_{1,2} = g_1 \vee g_2$ ，分号（；）后表示程序的三种可能性

大纲

1. 建模语言
2. 模型转换
3. 原型工具
4. 案例分析
5. 总结

基于 dReal 实现仿真验证

- dReal 是 CMU Edmund M. Clarke 团队开发的一个开源的 SMT 求解器，基于 OpenSMT 和 RealPaver 实现，能够处理含有可计算非线性实函数的一阶逻辑公式判定问题

dReal 判定结果

对于 SMT 公式 φ，dReal 将返回如下判定结果：

- UNSAT ：φ 是不可满足的，模型无数值扰动 δ
- δ-SAT ：φ^{δ} 是可满足的，模型有数值扰动 δ

- 数值扰动可以理解为模型中物理量在一定范围内的非确定变化情况，对应模型的数值精度控制，δ 数值越小表明精度越高

dReal：SMT 公式

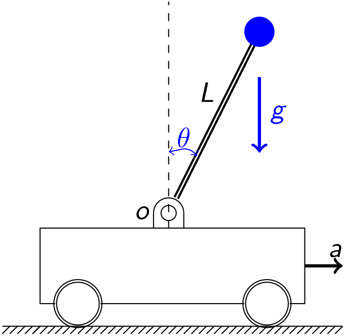
```
(set-logic QF_NRA_ODE)  ← 指定逻辑
(declare-fun height () Real)  ← 声明变量
...
(define-ode flow_1 ((= d/dt[height] velocity)
(= d/dt[velocity] (+ (- 0 9.8) (* (- 0 0.45) velocity)))
...
(assert (and (= mode_0 1) ... (= [velocity_0_t height_0_t
clock_0_t global_0_t](integral 0. time_0
[velocity_0_0 height_0_0 clock_0_0 global_0_0]  flow_1))
... (= mode_0 1)

(= mode_1 1)(= global_1_0 global_0_t)(= velocity_1_0
(* (- 0 0.9) velocity_0_t)) ...
...
(check-sat)
(exit)
```

大纲

案例分析：倒立摆

- 倒立摆数学模型

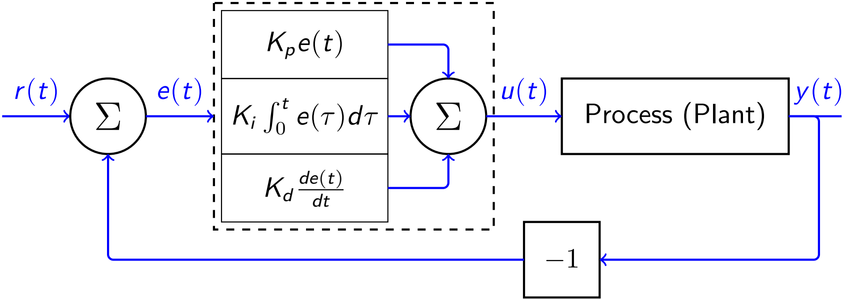


- L: 摆杆的长度，1 m
- θ: 摆杆与竖直方向的夹角
- a: 小车的加速度
- g: 重力加速度，$9.8\ \mathrm{m/s^2}$

$$L\frac{d^2\theta(t)}{dt^2} = g\sin[\theta(t)] - a(t)\cos[\theta(t)]$$

案例分析：倒立摆

- PID（比例 · 积分 · 微分）控制器的基本结构



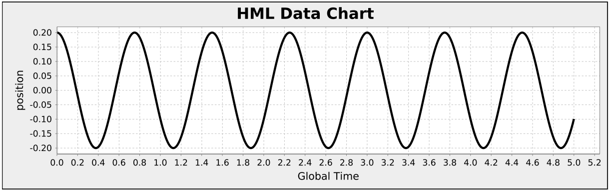
- r：目标值； e：误差； u：控制输出； y：过程输出（观测值）
- K_p：比例增益； K_i：积分增益； K_d：微分增益

案例分析：倒立摆 HML 模型，连续控制模式

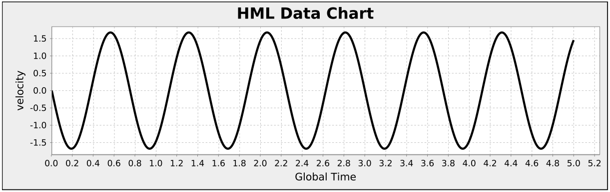
```
final float g=9.8, kp=80;← 常量
float position=0.2, velocity=0, clock=0, global=0; ← 变量
position in [-10, 10]; ← 设定取值范围
velocity in [-40, 40];
time in [0, 10];
clock in [0, 10];
global in [0, 10];
Template  Pendulum(float theta, float omega){
(dot  theta = omega)
||
(dot  omega = g*sin(theta) - (kp*theta)*cos(theta))
until  (false)
}
Main {
Pendulum(position, velocity)
}
```

