

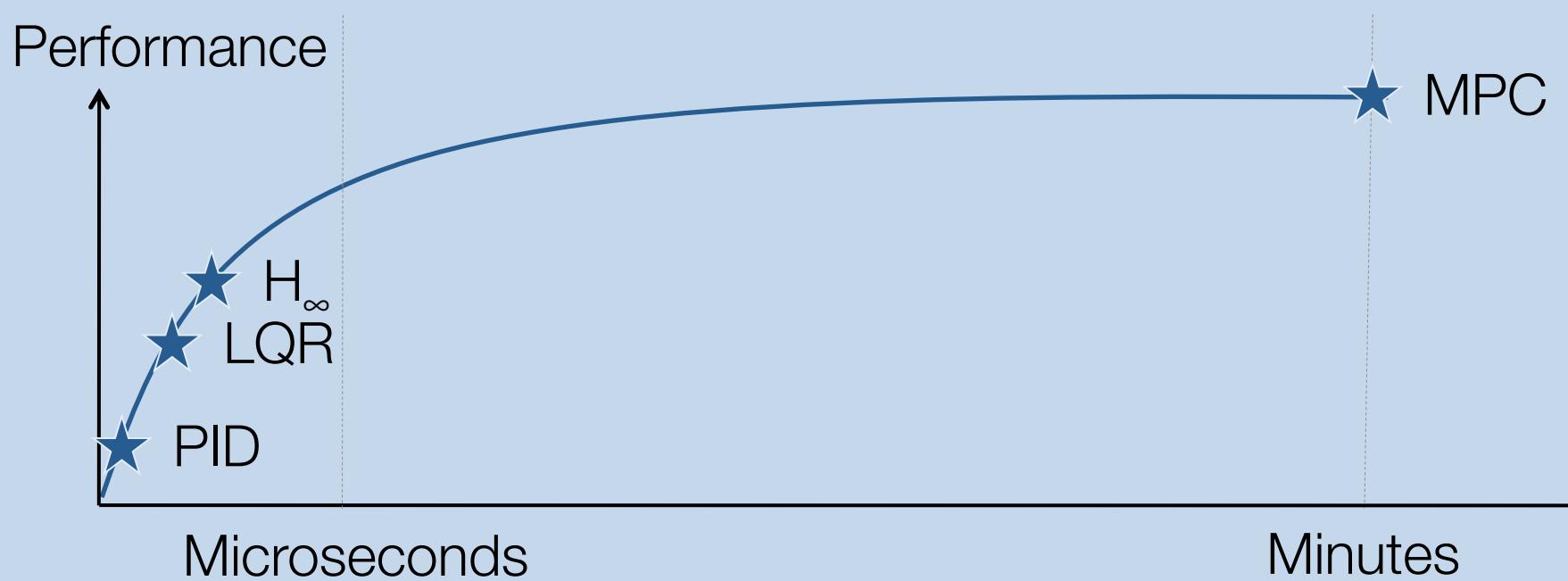
Real-Time Model Predictive Control

Colin Jones and Melanie Zeilinger



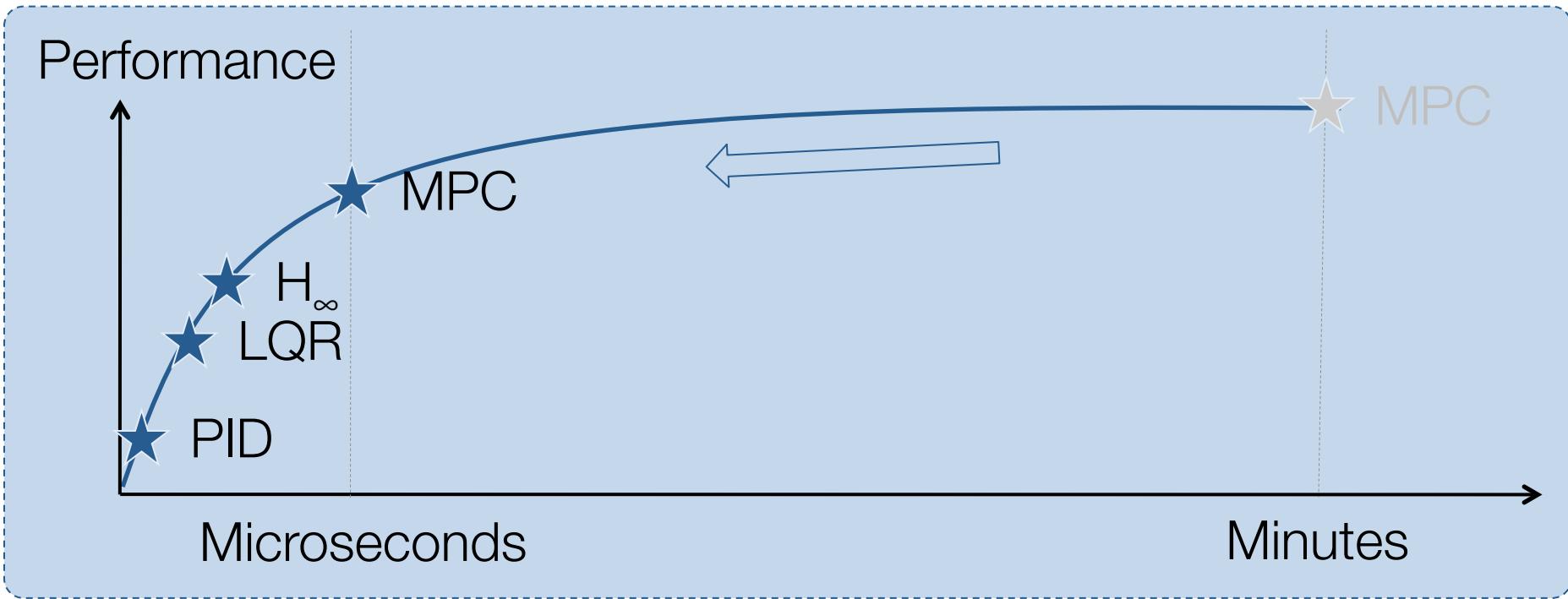
Automatic Control Laboratory, EPFL

Model Predictive Control : Computationally Challenged?



