Automatic Proving of Stability of the Cyber-Physical Systems in the Sense of Lyapunov with KeYmaera

Sergey Staroletov
Cyber-physical lab
Institute of Automation and Electrometry
Novosibirsk, Russia
serg_soft@mail.ru

***Abstract*—The use of control systems in human life requires new methods to ensure the quality of embedded software at all levels of its production. In this work, we refer to the level of a cyber-physical system.**

These systems here can be specified using the hybrid automata abstraction to represent both discrete-time and continuous-time transitions, such models are known as hybrid models and they are expressed using the Differential dynamic logic. For the verification of continuous-state systems, a lot of simple models have been built around the KeYmaera tool as its tutorials in train, car and robotic domains. The advantage of the formal verification method over the simulation is the possibility to play with parameter values and check the requirements automatically without looking to resulting plots of system behavior over time.

In this paper, we study how the hybrid theorem prover KeYmaera works while verifying control systems, in particular, using the Lyapunov method to prove stability of models encoded in a special notation of hybrid programs. We proceed to understand an example of a simple PD-controller and then move to a complex PID-controller model of a quadrotor stabilisation. We show that the process of such a verification can be carried out for rather complex systems initially taken in the form of engineering notation.

I. Introduction

We understand cyber-physical systems (CPS) as real-world systems that use a cyber part (computer/software) to control or operate a physical part (hardware/physical process) [1].

The idea of the article is based on engineering needs, coming from real-world CPSs, and comprises the analysis of hybrid programs to satisfy the needs. In this paper, we refer to *stability proving*, which means that for some given physical model of the system, the engineers want to know whether their model is *stable* (can predictably respond to parameter changes and return to an equilibrium point) or *unstable* which can prevent some violation of safety properties of a CPS.

In the study, we refer to well known methods and formalisms (PID control, Lyapunov stability, Lyapunov function, hybrod programs, dynamic logic, theorem proving) and apply them to solve the problem of automated proving of stability of a CPS, initially described in an engineering notation.

The main contributions of the paper are: (1) show the possibility of hybrid programs in KeYmaera notation to reflect real CPSs and their stability requirements; (2) discuss ways how to move from system states in a matrix notation to a formal notation of an automatic proof system in order to test the hypotheses of Lyapuniv stability.

The rest of the paper is organized as follows: in the *Background* section (Section II), we make a short description of the Differential dynamic logic abstraction as well as the declaration of hybrid programs to encode a CPS behavior, and the KeYmaera tool, then refer to the PID control scheme; in the next section (Section III) we study a PD-controller *example* and corresponding abstractions are used to check a tiny system; in the *main* section (Section IV), we discuss how to do machine-driven verification of stability of a complex model of quadrotor control using a given Lyapunov function. We present the model code using the KeYmaera input notation.

II. Background

A. Differential dynamic logic and hybrid programs

Differential dynamic logic (DDL, dL) [2] extends Pratt's dynamic logic by adding the following axioms [3]:

- Hoare's assignment rule;
- solution of the symbolic initial value problem;
- iteration axiom;
- modal modus ponens from modal logic;
- induction schema for loops;
- variation of Harel's convergence rule, suitably adapted to hybrid systems over $\mathbb{R}$;
- Barcan formula;
- vacuous modalities;
- Gödel's necessitation rule for modal logic;
- and an axiom for reducing differential equations with evolution domain constraints to equations without it.

According to A. Platzer [2], a dL formula that combines Boolean formulas and special operations on hybrid programs is defined as:

$$\neg P \mid P \wedge P \mid \forall x P \mid \exists x P \mid [\alpha] \mid \langle \alpha \rangle \tag{1}$$

where P is a meta-variable for the dL formulas, x is a meta-variable for first-order real-valued terms, [] is the box modality ("for all terminated runs"), $\langle\rangle$ is the diamond modality ("in a

terminated run"), α is a meta-variable for the hybrid programs (we do not describe standard Boolean things here). The syntax of hybrid programs α then is defined as follows:

$$\alpha ::= \; x := e \mid ?Q \mid x' = f(x) \& Q \mid \alpha \cup \alpha \mid \alpha; \alpha \mid \alpha^*$$

where x is a meta-variable for program variables, e is a meta-variable for the first-order real-valued terms, f is a meta-variable for the continuous real functions, x' is a time based derivative, and Q is a meta-variable for the first-order formulas over real numbers. The construct ';' means here the sequential composition, '$\cup$' is the non-deterministic choice, '?' is the condition operator, and '*' is the non-deterministic iteration (like Kleene-star) [4].

B. KeYmaera Tool

KeYmaera is an open-source (https://github.com/LS-Lab/KeYmaera-release) interactive theorem proving tool for CPSs encoded in the notation of hybrid programs. It combines deductive, real algebraic, and computer algebraic prover technologies [2].

The tool is based on the KeY project [5] and implemented in Java using software patterns [6]. Also some parts were developed in Scala. The proving process follows the analytic tableaux [7] scheme.

KeYmaera includes own code and the Orbital library for working with differential equations and real arithmetic, but it also has software interfaces to other tools handling real arithmetic and/or computer algebra for improved performance, for example, *Mathematica* by Wolfram Research.

