.ocml v0.02

Jonas Koenemann
2019/05/28

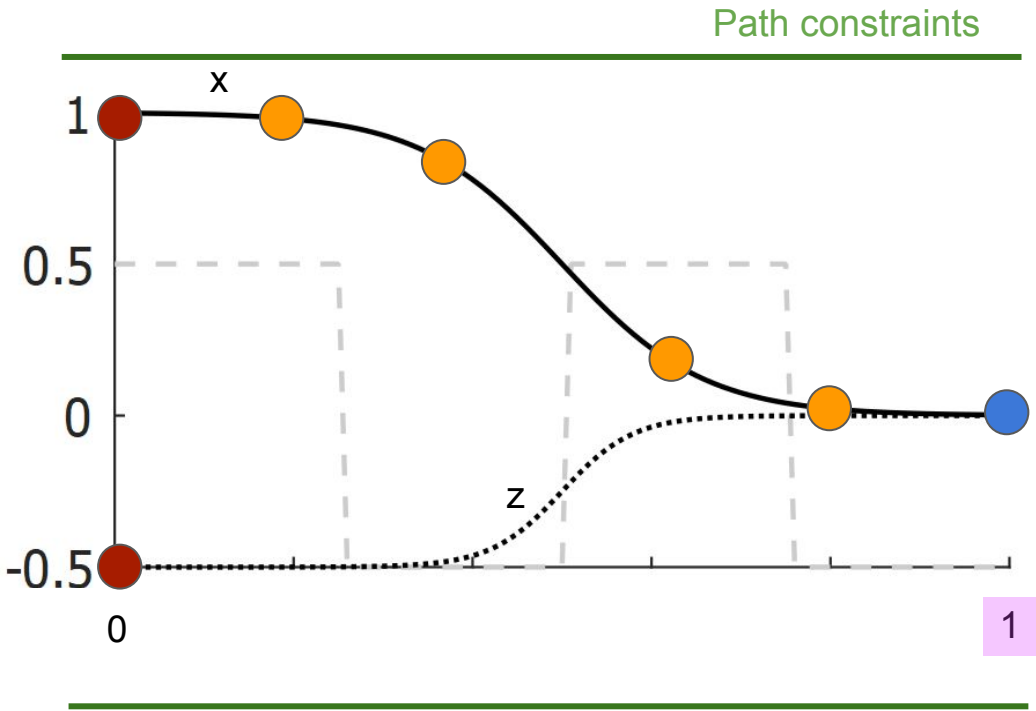
Follow progress at

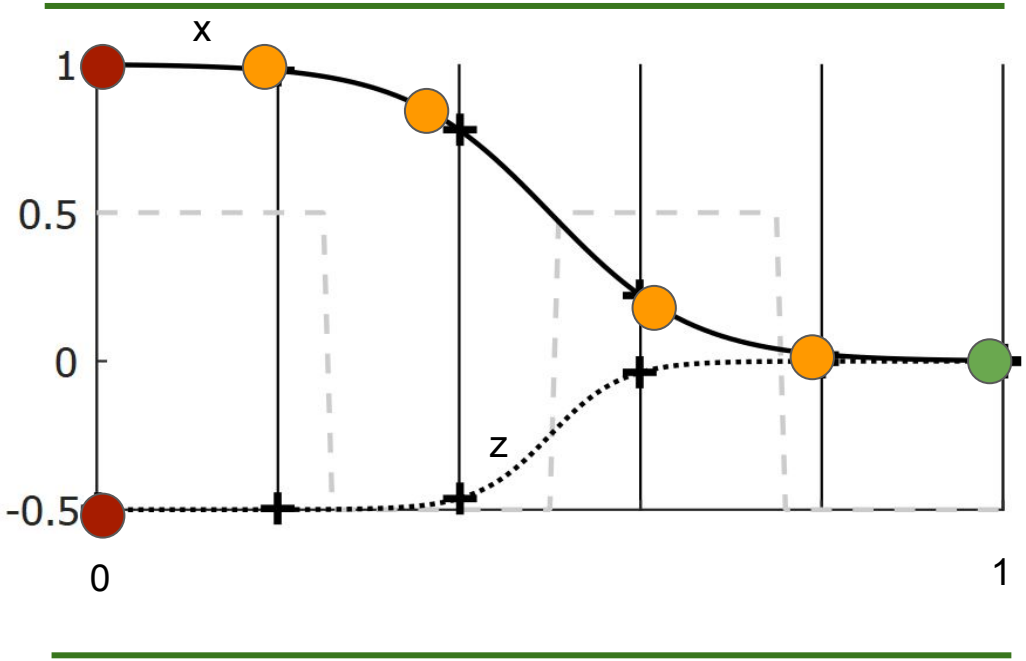
https://github.com/openocl/ocml/

Design choices

- Design inspired mostly by AMPL
- Start off with a minimal language
  - Minimize the number of keywords, functionalities
  - The language can be extended later
