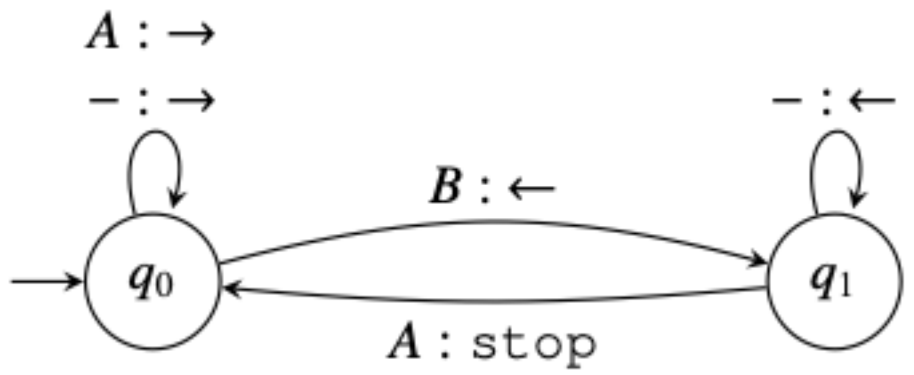
A Correctness Result for Synthesizing Plans With Loops in Stochastic Domains

Laszlo Treszkai & Vaishak Belle, ***University of Edinburgh***

Finite State Controllers

A	–	–	–	**B**



- FSCs, such as plans with loops, are powerful and compact representations of action selection widely used in robotics, video games and logistics

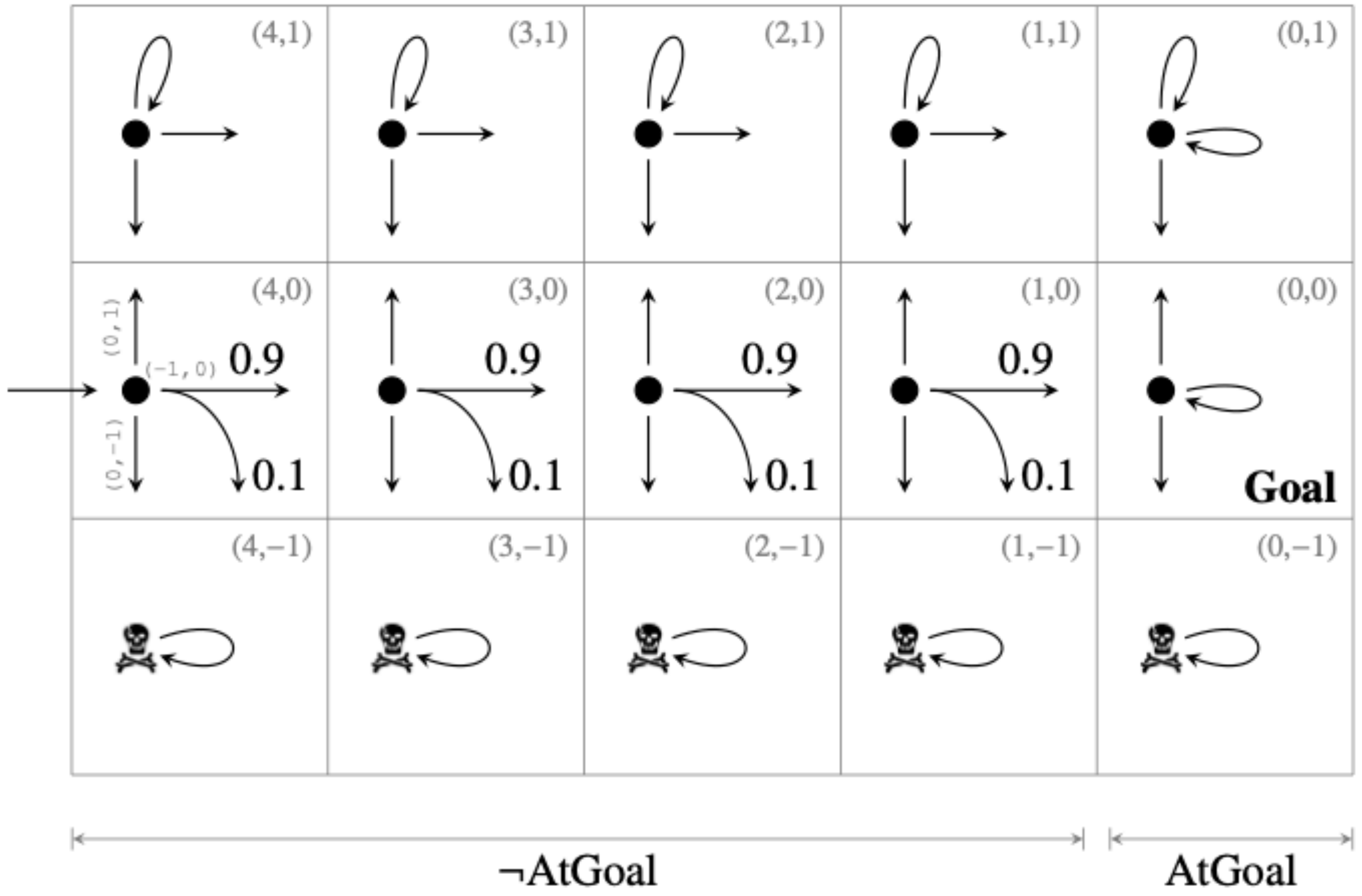
- Cleaning a table (with arbitrary number of objects), chopping tree of unknown thickness