A novel tool from the same research team, KeYmaera X [8] is a cloud-based successor of KeYmaera, offers a somewhat different syntax for the hybrid programs, but the idea of the proof process is the same.

C. Stability and Aleksandr Lyapunov's second method for stability

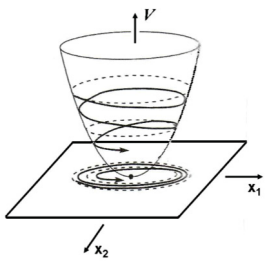

Fig. 1. Geometric interpretation of Lyapunov theorem [9]

Suppose $f' = x(t)$ and f has an equilibrium at x_e so that $f(x_e) = 0$. This equilibrium is said to be *Lyapunov stable*, if $\forall \varepsilon > 0$, $\exists \delta > 0$ such that, if $\|x(0) - x_e\| < \delta$, then for every $t \geq 0$ we have $\|x(t) - x_e\| < \varepsilon$. The equilibrium of the above system is said to be *asymptotically stable* if it is Lyapunov stable and $\exists \delta > 0$ such that if $\|x(0) - x_e\| < \delta$, then $\lim_{t\to\infty} \|x(t) - x_e\| = 0$.

The second method for stability makes use of a *Lyapunov function V(x)*, which represents an analogy to the potential energy function of classical dynamics. It is introduced as follows for a system $\dot{x} = f(x)$ having a point of equilibrium at $x = 0$. Consider a function $V : \mathbb{R}^n \to \mathbb{R}$ such that:

- $V(x) = 0$ iff $x = 0$;
- $V(x) > 0$ iff $x \neq 0$;
- $V'(x) = \frac{d}{dt} V(x) = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} f_i(x) = \nabla V \cdot f(x) \leq 0 \; \forall x \neq 0$.
- Note: for asymptotic stability, $V'(x) < 0$ for $x \neq 0$ is required.

Then $V(x)$ is called a *Lyapunov function* and the system is said to be stable in the sense of Lyapunov [10].

D. PID Control

The PID (**P**roportional, **I**ntegral, **D**ifferential) controller is a feedback system for correcting the state of a controlled object. When controlling the object, we calculate an error or difference between current and desired state (for example, between current and set altitude), then based on current error, we calculate the impact based on three parts with given coefficients.

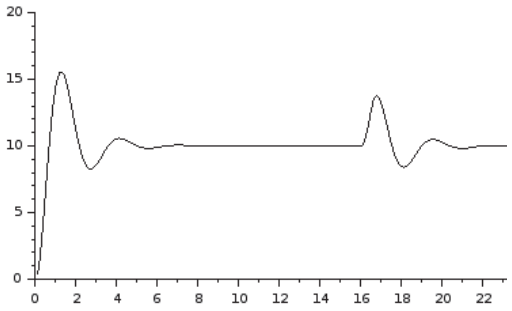

Fig. 2. An example of PID stabilization with a disturbance

- The proportional part (P) is responsible for the proportional reduction of the error (the representation of the present).
- The integral part (I) is a statistical change of the error (the representation of the past).
- The differential part (D) is the change of the error, its tendency to 0 (the representation of the future) [11].

PID-control is some kind of abstraction when the control process is based firstly on the difference (error) but not on attempts to describe the exact physical model of the system, for example, during a quadrotor flight, some changeable wind can blow, the rotation of the propellers may be unstable, the center of mass can be shifted and so on, but in such situations PID-controller will detect the deviation and try to make an impact to change it (see Fig. 2).

The analytic equation for the PID-control scheme:

$$u(t) = P + I + D = K_p\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de}{dt} \qquad (2)$$

where K_p, K_i and K_d are the PID coefficients. If some coefficient is zero, the system while controlling can lose stability or

smoothness of behaviour. The corresponding controllers with some zero parts are then called as P-controller, PD-controller or PI-controller.

III. Studying a PD-controller Example

Next, we proceed to the machine-driven verification of a simple PD-controller (the simplification of PID-controller given from example [12]). The model of the system is represented in a way close to a Hoare's triple:

$$init => [controller](req) \quad (3)$$

Then, we decompose the system into the following parts: precondition, continuous PD-controller with given K_p, K_d and requirements [11]. The precondition:

$$init :== v \geq 0 \wedge c > 0 \wedge K_p = 2 \wedge K_d = 3 \wedge V(p, p_r, v) < c \quad (4)$$

where v is the velocity; c is a number that greater than zero; Kp is the proportional coefficient; Kd is the differential coefficient; $V(p, p_r, v)$ is a Lyapunov function (will be established later).

The continuous state:

$$controller :== p' = v, v' = -K_p \cdot (p - p_r) - K_d \cdot v \quad (5)$$

where p is the current position; p_r is the resulting position; and $p - p_r$ it the error (position difference). As for the requirement, it is proposed to try checking stability using the Lyapunov method:

$$req :== V(p, p_r, v) < c \quad (6)$$

where c is a positive constant and V is the following Lyapunov function candidate:

$$V(p, p_r, v) = 5/4 \cdot (p - p_r)^2 + (p - p_r) \cdot v/2 + v^2/4 \quad (7)$$

The key property of Lyapunov functions which we use in this article is that $\forall c > 0$, $V(x) \leq c$ is the invariant of the system [12]. Intuitively, this follows from the fact that along the system dynamics, the Lyapunov function is decreasing (see Fig. 1).

