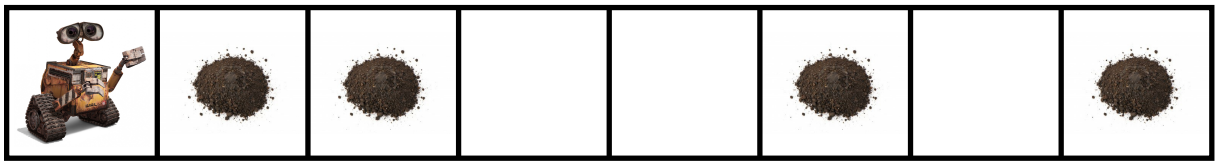
Policies that Generalize: Solving Many Planning Problems With the Same Policy

Blai Bonet
Universidad Simón Bolívar
Caracas, Venezuela

Hector Geffner
ICREA & U. Pompeu Fabra
Barcelona, Spain

IJCAI 2015, Buenos Aires

Example: Dust Cleaning

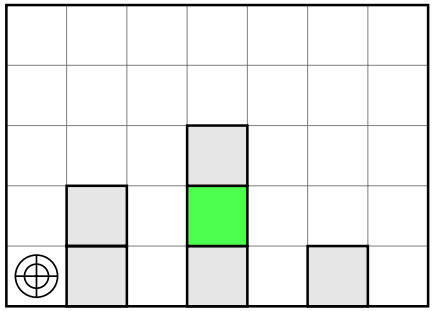


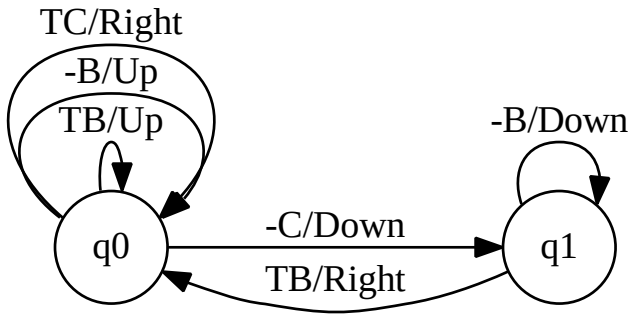
- **Problem:** clean all cells starting on the left
  - ▷ **Actions:** move left, move right, suck
  - ▷ **Sensors:** dirt, end of corridor
- **Solution** can be expressed as **memoryless policy:**
  - ▷ *if dirt, suck; if no dirt and not end, move-right*
- Policy is compact and **solves many variations**:
  - ▷ different grid sizes
  - ▷ different distribution of dirt
  - ▷ noisy actions that may fail

What's the common structure that explains generality of the policy?

Example 2: Visual Blocks

- **Problem** P: find **green block** using visual-marker (circle) that can move around one cell at a time
- **Observables:** Whether cell marked contains a green block (G), non-green block (B), or neither (C); and whether on table (T) or not (–)





- **Finite-state controller** obtained by running classical planner over transformed problem (Bonet et al., 2009)
- Controller solves this problem and **any configuration** with **any number** blocks

What is the common structure among these different problems?

Motivation

Why care about how policy for one problem generalizes to other problems?

- **Generalized planning:** solution that works for multiple problems sought
- **Transfer learning:** how solution to one problem used for solving another problem
- **Scalability:** why solve "large" problems if solutions to "smaller" problems ok?
- **Representation:** generalization imposes constraints on representation

Plan for the rest of the talk

- **Model:** Partially observable non-deterministic problems (PONDPs)
- **Solutions:** Memoryless policies that solve a problem
- **Reductions:** Mapping "large" problems into "smaller" ones
- **Theoretical Results:** When policy for P for Q on **structural ones**
- **Extensions:** Generalization to policies with finite memory (finite state controllers)

Model: PONDPs and POMDPs

Partially observable non-deterministic problems (PONDP) $P = \langle S, S_0, T, A, F, o \rangle$:

- S is a finite set of states s,
- S_0 is the set of possible **initial states** $S_0 \subseteq S$,
- T is the set of **goal states** $T \subseteq S$,
- $A(s) \subseteq A$ is set of **applicable actions** in $s \in S$,
- $F(a, s) \subseteq S$ denotes **set of successor states** $s \in S, a \in A(s)$
- o is the **observation function** $o(s) \in \Omega_o$, Ω_o set of observations

- PONDPs similar to POMDPs; noisy sensing easy to compile away
- Results for PONDPs apply to POMDPs where **goal** to be achieved with **certainty**
- Such **Goal POMDPs** more general than usual **discounted** version with **rewards**

Solutions: Memoryless Policies

- **Solution form** of PONDPs and POMDPs is **mapping** from **beliefs** into actions
- Yet such mappings tied to **particular state space** and don't **generalize** to others
- **Memoryless** policies and **finite-state controllers** not as powerful but generalize
- A **memoryless** policy is (partial) function μ mapping **observations** into actions
- The **action** in the state s is $\mu(o(s))$ where o is **observation function**
- Policy defines μ-trajectories $\langle s_0, s_1, \ldots \rangle$, $s_0 \in S_0$, $s_{i+1} \in F(\mu(o(s_i)), s_i)$, $i > 0$
- Policy μ **solves** problem P iff every **fair** μ-trajectory reaches a **goal state**

Fairness, Strong Cyclic Policies, Proper Policies

- A **transition** (s, a, s') is in P if s and s' are states in P, and $s' \in F(a, s)$
- A trajectory τ is **fair** if **infinite** occurrences of transitions (s, a, s') in τ, imply **infinite** occurrences of transitions (s, a, s'') that are also in P
- Solutions are thus **strongly cyclic (memoryless) policies**
- They correspond also to **proper** policies in (Goal) MDPs and POMDPs that ensure goal reached with **certainty**

Our question: When solution μ to P solves also Q on structural grounds?

Structure: Reductions

Definition: Function $h : S \to S'$ **reduces** $P = \langle S, S_0, T, A, F, o \rangle$ into $P' = \langle S', S'_0, T', A', F', o' \rangle$ if for all states s, s' in S:

R1. **Actions:** $A'(h(s)) \subseteq A(s)$ (no new actions)

R2. **Sensing:** $o(s) = o'(h(s))$ (observations preserved)

R3. **Dynamics:** if $s' \in F(a, s)$ then $h(s') \in F'(a, h(s))$ (transitions preserved)

R4. **Init:** if $s \in S_0$ then $h(s) \in S'_0$ (initial states preserved)

R5. **Goal:** if $s \notin T$ then $h(s) \notin T'$ (non-goal states preserved)

- Reductions h can map **multiple** states s and s' in P into **single** states $h(s) = h(s')$
- Reductions **embed** one problem into another preserving some **structure**
- Reductions are **not symmetric** like **bisimulations** though. This is key

Example: Reducing Counter 0 . . . 100 to Counter 0,1

- $P_n = \langle S, S_0, T, A, F, o \rangle$ is problem where state s is **counter** in $[0, n]$, $n \geq 1$
- Actions **increase** and **decrease** counter by 1 within interval $[0, n]$
- **Initial state** is uncertain, i.e., $S_0 = S$, and **goal** $s = 0$ is **observable**
- **Function** $h(0) = 0$ and $h(s) = 1$ for $s > 0$, does **not** reduce P_{100} into P_1
- **Transitions** like $(50, Dec, 49)$ in P_{100}, map into transitions $(h(50), Dec, h(40)) = (1, Dec, 1)$ that are **not** in P_1 in violation of **R3**
- Yet h **reduces** any P_n to problem P_1 **extended** with transition $(1, Dec, 1)$.
- Problem $P' = P_1 + \{(1, Dec, 1)\}$ **non-deterministic** as it contains also the transition $(1, Dec, 0)$.