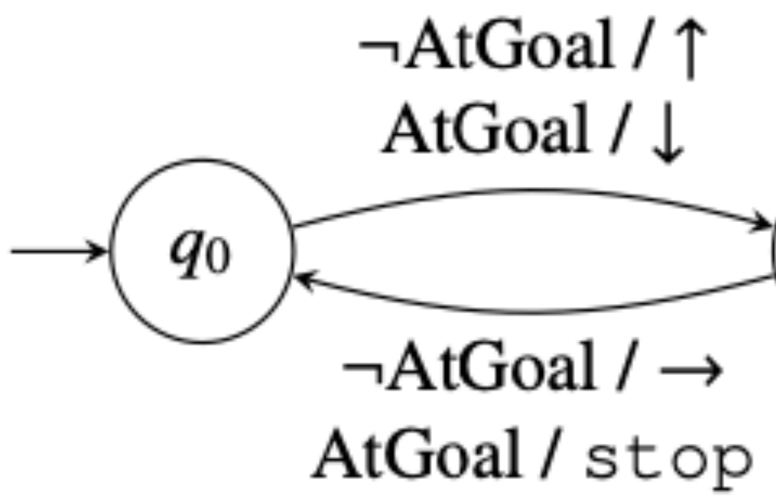
- Lots of work on algorithms for synthesis (e.g., AND/OR bounded search, abstraction)

What if the actions are noisy?

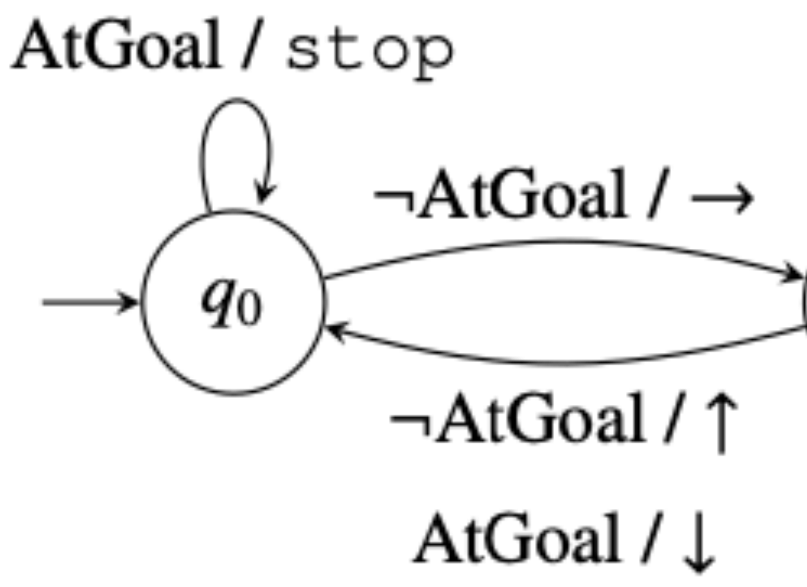
An agent that stands on the handrail of a bridge: on one side the sidewalk, on the other side the river, and the agent is n steps away is the goal. With every step taken on the handrail, the agent has a 0.1 probability of falling into the river (an absorbing state), and 0.9 probability of moving forward one step. However, can deterministically get onto sidewalk, where forward is also deterministic.