- MPC is an optimization-in-the-loop control law
 - Automatic translation from complex specification to controller
 - Reduces design and verification cost; manual synthesis errors
 - Optimizing at every sample => High performance control law
- The Myth : Suitable only for large-scale, high-cost systems

Model Predictive Control : Extreme speed



This workshop:

- Control extremely fast systems on low-cost hardware
- Critical requirement : Fast, real-time optimization

Outline : Introduction

- The intuition
- The math
- Automatic real-time synthesis : The goals and challenges

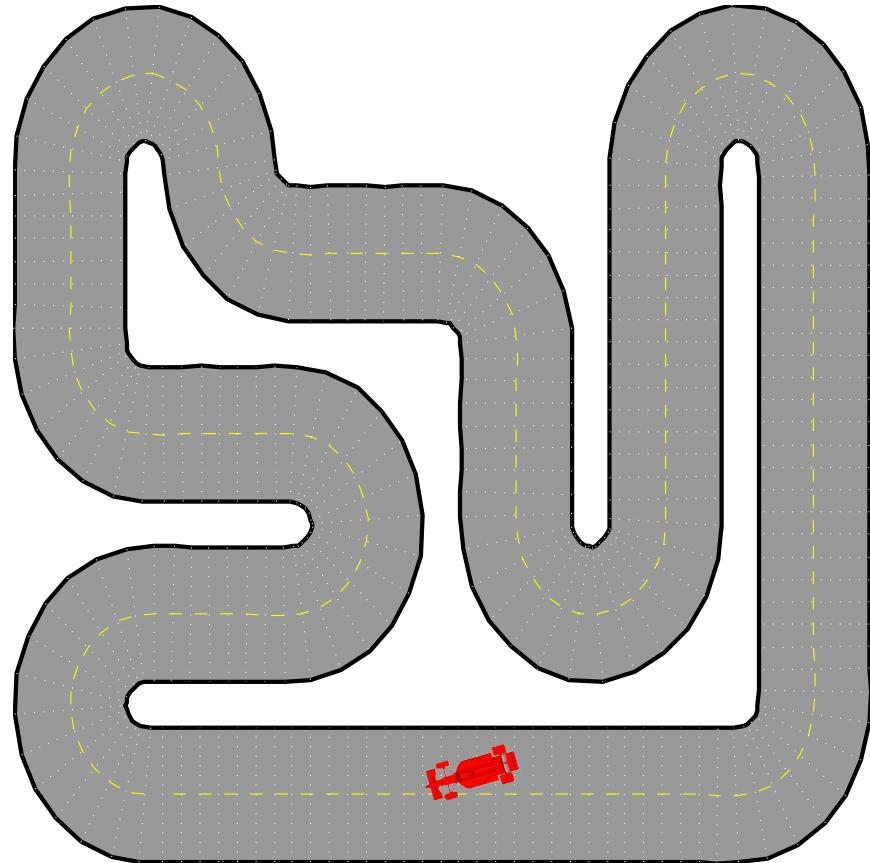
Optimization-based control: Conceptual Example

Constraints:

- Stay on road
- Don't skid
- Limited acceleration

Intuitive approach:

- Look forward and plan path based on
 - Road conditions
 - Upcoming corners
 - Abilities of car
 - etc

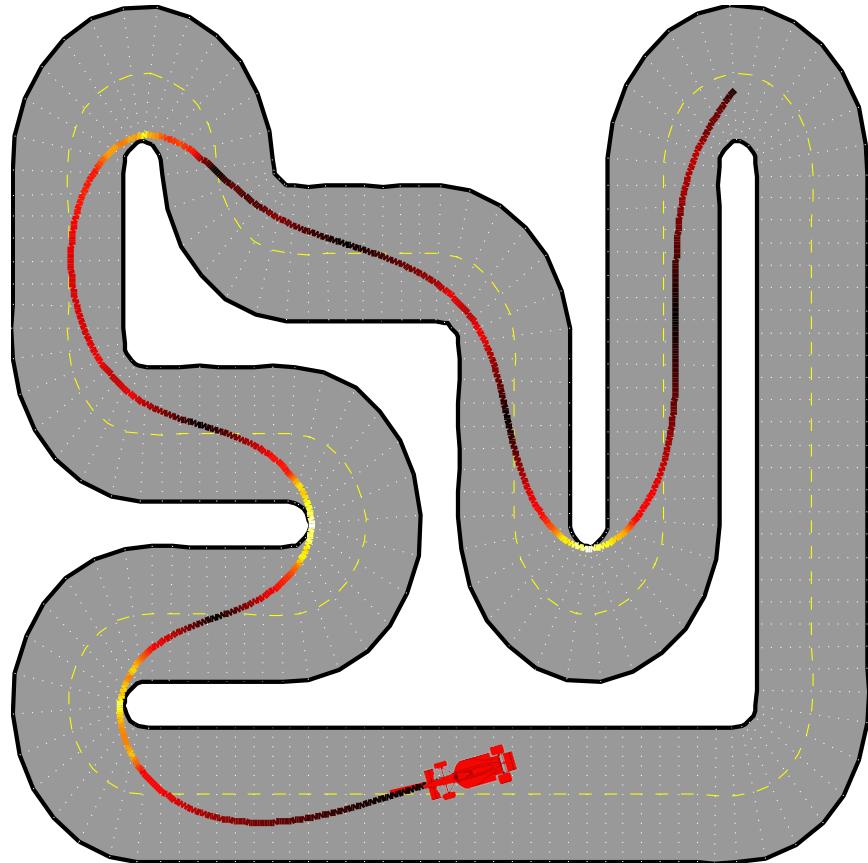


Optimization-based control: Conceptual Example

```
minimize(circuit time)  
while    avoid other cars  
        stay on road
```

...

- Solve optimization problem to compute minimum-time path
- What happens if something unexpected happens?
 - We didn't see a car around the corner!
 - Must introduce *feedback*

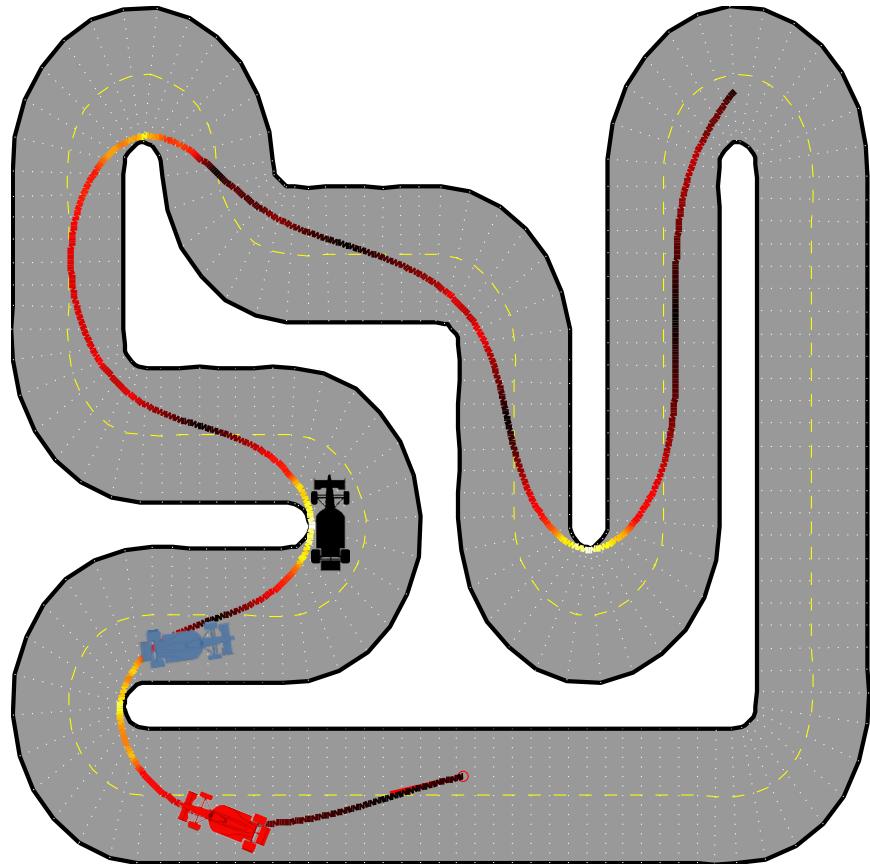


Optimization-based control: Conceptual Example

```
minimize(circuit time)  
while    avoid other cars  
        stay on road
```

...

