NICTA

Automatic Device Driver Synthesis

Alexander Legg
Leonid Ryzhyk
Adam Walker

(NICTA/UNSW)

John Keys
Arun Raghunath
Mona Vij

(INTEL)

Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners

Motivation

- The joys of driver development
  - Poor hardware documentation
  - Poor OS documentation
  - Debugging drivers is hard

A problem has been detected and windows has been shut down to prevent damage to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any windows updates you might need.

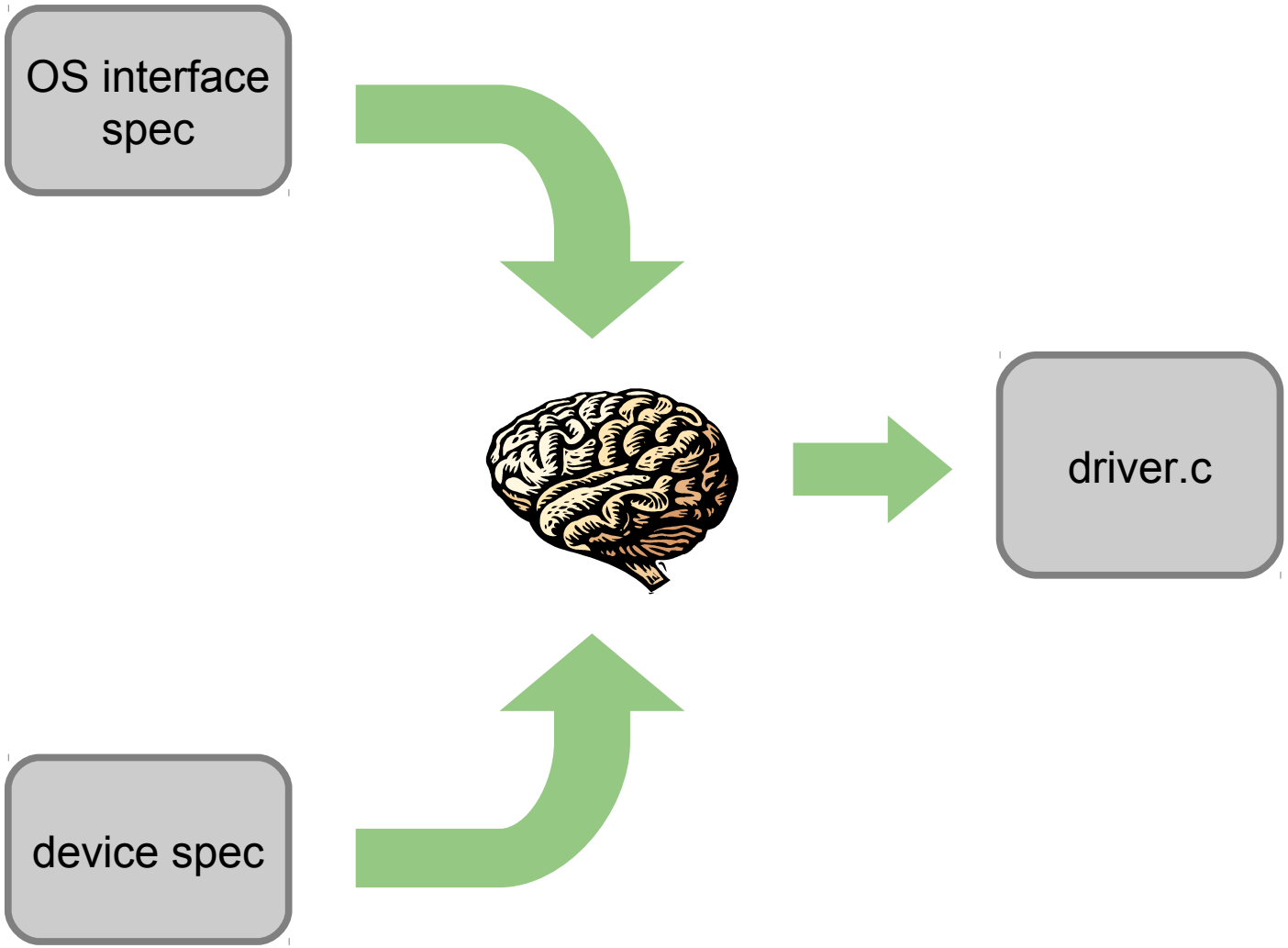
If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

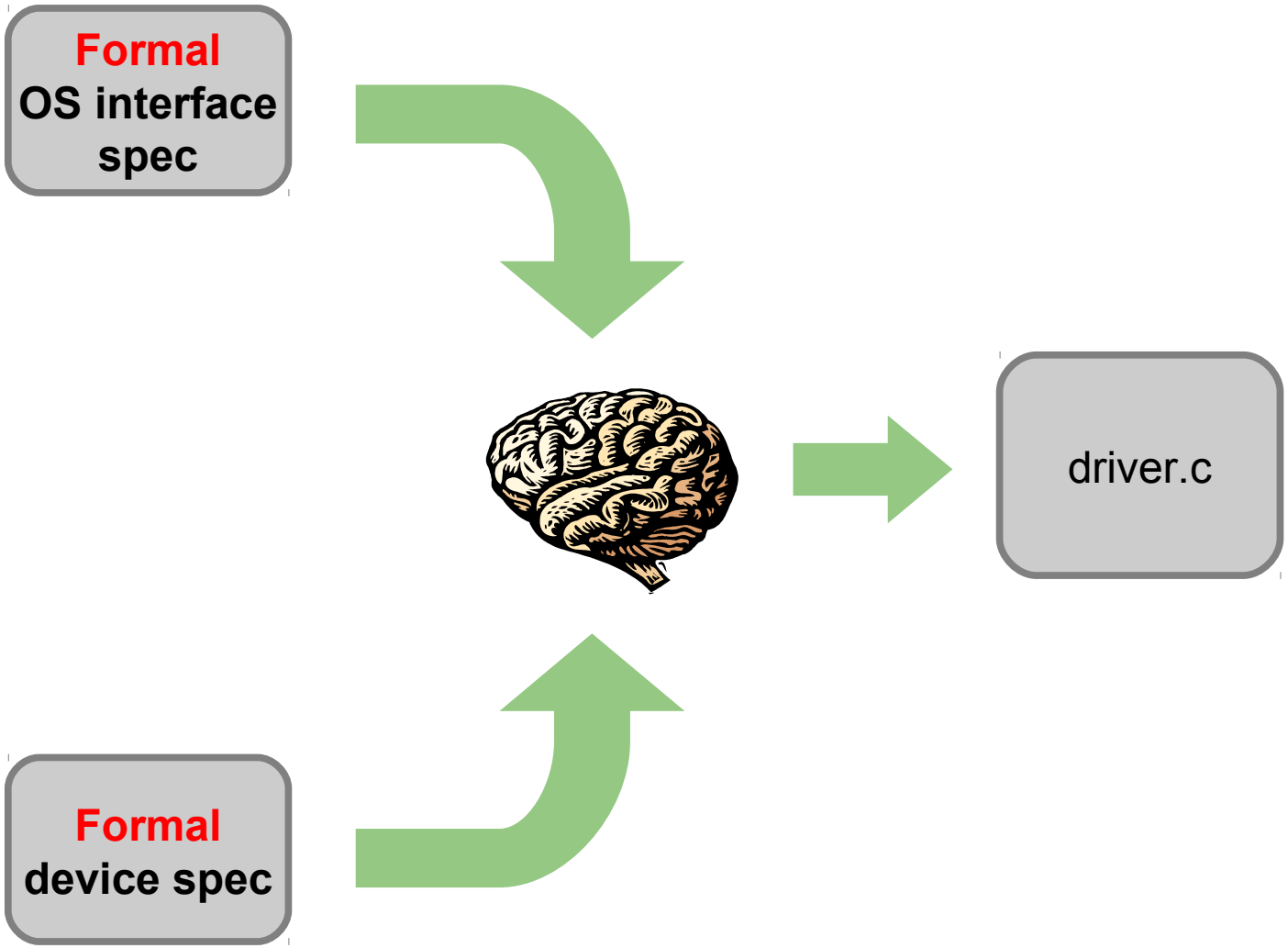
*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)

*** SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, Datestamp 3d6dd67c

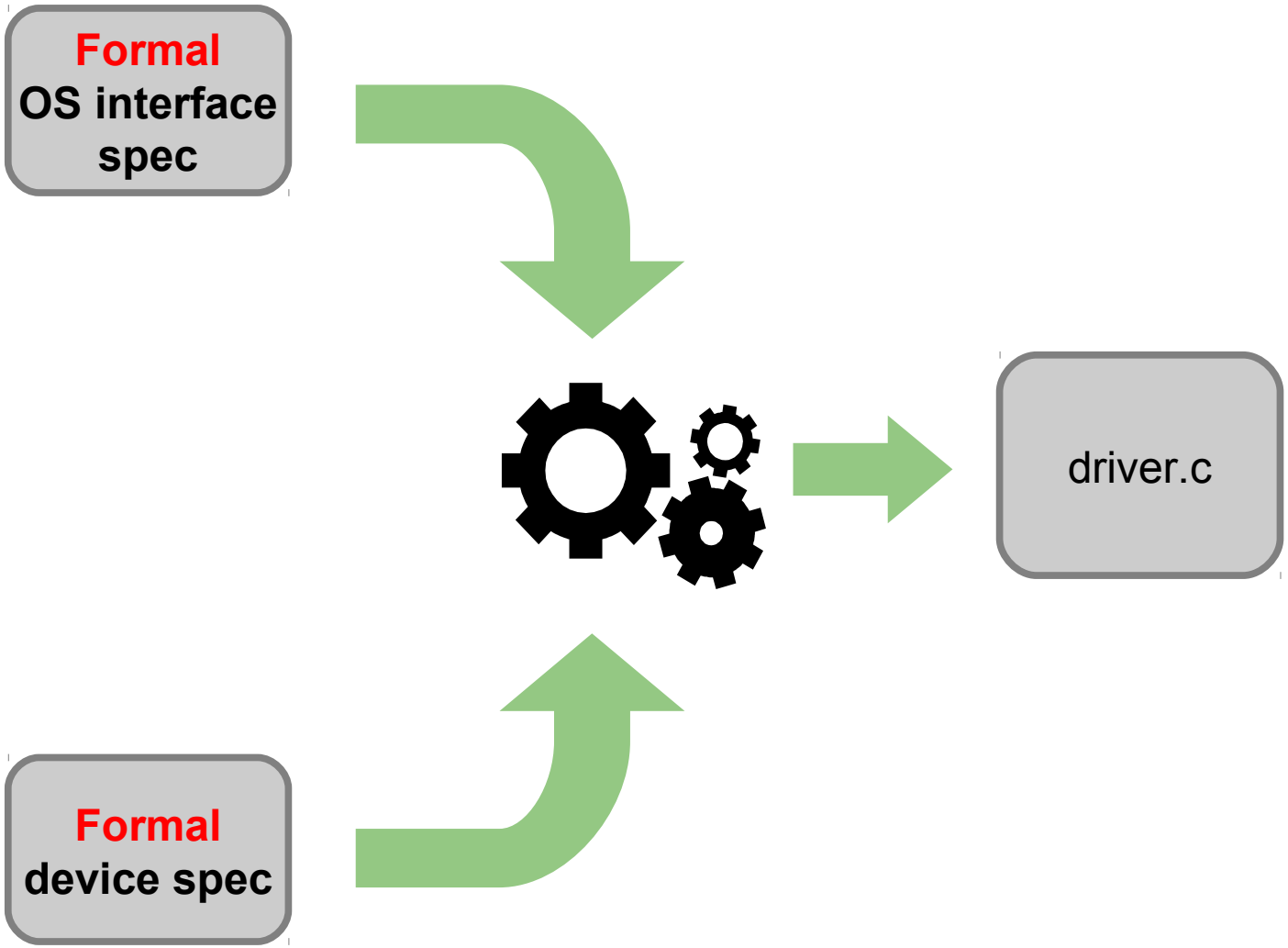
Conventional driver development



Driver synthesis with Termite



Driver synthesis with Termite



Termite is **not** ...

- … a domain-specific language for drivers

```
FEATURE device_read {
        OUTPUT char data;
        poll LSR.dr until (LSR.dr == 1) data = RBR
        ERROR (LSR.oe == 0x1 || LSR.pe == 0x1);
}
```

- … a register description language

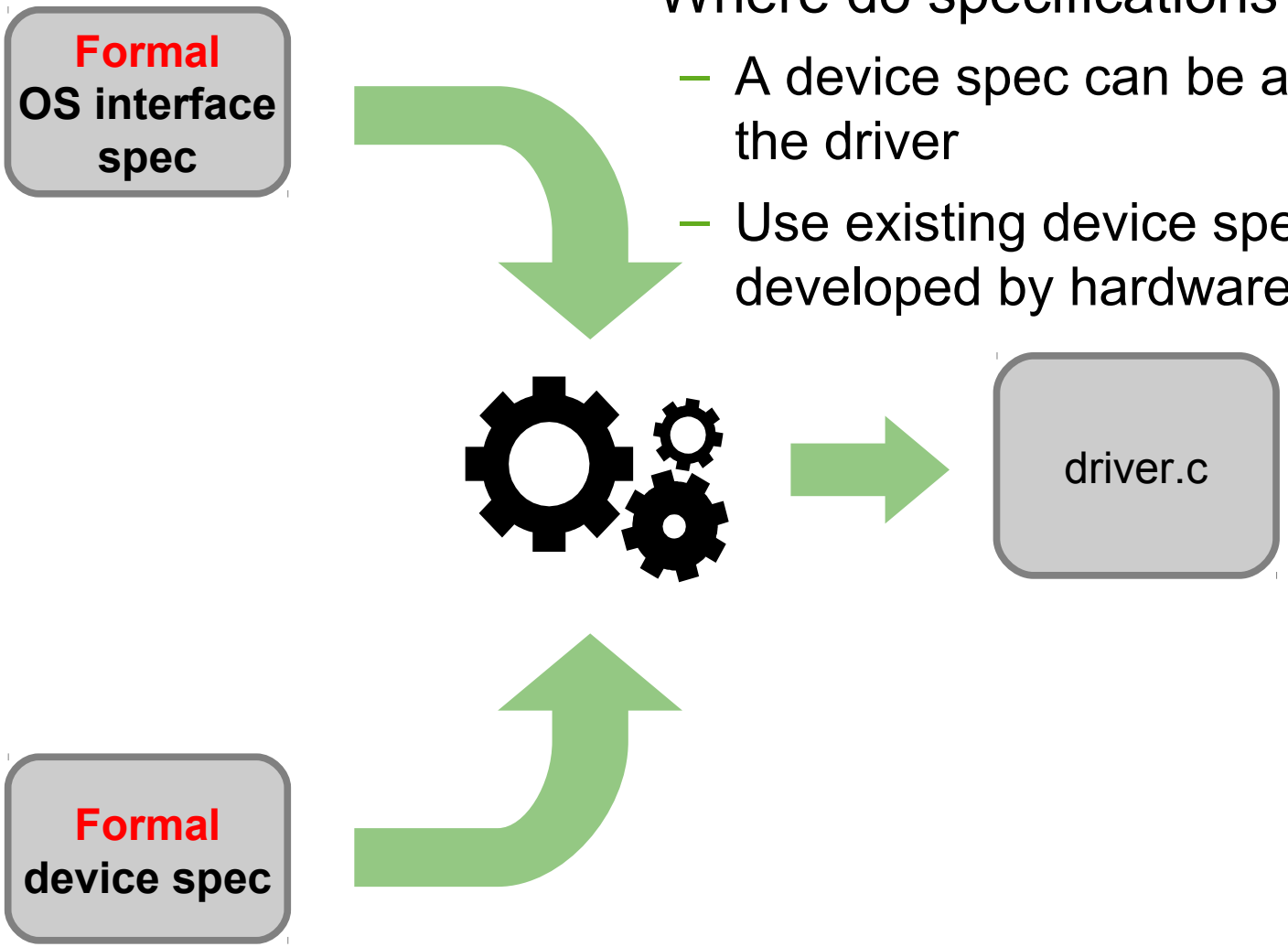
```
device logitech_busmouse(base : bit[8] port@{0..3})
{
        register sig_reg = base @ 1 : bit[8];
        variable signature = sig_reg, ... : int(8);
}
```

- … a theoretical exercise
  - the goal is to synthesise efficient drivers for real-world devices (network, storage, audio, etc.)

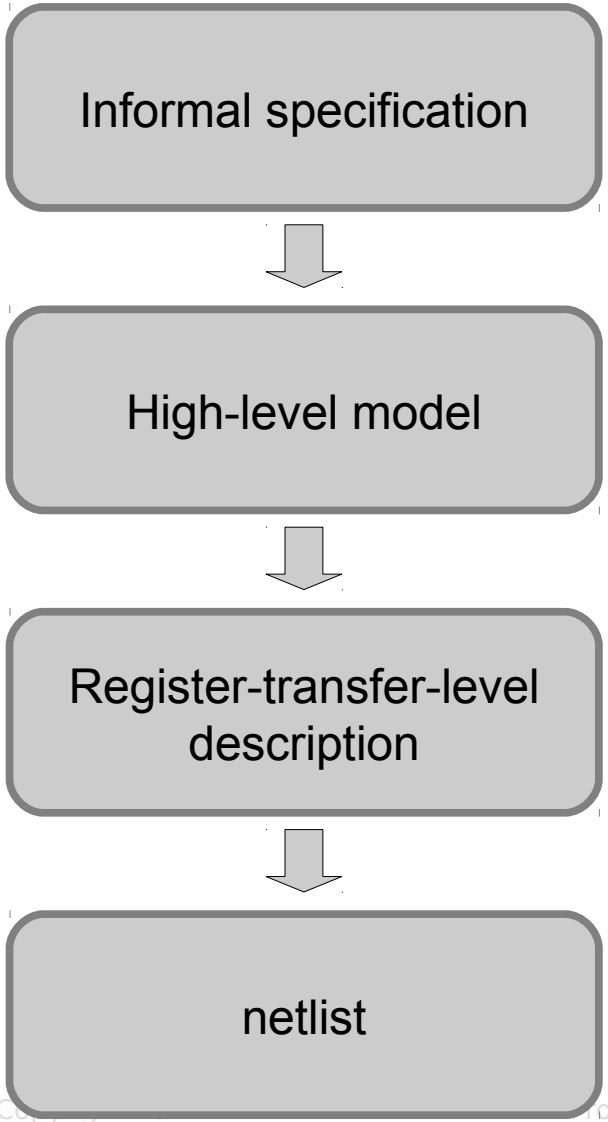
Driver synthesis with Termite

NICTA

- Where do specifications come from?
  - A device spec can be as complex as the driver
  - Use existing device specifications developed by hardware designers