Clearly, moving solely on handrail satisfies goal with $\Pr = 0.9^n$. But it can do better.


(4,1)
(3,1)
(2,1)
(1,1)
(0,1)
(4,0)
(3,0)
(2,0)
(1,0)
(0,0)
(0,1)
(−1,0)
(0,−1)
0.9
0.1
0.9
0.1
0.9
0.1
0.9
0.1
Goal
(4,−1)
(3,−1)
(2,−1)
(1,−1)
(0,−1)
¬AtGoal
AtGoal


¬AtGoal / ↑
AtGoal / ↓
q_0
q_1
¬AtGoal / →
AtGoal / stop

Pr = 1


AtGoal / stop
¬AtGoal / →
q_0
q_1
¬AtGoal / ↑
AtGoal / ↓

Pr = 0.9

How to handle noise?

Lots of approaches for FSCs, but many of them are either **approximate** or **do not properly handle non-terminating traces** (e.g., assume failure cannot happen infinitely many times)

Theorem AND-OR search algorithm fails if **at least one history that cannot be extended into a goal history**

Theorem AND-OR search algorithm fails if at least **one looping history**

Planning Problem

$$LGT \doteq \sum_{\{h|h \text{ is a goal history}\}} \Pr(h)$$

Given a planning problem $\mathscr{P} = \langle S, A, O, \Delta, \Omega, s_0, G \rangle$, an integer *N*, LGT* $\in$ **(0,1), find a finite-state controller with at most *N* states such that LGT ≥ LGT* for $\mathscr{P}$.** (Here, only Δ is stochastic.)

Theorem Given a planning problem $\mathscr{P}$, integer *N*, and LGT* $\in$ (0, 1), the search algorithm PANDOR is sound and complete: every FSC $\mathscr{C}$ returned is *N*-bounded and LGT ≥ LGT*, and if there exists an *N*-bounded controller that is LGT ≥ LGT*, then one such FSC will be found