- Solve optimization problem to compute minimum-time path
- What happens if something unexpected happens?
 - We didn't see a car around the corner!
 - Must introduce *feedback*

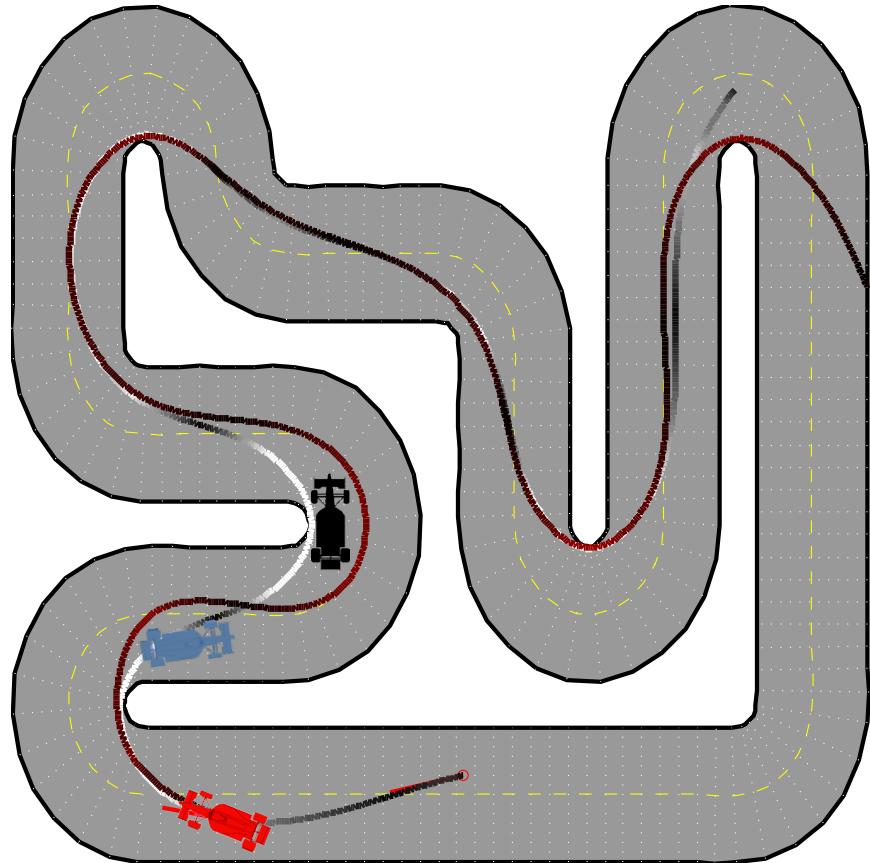


Optimization-based control: Conceptual Example

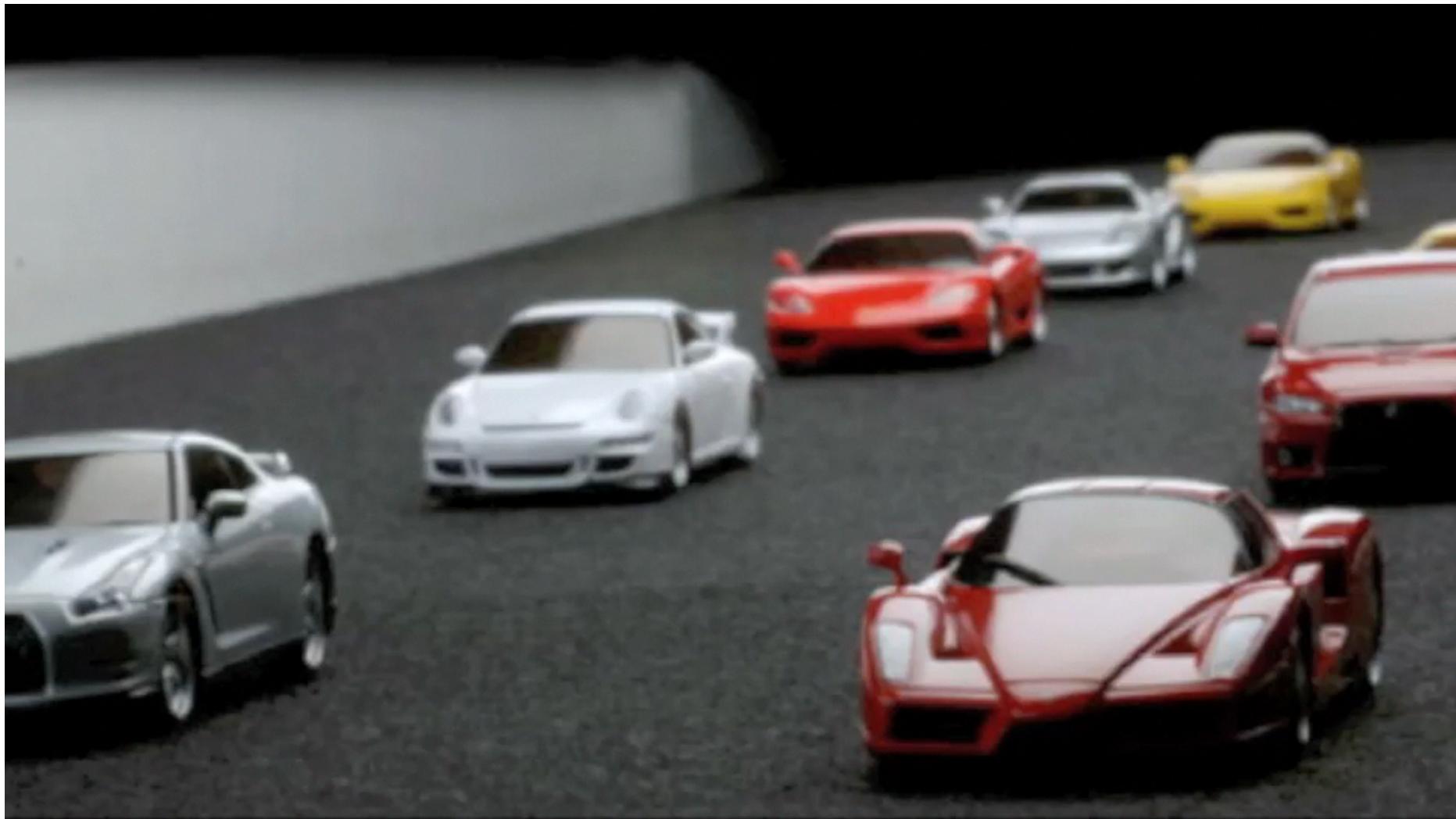
```
minimize(circuit time)  
while    avoid other cars  
        stay on road
```

...