Hardware Design Workflow



Hardware Design Workflow

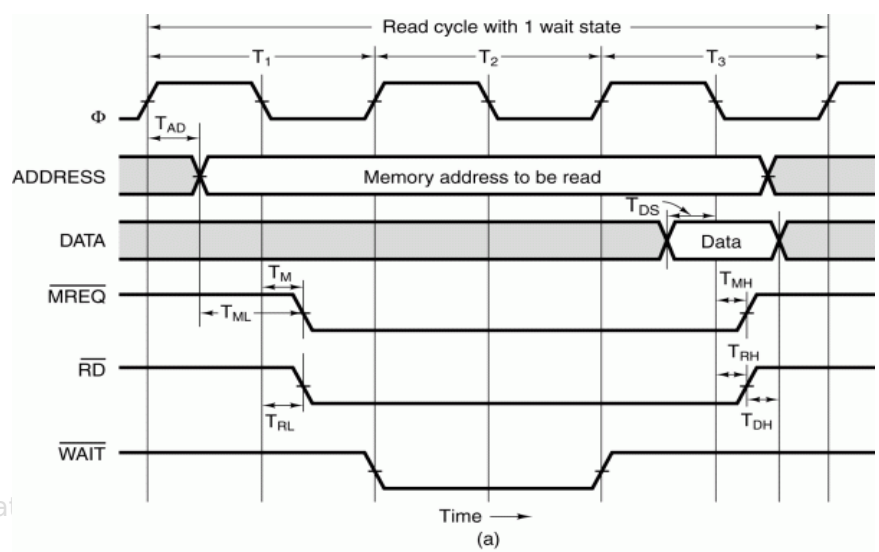
Informal specification

High-level model

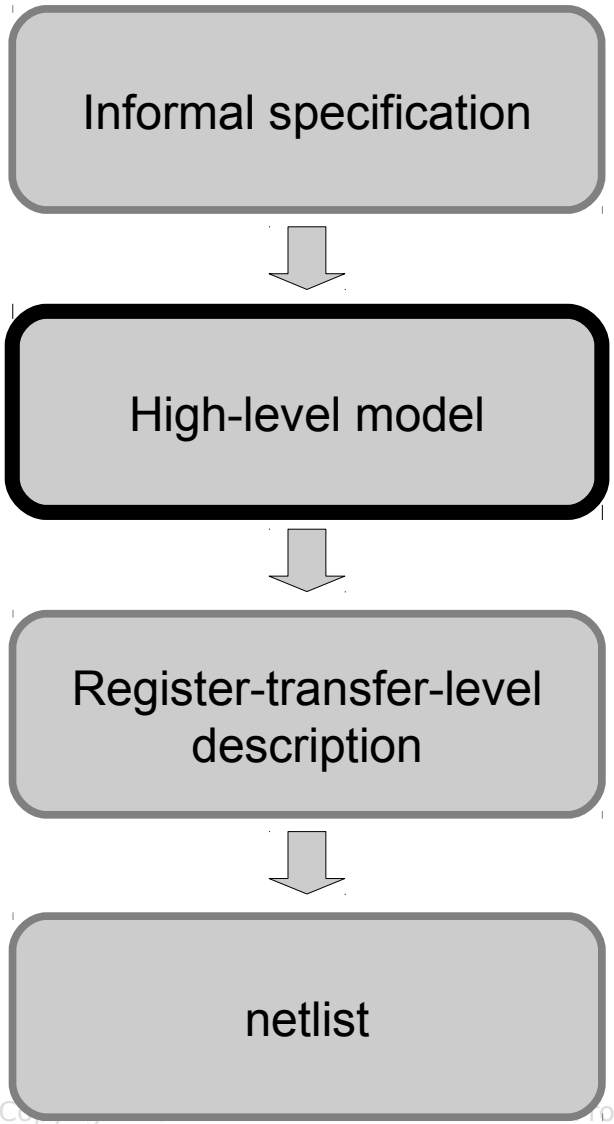
Register-transfer-level description

netlist

- Low-level description: registers, gates, wires.
- Cycle-accurate
- Precisely models internal device architecture and interfaces



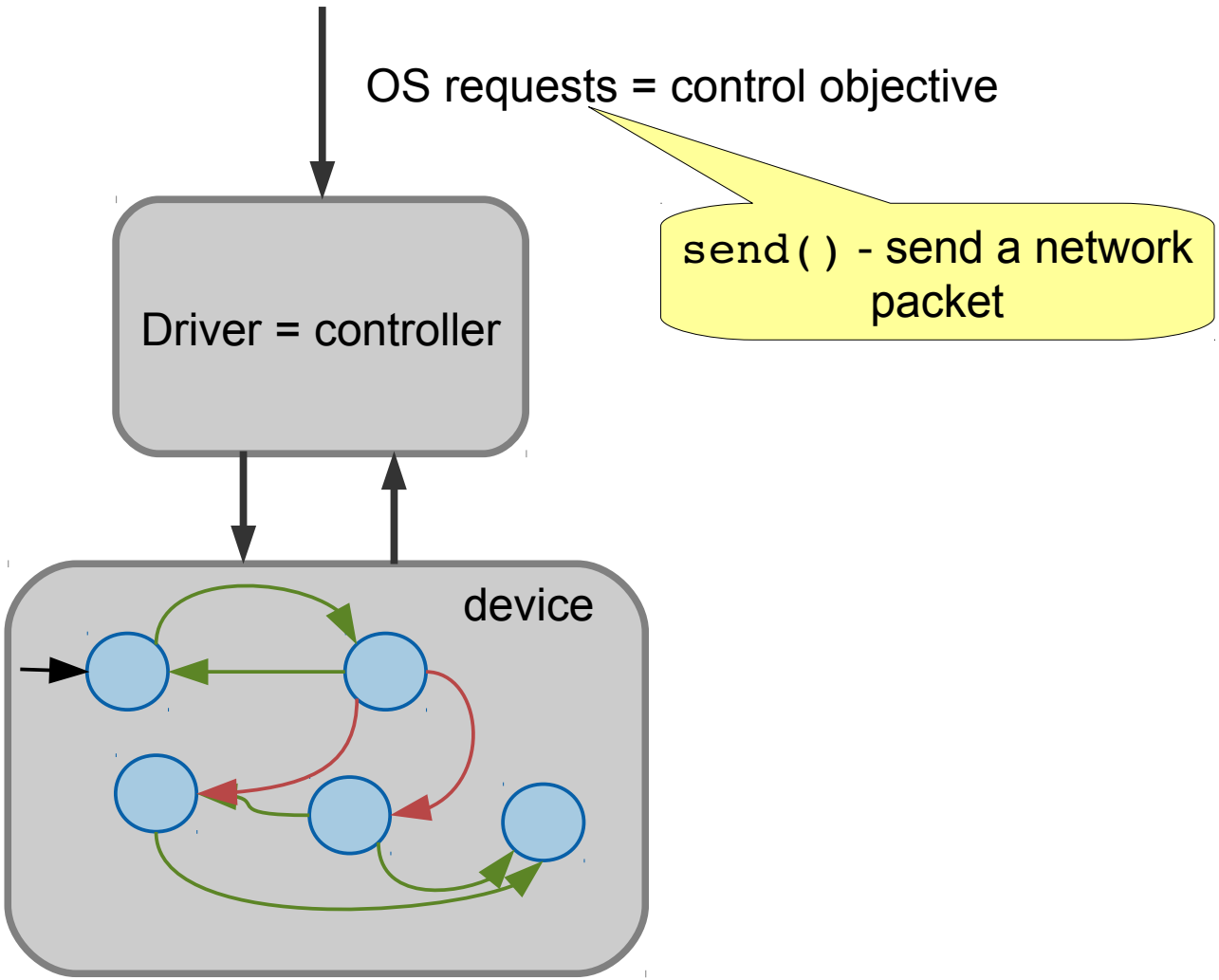
Hardware Design Workflow



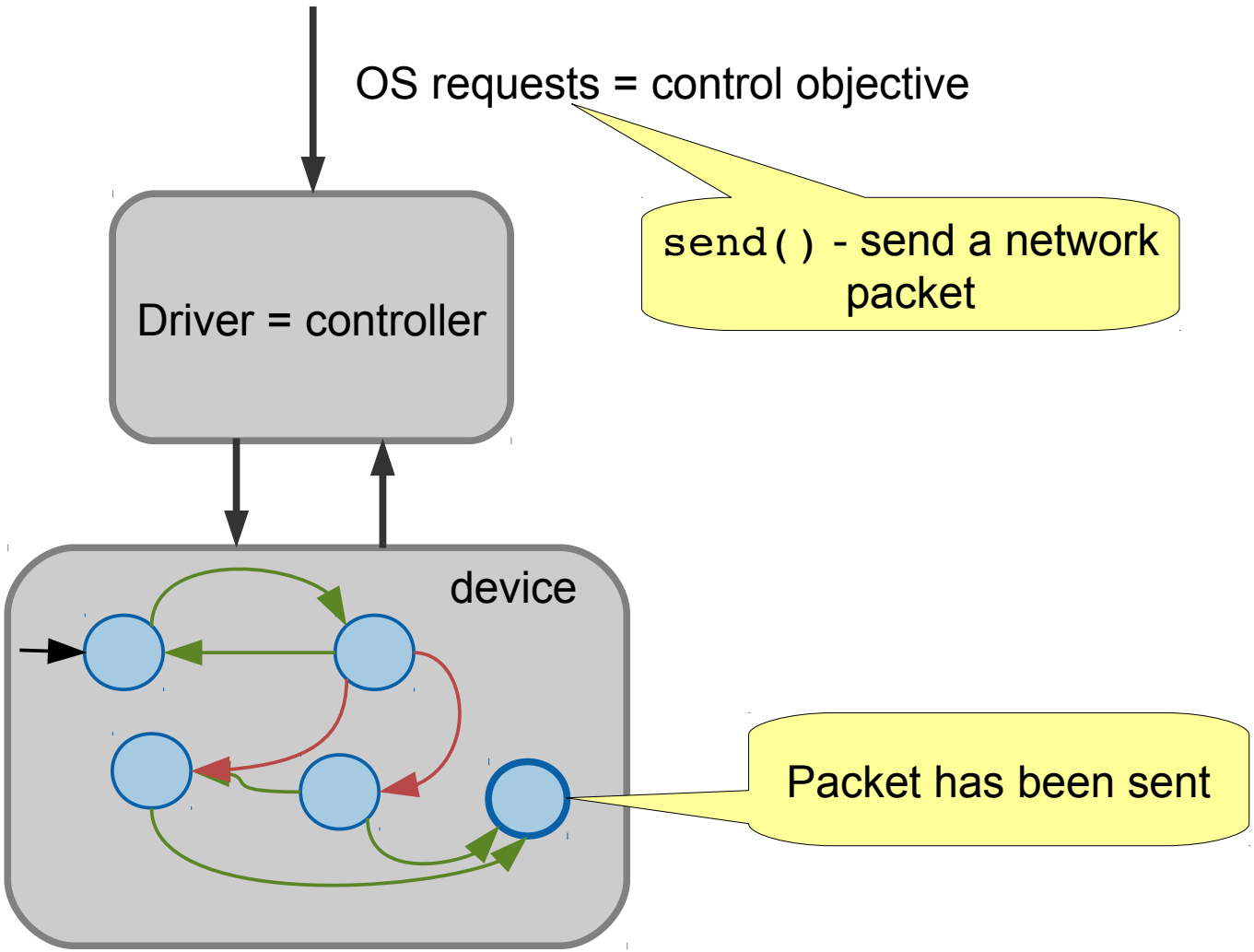
- Captures external behaviour
- Abstracts away structure and timing
- Abstracts away the low-level interface

```
bus_write(u32 addr, u32 val)
{
   ...
}
```