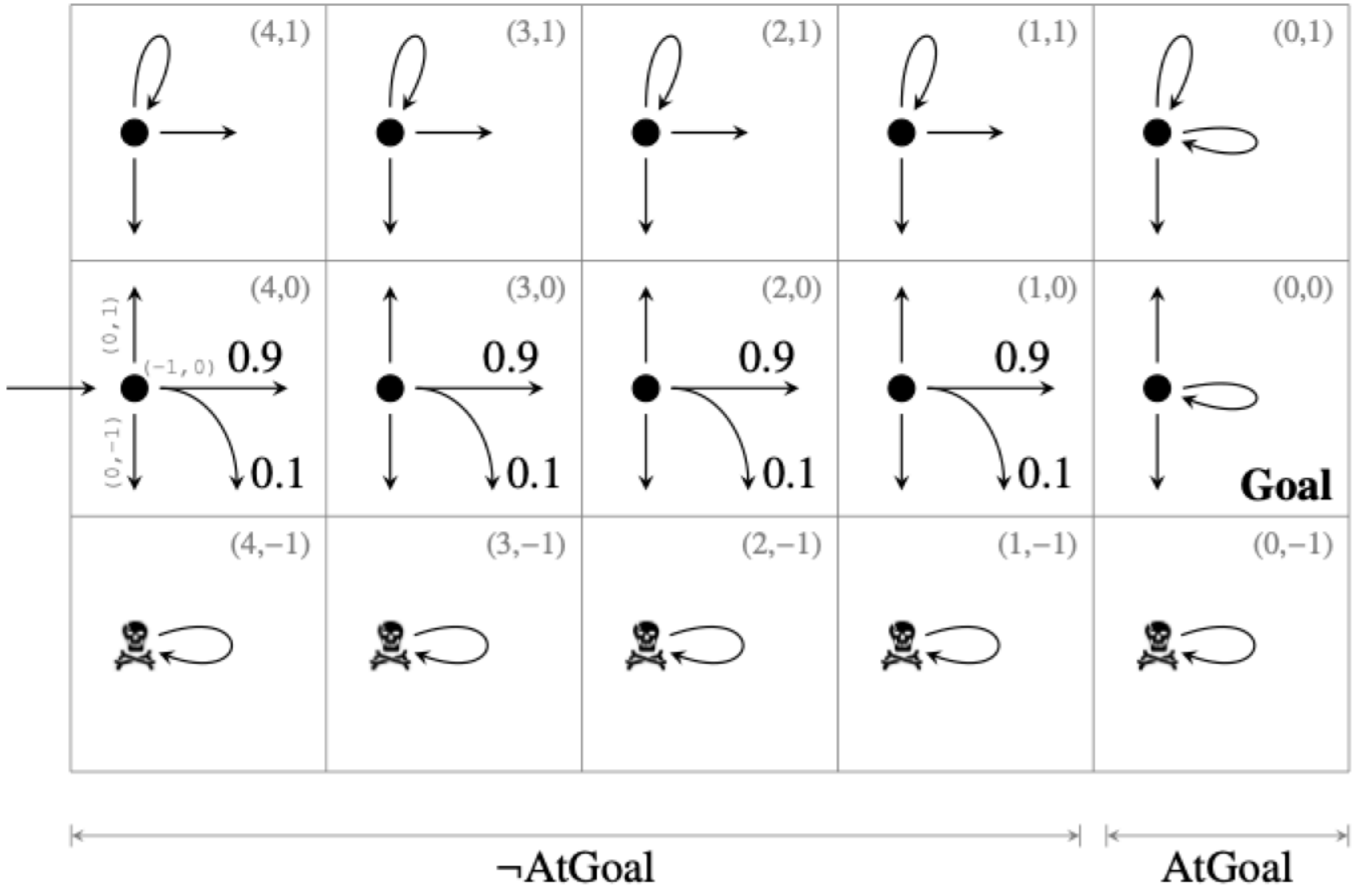
How? Consider the AND-OR algorithm

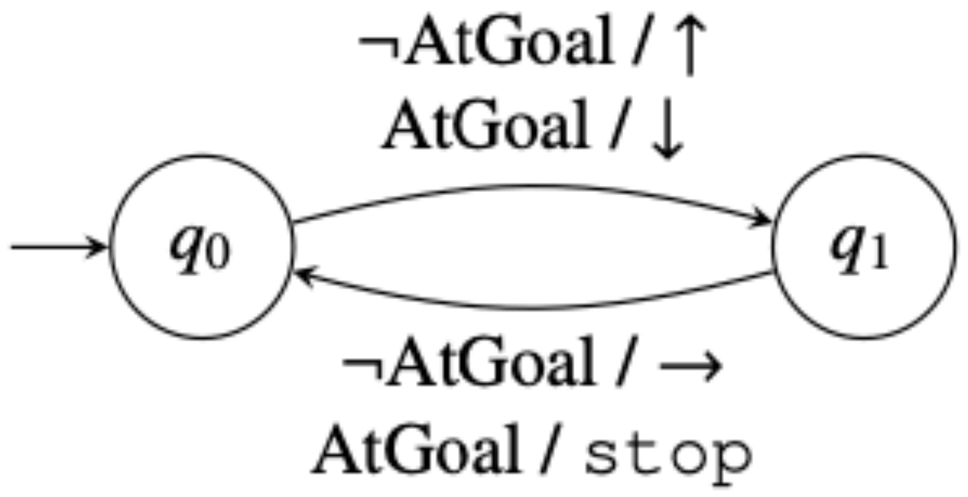
- Initially, the algorithm starts with the empty controller $\mathscr{C}$, at initial controller state q_0 & s_0
- AND function enumerates the outcomes of an action from a given combined state and history, and calls OR to synthesize a controller that is correct for every outcome
- The OR function enumerates the extensions of a controller for the current controller state and observation, and thus selects a next action for the current observation, and then calls AND to test for correctness recursively on the outcomes of the chosen action