- Solve optimization problem to compute minimum-time path
- What happens if something unexpected happens?
 - We didn't see a car around the corner!
 - Must introduce *feedback*



Putting it all together : OrcaRacer



Outline : Introduction

- The intuition
- The math
- Automatic real-time synthesis : The goals and challenges

Receding horizon control : Mathematical formulation

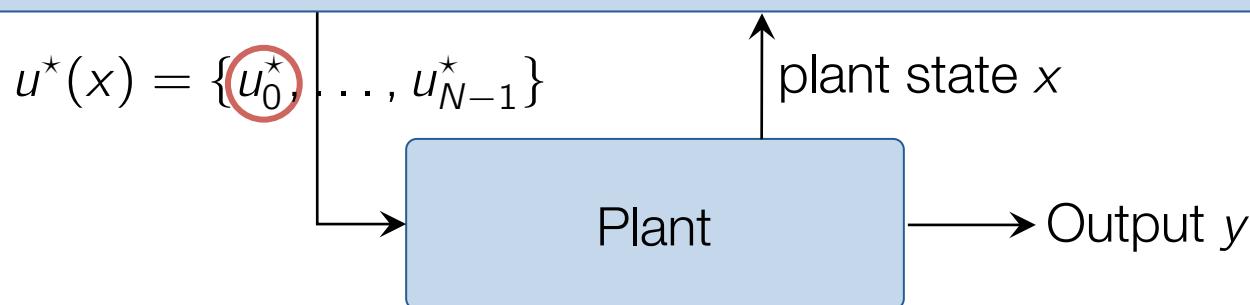
$$\begin{aligned}
u^*(x) := \operatorname{argmin} \quad & x_N^T Q_f x_N + \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i \\
\text{s.t.} \quad & x_0 = x \quad \text{measurement} \\
& x_{i+1} = Ax_i + Bu_i \quad \text{system model} \\
& Cx_i + Du_i \leq b \quad \text{constraints} \\
& R \succ 0, Q \succ 0 \quad \text{performance weights}
\end{aligned}$$

Cost function measures distance from origin

- Also possible to express much more complex constraints / objectives

Receding horizon control : Mathematical formulation

$$\begin{aligned}
u^*(x) := \operatorname{argmin} \quad & x_N^T Q_f x_N + \sum_{i=0}^{N-1} x_i^T Q x_i + u_i^T R u_i \\
\text{s.t.} \quad & x_0 = x \quad \text{measurement} \\
& x_{i+1} = Ax_i + Bu_i \quad \text{system model} \\
& Cx_i + Du_i \leq b \quad \text{constraints} \\
& R \succ 0, Q \succ 0 \quad \text{performance weights}
\end{aligned}$$



Each sample time:

1. Measure / estimate state
 2. Solve optimization problem for entire planning window
 3. Implement only the *first* control action

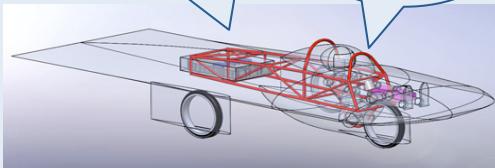
Outline : Introduction

- The intuition
- The math
- Automatic real-time synthesis : The goals and challenges