Basic Theoretical Result

Reduction h of P into P' **not** sufficient for solutions μ of P' to **generalize** to P. Yet, every μ-trajectory τ in P **induces** a μ-trajectory $h(\tau)$:

Lemma: For a **reduction** h of P into P', if $\tau = \langle s_0, a_0, s_1, a_1, \ldots \rangle$ is a μ-trajectory in P, then $h(\tau) = \langle h(s_0), a_0, h(s_1), a_1, \ldots \rangle$ is μ-trajectory in P'.

E.g., μ-trajectory $\tau = (4, 3, 2, 1, 0)$ in P_{100} for μ : "decrement until goal" maps into $h(\tau) = (h(4), h(3), h(2), h(1), h(0)) = (1, 1, 1, 1, 0)$ that is a μ-trajectory in P'.

Reduction $h : P \to P'$ ensures **generalization** when it preserves **fairness:**

Theorem: If μ **solves** P' and h reduces P into P' so that **fair** μ-trajectories τ in P are mapped into trajectories $h(\tau)$ that are **fair** in P', μ **solves** P

Example: Counters Revisited

- h **reduces** P_n into $P' = P_1 + \{(1, Dec, 1)\}$
- μ **solves 2-state** problem P'
- **Theorem** implies that μ **solves** P_n if $h(\tau)$ is **fair** in P' given that τ is a (fair) μ-path in P_n

  E.g., if μ-path τ is $(i, i-1, i-2, \ldots, 1, 0)$, $h(\tau)$ is $(1, 1, 1, \ldots, 1, 0)$ that is **fair**
- Theorem explains **why** μ works for **any** P_n given that it works for P'
- It also explains how generalization fails. E.g., if transition $(40, Dec, 39)$ **changed** to $(40, Dec, 50)$ in P_n, h reduces P_{100} to P' **but** does **not** preserve fairness

Non-Determinism and the Structure of Abstract Problems

- Write $P + E$ to denote problem P **extended** with set of **transitions** E
- Extension **admissible** given μ if **reachable states** in P and $P + E$ the same
- Clearly, if μ **solves** P and E **admissible**, μ **solves** $P + E$. Also:

Theorem: Policy μ generalizes from a **deterministic** problem P into Q if Q **reduces** into an admissible extension $P + E$ through h such that $h(\tau)$ contains a **finite number of transitions from** E for any μ-trajectory τ in Q.

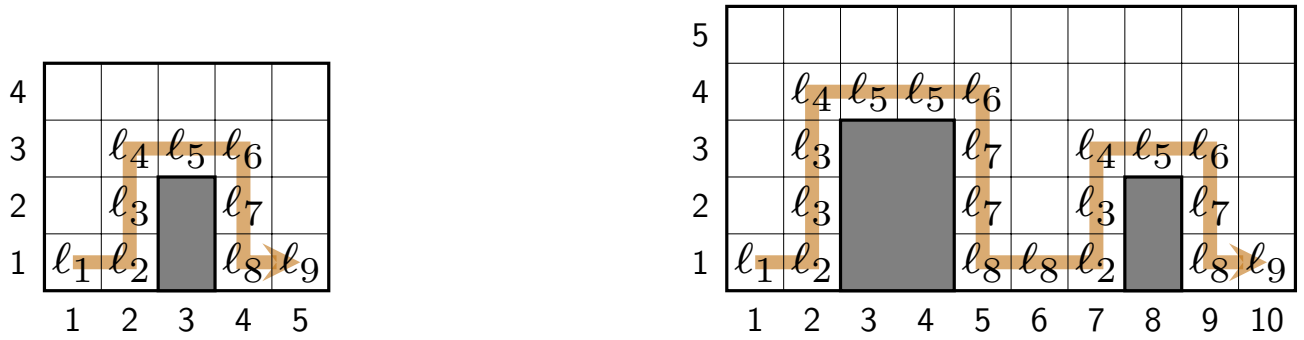
- **Intuition:** Extra transitions E can be used to reduce Q into $P + E$, but they must be **transient** in the induced executions $h(\tau)$
- Results generalized if conditions on P and Q applied to P_μ and Q_μ instead where P_μ is P $\mu(o(s))$ as only **applicable** action in each state s
- We will say that h reduces Q into P **given** μ when h reduces Q_μ into P_μ.

Example: Dust Cleaning Revisited

P_2 P_8

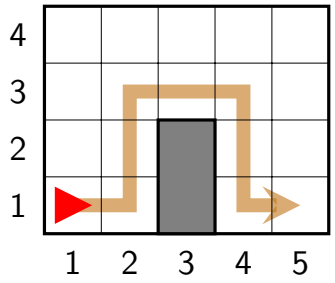
- P_n is **cleaning problem** over $1 \times n$ grid, robot on left, dirt distribution unknown
- **States** are tuples $\langle i, d_1, \ldots, d_n \rangle$: i is robot location, d_i status of cell i
- Policy μ : "if dirt, suck; else if not end, right" **generalizes** from P_2 to P_n
- **Reduction** h maps state $\langle i, d_1, \ldots, d_n \rangle$ into $\langle 1, d_i, d_n \rangle$ if $i < n$ else to $\langle 2, 0, d_n \rangle$
- **Extension** E contains **two extra transitions** by which:
  - ▷ Action $Right$ can "fail" in leftmost cell leaving robot in same cell, and making cell **clean** or **dirty.**
- Extra transitions account for **generalization** to **any grid size**, **any dirt distribution** using **theorem**

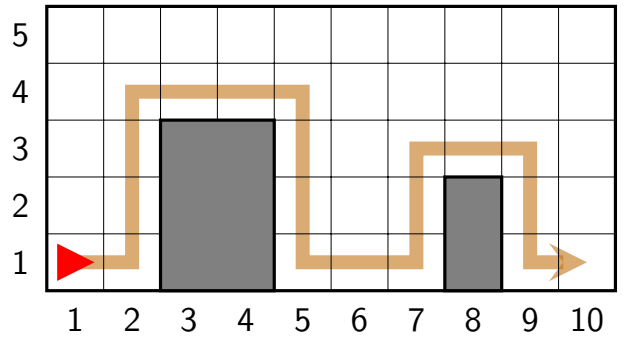
Example: Wall Following



- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - ▷ $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