Using the KeYmaera tool, it is possible to fully automatically verify stability of the CPS that models the PD-controller described in the hybrid program notation, as it is shown later:

Listing 1. A fragment of hybrid program to check stability of a PD-controller using the evolution statement

```
\problem {
\[
R p, v, a, S, Kp, Kd, c, r
\] (((v >= 0) & (Kp = 2))
& (Kd = 3) & (c >  0) & ((5 / 4)
* (p - r) ^ 2 + ((p - r) * v) / 2
+ (v ^ 2) / 4  <  c)
-> (\[
{p' = v, v' = (-Kp) * (p - r) -
Kd * v, v >= 0}
\] ((5 / 4) * (p - r) ^ 2) +
  ((p - r) * v) / 2
+ (v ^ 2) / 4 <  c))
}
```

The program is devoid of any cycles and contains one continuous state representing the physical model of the CPS, defined by the system of two equations. The program in the evolution statement is placed inside the modal operator *always* ([] – the box modality), while the rule for the Lyapunov function is set in the post- and precondition. Executing the evolution statement means for the modelled system to stay in the continuous state as long as it wishes (the time to stay is chosen non-deterministically) [13].

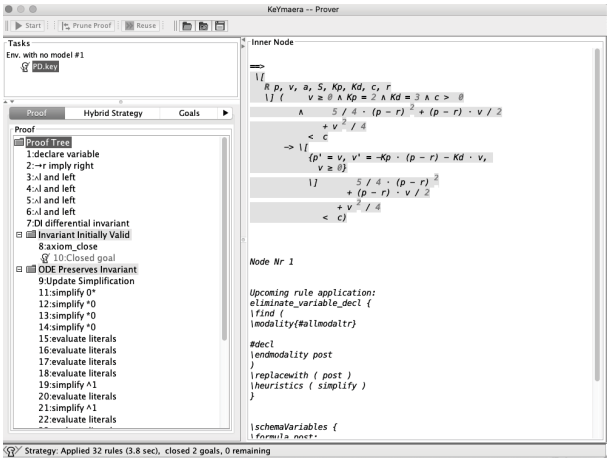

Fig. 3. Successful verification of the PD-controller model in KeYmaera

Next, we modify the previous program to solve the stabilization problem of the PD-controller using methods of computational mathematics, without encoding continuous states in the hybrid program, by using an explicit time cycle $t <= 30$ as well as setting an arbitrarily small time conversion value dt:

Listing 2. A fragment of Hybrid program using computational mathematics

```
\programVariables {
R t;
}
\problem {
\[
R p, v, a, S, Kp, Kd, c, r, dt, v0, p0,
dp, dv
\] (((v >= 0) & (Kp = 2))
& (Kd = 3) & (c >  0) & ((5 / 4)
* (p - r) ^ 2 + ((p - r) * v) / 2
+ (v ^ 2) / 4  <  c) & (t >= 0) &
(t <= 30) &
(dt > 0) & (dt < 0.01)
-> (\[
t := 0;
(while(t < 30 & v >= 0)
/* save old values */
v0 := v;
p0 := p;
/* calculate derivatives */
/* p' = v */
```

```
dp := v0;
/* v' = (-Kp) * (p - r) - Kd * v,
 v >= 0; */
dv := (-Kp) * (p0 - r) - Kd * v0;
/* get current values */
v := dv * dt + v0;
p := dp * dt + p0;
/* increase t */
t := t + dt
end)
\]
(
(5 / 4) * (p - r) ^ 2) + ((p - r) * v) / 2
+ (v ^ 2) / 4 <  c))
}
```

The system can also be proven automatically. Stability is proving here at the end of the computation cycle, i.e. after 30 seconds of the evolution of the system.

IV. Real-World Example: Modeling and Verification of a Quadrotor PID-controller

A. The model

In Fig. 4 we show a quadrotor with its position according to the axes OX, OY, OZ and its orientation (or so-called Euler angles) – roll, pitch and yaw.

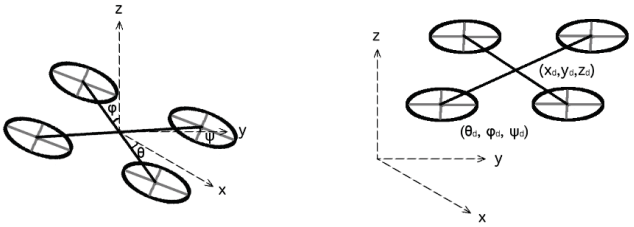

Fig. 4. Initial and desired stable state of the quadrotor

We borrowed a PID-controller model from the work by Camarillo-Gómez et al. [14], then we added some consistency, defined some constants and make the model suitable for the verification. Constants are defined in Table I. The system state includes a current position as well as current Euler angles:

$$q = [x, y, z, \theta, \phi, \psi]^T \tag{8}$$

Desired system state is the vector of position and orientation:

$$q_r = [x_r, y_r, z_r, \theta_r, \phi_r, \psi_r]^T \tag{9}$$

Mt represents the mass and inertia tensor:

$$Mt = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & Ixx & Ixy & Ixz \\ 0 & 0 & 0 & Iyx & Iyy & Iyz \\ 0 & 0 & 0 & Izx & Izy & Izz \end{bmatrix} \tag{10}$$