Driver synthesis as controller synthesis



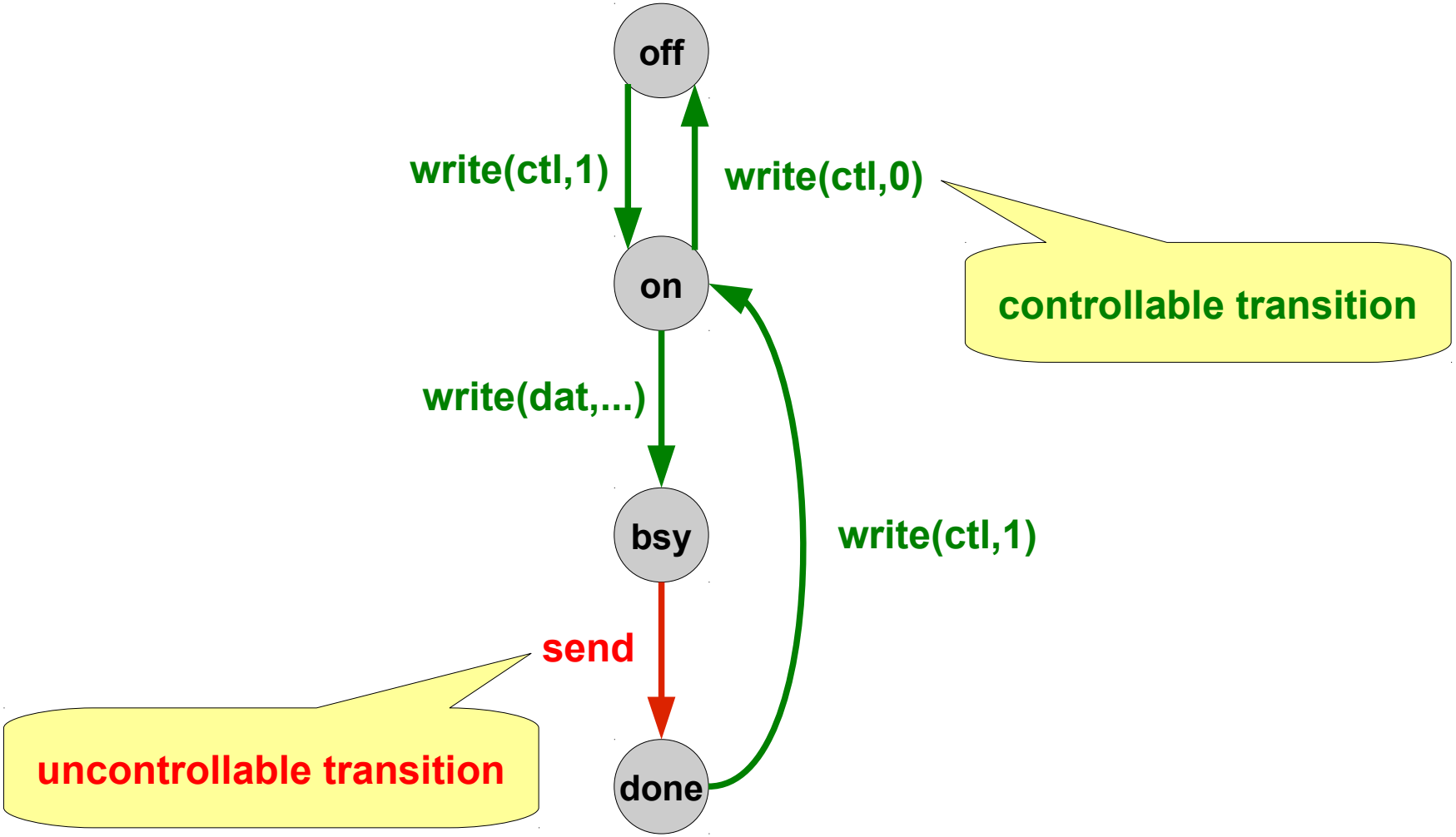
Driver synthesis as controller synthesis



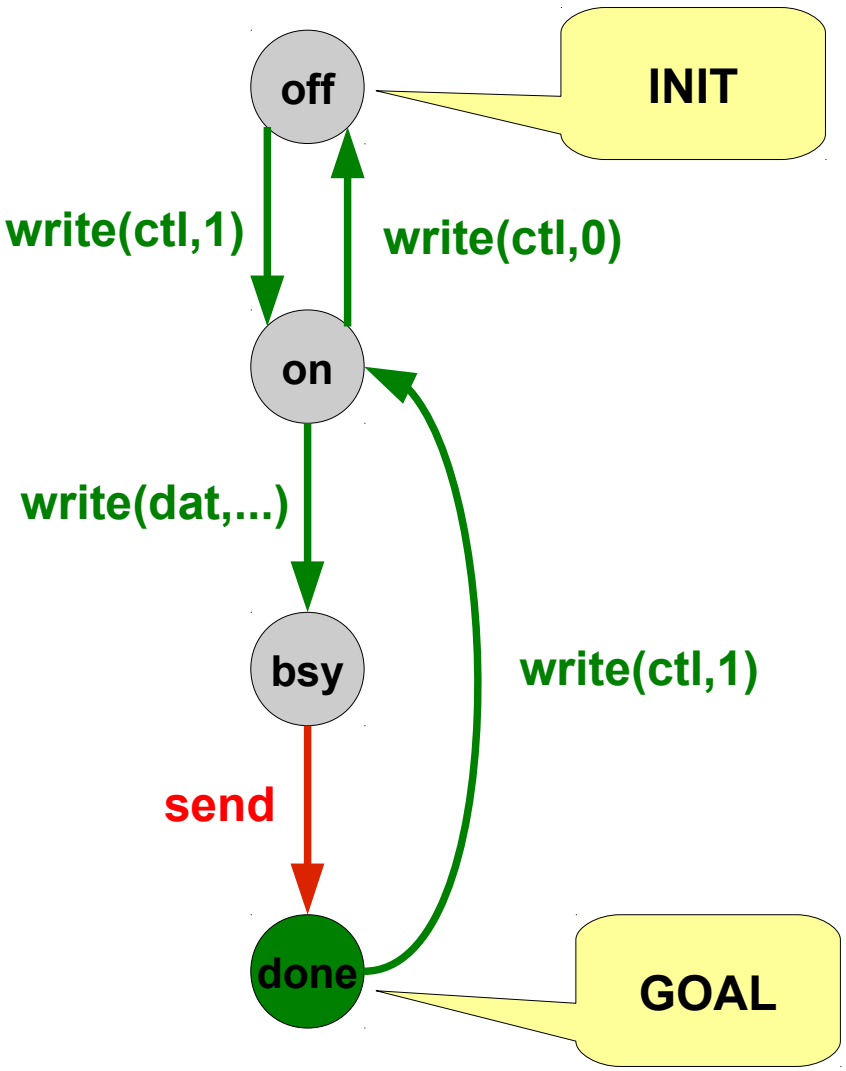
Game theory

- Game theory
  - Provides a theoretical framework for verification and synthesis of reactive systems
  - Provides a classification of games
  - Complexity bounds for various types of games
  - Algorithms for finding winning strategies