  - Requires reformulations on the user side
- Multiple phases
- Continuous time formulation with **point costs** and **point constraints**

Point cost and constraints




x
1
0.5
0
-0.5
z
0
1

$$\min_{x,z,u,t_{\max}} \quad \sum_{i=1}^{N_l} l_i(x(t_i), z(t_i)) \qquad \text{with } t_i = \hat{t}_i t_{\max}$$

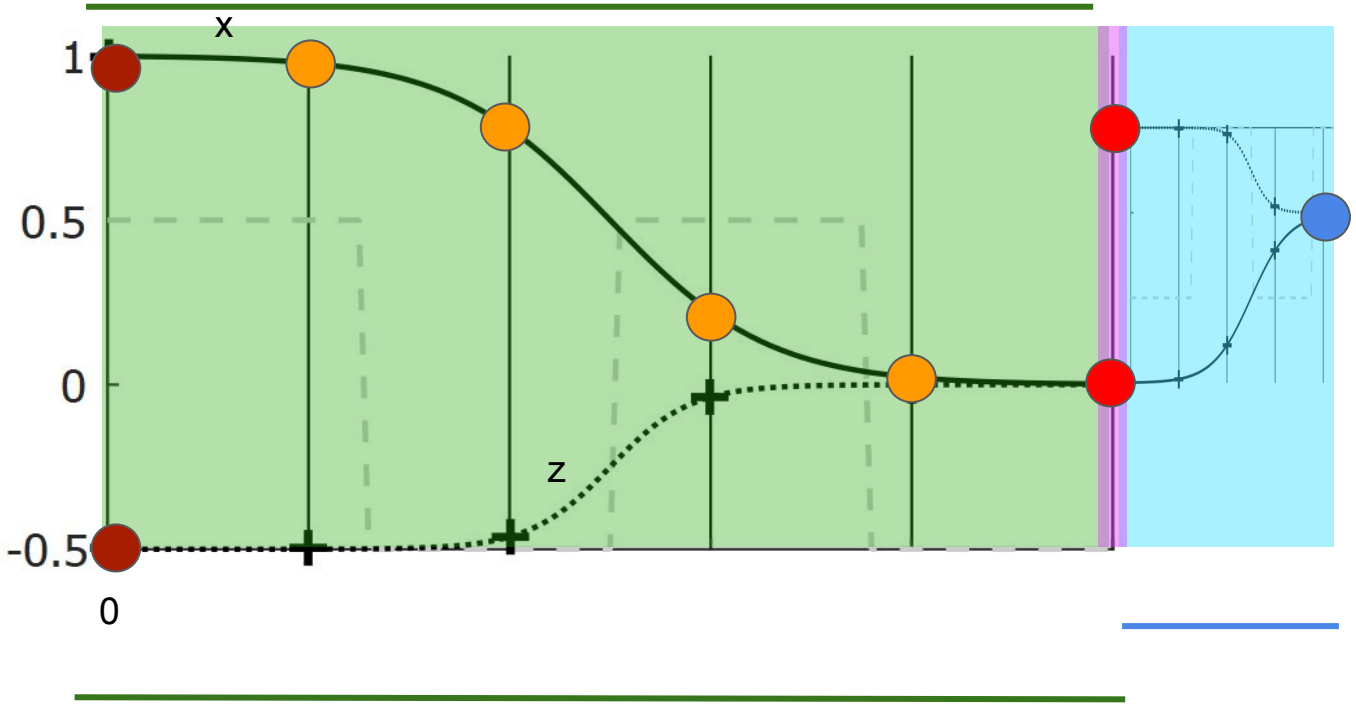
s.t.