PID coefficients (proportional, integral differential gain for OX, OY, OZ axes as well as for roll, pitch, yaw stabilization are given in the diagonal matrices:

$$Kp = \begin{bmatrix} Kp_x & 0 & 0 & 0 & 0 & 0 \\ 0 & Kp_y & 0 & 0 & 0 & 0 \\ 0 & 0 & Kp_z & 0 & 0 & 0 \\ 0 & 0 & 0 & Kp_\theta & 0 & 0 \\ 0 & 0 & 0 & 0 & Kp_\phi & 0 \\ 0 & 0 & 0 & 0 & 0 & Kp_\psi \end{bmatrix} \tag{11}$$

$$Ki = \begin{bmatrix} Ki_x & 0 & 0 & 0 & 0 & 0 \\ 0 & Ki_y & 0 & 0 & 0 & 0 \\ 0 & 0 & Ki_z & 0 & 0 & 0 \\ 0 & 0 & 0 & Ki_\theta & 0 & 0 \\ 0 & 0 & 0 & 0 & Ki_\phi & 0 \\ 0 & 0 & 0 & 0 & 0 & Ki_\psi \end{bmatrix} \tag{12}$$

$$Kd = \begin{bmatrix} Kd_x & 0 & 0 & 0 & 0 & 0 \\ 0 & Kd_y & 0 & 0 & 0 & 0 \\ 0 & 0 & Kd_z & 0 & 0 & 0 \\ 0 & 0 & 0 & Kd_\theta & 0 & 0 \\ 0 & 0 & 0 & 0 & Kd_\phi & 0 \\ 0 & 0 & 0 & 0 & 0 & Kd_\psi \end{bmatrix} \tag{13}$$

Vector of the difference (error) between the current and desired system states:

$$q_d = q - q_r = [x - x_r, y - y_r, z - z_r, \theta - \theta_r, \phi - \phi_r, \psi - \theta_r]^T \tag{14}$$

Vector of the derivatives:

$$q_v = [x', y', z', \theta', \phi', \psi']^T \tag{15}$$

System state for PID-controlled plant is defined as the $\mathbb{R}^{18}$ vector:

$$\Xi = [\zeta, q_d, q_v]^T \tag{16}$$

where $\zeta' = q_d$ is the integral part rule.

The closed–loop equation of the system is obtained [14]:

$$q'' = \begin{bmatrix} (Kp_x \cdot x_d - Kd_x \cdot x' + Ki_x \cdot \zeta_x)/m \\ (Kp_y \cdot y_d - Kd_y \cdot y' + Ki_y \cdot \zeta_y)/m \\ (Kp_z \cdot z_d - Kd_z \cdot z' + Ki_z \cdot \zeta_z - m \cdot g)/m \\ (Kp_\phi \cdot \phi_d - Kd_\phi \cdot \phi' + Ki_\phi \cdot \zeta_\phi)/Ixx \\ (Kp_\theta \cdot \theta_d - Kd_\theta \cdot \theta' + Ki_\theta \cdot \zeta_\theta)/Iyy \\ (Kp_\psi \cdot \psi_d - Kd_\psi \cdot \psi' + Ki_\psi \cdot \zeta_\psi)/Izz \end{bmatrix} \tag{17}$$

The Lyapunov function proposed in the work [14] for the stability analysis is:

TABLE I. PARAMETERS OF THE QUADROTOR SYSTEM

Symbol	Description	Value	Unit
m	Mass of the quadrotor	2.03	kg
g	Earth gravity	9.8	m/s^2
$K\{p,i,d\}\{x,y,z\}$	PID gain coefficients for position	1	-
$K\{p,i,d\}\{\theta,\phi,\psi\}$	PID gain coefficients for orientation	1	-
Ixx	Moment of inertia around X axis	0.002547	$kg \cdot m^2$
Iyy	Moment of inertia around Y axis	0.003613	$kg \cdot m^2$
Izz	Moment of inertia around Z axis	0.001074	$kg \cdot m^2$
Ixy	Moment of inertia around XY axises	0.00308	$kg \cdot m^2$
Ixz	Moment of inertia around XZ axises	0.0018105	$kg \cdot m^2$
Iyx	Moment of inertia around YX axises	0.00308	$kg \cdot m^2$
Izy	Moment of inertia around ZY axises	0.0023435	$kg \cdot m^2$
Izx	Moment of inertia around ZX axises	0.0018105	$kg \cdot m^2$
Iyz	Moment of inertia around YZ axises	0.0023435	$kg \cdot m^2$
ε	A constant to adjust stability	–	-
$\lambda_{min}\{Kd\}$	Minimum eigenvalue of Kd matrix	1	-
$\lambda_{min}\{Kp\}$	Minimum eigenvalue of Kp matrix	1	-
$\lambda_{max}\{Ki\}$	Maximum eigenvalue of Ki matrix	1	-
$\lambda_{min}\{Mt\}$	Minimum eigenvalue of Mt matrix	$9.90422 \cdot 10^{-20}$	-
$\lambda_{max}\{Kp\}$	Maximum eigenvalue of Mt matrix	2.03	-