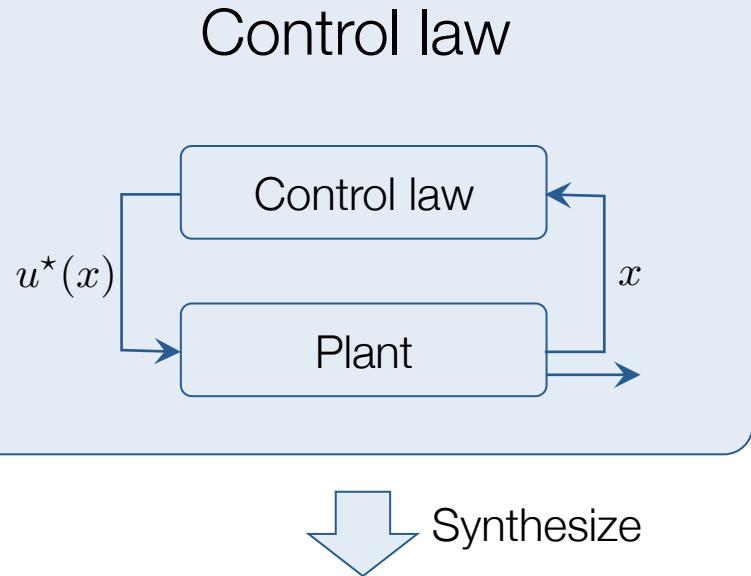
Control : The Holy Grail

Formal specification

Maximize fuel economy



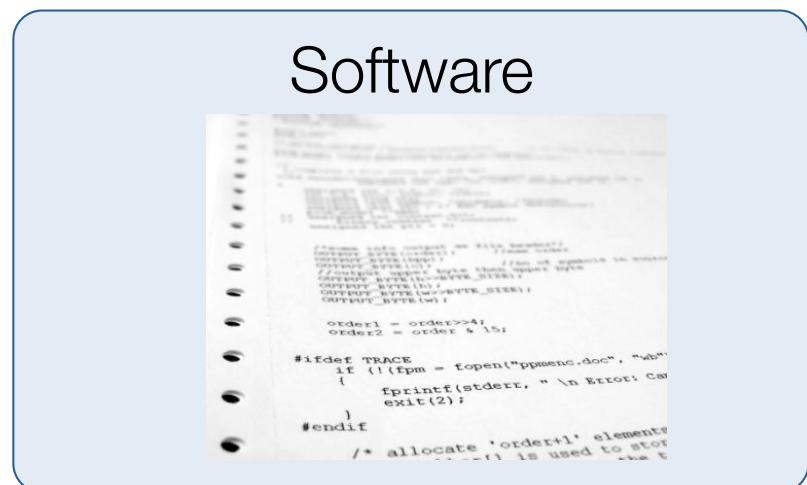
Don't slip



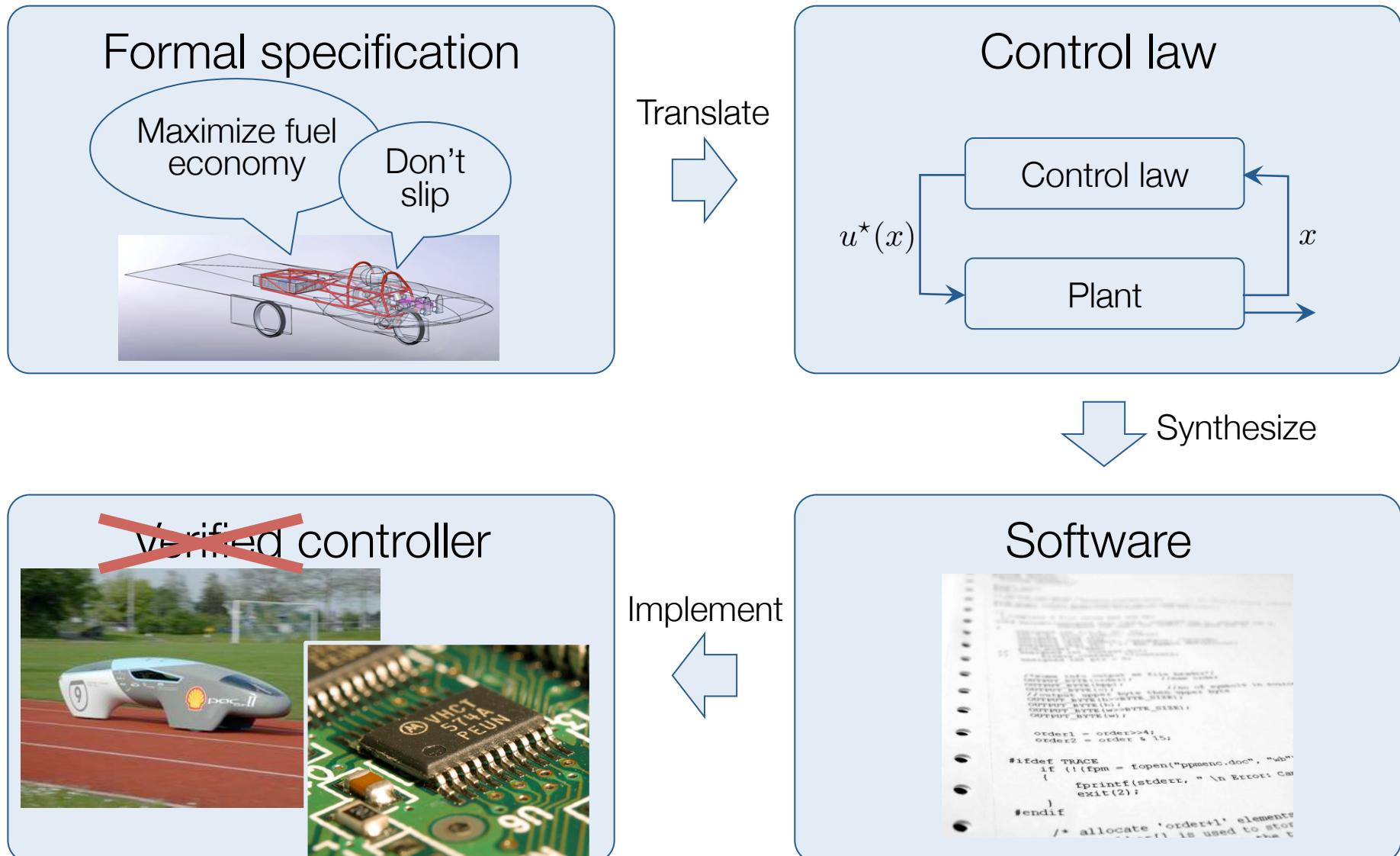
Verified system



Implement



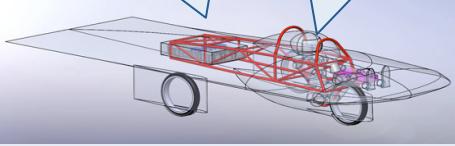
The challenge : Real-time constraints



Key principle : Computation is part of the spec

Formal specification

Maximize fuel economy



Don't slip

Use this processor



Translate



Control law



Verified system



Implement



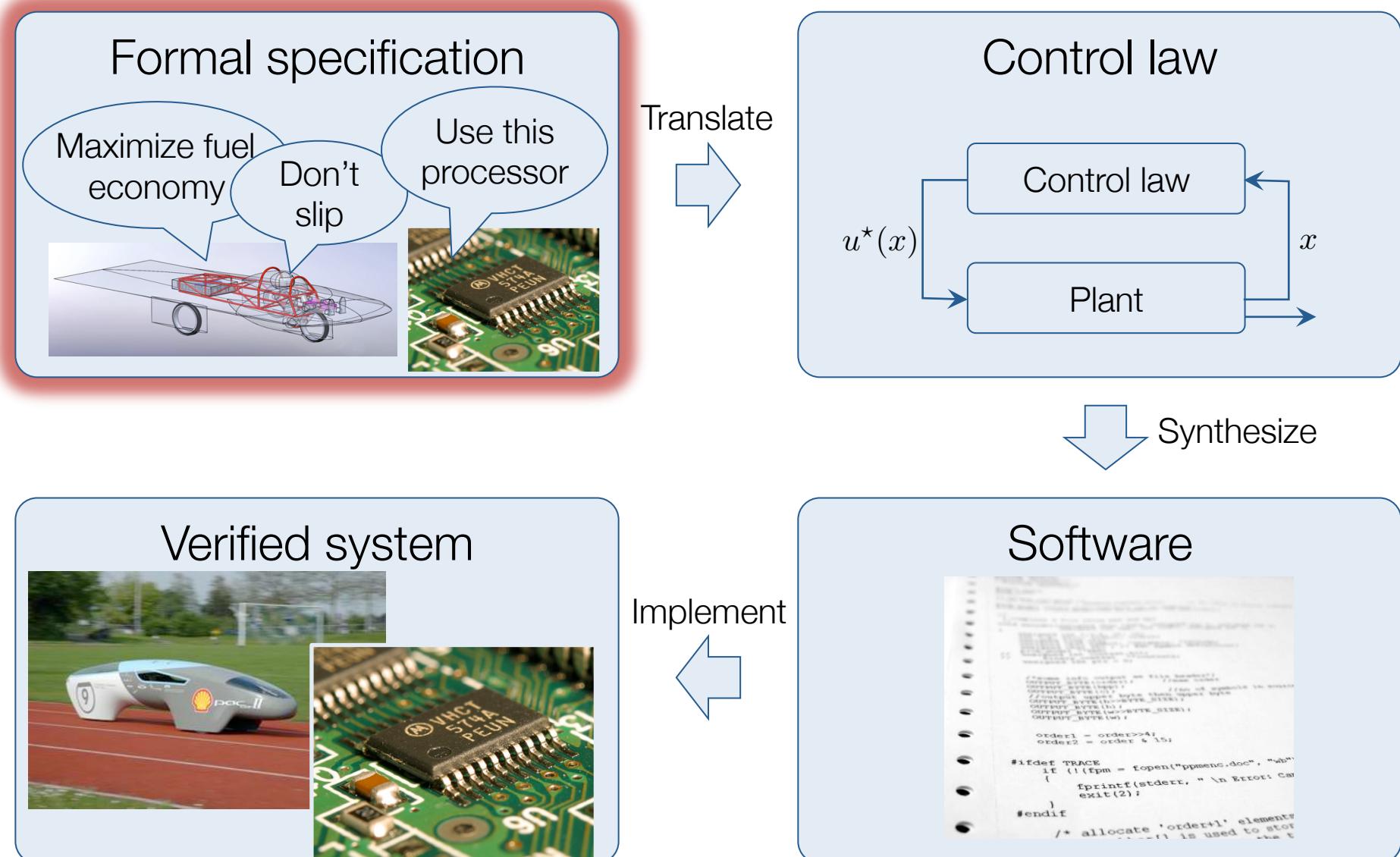
Software

```
/* This code is generated by the C2C tool. It contains
 * comments and annotations that are not part of the original
 * specification. These annotations are used to help
 * the tool understand the context of the code.
 *
 * The code is intended to be used in a real-time
 * environment where the plant's state is updated
 * periodically. The control law is implemented
 * using a discrete-time model predictive control
 * (MPC) approach.
 *
 * The code uses a fixed-point arithmetic library
 * to handle numerical precision issues. The
 * variables are scaled by powers of two to
 * prevent overflow.
 *
 * The code includes error handling and trace
 * functionality. It checks for errors in the
 * input data and prints error messages to
 * standard error if they occur. It also
 * prints trace information to a file named