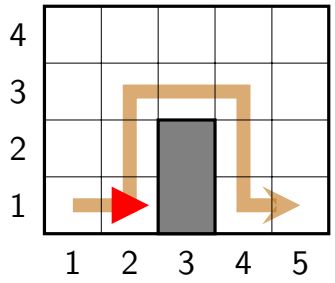
Example: Wall Following

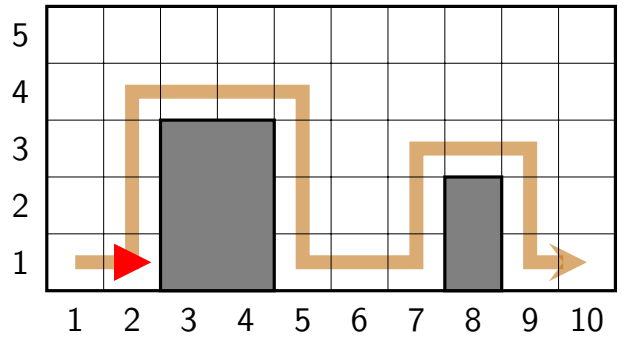




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when  , Left when  , and Right-Forward when
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

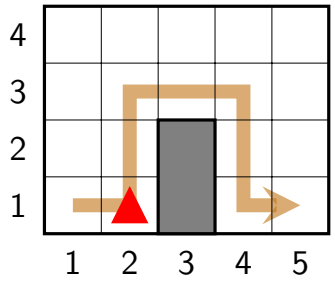
Example: Wall Following

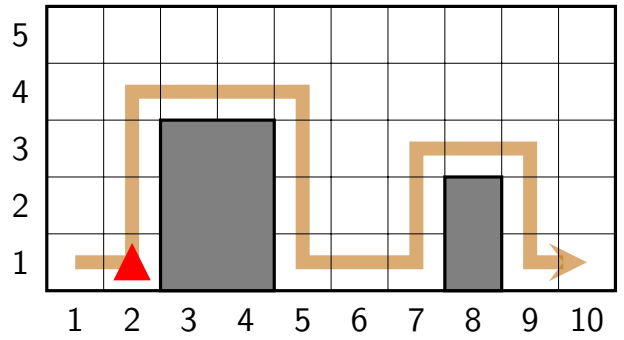




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

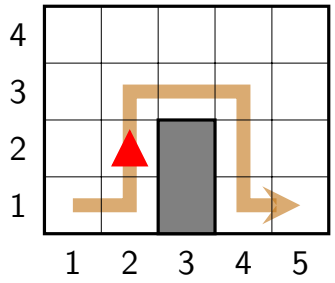
Example: Wall Following

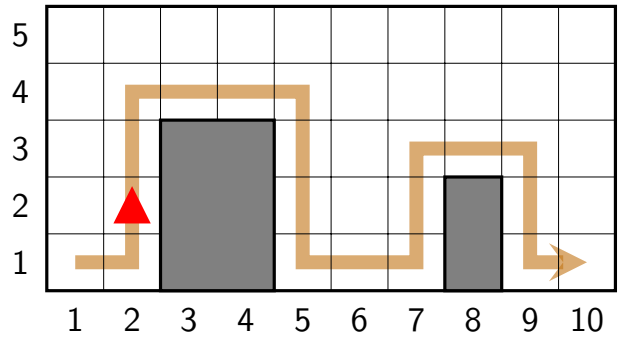




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

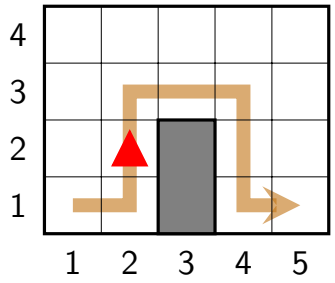
Example: Wall Following

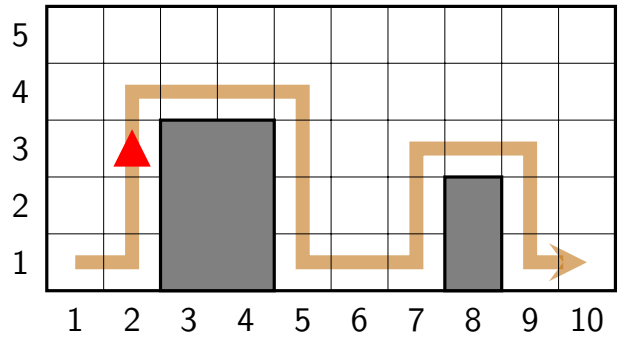




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

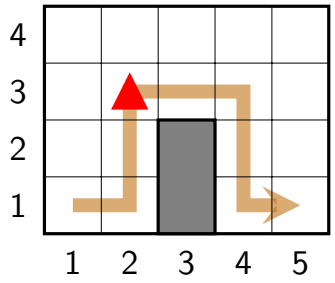
Example: Wall Following

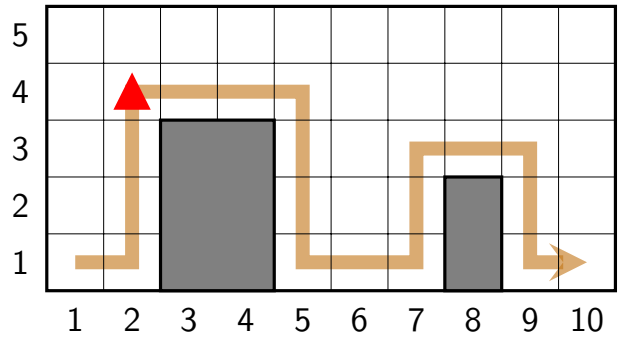




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

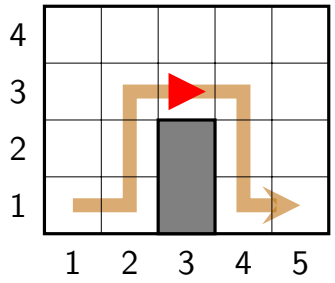
Example: Wall Following

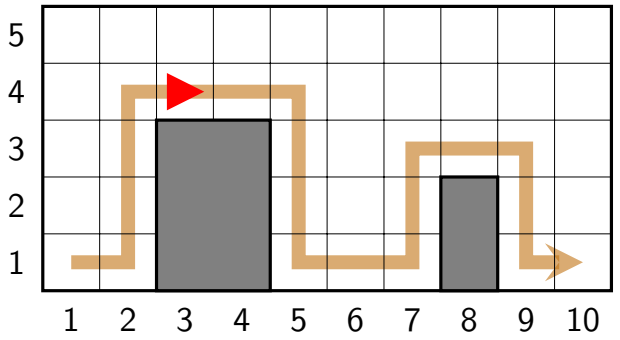




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

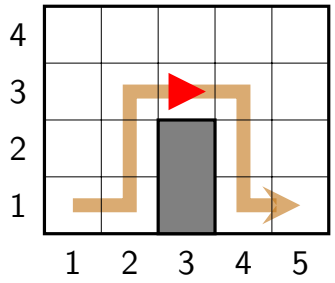
Example: Wall Following

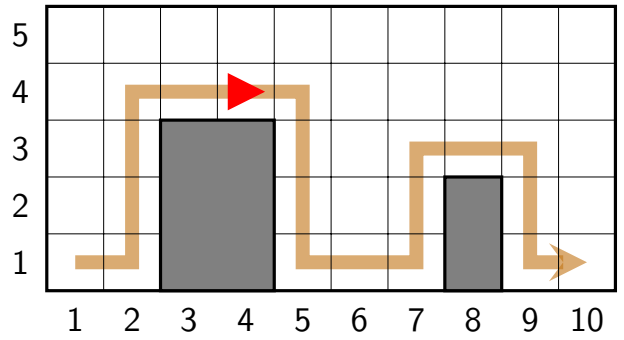