$$V(q_d,q_v,\gamma) = \frac{1}{2}\cdot\begin{bmatrix}\gamma\\ q_d\\ q_v\end{bmatrix}^T\cdot\begin{bmatrix}\frac{1}{\varepsilon}\cdot Ki & 0 & 0\\ 0 & \varepsilon\cdot Kd & -\varepsilon\cdot Mt\\ 0 & -\varepsilon\cdot Mt & Mt\end{bmatrix}\cdot\begin{bmatrix}\gamma\\ q_d\\ q_v\end{bmatrix} + \frac{1}{2}\cdot q_d^T\cdot\left[Kp-\frac{1}{\varepsilon}\cdot Ki\right]\cdot q_d^T + U(q_r-q_d) - U(q_r) + q_d^T\cdot\begin{bmatrix}0\\0\\m\cdot g\\0\\0\\0\end{bmatrix} \tag{18}$$

Where

- U represents the potential energy of the quadrotor;
- $\gamma = \varepsilon\cdot\mu + q_d$ and $\mu' = q_r$;
- ε is chosen according to the following law:

$$\frac{\lambda_{min}\{Kd\}\cdot\lambda_{min}\{Mt\}}{\lambda^2_{max}\{Mt\}} > \varepsilon > \frac{\lambda_{max}\{Ki\}}{\lambda_{min}\{Kp\}} \tag{19}$$

(see Table 1 for the description).

B. Verification of the model

Here we construct a hybrid program for the quadrotor system.

At first, after multiplying the matrices in (18) we get:

$$\begin{aligned}V = \frac{1}{2}\Big(&\theta'\cdot\big(-\varepsilon\cdot \mathrm{Ixx}\cdot(\theta-\theta_r) - \varepsilon\cdot \mathrm{Iyx}\cdot(\phi-\phi_r) - \varepsilon\cdot \mathrm{Izx}\\ &\cdot(\psi-\psi_r) + \mathrm{Ixx}\cdot\theta' + \mathrm{Iyx}\cdot\phi' + \mathrm{Izx}\cdot\psi'\big) + (\theta-\theta_r)\\ &\cdot\big(-\varepsilon\cdot\mathrm{Ixx}\cdot\theta' - \varepsilon\cdot\mathrm{Iyx}\cdot\phi' - \varepsilon\cdot\mathrm{Izx}\cdot\psi' + \varepsilon\cdot Kd_\theta\cdot(\theta-\theta_r)\big)\\ &+\phi'\cdot\big(-\varepsilon\cdot \mathrm{Ixy}\cdot(\theta-\theta_r) - \varepsilon\cdot \mathrm{Iyy}\cdot(\phi-\phi_r) - \varepsilon\cdot \mathrm{Izy}\\ &\cdot(\psi-\psi_r) + \mathrm{Ixy}\cdot\theta' + \mathrm{Iyy}\cdot\phi' + \mathrm{Izy}\cdot\psi'\big) + (\phi-\phi_r)\\ &\cdot\big(-\varepsilon\cdot\mathrm{Ixy}\cdot\theta' - \varepsilon\cdot\mathrm{Iyy}\cdot\phi' - \varepsilon\cdot\mathrm{Izy}\cdot\psi' + \varepsilon\cdot Kd_\phi\cdot(\phi-\phi_r)\big)\\ &+\psi'\cdot\big(-\varepsilon\cdot \mathrm{Ixz}\cdot(\theta-\theta_r) - \varepsilon\cdot \mathrm{Iyz}\cdot(\phi-\phi_r) - \varepsilon\cdot \mathrm{Izz}\\ &\cdot(\psi-\psi_r) + \mathrm{Ixz}\cdot\theta' + \mathrm{Iyz}\cdot\phi' + \mathrm{Izz}\cdot\psi'\big) + (\psi-\psi_r)\\ &\cdot\big(-\varepsilon\cdot\mathrm{Ixz}\cdot\theta' - \varepsilon\cdot\mathrm{Iyz}\cdot\phi' - \varepsilon\cdot\mathrm{Izz}\cdot\psi' + \varepsilon\cdot Kd_\theta\cdot(\psi-\psi_r)\big)\\ &+(x-x_r)\cdot\big(\varepsilon\cdot Kd_x\cdot(x-x_r) - m\cdot\varepsilon\cdot x'\big) + (y-y_r)\\ &\cdot\big(\varepsilon\cdot Kd_y\cdot(y-y_r) - m\cdot\varepsilon\cdot y'\big) + (z-z_r)\\ &\cdot\big(\varepsilon\cdot Kd_z\cdot(z-z_r) - m\cdot\varepsilon\cdot z'\big) + \frac{\gamma_\theta^2\cdot Ki_\theta}{\varepsilon} + \frac{\gamma_\psi^2\cdot Ki_\psi}{\varepsilon}\\ &+\frac{{\gamma_x}^2\cdot Ki_x}{\varepsilon} + \frac{{\gamma_y}^2\cdot Ki_y}{\varepsilon} + \frac{{\gamma_z}^2\cdot Ki_z}{\varepsilon} + \frac{\gamma_\phi^2\cdot Ki_\phi}{\varepsilon} + x'\\ &\cdot\big(m\cdot x' - m\cdot\varepsilon(x-x_r)\big) + y'\cdot\big(m\cdot y' - m\cdot\varepsilon(y-y_r)\big)\\ &+z'\cdot\big(m\cdot z' - m\cdot\varepsilon(z-z_r)\big)\Big)\\ &+\frac{1}{2}\left((\theta-\theta\mathrm{r})^2\left(Kp_\theta - \frac{Ki_\theta}{\varepsilon}\right) + (\psi-\psi\mathrm{r})^2\left(Kp_\theta - \frac{Ki_\theta}{\varepsilon}\right)\right.\\ &+(x-\mathrm{xr})^2\left(Kp_x - \frac{Ki_x}{\varepsilon}\right) + (y-y_r)^2\left(Kp_y - \frac{Ki_y}{\varepsilon}\right)\\ &\left.+(z-z_r)^2\left(Kp_z - \frac{Ki_z}{\varepsilon}\right) + (\phi-\phi\mathrm{r})^2\left(Kp_\phi - \frac{Ki_\phi}{\varepsilon}\right)\right)\\ &+\mathrm{mg}\cdot(z-z_r) + U\end{aligned} \tag{20}$$