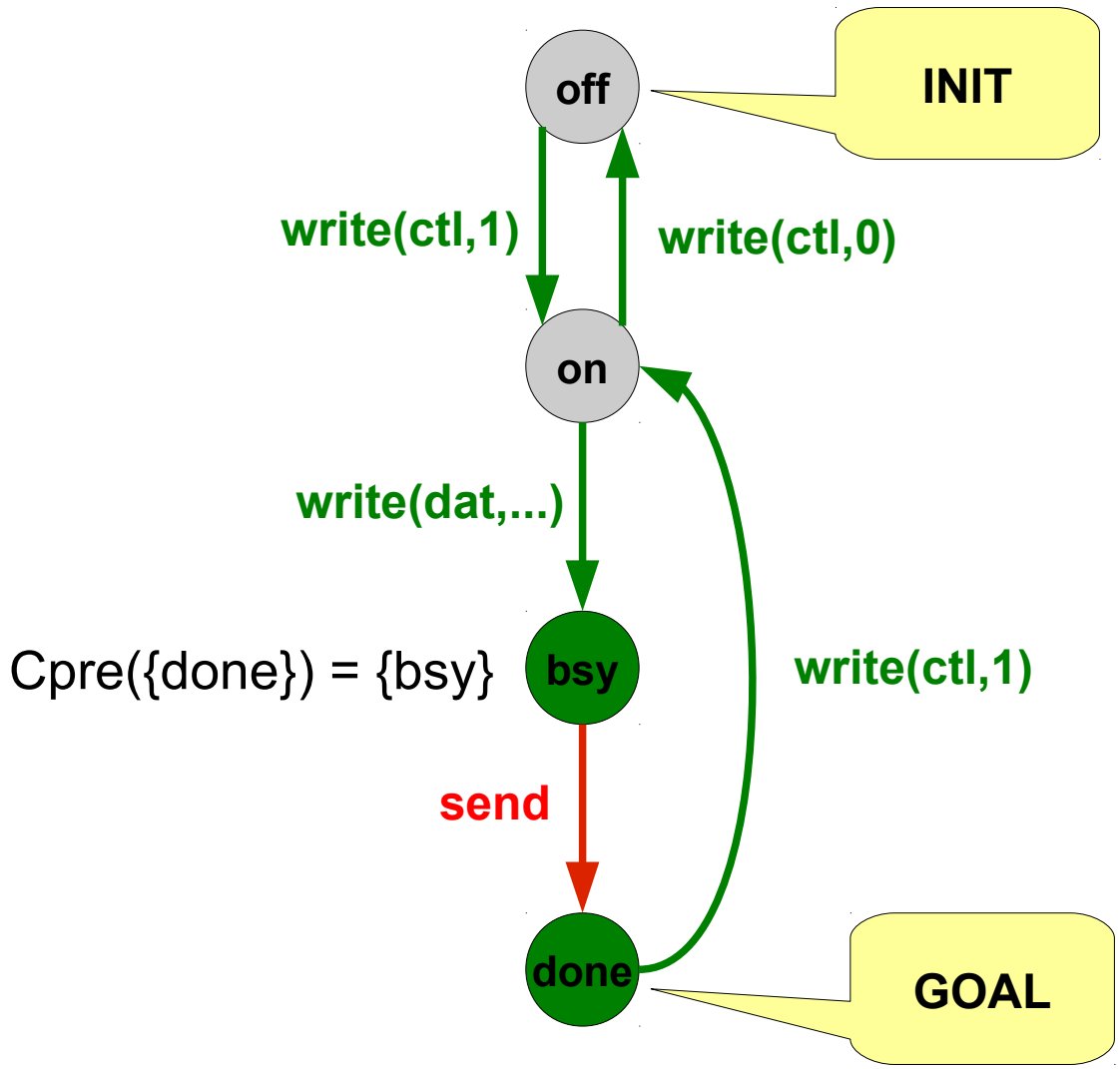
Example: trivial network adapter



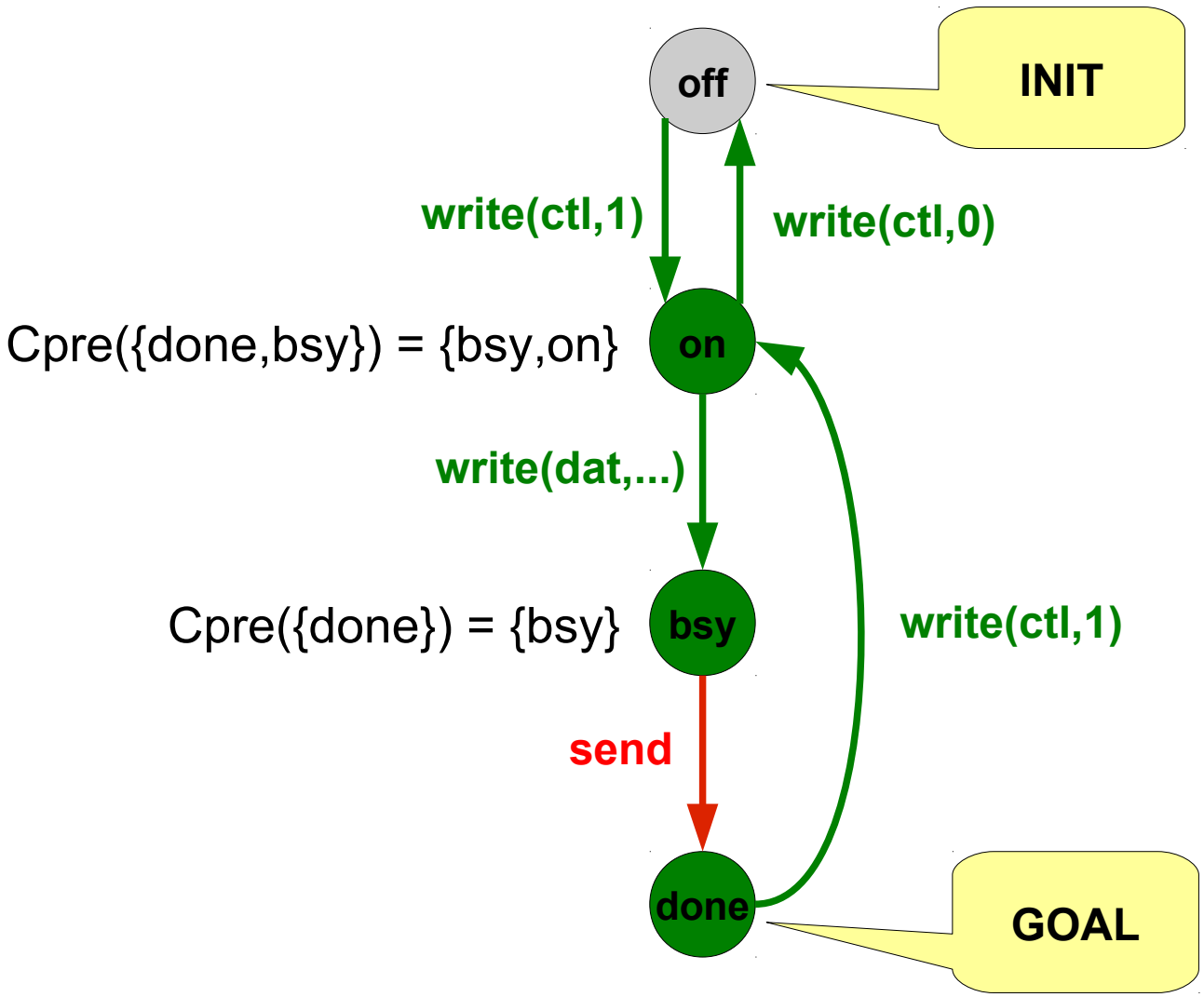
Computing the winning set



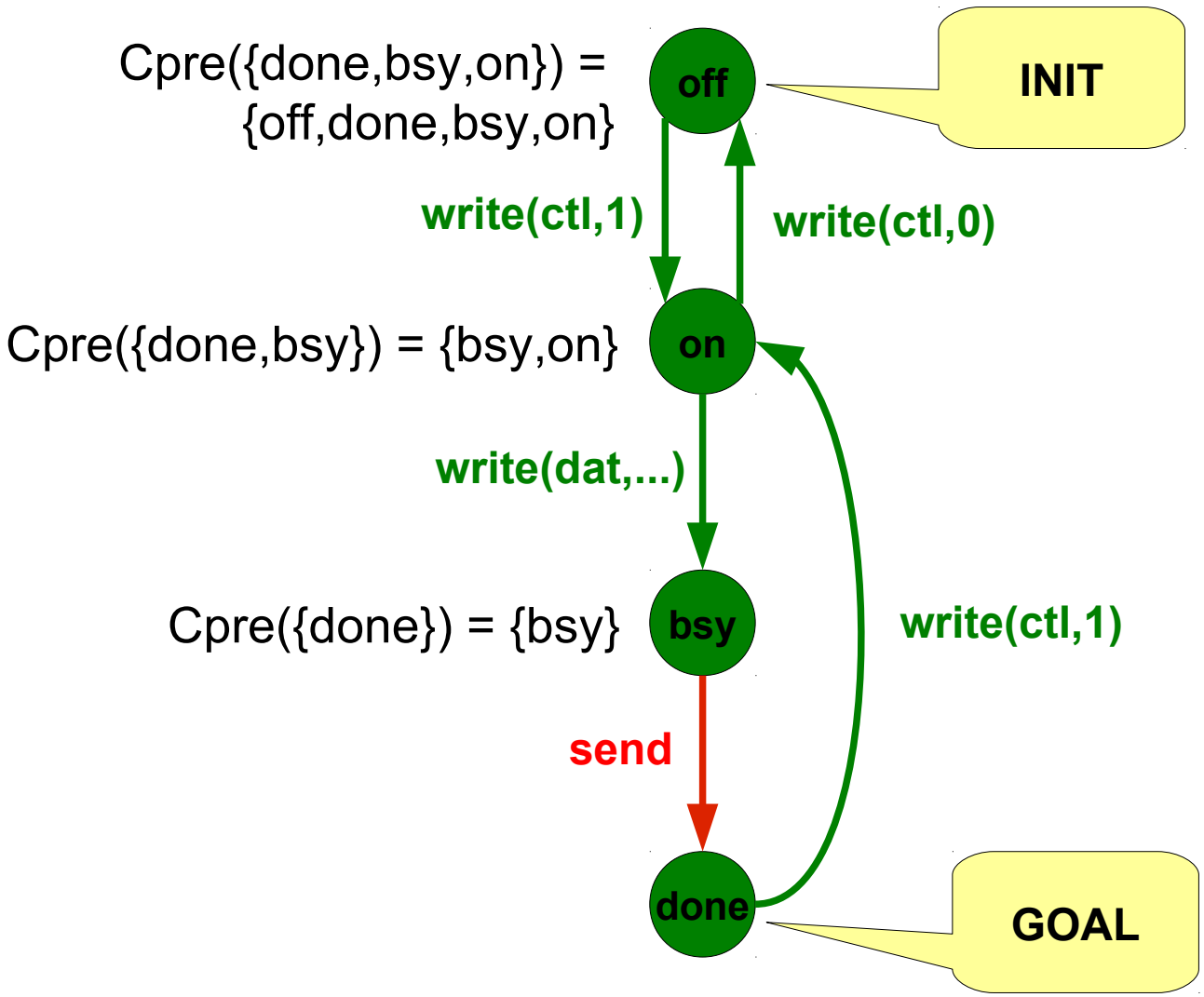
Computing the winning set



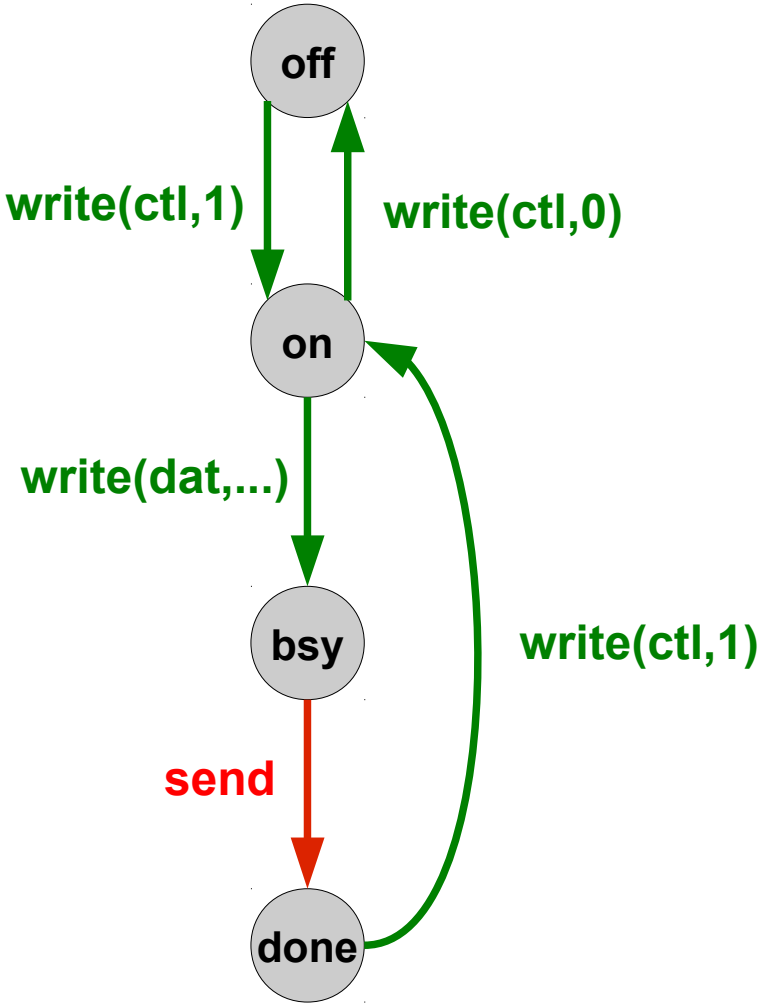
Computing the winning set