 * ppmenc.doc.
 */

// Define the plant's state and control inputs
// ...
// Define the control law's parameters
// ...
// Define the optimization problem
// ...
// Implement the control law
// ...
// Implement the plant
// ...
// Implement the error handling
// ...
// Implement the trace functionality
// ...
// Implement the scaling and fixed-point arithmetic
// ...
// Implement the main loop
// ...

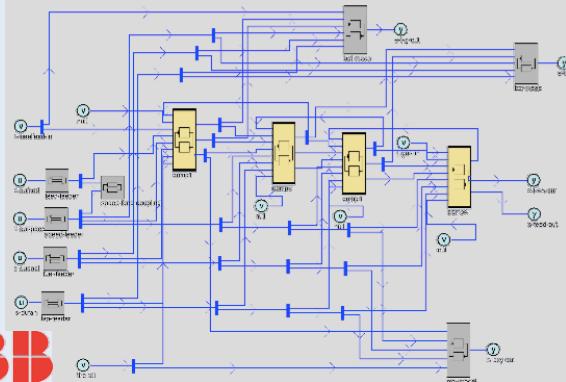
// Function prototypes
// ...
// Variable declarations
// ...
// Main function
int main()
{
    // Initialize the plant and control law
    // ...
    // Enter the main loop
    while (true)
    {
        // Get the plant's state
        // ...
        // Compute the control signal
        // ...
        // Set the plant's state
        // ...
        // Check for errors
        // ...
        // Print trace information
        // ...
    }
}
```

Key principle : Computation is part of the spec



Translate formal spec => MPC control law

Formal specification



Translate

Control law

$$u^*(x) = \operatorname{argmin}_{u_i} \sum_{i=0}^{N-1} l(x_i, u_i)$$

s.t. $x_{i+1} = f(x_i, u_i)$
 $(x_i, u_i) \in \mathcal{X} \times \mathcal{U}$
 $x_0 = x$

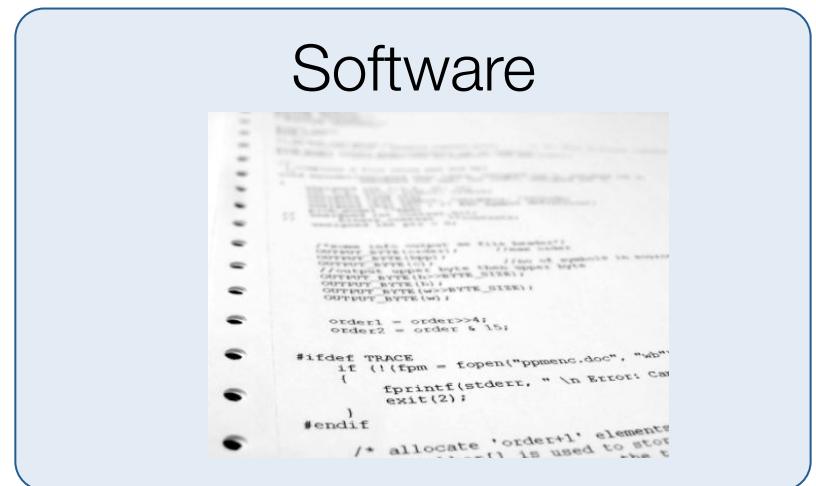
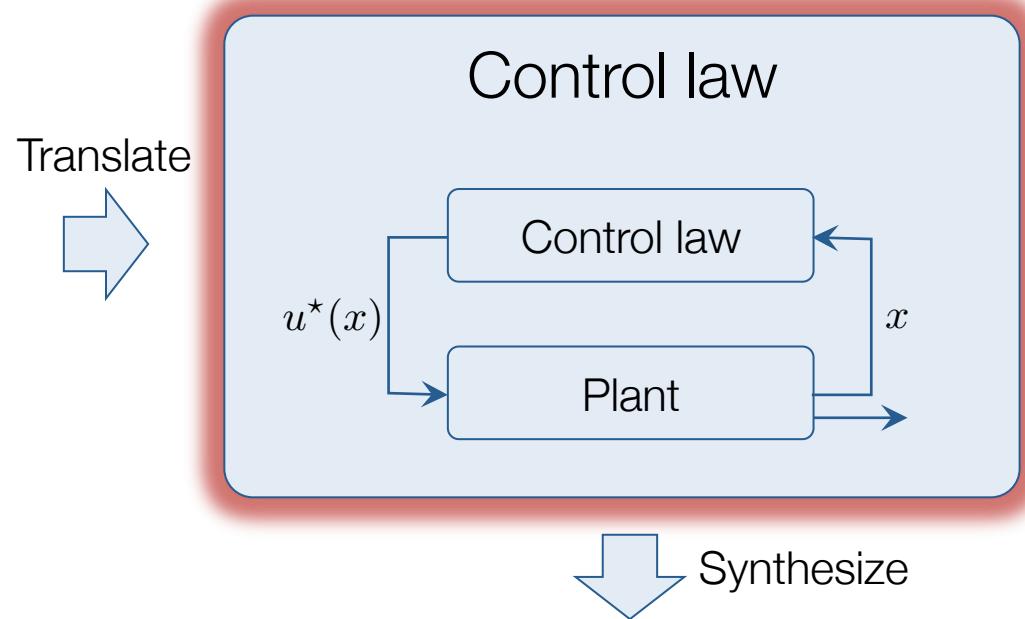
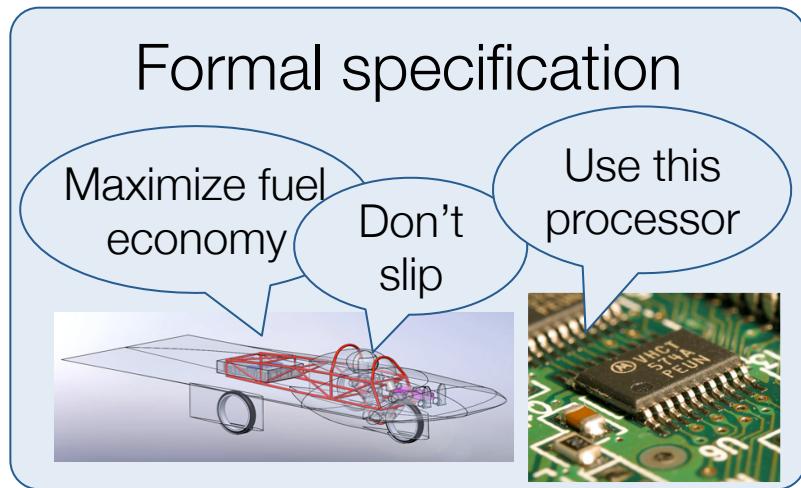
Control problem compilers:

- HYSDEL *Kvasnica, et al*
- MPT *Kvasnica, et al*
- OPTIMICA *Modelon*
- ACADO *Houska, et al*
- ...

MPC control law:

- Optimal trajectory subject to system limitations
- Re-optimize at each sample
⇒ Feedback control

Key principle : Computation is part of the spec



MPC : The fine print

'Classic' MPC has no guarantee of stability or constraint satisfaction

but...

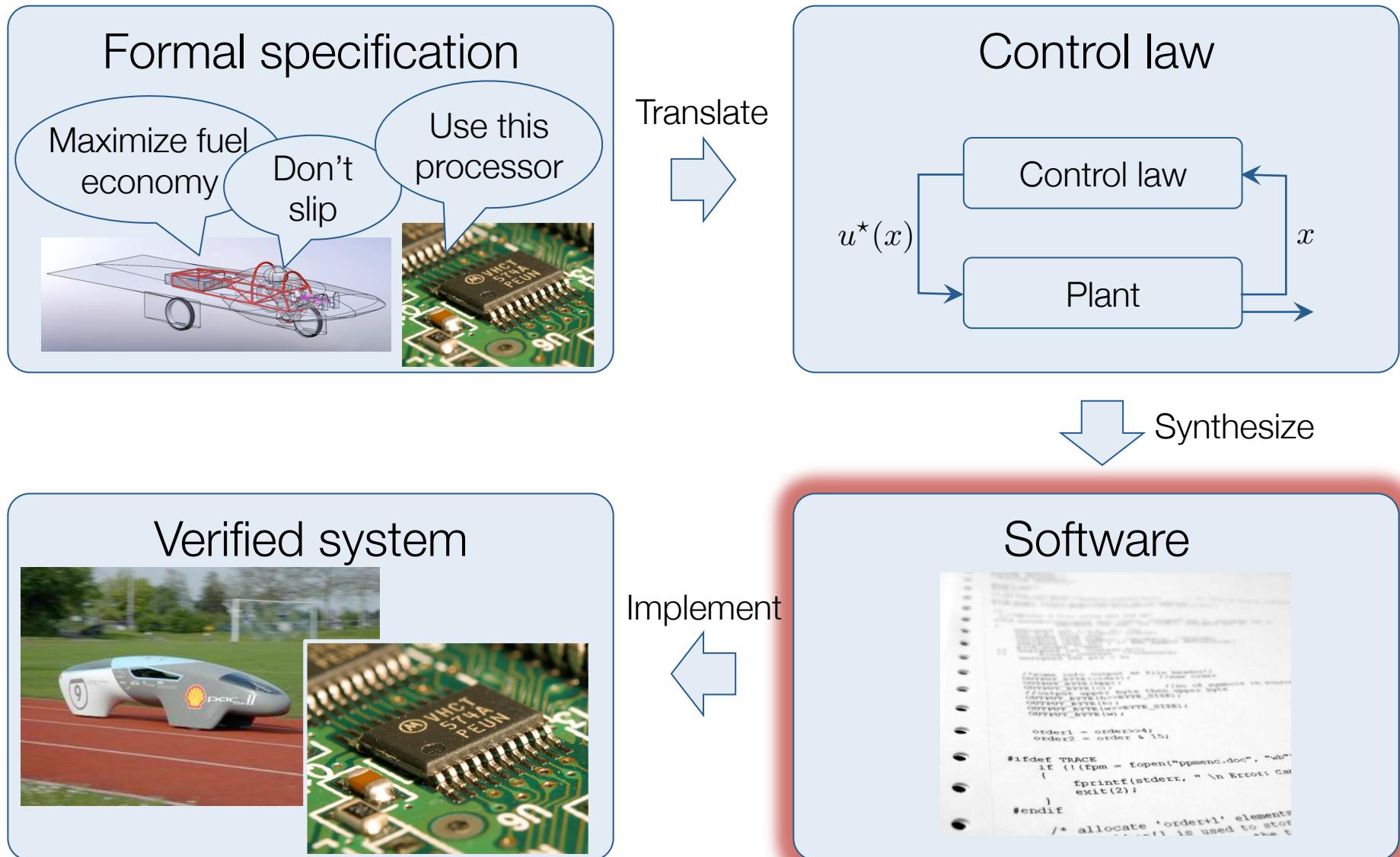
Theory is now reasonably well-developed

- Automatic augmentation of problem for range of systems ensuring
 - (Robust) constraint satisfaction
 - Stability
- Covered in first part of the lecture

$$J^*(x_0) = \min_{u_i} V_N(x_N) + \sum_{i=0}^{N-1} l(x_i, u_i)$$

s.t. $x_{i+1} = f(x_i, u_i)$
 $(x_i, u_i) \in \mathcal{X} \times \mathcal{U}$
 $x_N \in \mathcal{X}_N$