Then, the hybrid program, as it was discussed, should be considered in the form

$$init => [PID-controlled-law](req)$$

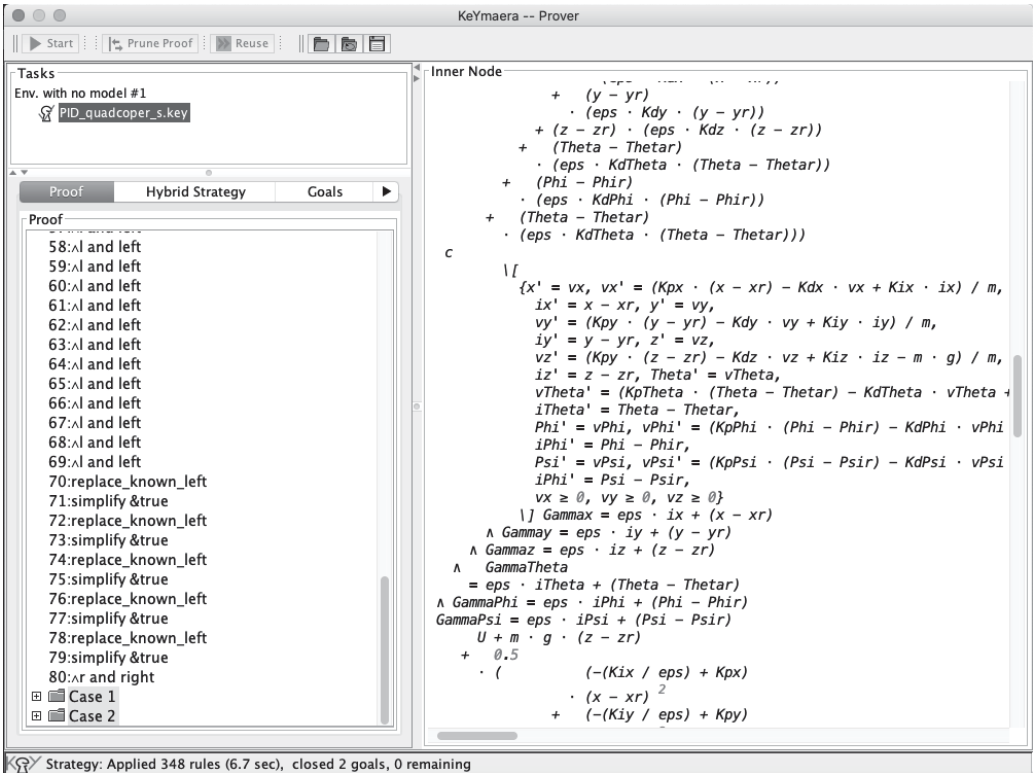

Fig. 5. Successful verification of the PID-controller model in KeYmaera

- For the precondition, we setup initial values mostly according to Table 1 and then test stability at the initial point $[x_0, y_0, z_0, \theta_0, \phi_0, \psi_0]$. For this, we use rule $V_0 < c, \forall c > 0$, where V_0 is the state of V (20) at the initial point with all derivatives set to 0.
- For the plant, we follow (17), and add rules for integral parts according to (16). Second derivatives are eliminated by introducing speed variables as derivatives of q and variable substitution.
- For the invariant, we use rule $V < c, \forall c > 0$ where V is introduced in (20). According to the nature of the evolution state in a hybrid program, it leaves the state with an ODE system solution at a random time and here we are supposed to check stability of the system. So, the given postcondition becomes an invariant of the system to check the fact that "along the system dynamics, the Lyapunov function is decreasing" which leads to system stability.