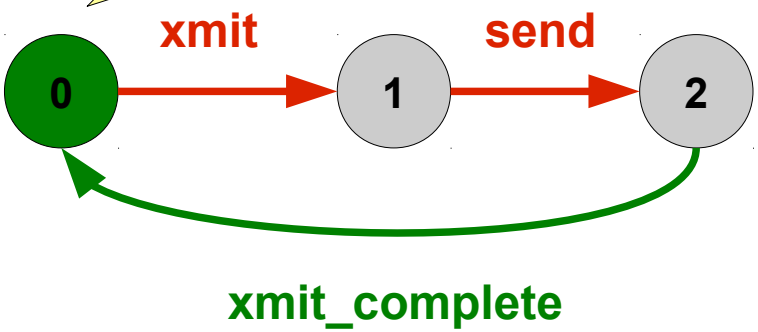
Computing the winning set



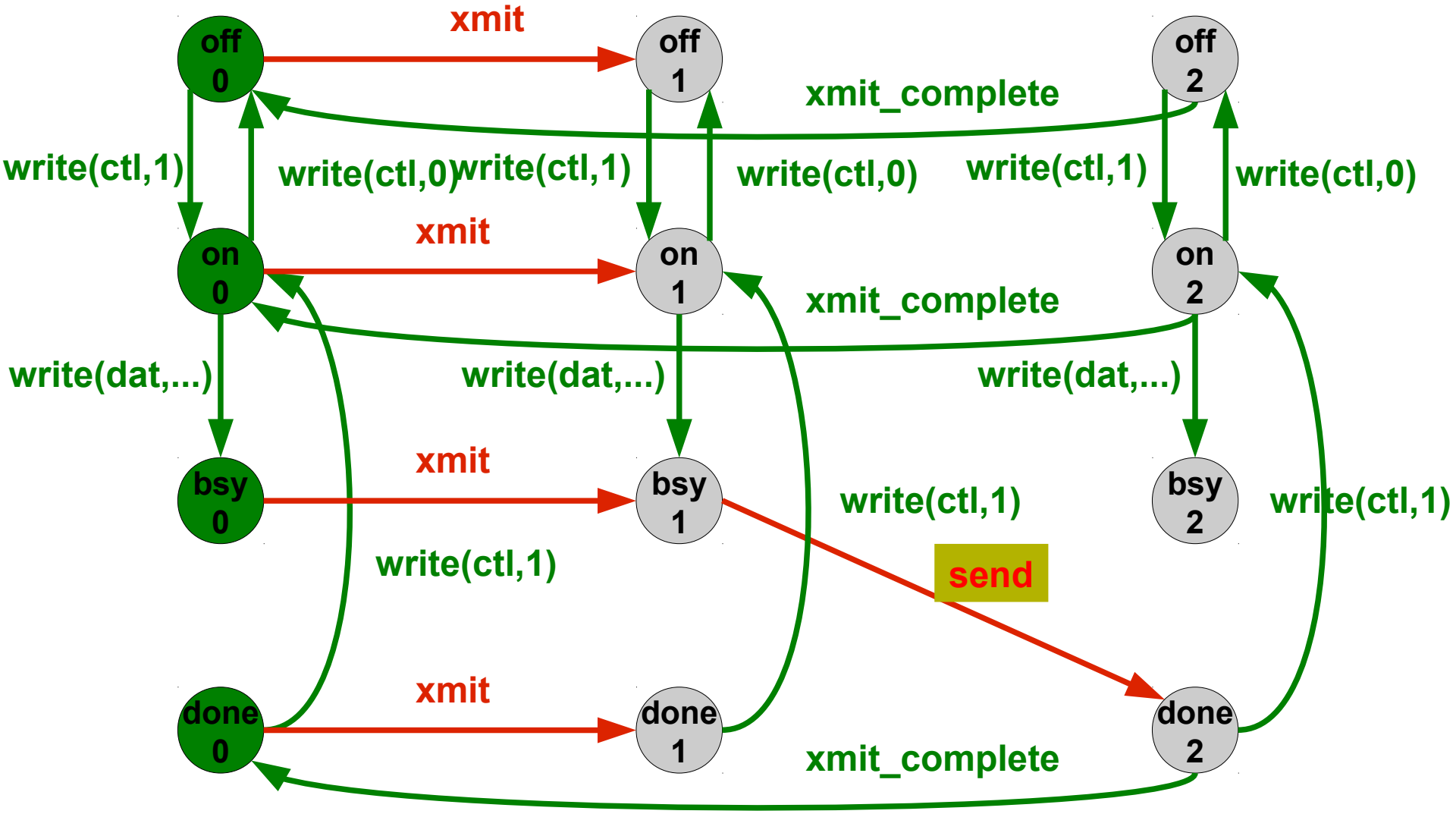
OS specification



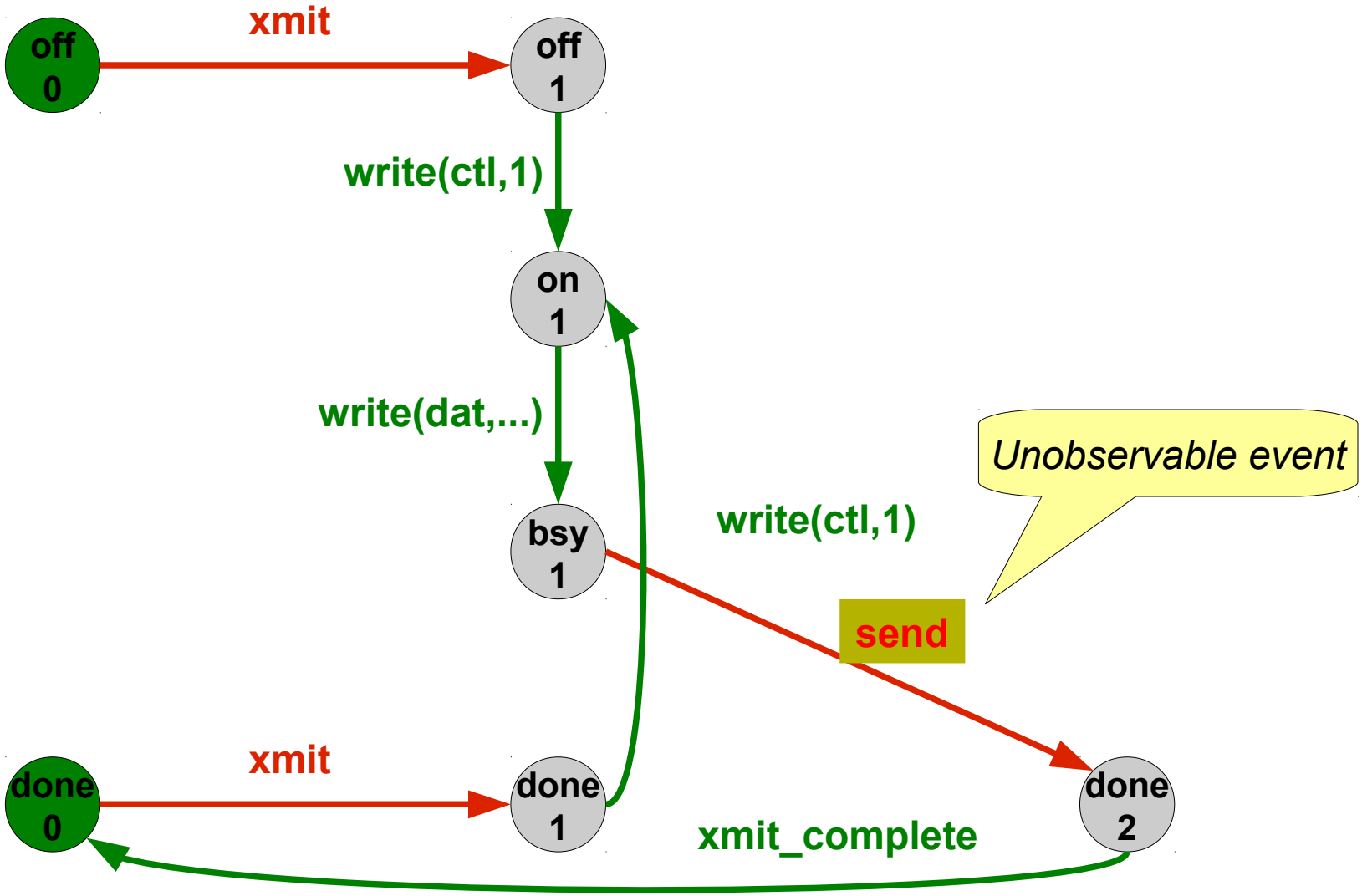
Game objective:
The driver must be in state 0
infinitely often
(aka Büchi objective)



