CSP2Turtle: Verified Turtle Robot Plans

Dara MacConville, Marie Farrell, Matt Luckcuck, Rosemary Monahan

dara.macconville.2018@mumail.ie

Maynooth University
National University of Ireland Maynooth

SFI Centre for Research Training in Foundations of Data Science

Our Goal

- Model a simple robot and environment
- Give it plans to follow
- Verify those plans align with the environment before running

Turtle

- A Python graphics library in which you control a "turtle" that draws as it moves along the screen
- Based off Logo and actual real robots
- Can do various movements (forward(), backward()), change direction (left(), right())
- Can control drawing (penup(), pendown())

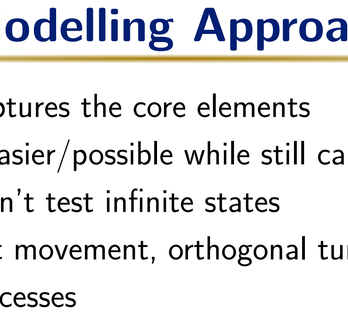


Figure 1: A Hilbert curve drawn by the turtle

Communicating Sequential Processes (CSP)

- A formal language for modelling concurrent systems
- Made up of processes and events
- Event are communicated by the environment and processes react to them, e.g. $P0 = forward \rightarrow left \rightarrow Stop$
- Trace of a process: the sequence of events that happen throughout its lifetime
- Q *trace-refines* P ($P \sqsubseteq Q$) if every trace of Q is a trace of P, e.g. $P1 = forward \rightarrow left \rightarrow P1$, then $P1 \sqsubseteq P0$
- Build up more complicated systems out of basic operators, recursions, etc

Simple Prefix	$a \rightarrow P$	Communicate event a, then act like process P
External Choice	$P \Box Q$	Offer a choice between two processes P and Q
Interleaving	$P \mid\mid\mid Q$	Processes P and Q run in parallel with no synchronisation

Modelling Approach

- Simplified turtle that captures the core elements
- Makes implementation easier/possible while still capturing essentials
- FDR (model checker) can't test infinite states
- Bounded grid world, unit movement, orthogonal turning
- Plans as simple CSP processes
- Make CSP events mirror Turtle functions (fd, bk, lt, rt, pu, pd)

Paper

Bibliography

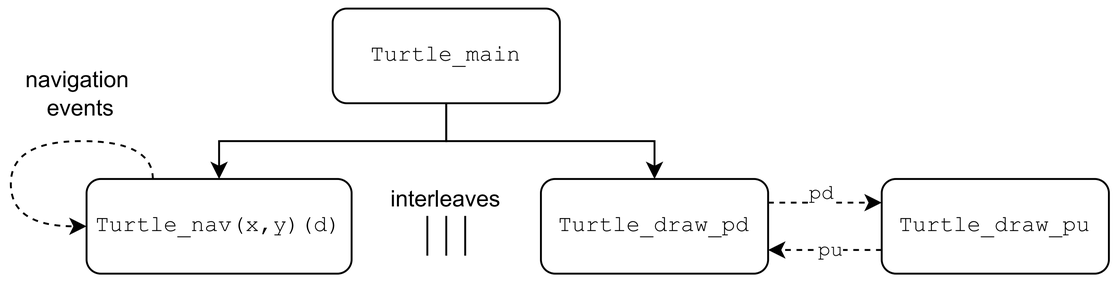
[1] D. MacConville, M. Farrell, M. Luckcuck, and R. Monahan. Csp2turtle: Verified turtle robot plans. *Robotics*, 12(2), 2023.

Funding Acknowledgement

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 18/CRT/6049. The opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Science Foundation Ireland.

CSP Model Architecture

- Main process made up of multiple constituent processes
- Modular design allows for greater extensibility
- Main process runs the navigation and drawing processes independently, they don't need to communicate
- Each handle their respective events



CSP2Turtle Toolchain

- Inputs:
  - Navigation plan
  - World specification: dimensions, goal, obstacles, known with certainty
- Checks:
  - Is the plan executable?
  - Is the goal reachable?
- Outputs:
  - Results: Feedback on plan and goal targets, with possible paths to the goal displayed.
- Interactive or File Modes

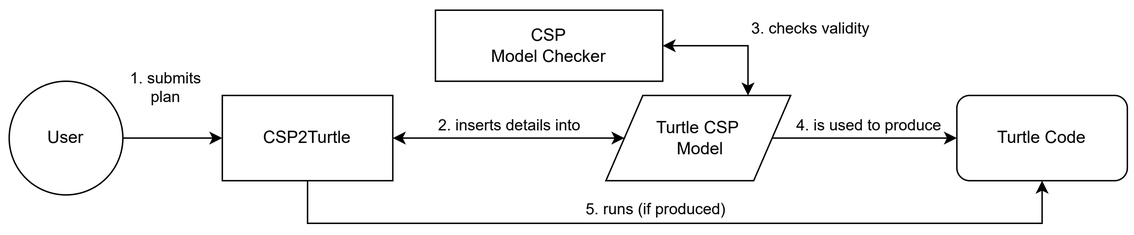
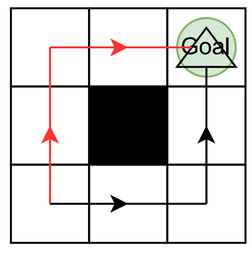


Figure 2: Stages and components of our toolchain.

Example Usage




```
Interactive Mode
Enter starting position as x, y: 0, 0
Enter H, V (Horizontal, Vertical): 3, 3
Enter plan: (fd -> fd -> lt -> SKIP
          [] lt -> fd -> fd -> rt -> SKIP) ; fd -> fd
Enter goal location as x, y: 2, 2
Enter obstacles as (x1, y1), (x2, y2), etc: (1, 1)
Assertion succeeded: Plan reaches goal ✅
One path to goal is: fd -> fd -> rt -> bk -> bk
```

Figure 3: A usage example of CSP2Turtle's Interactive Mode, where all possible paths lead to the goal and CSP2Turtle accepts the plan.

Future Work

Python Model Checker: General purpose language, also widely used in critical systems like robotics, e.g. ROS

Literature/Tool Review: Java Path Finder, JBMC, SPIN, etc ...

Accompanying Case Study: Especially interested in robotics and autonomous systems

University of Derby · Manchester 1824 The University of Manchester · Science Foundation Ireland sfi For what's next