The complete model is available on GitHub [15].

According to our experiment, the model of quadrotor system is provable with KeYmaera (Fig. 5). Verification was completed automatically in about 7 seconds and 348 rules of dL were applied.

In Fig. 6 we show a generated detailed description of the hybrid system using our intermediate tool.

V. Related Work

The main historical stages of automatic control are given by Bennett [16]. For some pre-defined types of systems, automated methods to check stability have been proposed by Oehlerking et al. [17], however, such methods have not been widely used in the CPS development. With the increasing complexity of systems, a fuzzy control method has been proposed recently by Tanaka et al. [18]. Some ideas of the transition of PD-controllers models described in the Simulink notation, to hybrid programs are given by Baar and Schulte [19]. For multicopters, in book [20] Quan described most of the issues of their design as well as modeling of the dynamics. Some verification techniques for hybrid systems are given by Tomlin et al. [21] and the most complete logical foundations of CPS are discussed by Platzer in [22]. In the very recent preprint [23] they have also proposed some lemmas for stability proving.

VI. Conclusions

In this paper, we discuss the theorem proving process to verify the system stability in the sense of Lyapunov using KeYmaera input syntax. We show that it is possible to prove complex models, designed using the Control theory (close to real engineers). The automatic theorem proving, in contrast to manual proving, could help the engineers to verify systems with different parameters and automatically try various Lyapunov functions without having to calculate their derivatives. The latter tells us that it is possible to create a static stand for

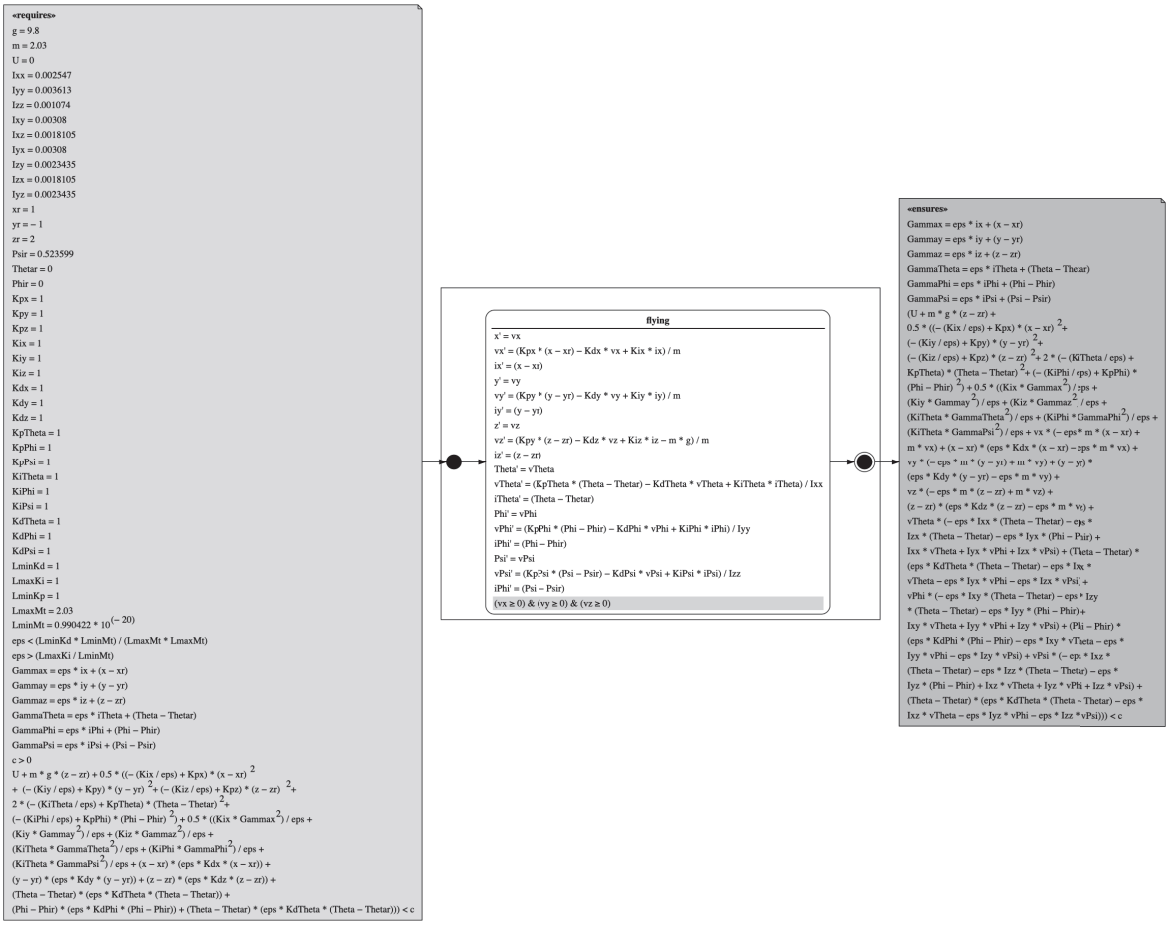

Fig. 6. Precondition and invariant for the hybrid flying state

the selection and testing of Lyapunov functions. We can also note that although dL is a powerful mathematical basis for describing and checking such systems, an intermediate DSL for the possibility of an easier description of input models would be useful for the users of KeYmaera tool. In particular, this problem is considered in [13], however, as it can be seen from the present article, the matrix operations must also be included into such a language.