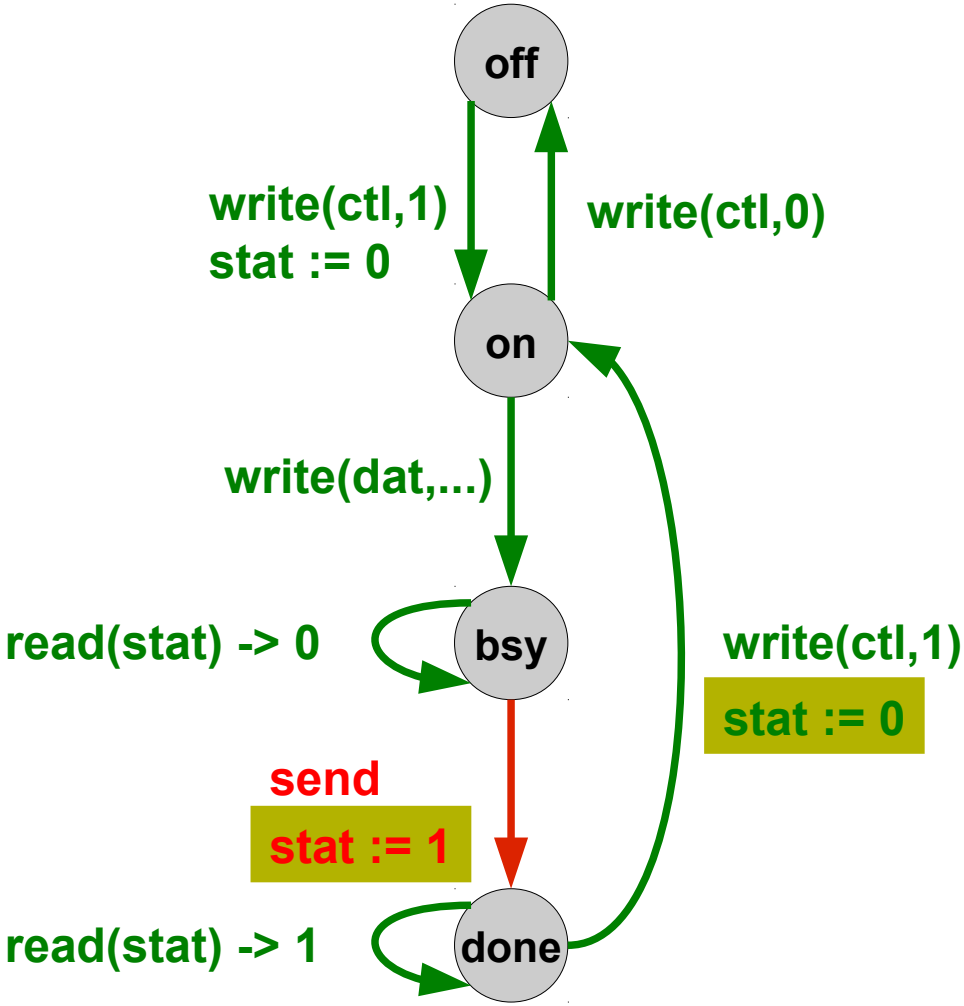
Game automaton



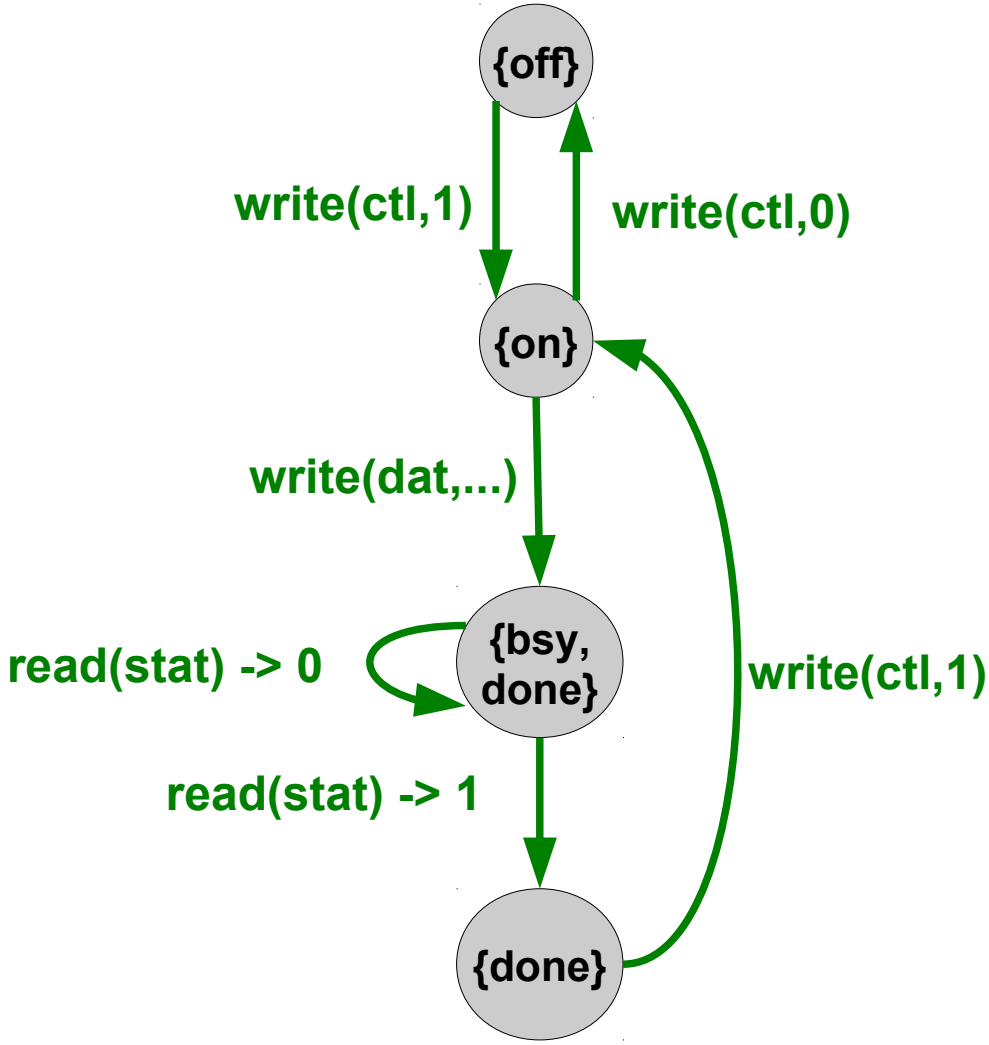
Winning strategy



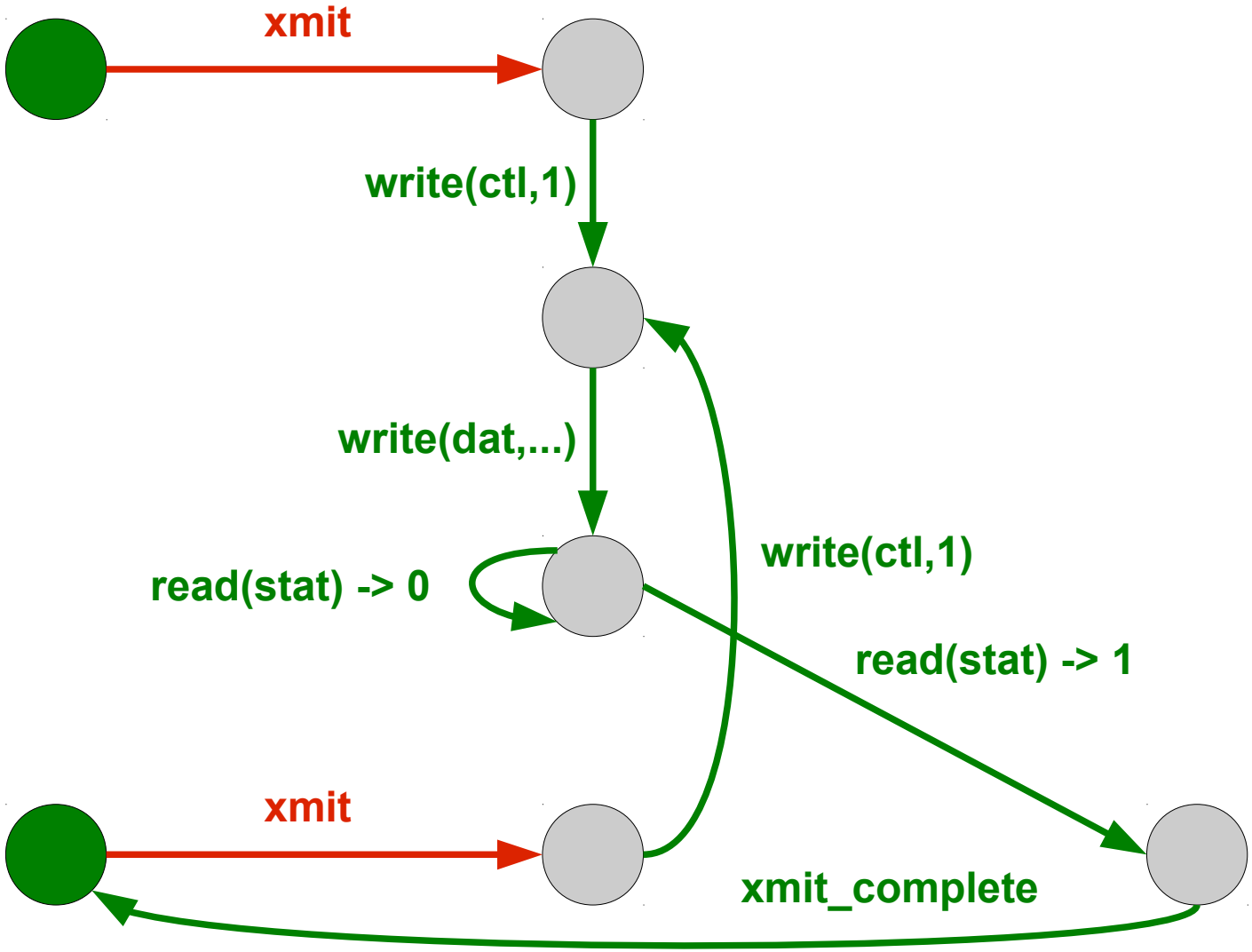
Synthesis with imperfect information



Synthesis with imperfect information



Strategy with imperfect information



Challenges

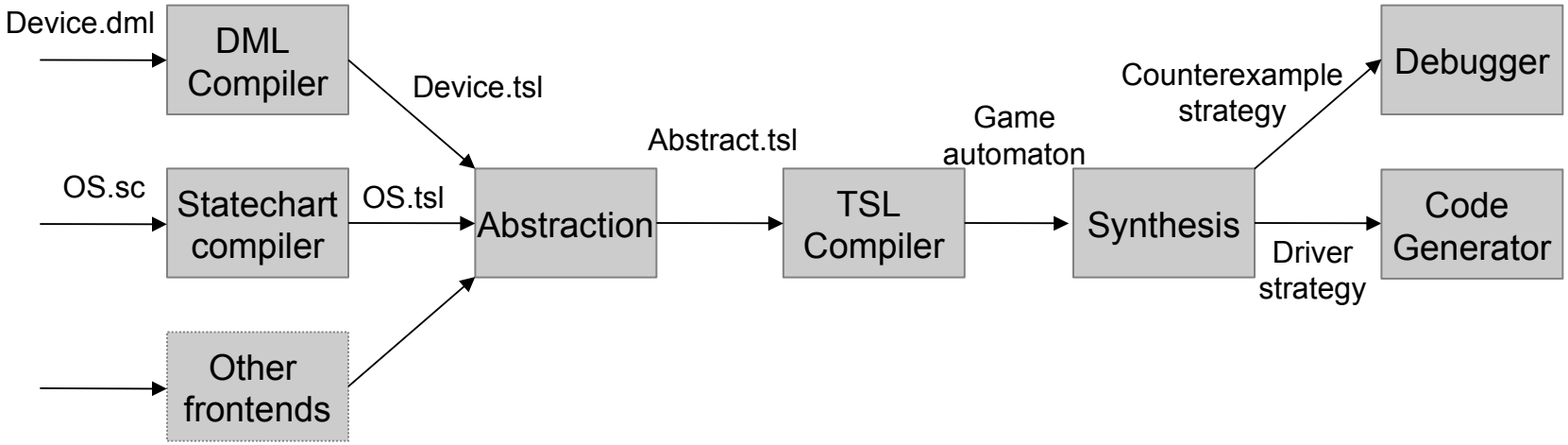
- State explosion (2^{320} states in the IDE controller device)
  - Classical game theory algorithms do not scale
  - Predicate abstraction (2^{48} states after abstraction)
  - Symbolic state space representation (BDD size: 3226 nodes)

