

# Designing Agents with Task-Specific Minimal Representation

Joshua Hernandez

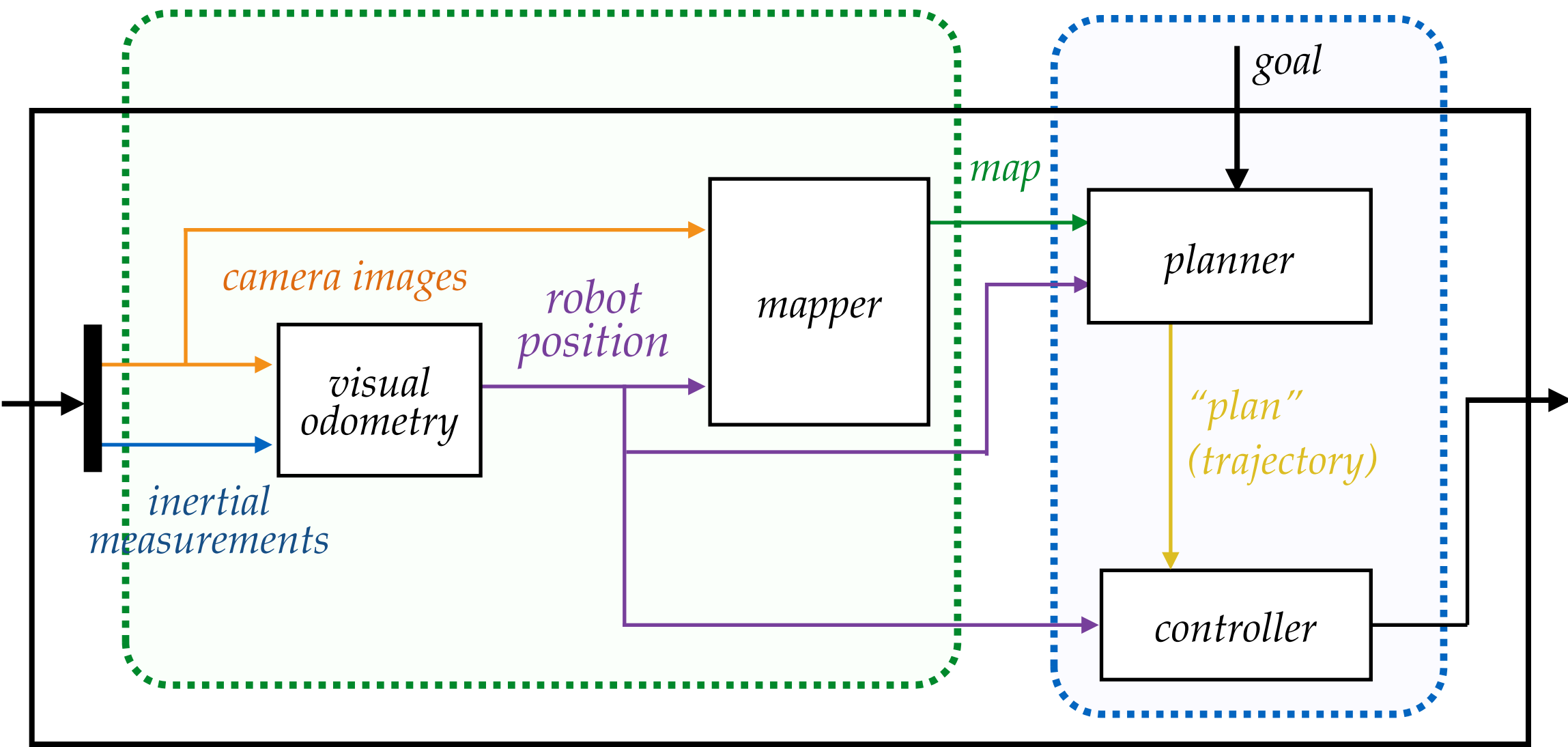
University of California, Los Angeles

*jheez@ucla.edu*

October 27, 2014

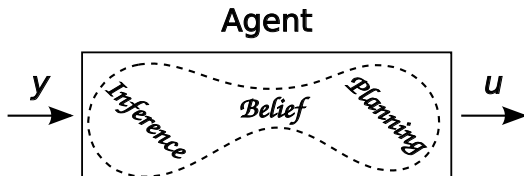
“inference”

“control”



# Separate Inference and Control

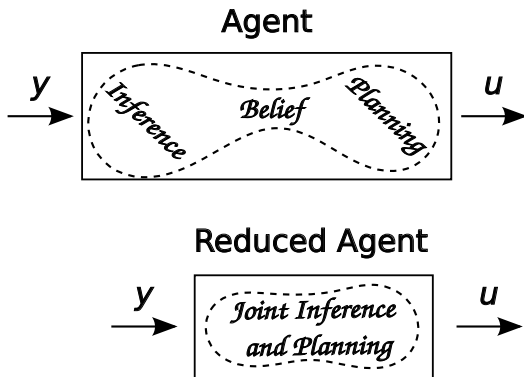
It's inefficient to do inference and control separately



Canonical belief spaces can become a serious informational bottleneck between inference and planning modules

# Joint Inference and Control

One approach is reduction



# Representation Reduction

There exists a great deal of literature on the subject of representation reduction

- Continuous case: Dimensionality Reduction, Information Bottleneck...
- Discrete case: FSM Reduction

I will focus on the latter discrete case.

