OmegaThreads: Symbolic Controller Design for ω-regular Objectives

Mahmoud Khaled and Majid Zamani | HSCC 2021 : Poster/Demo Session, May 19th 2018

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN | University of Colorado Boulder | Technische Universität München

CPS-IoT Week 2021

OmegaThreads: Computational Awareness and the Support for Practical Specifications

Problem: Symbolic control methods exhibit exponential time complexity w.r.t. state dimensions. The existing tools support limited fragments of LTL specifications.
Contributions: We introduce `OmegaThreads` to address the above two issues of Symbolic Control. Two main concepts make this possible:

1. **Computational awareness:** having parallel algorithms and variable computing resources allows controlling the complexity of abstraction/synthesis algorithms.
2. **Accepting ω-regular objectives:** having specifications as parity automata allows constructing parity games whose solutions are dynamic symbolic controllers.

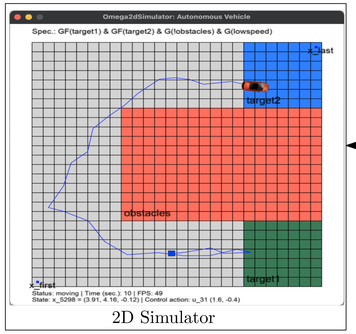
Download | Install | Run

`OmegaThreads` is open-source and available on GitHub:

https://github.com/mkhaled87/pFaces-OmegaThreads

- Based on `pFaces` and supports `CPUs`, `GPUs` and `HWAs`.
- A `Docker` version is available.

Internal Design and Workflow of OmegaThreads



Config File (Text)

```
system {
    states { bounds, quant. }
    controls { bounds, quant. }
    dynamics {timestep, code_file}
}

specifications {
    ltl_formula = "....." ;
}
```

Dynamics File (C99)

```
// Model Post Function
void model_post(
    concrete_t* post_x_lb, concrete_t* post_x_ub,
    const concrete_t* x, const concrete_t* u
) {
    // Do your Calculations
    ...

    // Set Post States
    post_x_ub[:] = ...;
    post_x_lb[:] = ...;
}
```

Example: Drone on Battery

System: Hybrid, impulsive nonlinear system.

Task: design a controller for a drone operated on a battery to:

1. perform infinitely-often two different pickup-and- delivery tasks,
2. fully complete a task before starting another one,
3. avoid some obstacles labelled on the states set,
4. never let the battery reach a low state, and
5. from time to time, charge the battery fully.

Specifications:

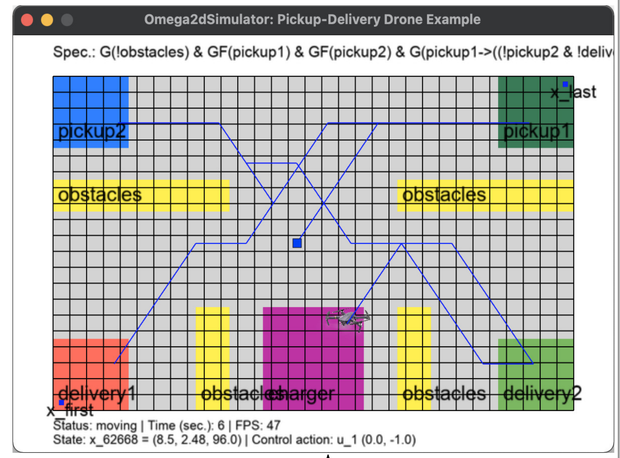
$$\begin{aligned}\psi := &\ \square(!\text{Obstacles}) \wedge \square\lozenge(\text{Pickup1}) \wedge \square\lozenge(\text{Pickup2}) \\ &\wedge \square(\text{Pickup1} \implies (!\text{Pickup2} \wedge !\text{Delivery2 U Delivery1})) \\ &\wedge \square(\text{Pickup2} \implies (!\text{Pickup1} \wedge !\text{Delivery1 U Delivery2})) \\ &\wedge \square(!\text{low\_battery}) \wedge \square\lozenge(\text{full\_battery}),\end{aligned}$$

System Dynamics

$$\Sigma : \begin{cases} \dot{x}(t) = f(x(t), u) & t \in \mathbb{R}_{\geq 0} \setminus \Omega \\ x(t) = g(x(^{-}t), u) & t \in \Omega \end{cases},$$

$$f\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\right) = \begin{bmatrix} u_1 \cos(u_2) \\ u_1 \sin(u_2) \\ 0 \end{bmatrix}, \ g\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ x_2 \\ B(x) \end{bmatrix},$$

$$B\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = \begin{cases} \min\{99,\ x_3 + 20\}, & \text{if } (x_1, x_2) \in \text{Charger}, \\ \max\{0,\ x_3 - 1\}, & \text{otherwise}, \end{cases}$$



OmegaThreads