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}t}x(t) = f(x(t), z(t), u(t))\,, && \text{for } t \in [0, t_{\max}]\,,\\
&0 = g(x(t), z(t), u(t))\,, && \text{for } t \in [0, t_{\max}]\,,\\
&r_j(x(t_j), z(t_j) \leq 0\,, && t_j = \hat{t}_j t_{\max}\,, \forall j \in \{1, \ldots, N_r\}\,,\\
&h(x(t), z(t)) \leq 0\,, && \text{for } t \in [0, t_{\max}]\,,\\
&x : [0, t_{\max}] \to \mathbb{R}^{N_x}\,,\\
&z : [0, t_{\max}] \to \mathbb{R}^{N_z}\,,\\
&u : [0, t_{\max}] \to \mathbb{R}^{N_u}\,,
\end{aligned}$$

given

Point cost pairs:	$(\hat{t}_i, l_i)$,	$\hat{t}_i \in [0,1]\,, \forall i \in \{1, \ldots, N_l\}$
Point constraint pairs:	$(\hat{t}_j, r_j)$,	$\hat{t}_j \in [0,1]\,, \forall j \in \{1, \ldots, N_r\}$
Dynamics functions	f, g	
Path constraints function	h	

Stages and phases



$$\min_{\substack{x_0,\dots,x_N \\ z_0,\dots,z_N \\ u_0,\dots,u_N \\ t_N}} \quad \sum_{k=1}^{N_\phi} \sum_{i=1}^{N_{k,l}} l_{k,i}(x_k(t_{k,i}), z_k(t_{k,i})) \qquad \text{with } t_{k,i} = \hat{t}_{k,i} t_{k+1}$$

s.t.

$$\forall k \in \{0, \dots, N-1\} :$$

$$\frac{\mathrm{d}}{\mathrm{d}t} x_k(t) = f_k(x_k(t), z_k(t), u_k(t)), \qquad \text{for } t \in [t_k, t_{k+1}),$$

$$0 = g_k(x_k(t), z_k(t), u_k(t)), \qquad \text{for } t \in [t_k, t_{k+1}),$$

$$x_{k+1}(t_{k+1}) = \phi_k(x_k(t_{k+1})), \qquad \text{(phase continuity)}$$

$$r_{k,j}(x_k(t_{k,j}), z_k(t_{k,j}) \leq 0, \qquad t_j = \hat{t}_{k,j} t_{k+1}, \forall j \in \{1, \dots, N_{k,r}\},$$

$$h_k(x_k(t), z_k(t)) \leq 0, \qquad \text{for } t \in [t_k, t_{k+1}),$$

$$x_k : [t_k, t_{k+1}] \to \mathbb{R}^{N_{k,x}},$$
$$z_k : [t_k, t_{k+1}] \to \mathbb{R}^{N_{k,z}},$$
$$u_k : [t_k, t_{k+1}] \to \mathbb{R}^{N_{k,u}},$$

Language specification

problem <name> [<end_time>]:

or

phase <name> [<end_time>]:

```
variable.state <name> [(<s1>,<s2>)] [>=<lb>] [<=<ub>]

variable.algebraic <name> [(<s1>,<s2>)] [>=<lb>] [<=<ub>]

variable.control <name> [(<s1>,<s2>)] [>=<lb>] [<=<ub>]
```

```
constraint.differential <name> diff(<name>) = <expr(x,z,u)>

constraint.algebraic <name> = <expr(x,z,u)>

constraint.path <name> <expr(x,z)> <= <expr(x,z)>
constraint.path <name> <expr(x,z)> >= <expr(x,z)>

constraint.point <location> <name> <expr(x,z)> = <expr(x,z)>
constraint.point <location> <name> <expr(x,z)> <= <expr(x,z)>
constraint.point <location> <name> <expr(x,z)> >= <expr(x,z)>

constraint.phase <name> <state> = <expr(phase.x,phase.z,x,z)>
```

```
minimize.point <location> <name> = <expr(x,z)>
```

