Translating Specifications

Translating Specification

Expressing properties using either deontic logic and or ctl(*), and translating them to the other:

- **P** is a property of a given system
- A Kripke structure is implicitly defined

Translating Specification

DEONTIC ➨ CTL(*)

- **O(p)** → **AX p** is enough
- **P(p)** → **EX p** is enough
- **F(p)** → **AG ¬p**

Deontic logic is less expressive so everything is fine…

Translating Specification

CTL(*) ➔ DEONTIC LOGIC

Target logic is less expressive

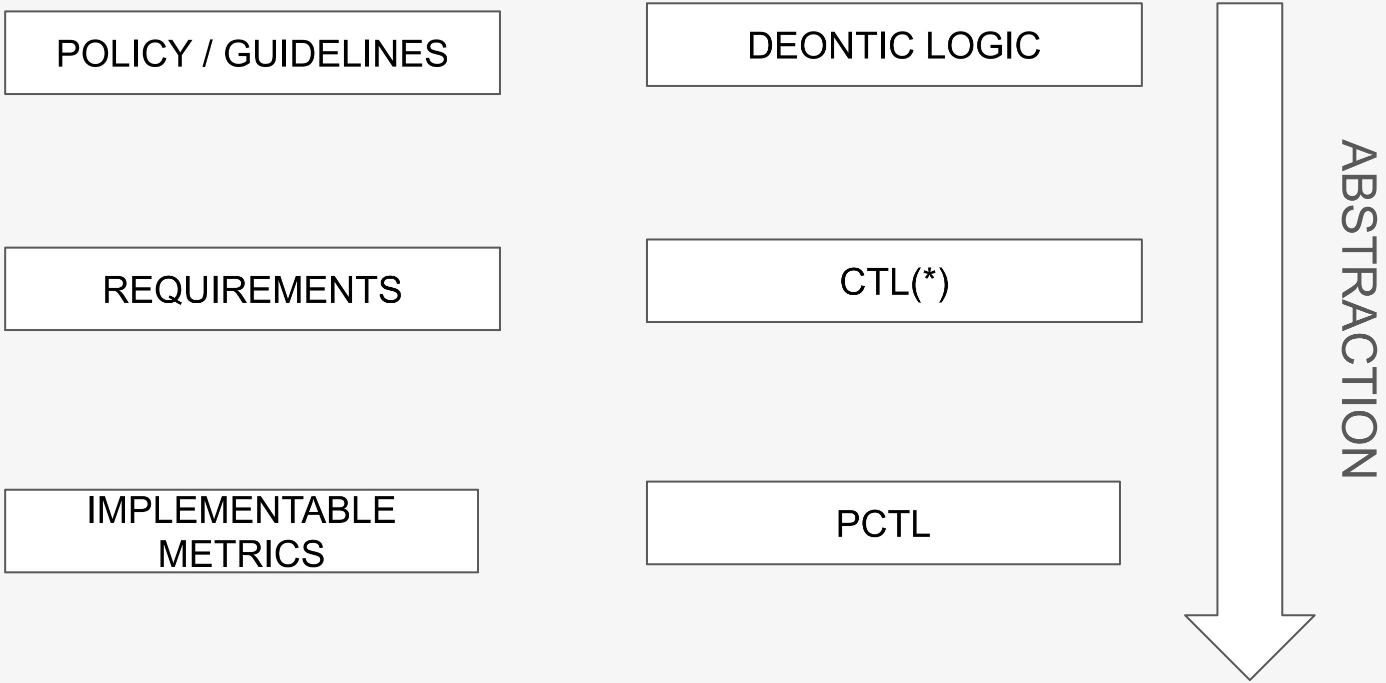
➔ INFORMATION LOSS

Defining a property p’ that works well is a solution..

Abstraction pipeline

How all of this could work in a real setting.

Abstraction pipeline



POLICY / GUIDELINES

DEONTIC LOGIC

REQUIREMENTS

CTL(*)

IMPLEMENTABLE METRICS

PCTL

ABSTRACTION

Concrete example

Putting all of that to use.

Concrete example

Let’s design a calculator :

- The calculator takes user input
- User input is a negative integer
- The calculator should not crash

Concrete example

- O(the calculator takes user input)

- P(user input is a negative integer)

- F(the calculator crashes)

Concrete example

- AΦ (user input is taken)
- EX(user input is a negative integer)
- AG ¬(the calculator crashes)

Concrete example

- P (user input is taken) = 1 (by design)

- P (the calculator crashes) < 0.05 (setting a threshold)

What’s left

The end of everything

What’s left

- Recap
- Big result presentation
- The end report

THANK YOU

Any questions? Remarks ?