Challenges

- Synthesis with imperfect information
  - Generalisation of a perfect information strategy
- Efficient C code generation
  - Control flow graph minimisation
- Support for DMA
  - wip

Demo

Termite toolkit



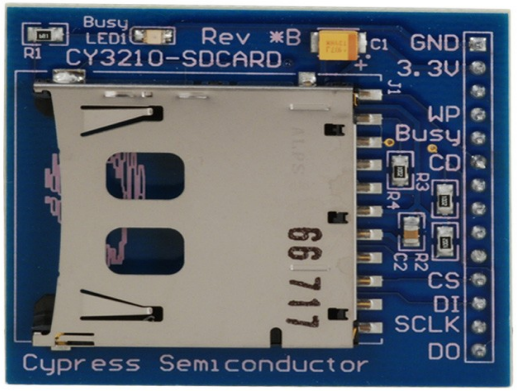
Key Research Outcomes (to date)

Successfully synthesised drivers:

IDE disk controller

W5100 Eth shield

Asix AX88772
USB-to-Eth adapter



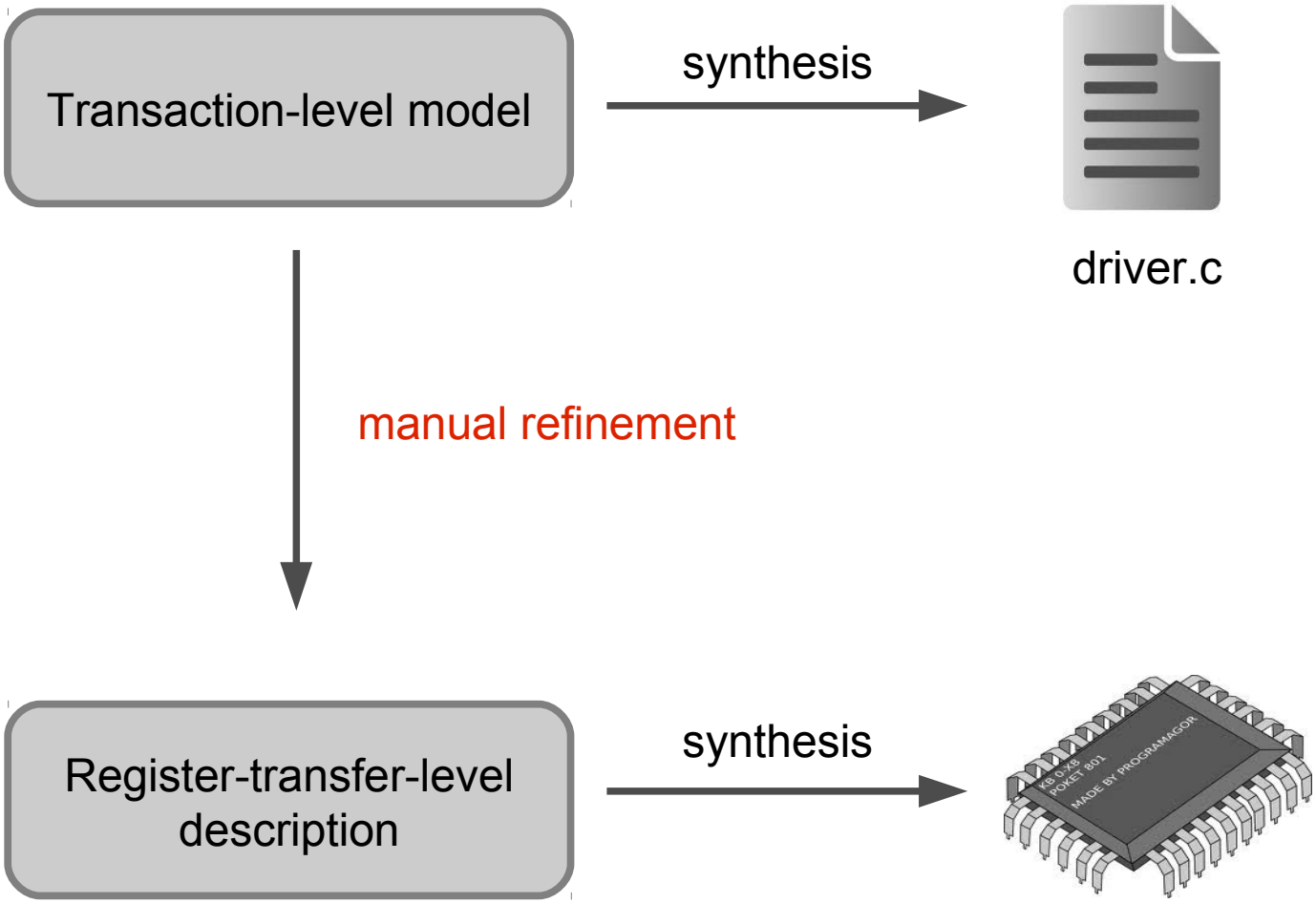
SD host controller

Synthesis of provably correct drivers

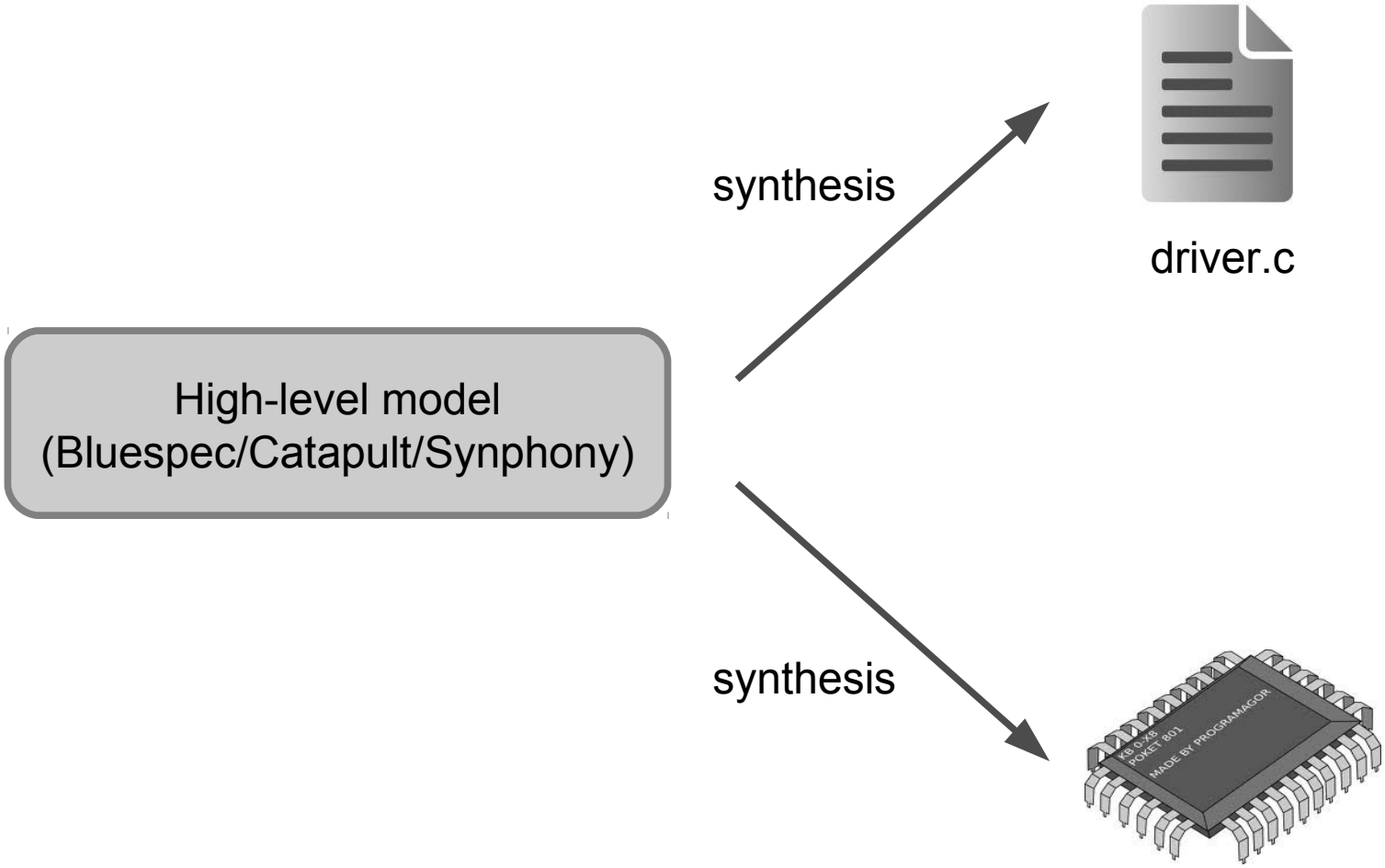
NICTA

- **Synthesised => Correct ??**
  - Bugs in the tool
  - Bugs in the input spec
- Dealing with bugs in the synthesis tool
  - Generate proof of correctness along with the driver
- Dealing with buggy specs

Bridging the gap between hardware and driver synthesis



Bridging the gap between hardware and driver synthesis



Conclusions

NICTA

- Automatic device-driver synthesis:
  - correct-by-construction device drivers at a fraction of the cost of manual development
  - practical alternative to traditional driver development