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

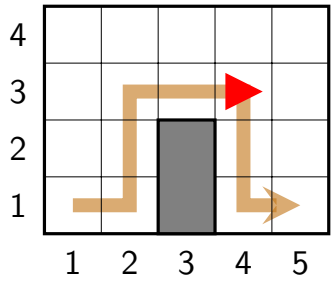
Example: Wall Following

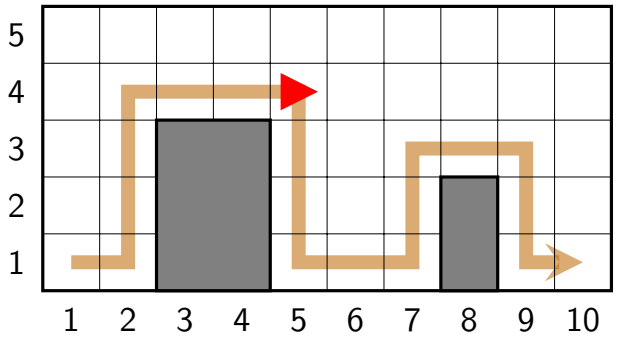




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

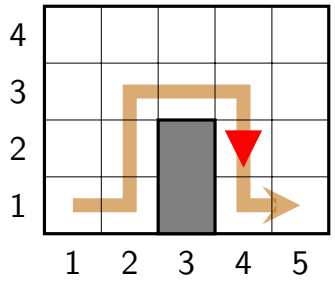
Example: Wall Following

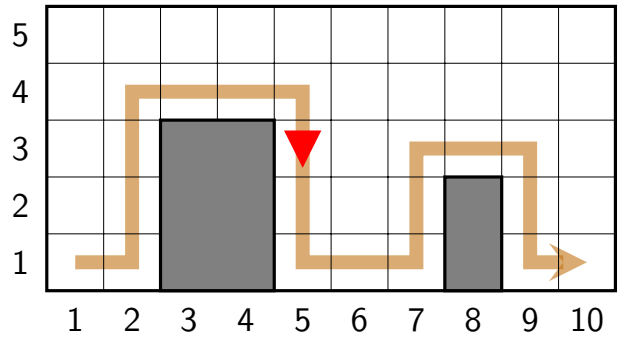




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

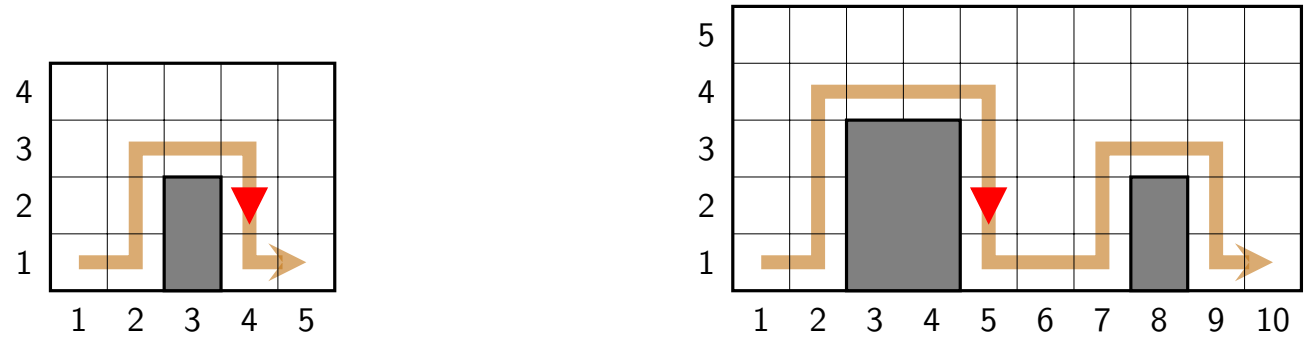
Example: Wall Following





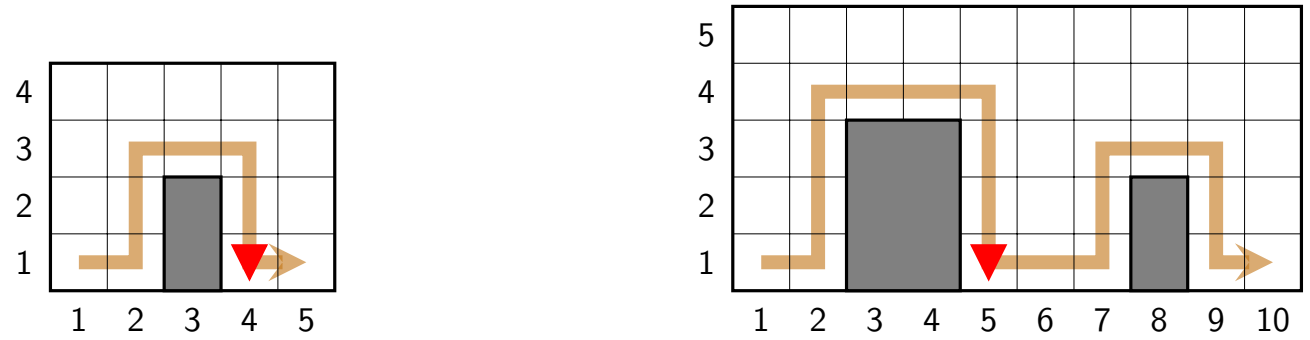
- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

Example: Wall Following



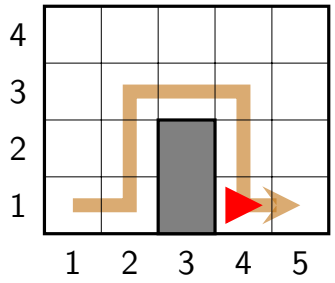
- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - ▷ $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

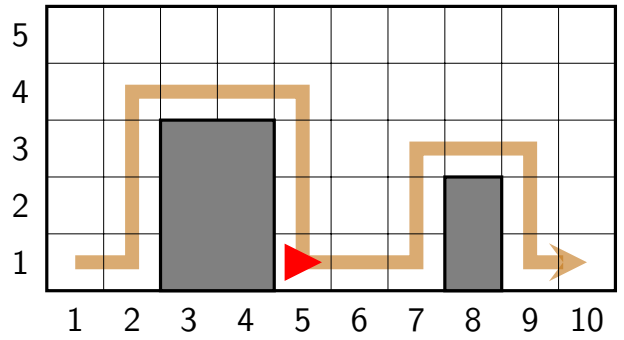
Example: Wall Following



- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - ▷ $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

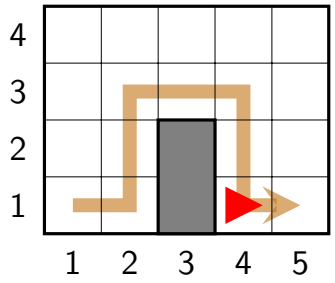
Example: Wall Following

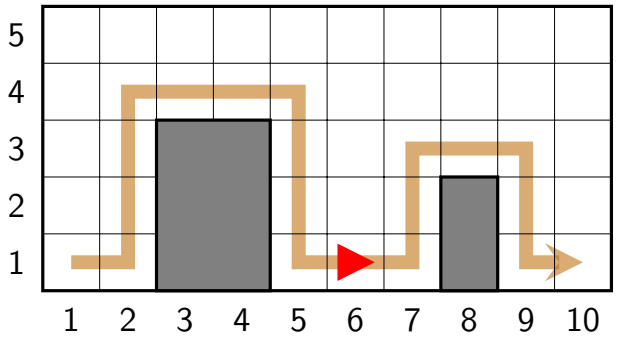




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when  , Left when  , and Right-Forward when
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