# FSM

## Definition (Finite State Machine)

A **finite state machine** is a tuple

$$\langle \Sigma, \Gamma, S, s_0, \delta, \omega \rangle ,$$

where

- $\Sigma$  is an input alphabet,
- $\Gamma$  is an out alphabet,
- $S$  is a finite set of states,
- $s_0 \in S$  is an initial state,
- $\delta : S \times \Sigma \rightarrow S$  is a state-transition function,
- $\omega : S \times \Sigma \rightarrow \Gamma$  is an output function.

# ISFSM

## Definition (Incompletely-Specified Finite State Machine)

An **incompletely-specified finite state machine** is a tuple

$$\langle \Sigma, \Gamma, S, s_0, \delta, \omega \rangle ,$$

where

- $\Sigma$  is an input alphabet,
- $\Gamma$  is an out alphabet,
- $S$  is a finite set of states,
- $s_0 \in S$  is an initial state,
- $\delta : S \times \Sigma \rightarrow S \cup \{\phi\}$  is a state-transition function,
- $\omega : S \times \Sigma \rightarrow \Gamma \cup \{\epsilon\}$  is an output function,

and  $\phi$  and  $\epsilon$  denote unspecified outputs.

## Definition (Obedience)

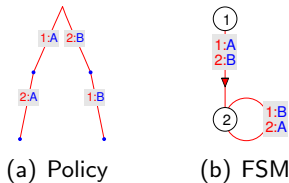
Given the decision policy  $P = \langle \mathcal{U}, T : \text{Sequences}(\mathcal{Y}) \rightarrow \mathcal{U} \cup \{\epsilon\} \rangle$ , we say that an ISFSM  $\langle \Sigma, \Gamma, S, s_0, \delta, \omega \rangle$  **obeys** the policy  $P$  if for every finite sequence  $y_1, \dots, y_n \in \mathcal{Y}$ , there exists a sequence  $s_0, \dots, s_{n-1} \subseteq S$  such that

$$s_i = \delta(s_{i-1}, y_i) \text{ for all } i = 1, \dots, n$$

and

$$T(y_1, \dots, y_n) = \omega(s_{n-1}, y_n) \quad \text{or} \quad T(y_1, \dots, y_n) = \epsilon.$$

## Example (Obedience)



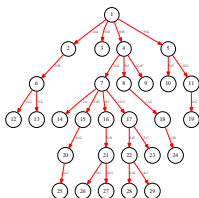


## Definition (Equivalent FSMs)

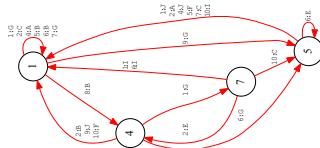
Two FSMs are equivalent if they obey all the same policies.

Equivalent FSMs may have very different forms

## Example (Reduced FSMs)



(c) Canonical FSM



(d) Equivalent reduced FSM

# Problem

## Problem (FSM minimization)

*Given a decision policy (or an ISFSM) how do we find an obedient (or equivalent) ISFSM with the smallest possible state set?*

It turns out,

- It is easy for CSFSM (polynomial in  $|S|$ ).
- It is hard (NP-hard) for ISFSM.

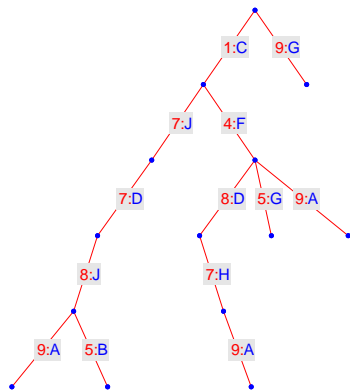
# Previous Work

## Previous work: Heuristics for ISFSMs

- C. P. Pfleeger. “State reduction in incompletely specified finite-state machines”. In: *IEEE Trans. Comput* 22.12 (1973), pp. 1099 –110
- A. Grasselli and F. Luccio. “A method for minimizing the number of internal states in incompletely-specified sequential networks”. In: *IEEE Trans. Comput* 13.3 (1965), pp. 350 –359
- J. Pena and A. Oliveira. “A new algorithm for the reduction of incompletely specified finite state machines”. In: *IEEE/ACM International Conference on Computer-Aided Design* (1998), pp. 4482 –489
- S. Goren and F. Ferguson. “On state reduction of incompletely specified finite state machines”. In: *Computers and Electrical Engineering* 33.1 (2007), pp. 58 –69

# Bit-at-a-time (Censi)

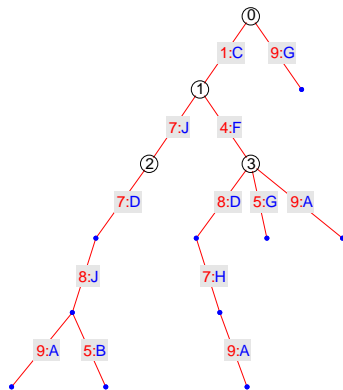
Proposed by Andrea Censi, MIT-LIDS: Greedily separate ambiguous contexts along decision tree.