.

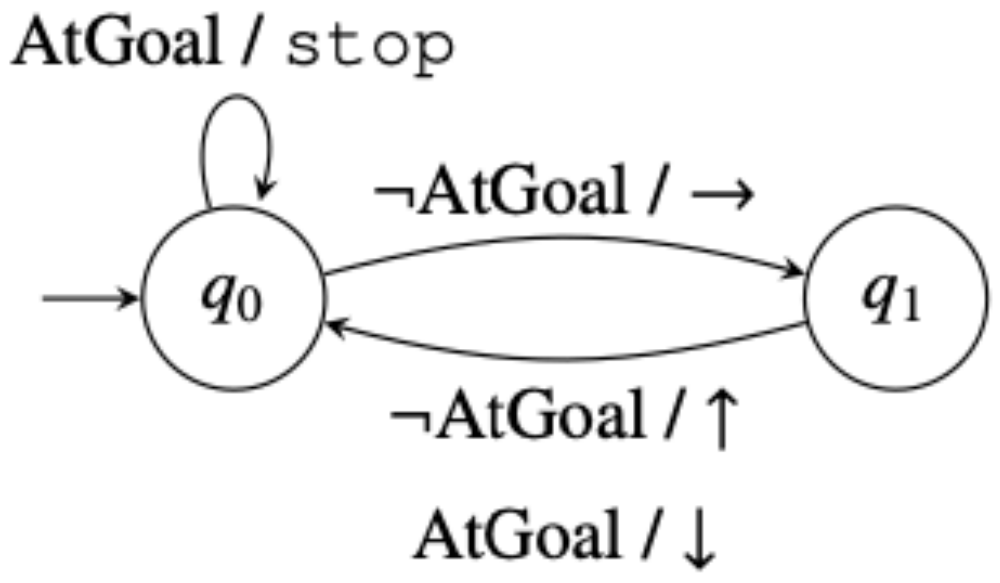
The extension: key idea

- Maintain an upper and lower bound for the LGT of the current controller
- Whenever a failing run is encountered, the upper bound is decreased by the likelihood of this run; similarly, a goal run increases the lower bound on LGT
- When the lower bound exceeds the desired correctness likelihood (i.e., LGT*), the current controller is guaranteed to be “good enough”, and the algorithm returns with success
- When the upper bound is lower than LGT*, none of the extensions of the controller is sufficiently good, and we revert the program state to the point of the last non-deterministic choice point
- **Need to carefully keep track of looping histories** (involved)