matrices, expressions, operators, functions

```
[1 2 3; 4 5 6; 7 8 10]   zeros     ones

+    -    *    /    ^    '    .+   .-   .*   ./   .^

cos  sin  tan  atan2 …

mul  dot  cross     norm      sum  sqrt

vertcat   horzcat

jacobian  jtimes

reshape   repmat    transpose      polyval

triu      inv  det  trace

mldivide  mrdivide
```

```
problem cartpole:

variable.state s, >= -5, <= 5
variable.state theta, >= -2*pi, <= 2*pi
variable.state v
variable.state omega
variable.state time, >= 0, <= 10
variable.control F, >= -12, <= 12

constraint.point 0.0 s = 0
constraint.point 0.0 theta = pi
constraint.point 0.0 v = 0
constraint.point 0.0 omega = 0
constraint.point 0.0 time= 0
```

```
constraint.differential ds diff(s) = v

constraint.differential dtheta diff(theta) = omega

constraint.differential domega diff(omega) =
  (9.81 * sin(theta) + cos(theta) *
   (-F - 0.05 * omega^2 * sin(theta)) / 1.1) /
  (0.5 * (4/3 - 0.1 * cos(theta)^2 / 1.1))

constraint.differential dv diff(v) =
    (F + 0.05 * (omega^2 * sin(theta) -
     domega * cos(theta)) / 1.1

constraint.differential dtime diff(time) = 1

minimize.arrival time_cost = time
```

```
phase before_contact:

variable.state s, >= 0, <= 1
variable.state v

constraint.differential ds diff(s) = v
constraint.differential dv diff(v) = -9.81

constraint.point 0.0 s = 1
constraint.point 0.0 v = 0

constraint.point 1.0 s = 0
```

```
phase after_contact 0.452:

variable.state s, >= 0, <= 1
variable.state v,
variable.state iF
variable.control F

constraint.differential ds diff(s) = v
constraint.differential dv diff(v) = -9.81 + F
constraint.differential diF diff(iF) = F^2

constraint.phase s = before_contact.s
constraint.phase v = -before_contact.v / 2
constraint.phase iF = 0

constraint.point 1.0 s = 1
constraint.point 1.0 v = 0

minimize.arrival force iF
```

```
problem path_following:

data wp, (2,1), path.dat                    # maybe: data wp (2,1) tcp://localhost:1234

variable.state p, (2,1)
variable.control v, (2,1)

constraint.point 0.0 p = [0;0]

constraint.differential dp diff(p) = v

minimize.point 0.2 c1 = norm(p - wp{1})^2
minimize.point 0.4 c2 = norm(p - wp{2})^2
minimize.point 0.6 c3 = norm(p - wp{3})^2
minimize.point 0.8 c4 = norm(p - wp{4})^2
minimize.point 1.0 c5 = norm(p - wp{5})^2
```