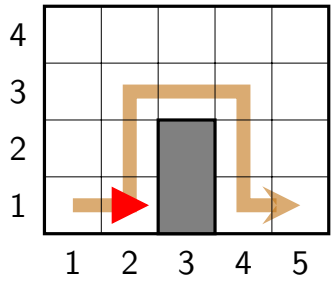
Example: Wall Following

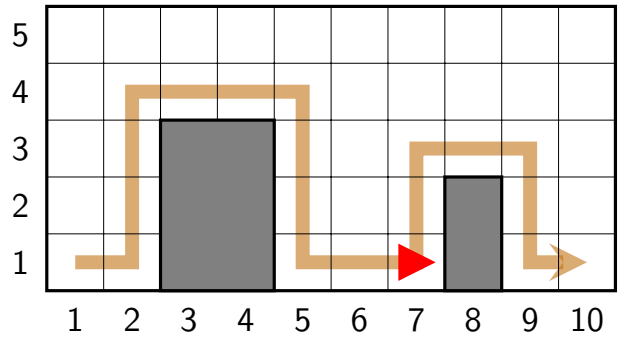




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

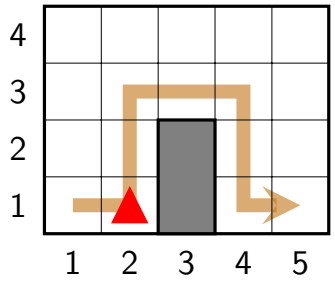
Example: Wall Following

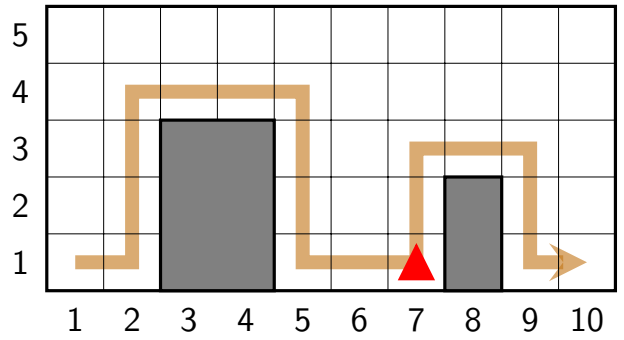




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

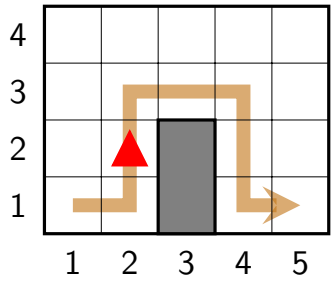
Example: Wall Following

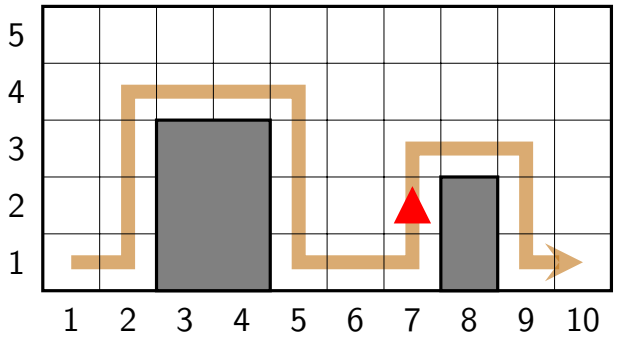




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

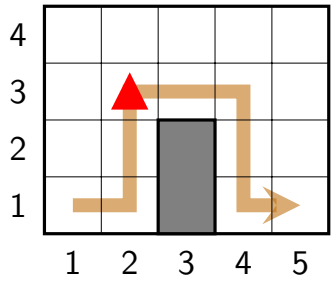
Example: Wall Following

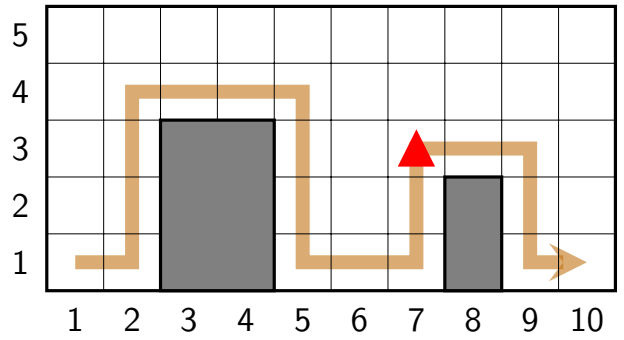




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

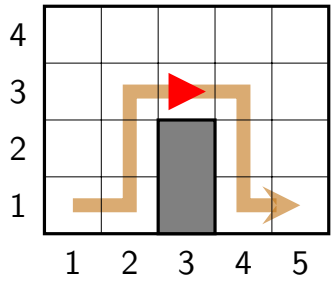
Example: Wall Following

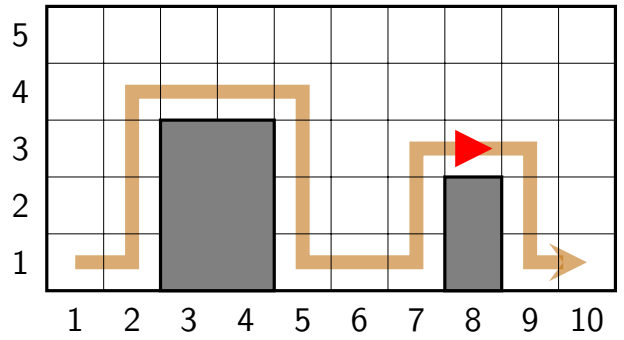




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

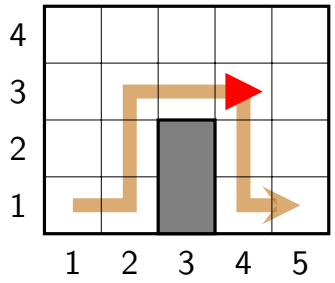
Example: Wall Following

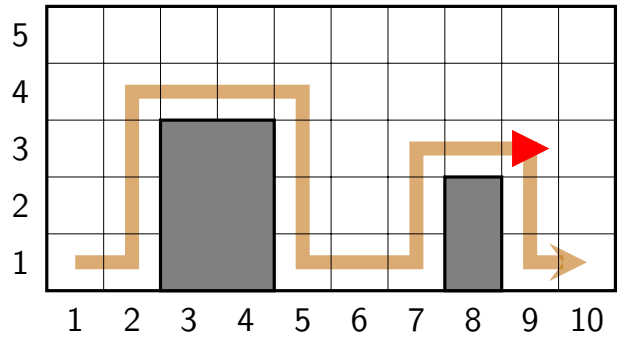




- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

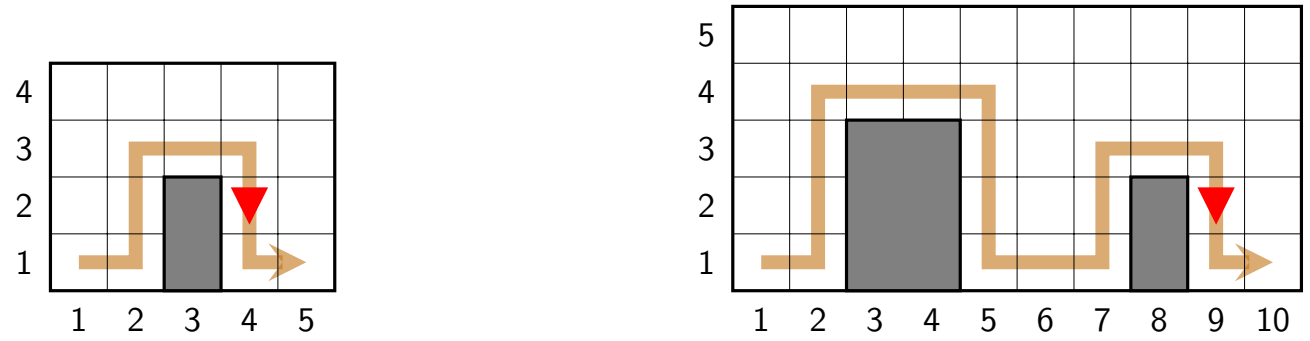
Example: Wall Following





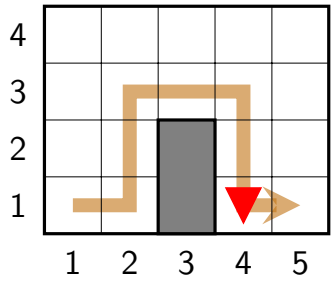
- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

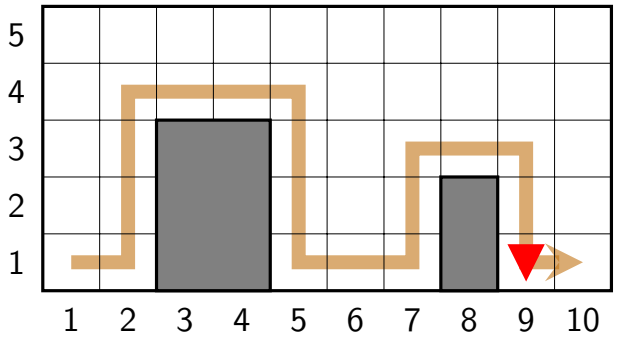
Example: Wall Following



- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

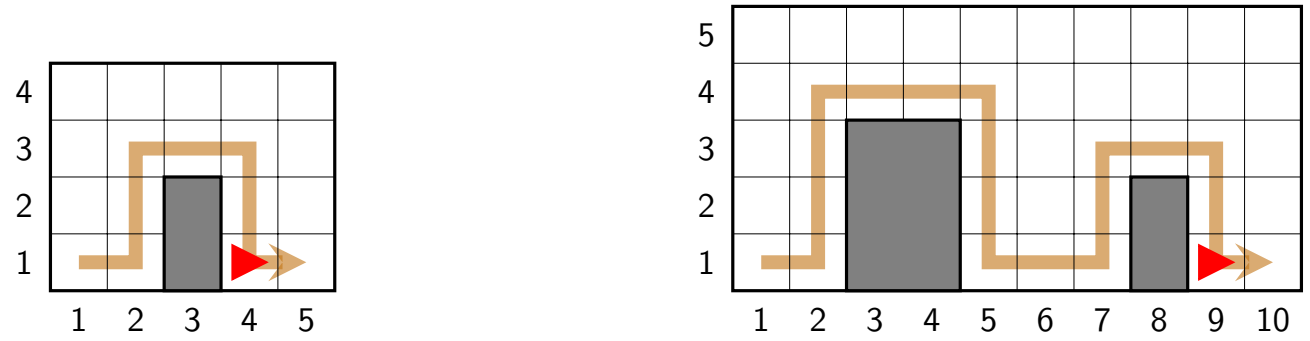
Example: Wall Following





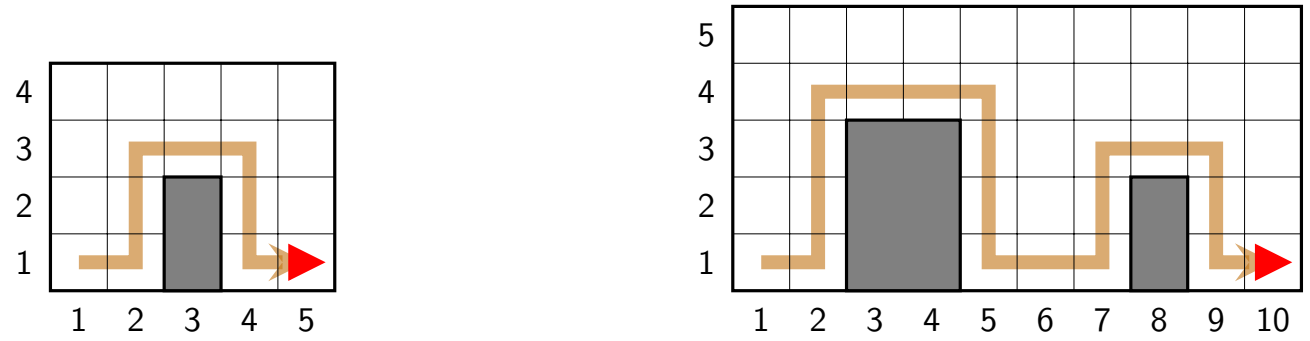
- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

Example: Wall Following



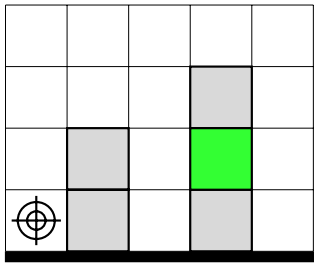
- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - ▷ $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

Example: Wall Following

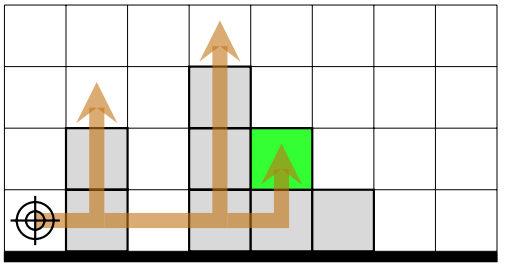


- **Problems** P, Q: navigating around **observable** walls for **observable** goal
- **States** are (c, d) where c is cell and d is direction
- **Policy** μ: Forward when ▷⋮, Left when ▷|, and Right-Forward when ▷⋰
- **Reduction** h maps pairs (c, d) in Q into $(\ell(c), d)$ in $P + E$ with $\ell(c)$ shown
- E contains **extra transitions** (s, a, s') where a is Forward, and $\langle s, s' \rangle$ is
  - $\langle (\ell_8, e), (\ell_2, e) \rangle$ or $\langle (\ell_i, d), (\ell_i, d) \rangle$ for $i = 3, 5, 7, 8$ and any heading d
- They enable **self-loops** and **re-start** to (ℓ_8, e) when past column at (ℓ_8, e)
- They account for **generalization** of μ to problems with **any number of columns** of **any height** and **width** using **theorem**