Key principle : Computation is part of the spec



MPC Controller Synthesis

Large-scale, slow

Generate problem data

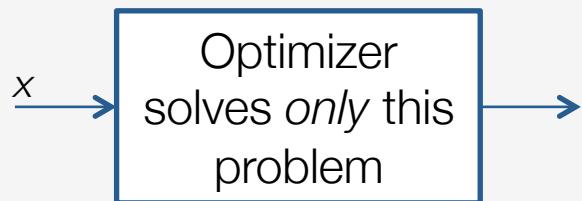
offline
online



Medium-scale, fast

Generate custom code
to solve problem

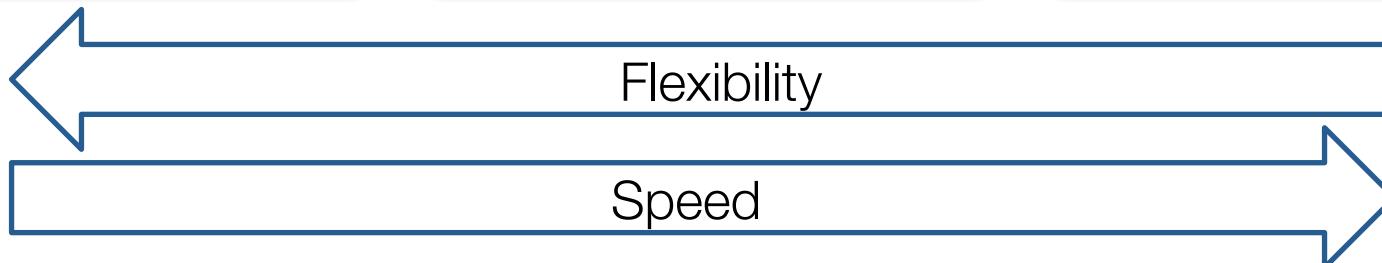
offline
online



Small-scale, ultra-fast

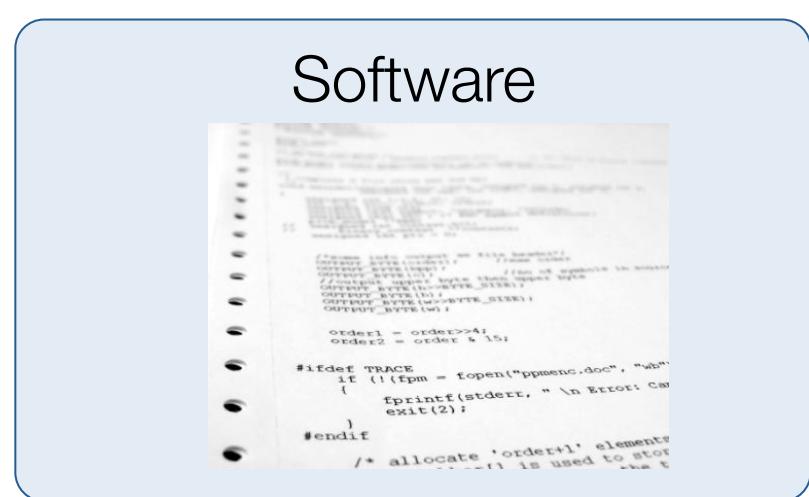
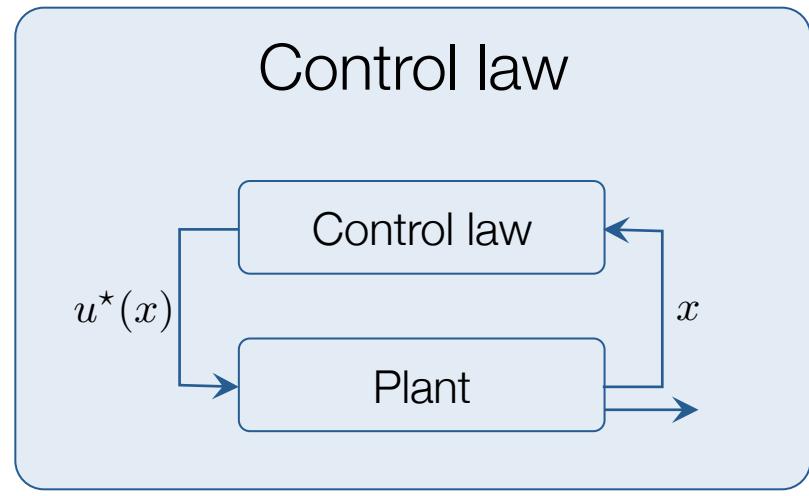
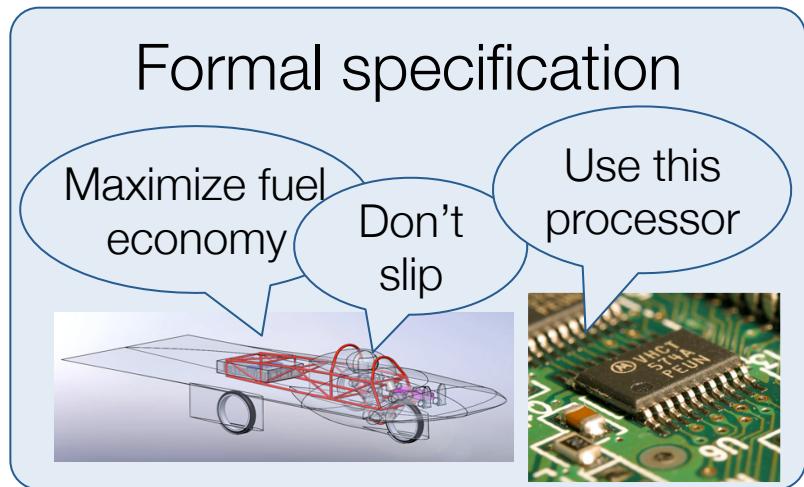
Pre-compute entire
control law

offline
online



- Ideal approach is problem specific

Key principle : Computation is part of the spec



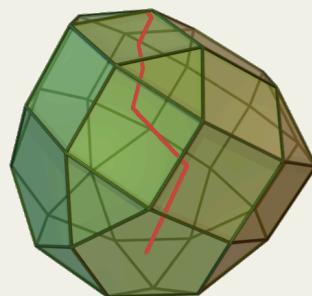
Real-time synthesis : Complexity as a specification

MPC problem

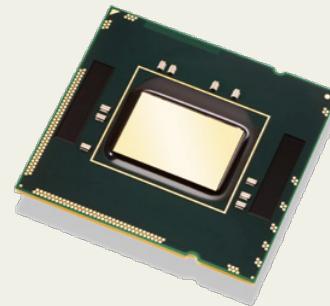
$$J^*(x_0) = \min_{u_i} V_N(x_N) + \sum_{i=0}^{N-1} l(x_i, u_i)$$

s.t. $x_{i+1} = f(x_i, u_i)$
 $(x_i, u_i) \in \mathcal{X} \times \mathcal{U}$
 $x_N \in \mathcal{X}_N$

Computational
method



Embedded
Processor



Real
time!



- Hardware platform bounds computation *time* and *storage*
- Complexity is a function of the problem
 - Uncertain and difficult to estimate and/or bound apriori
- Active area of research : Real-time MPC
 - Sub-optimal, but stabilizing controller in specified time
- 3rd lecture this afternoon