VII. Acknowledgment

The author gratefully acknowledges Prof. Dr. Thomas Baar from HTW Berlin for involvement in this area and valuable remarks on this paper. The author also thanks the four anonymous reviewers for their comments.

References

[1] S. Staroletov, N. Shilov, V. Zyubin, T. Liakh, A. Rozov, I. Konyukhov, I. Shilov, T. Baar, and H. Schulte, "Model-driven methods to design of reliable multiagent cyber-physical systems," in *Proc. of the Conference on Modeling and Analysis of Complex Systems and Processes (MACSPro 2019)*, 2019.

[2] A. Platzer, *Logical analysis of hybrid systems: proving theorems for complex dynamics*. Springer Science & Business Media, 2010.

[3] ——, "The complete proof theory of hybrid systems," in *2012 27th Annual IEEE Symposium on Logic in Computer Science*. IEEE, 2012, pp. 541–550.

[4] S. Staroletov and N. Shilov, "Applying model checking approach with floating point arithmetic for verification of air collision avoidance maneuver hybrid model," in *International Symposium on Model Checking Software*. Springer, 2019, pp. 193–207.

[5] W. Ahrendt, T. Baar, B. Beckert, R. Bubel, M. Giese, R. Hähnle, W. Menzel, W. Mostowski, A. Roth, S. Schlager *et al.*, "The KeY tool," *Software & Systems Modeling*, vol. 4, no. 1, pp. 32–54, 2005.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design patterns. isbn: 0-201-63361-2," 1996.

[7] R. Goré, "Tableau methods for modal and temporal logics," in *Handbook of tableau methods*. Springer, 1999, pp. 297–396.

[8] N. Fulton, S. Mitsch, J.-D. Quesel, M. Völp, and A. Platzer, "KeYmaera X: An axiomatic tactical theorem prover for hybrid systems," in *International Conference on Automated Deduction*. Springer, 2015, pp. 527–538.

[9] A. A. Ahmadi, "Non-monotonic Lyapunov functions for stability of nonlinear and switched systems: theory and computation," Ph.D. dissertation, Massachusetts Institute of Technology, 2008.

[10] A. M. Lyapunov, "The general problem of motion stability," *Annals of Mathematics Studies*, vol. 17, 1892.

[11] S. M. Staroletov, M. S. Amosov, and K. M. Shulga, "Designing robust quadcopter software based on a real-time partitioned operating system and formal verification techniques," *Proceedings of the Institute for System Programming of the RAS*, vol. 31, no. 4, pp. 39–60, 2019.

[12] J.-D. Quesel, S. Mitsch, S. Loos, N. Aréchiga, and A. Platzer, "How to model and prove hybrid systems with KeYmaera: a tutorial on safety," *International Journal on Software Tools for Technology Transfer*, vol. 18, no. 1, pp. 67–91, 2016.

[13] T. Baar, "A metamodel-based approach for adding modularization to KeYmaera's input syntax," in *International Andrei Ershov Memorial*

Conference on Perspectives of System Informatics. Springer, 2019, pp. 125–139.

[14] K. A. Camarillo-Gómez, G. I. Pérez-Soto, J. Rodríguez-Reséndiz *et al.*, "Comparison of PD, PID and sliding-mode position controllers for V–tail quadcopter stability," *IEEE Access*, vol. 6, pp. 38 086–38 096, 2018.

[15] S. Staroletov, *Hybrid programs in .key (KeYmaera syntax) to prove the stability of PD and PID. DOI:10.5281/zenodo.3737545*, 2020.

[16] S. Bennett, "A brief history of automatic control," *IEEE Control Systems Magazine*, vol. 16, no. 3, pp. 17–25, 1996.

[17] J. Oehlerking, H. Burchardt, and O. Theel, "Fully automated stability verification for piecewise affine systems," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2007, pp. 741–745.

[18] K. Tanaka, T. Hori, and H. O. Wang, "A multiple lyapunov function approach to stabilization of fuzzy control systems," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 4, pp. 582–589, 2003.

[19] T. Baar and H. Schulte, "Safety analysis of longitudinal motion controllers during climb flight," *Modeling and Analysis of Information Systems*, vol. 26, no. 4, pp. 488–501, 2019.

[20] Q. Quan, *Introduction to multicopter design and control*. Springer, 2017.

[21] C. J. Tomlin, I. Mitchell, A. M. Bayen, and M. Oishi, "Computational techniques for the verification of hybrid systems," *Proceedings of the IEEE*, vol. 91, no. 7, pp. 986–1001, 2003.

[22] A. Platzer, *Logical foundations of cyber-physical systems*. Springer, 2018, vol. 662.

[23] Y. K. Tan and A. Platzer, "Deductive stability proofs for ordinary differential equations," *arXiv preprint arXiv:2010.13096*, 2020.