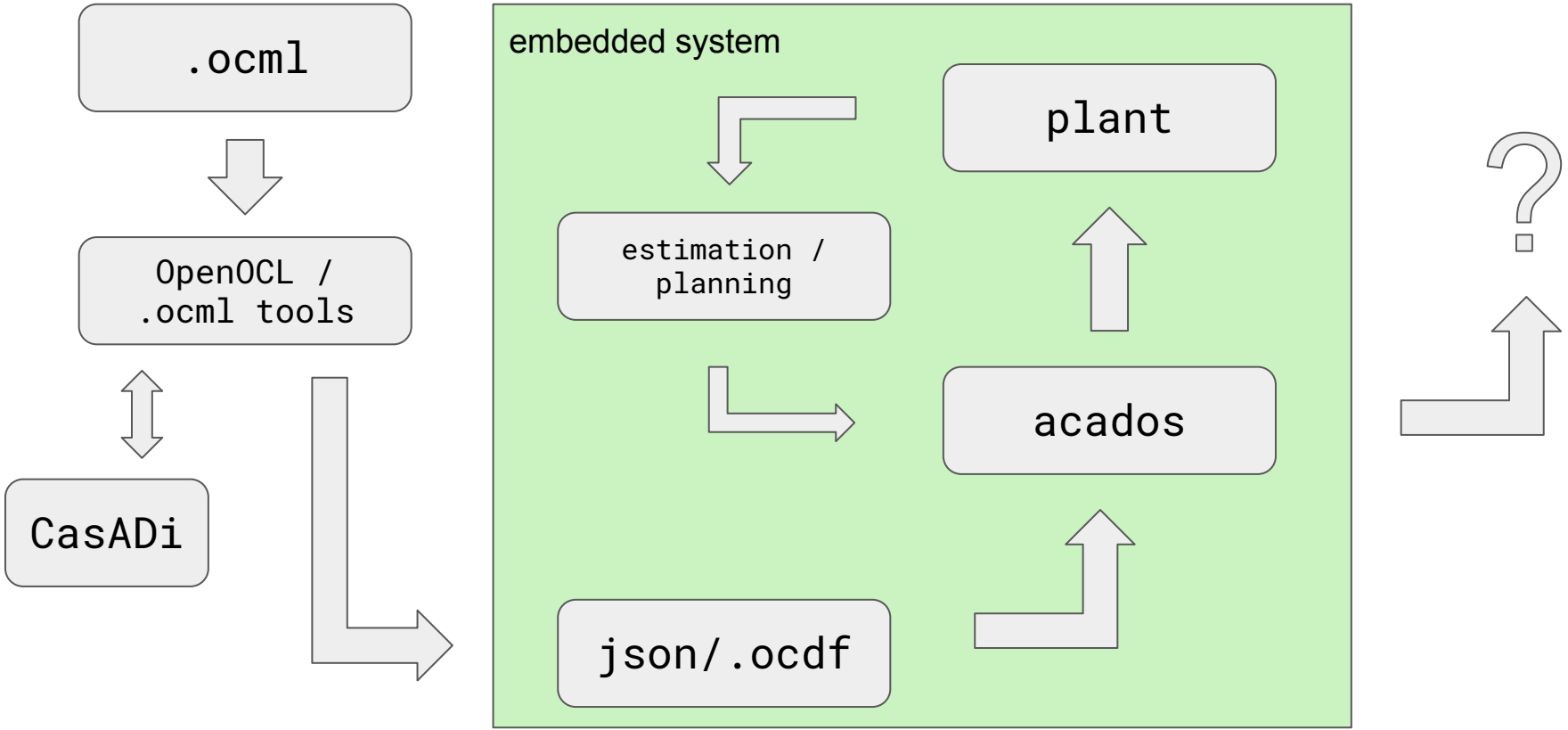
Extensions

```
constraint.differential.linear <name> diff(<state>) =
        <matrix> * <state> + <matrix> * <control>

constraint.differential.linear <name> next(<state>) =
        <matrix> * <state> + <matrix> * <control>

constraint.linear <name> <matrix> * <variable> <= <vector>

minimize.initial <name> = <expr(x)>

minimize.lagrange <name> = <expr(x)>

minimize.control <name> = <expr(u)>

let <name>.scaling = <scalar>

constraint.terminal <name> <state> = <value>
```

Extensions

```
variable.state <name> [(<s1>,<s2>)] [>=<lb>] [<=<ub>] [:=<ig>]

variable.control F, >= -12, <= 12
variable.control -12 <= F <= 12

variable.state omega, constraint.point 0.0 omega = 0
variable.state omega, 0= 0

minimize.arrival cost c
maximize.arrival reward r

# comments

ocml v0.02
```

Embedded optimization workflow



.ocml v0.02

Jonas Koenemann
2019/05/28

Trajectory optimization workflow