Dependent types, 2-categories and Cyber-Physical System Applications

Georgios Bakirtzis Jacques Carette

October 28, 2020

These are the three papers we talked about [?, ?, ?].

In relation to CPS we talked about vertical decomposition in the form of types.

From the Lawvere understanding of a functor translating syntax to semantics to a 2-categorical perspective: one category models syntax while another models semantics.

This allows us to completely decouple how we name things – which is important in system design – and what the naming corresponds to.

We talked specifically about how a sensor in a control system can be one of many things even in the sense of behavior.

My general idea is to talk about decomposition the same way Carette talks about mathematical constructs (e.g., magma $\rightarrow$ abelian group) and also for the physical side talk about units as they are presented in Drasil.

I think merging those two idea for "cyber-physical" modeling would be a good first paper.

Will we address operads in terms of how this is different (we definitely want to address monoidal categories + extra structure)

1 Make the UAV types follow Carette

The following is very rough.

From the controls point of view there is often no external input to the sensor. This is because sensor is this context measures the system. However, while this is true almost always it is not complete with respect to the implementation of the system. The sensors play a congruent but important role, they take measurements of both the system and the environment, thereby allowing the plane to actually fly even without an internal model of the system within the controller. From a systems and software engineering point of view we want to model both the estimated state produced by the sensor and the translation of the environment to distinct physical measurements to digital signals. We could envision the type signature for the sensor as taking in the environment and state and producing a new state,

$$L : e, s \longrightarrow s'.$$

The environment e for the UAV application is a measurement of physical quantities corresponding to position, attitude or otherwise orientation, and speed. The states, s, s' represent the attitude and the speed at a particular point in time.

UAV systems are often controlled using proportional–integral–derivative (PID) controller by sending an electric signal to small motors, called servos, which in turn control the position of the plane by manipulating the aileron, rudder, elevator, and throttle. In practice, an external signal either through the ground control station or an actual pilot is given to describe the desireable flight path. In UAVs this is a set of waypoints that is sent outside the UAV.

The job of the controller is to take this desired state and apply the correct actions to fly the plane from waypoint A to waypoint B and beyond. An additional predicted state is used for error correction. We can model the controller through the type signature

$$C : (s', d) \longrightarrow \mathrm{c}$$

taking in a predicted state s', a desired state d, and producing some control action through servos c.

By controlling the different control surfaces servos ultimate control the plane's *attitude* and correspondingly the pitch, roll, and yaw through

$$D : c \longrightarrow \mathrm{s}.$$

Dynamics does a lot more in terms of airframe, position, attitude, etc. However, at this level of abstraction it is useful to think about dynamics as taking some input from the electromechanical subsystems, which is captured in the motors that control the plane, and produces a new state. This new state in the case of a UAV takes the form of movement of the plane from one waypoint to another. The compositional approach to CPS design is flexible in terms of further decompositions, meaning it is possible to decompose the mechanical elements of the UAV, for example, the airframe can be decomposed to wings, wheels, propeller, et cetera.

References

[1] J. Carette and R. O'Connor. Theory presentation combinators. In *International Conference on Intelligent Computer Mathematics*. Springer, 2012.

[2] D. R. Smith. Mechanizing the development of software. *NATO ASI Series F Computer and Systems Sciences*, 1999.

[3] D. Szymczak, S. Smith, and J. Carette. Position paper: A knowledge-based approach to scientific software development. In *Proceedings of the 2016 IEEE/ACM International Workshop on Software Engineering for Science (SE4Science)*. IEEE, 2016.