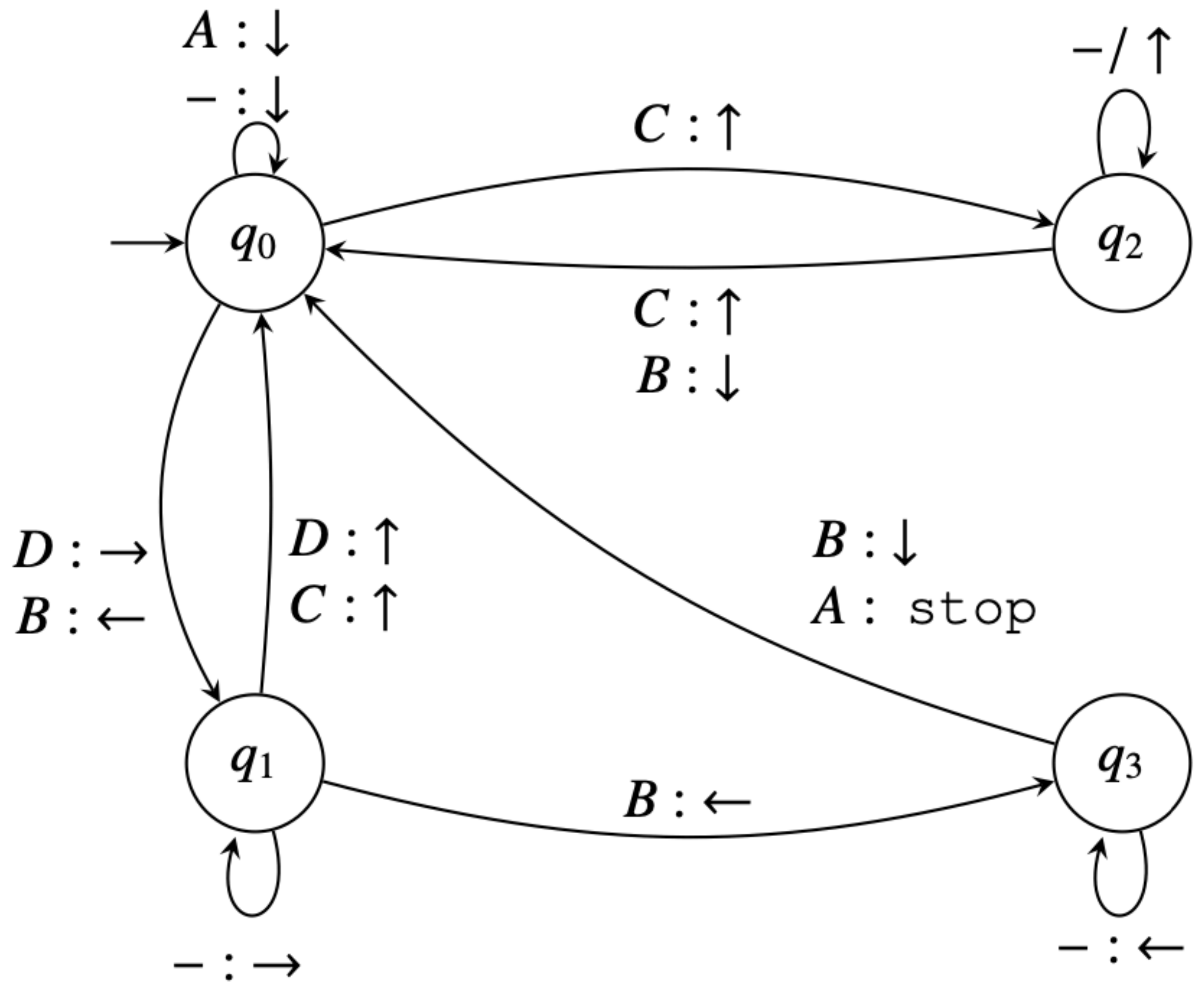
$\Pr = 1$



$\Pr = 0.9$

github.com/treszkai/pandor

A	–	–	–	**B**
–				–
–				–
–				–
D	–	–	–	**C**



github.com/treszkai/pandor

Conclusions

- New theoretical results on a **generic technique** for synthesizing FSCs in **stochastic environments**, allowing for highly granular specifications on termination and goal satisfaction
    - Builds on the generic AND-OR bounded search, a generic technique for deterministic environments
- Proved the **soundness** and **completeness** of that synthesis algorithm