Example: Visual Blocks

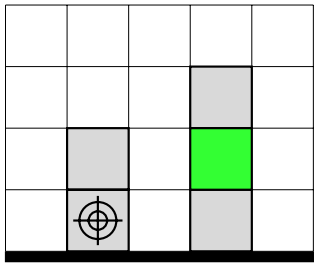


initial state with green block
in the middle of 2nd tower

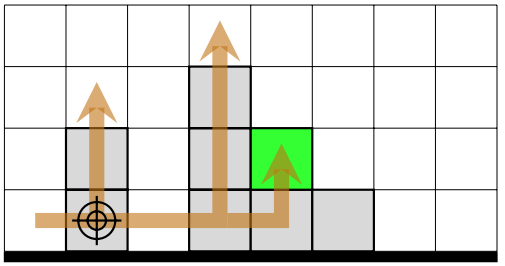


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks

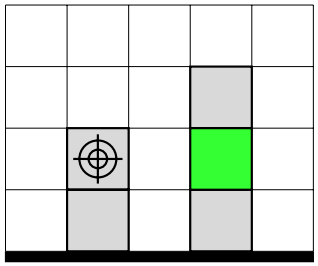


initial state with green block
in the middle of 2nd tower

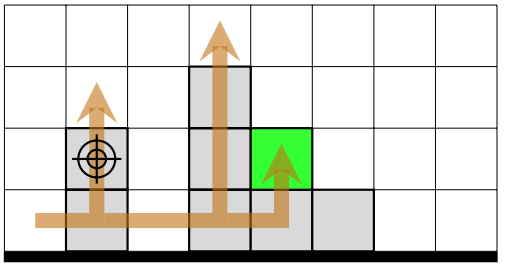


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks

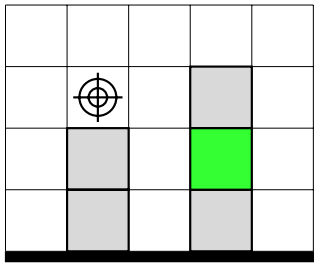


initial state with green block
in the middle of 2nd tower

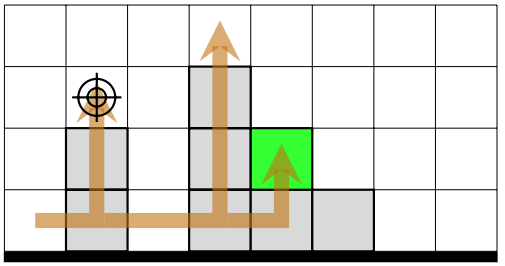


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks

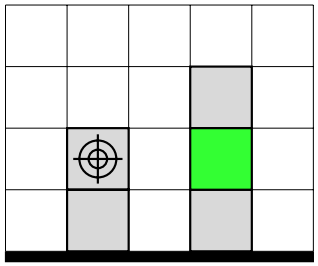


initial state with green block
in the middle of 2nd tower

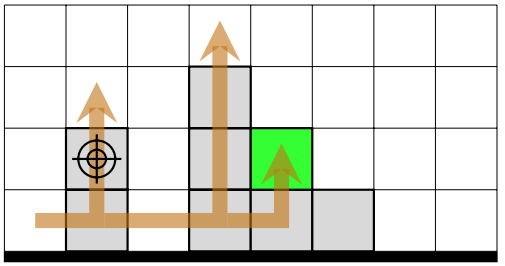


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks

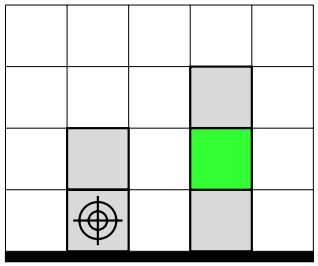


initial state with green block
in the middle of 2nd tower

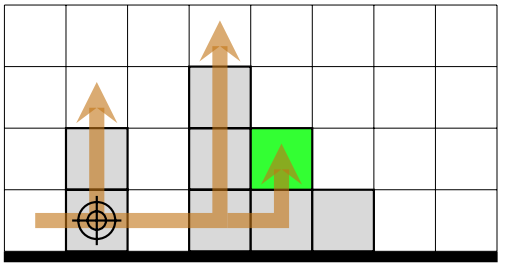


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks

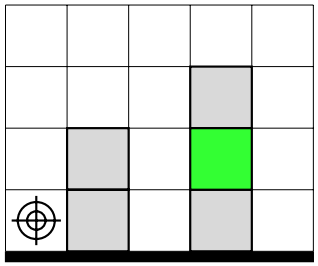


initial state with green block
in the middle of 2nd tower

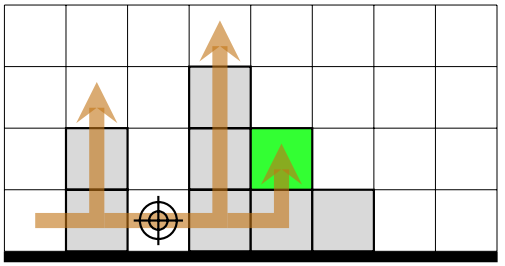


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks

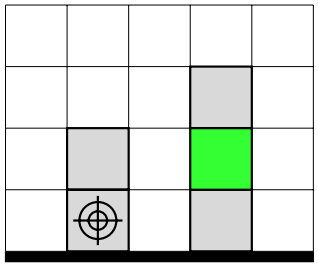


initial state with green block
in the middle of 2nd tower

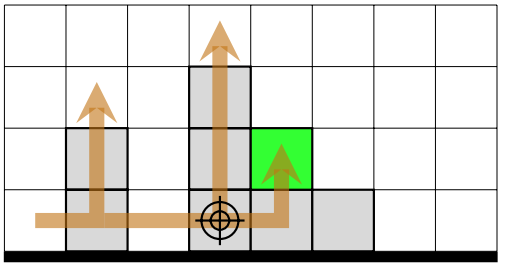


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks

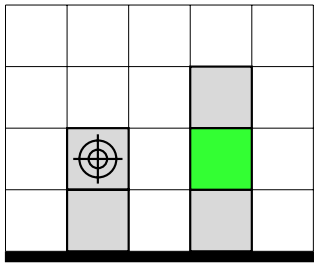


initial state with green block
in the middle of 2nd tower

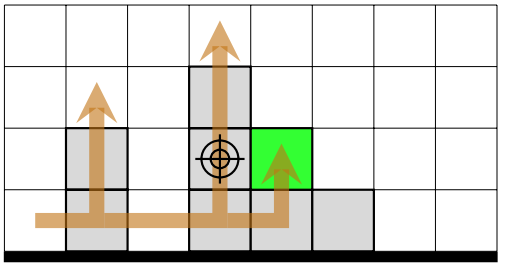


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks

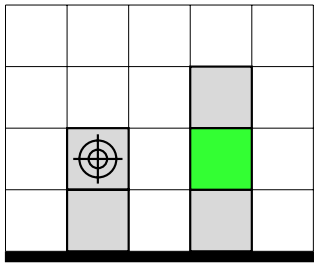


initial state with green block
in the middle of 2nd tower

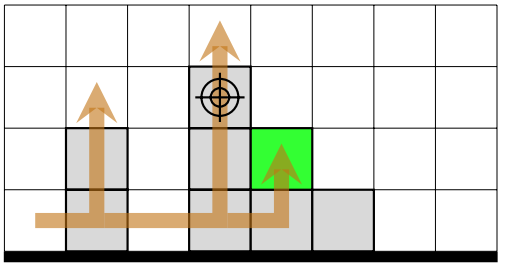


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks

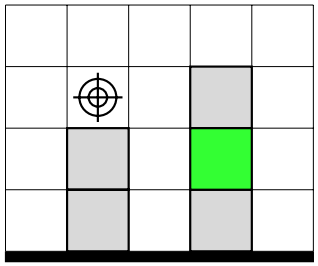


initial state with green block
in the middle of 2nd tower

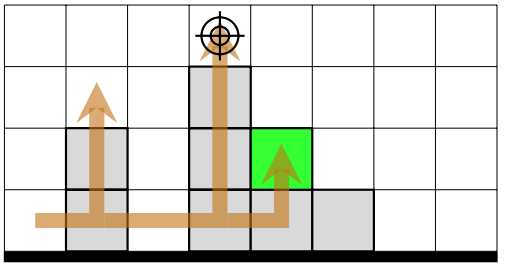


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks

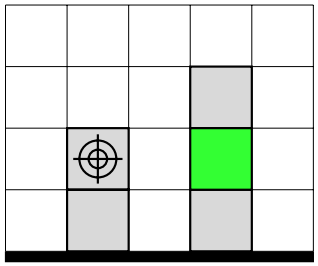


initial state with green block
in the middle of 2nd tower

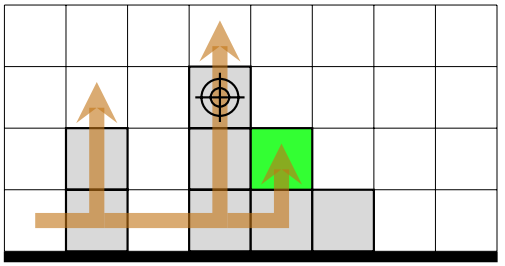


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks

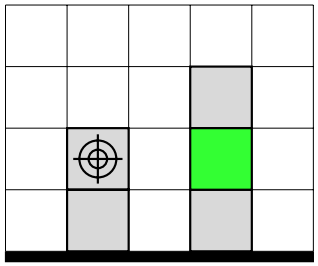


initial state with green block
in the middle of 2nd tower

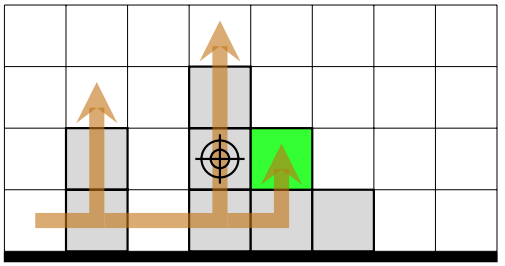


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks

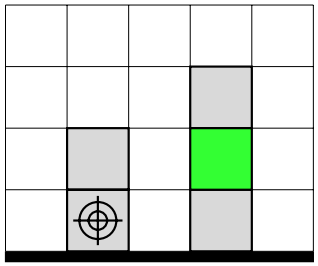


initial state with green block
in the middle of 2nd tower

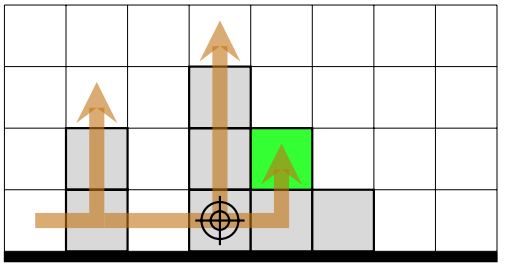