案例分析：倒立摆连续控制模式仿真结果

- 倒立摆的位置。参数 $K_p = 80$，$K_i = 0$，$K_d = 0$。仿真时间 5 s

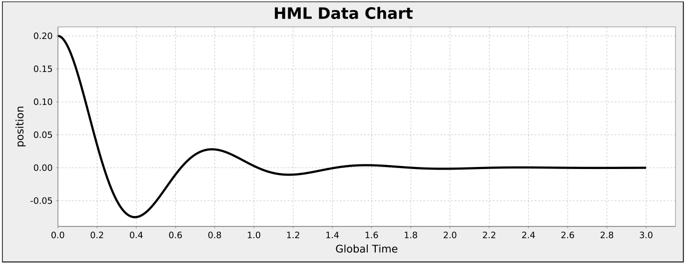


- 倒立摆的角速度。参数 $K_p = 80$，$K_i = 0$，$K_d = 0$。仿真时间 5 s



案例分析：倒立摆连续控制模式仿真结果

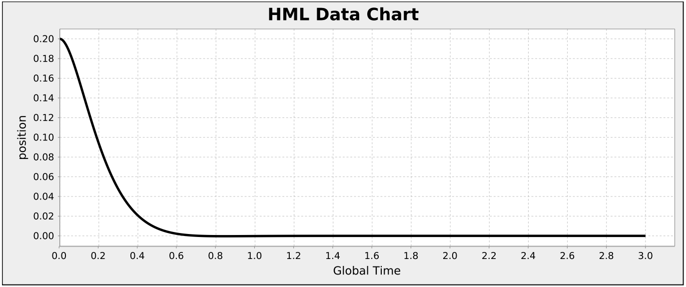
- 倒立摆的位置。参数 $K_p = 80$，$K_i = 0$，$K_d = 5$。仿真时间 3 s



- 位置振荡随着时间逐渐减弱，在 2 秒后趋于稳定

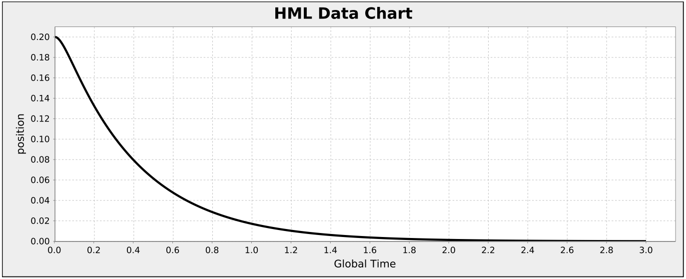
案例分析：倒立摆连续控制模式仿真结果

- 倒立摆的位置。参数 $K_p = 80$，$K_i = 0$，$K_d = 15$。仿真时间 3 s



- 位置没有出现振荡，在 0.6 秒之后趋于稳定

- 倒立摆的位置。参数 $K_p = 80$，$K_i = 0$，$K_d = 30$。仿真时间 3 s



- 位置没有出现振荡，与前一参数设定相比位置趋于稳定的速度变慢

案例分析：倒立摆离散控制 HML 模型

```
Template  Control(float a, float theta, float omega){
  while (true) {
    a = kp*theta  +  kd*omega; wait(0.08)
  }
}
Template  Pendulum(float theta, float omega, float a){
  (dot  a = 0) || (dot  theta = omega)
  || (dot  omega = g*sin(theta) - a*cos(theta))
  until  (false)
}
Main {
  {
  Pendulum(position, velocity, acc)
  || Control(acc, position, velocity)
  }
}
```

案例分析：倒立摆离散控制模式

1. 验证倒立摆的位置和角速度可以处于 $-0.01 \sim +0.01$：

$$(position \in [-0.01, 0.01]) \wedge (velocity \in [-0.01, 0.01])$$

验证结果：SAT （δ-SAT），可满足

案例分析：倒立摆离散控制模式

1. 验证倒立摆的位置和角速度可以处于 $-0.01 \sim +0.01$：

$$(\mathit{position} \in [-0.01, 0.01]) \wedge (\mathit{velocity} \in [-0.01, 0.01])$$

验证结果：SAT （δ-SAT），可满足

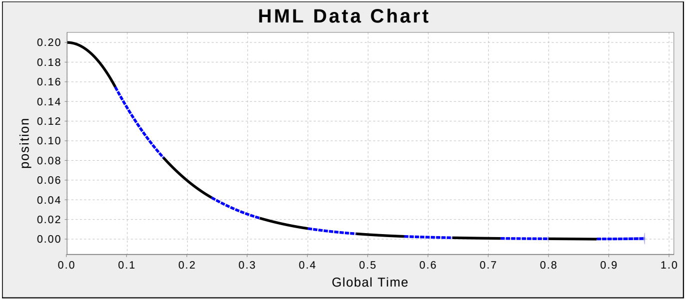
2. 判定倒立摆的位置和速度是否会超出期望的范围：

$$(\mathit{position} > 0.01 \vee \mathit{position} < -0.01) \vee (\mathit{velocity} > 0.01 \vee \mathit{velocity} < -0.01)$$

验证结果：UNSAT（即不可满足的），因此，倒立摆的位置和速度将稳定于范围 $[-0.01, 0.01]$ 内

案例分析：倒立摆离散控制仿真结果

- 离散控制下倒立摆的位置。采样时间 0.08 s，仿真时间 0.96 s



- 通过仿真与验证，倒立摆的位置在离散控制下将在 0.6 秒后趋于稳定

大纲

总结

- 混成系统建模语言 HML 的模型适合以逻辑公式的方式表达，并可基于 SMT 约束求解技术进行自动分析和验证

总结

- 混成系统建模语言 HML 的模型适合以逻辑公式的方式表达，并可基于 SMT 约束求解技术进行自动分析和验证
- 基于混成关系理论的基本思想，将 HML 模型转换为 SMT 公式，开发了原型工具，实现了 HML 模型仿真和验证的自动化

总结

- 混成系统建模语言 HML 的模型适合以逻辑公式的方式表达，并可基于 SMT 约束求解技术进行自动分析和验证
- 基于混成关系理论的基本思想，将 HML 模型转换为 SMT 公式，开发了原型工具，实现了 HML 模型仿真和验证的自动化
- 通过案例分析，HML 仿真和验证原型工具的可行性和有效性在一定程度上得以检验