Heptagon/BZR program

Heptagon/BZR in one slide!

state +
uncont. inputs

controller

controllable vars

automaton model

execute (outputs)

managed system

monitor
(uncontrollable inputs)

Known usages

- Discrete control of real-time tasks [Aboubekr et al., 2011]
- Control of reconfiguration in the Fractal framework [Bouhadiba et al., 2011]
- Coordination of autonomic systems administration loops [Gueye et al., 2012; Delaval et al., 2013]
- Control of reconfigurable hardware architecture [An et al., 2013]
- Control of execution of ECA rules [Cano et al., 2014]
- Fault tolerance of multi-tasks systems [Sun et al., 2015]
- Back-end of the Ctrl-F language: reconfiguration control in a component model [Alvares et al., 2015]
- Model-based design of correct controllers for dynamically reconfigurable architectures [An et al., 2016]
- Control for DPR FPGA [Gueye et al., 2018]
- Control for applications and middleware deployment in IoT and smart buildings [Sylla et al., 2018]
- Control for cybersecurity in industrial control systems [Delaval et al., 2020]
Is all this useful? How can we know?

Each of these results: work on a case study (± real...), model the system with Heptagon/BZR, apply DCS to obtain a controller

→ generic method
Is all this useful? How can we know?

Each of these results: work on a case study (± real...), model the system with Heptagon/BZR, apply DCS to obtain a controller → generic method

But... how to evaluate the *usefulness* of this method?
Is all this useful? How can we know?

Each of these results: work on a case study (± real...), model the system with Heptagon/BZR, apply DCS to obtain a controller → generic method

But... how to evaluate the usefulness of this method?

Several levels of answers:

- “it is not and/or we don’t care it’s research” (idealistic answer)
- Lenaïc Terrier’s internship (with A. Demeure, IIHM): evaluation of a DSL with qualitative/quantitative experiments with actual users (and IHM experimental methods)
- “Engineering distance” for being actually useful (materialist answer): what tools do we need? Diagnosis, debugging, integration with IDEs...
- Evaluation tools for the produced controller
What do/can we evaluate?

- the **language/tools**
  - expressivity, performances, integration or completeness of toolchain...

- the **result: the produced/synthesized controller**
  - Comparison with what? Program without controller? Program verifying the synthesis objectives, or not?
  - Evaluation level: simulation / online execution
  - Evaluation method: experiment with random inputs

→ try on Cybersecurity example: simulation of controlled system, evaluation of “how long” the system stays in nominal/degraded mode