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks

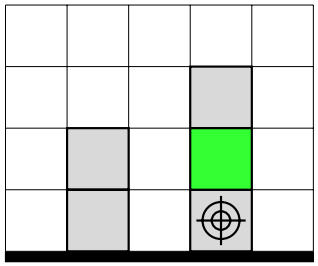


initial state with green block
in the middle of 2nd tower

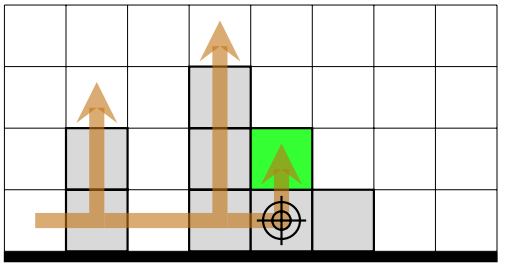


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks

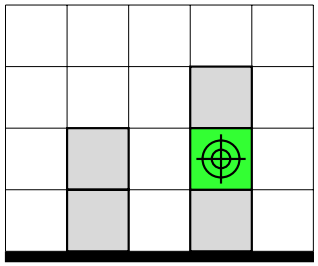


initial state with green block
in the middle of 2nd tower

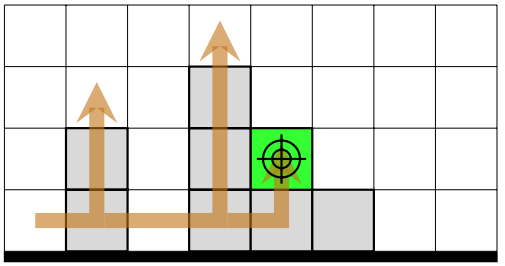


- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Example: Visual Blocks



initial state with green block
in the middle of 2nd tower



- **Problems** P, Q: place **visual marker** on bottom-left on **hidden** green block
- **Sense** if marker at table level, and on top of green/non-green block or air
- **States** are pairs (c, g) where c is **marker** cell and g is **hidden** (green block) cell
- **Initial states** in P are (c, g_i) where $c = (1, 1)$ and g_i is goal cell in second tower
- For given controller, Q **reduced** to $P + E$ by h that preserves observations and if marker is in goal/non-goal/empty tower
- **Extra transitions** E required for allowing such correspondence (27 in total)
- **Generalization** to **any number of towers** of **any height** using **theorem**

Related Work

Generalized planning: solution that works for multiple problems sought

- Problem is **hard** as the set of problems for which a **common solution** is sought must be **explicitly represented** and **solved**
- Our approach is slightly different: solution to **one problem** is shown to solve **many other structurally similar problems**

Transfer learning: how solution to one problem used for solving another problem

- Usually cast in **discounted reward setting (MDPs)** where there is **no crisp, qualitative notion of solution**
- Our approach focuses instead on **reaching goal with certainty** in partially observable setting
- **Probability values** are not relevant, and **cost/reward** considerations ignored
- Approaches applies directly to probabilistic **Goal POMDPs**

Conclusions

- We study conditions under which **controller** that **solves** a POND problem P **solves structurally similar** problems Q.
- Three key **concepts**: **reductions** h, **fairness**, abstraction through **non-determinism**
- Interesting **lessons** about **scalability** and **representation** (actions and observations must be shared)
- Account applies directly to **POMDPs** where goals to be achieved with certainty
- Work is useful for **understanding** general scope of controllers; further work required to gain **computational benefits** of it