(a) Decision Tree

# Bit-at-a-time (Censi)

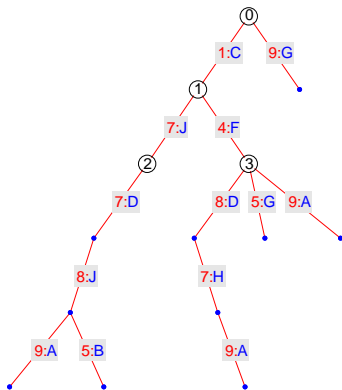
Proposed by Andrea Censi, MIT-LIDS: Greedily separate ambiguous contexts along decision tree.



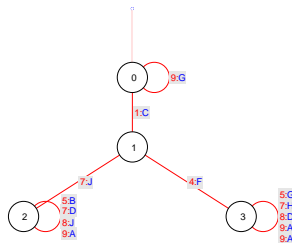
(a) Decision Tree

# Bit-at-a-time (Censi)

Proposed by Andrea Censi, MIT-LIDS: Greedily separate ambiguous contexts along decision tree.



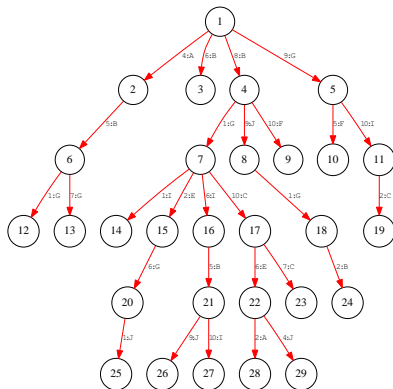
(a) Decision Tree



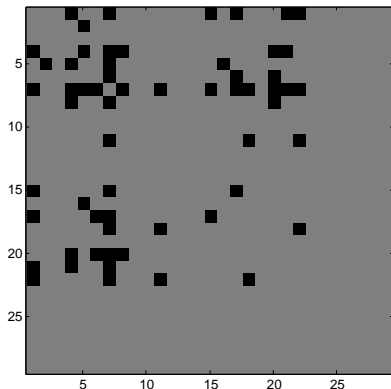
(b) Reduced FSM

# Greedy Clique Covering

Greedily combine compatible states



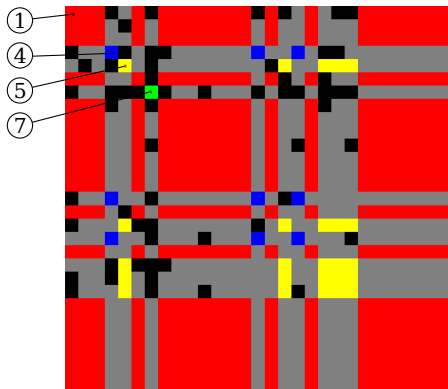
(c) Decision Tree



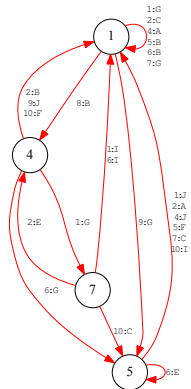
(d) Compatibility Matrix

# Greedy Clique Covering

Greedily combine compatible states



(e) Greedy Clique Covering



(f) Reduced Rep



# Alberto and Simão, '09

A. Simão A. Alberto. “Minimization of Incompletely Specified Finite State Machines Based on Distinction Graphs”. In: *Latin American Test Workshop* (2009), pp. 1 –6

## Method

- 1 Construct equivalence graph
- 2 Select maximal anticlique in equivalence graph
- 3 Merge remaining states with members of the anticlique graph

# 1. Equivalence Graph

Construct an equivalence graph using Hopcroft's algorithm<sup>1</sup> (its complement, the distinction graph, is shown).

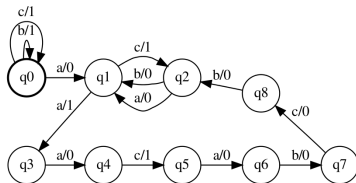


Figure 1: An incompletely-specified FSM

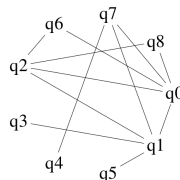


Figure 2: Distinction graph for the FSM in Figure 1

<sup>1</sup>J. Hopcroft. "An  $n \log n$  algorithm for minimizing states in a finite automaton". In: *Theory of machines and computations* (1971), pp. 198 –196.

## 2. Maximal Anticlique

The next step requires that we find a maximal anticlique of the equivalence graph.

- Cliques on the equivalence graph identify sets of states that can be collapsed into a single state. The minimal clique-covering, that is the smallest collection of disjoint cliques that covers the equivalence graph, corresponds to a minimal reduction of the FSM.
- Finding a maximum anticlique is also NP-hard (same paper), but this algorithm only *approximates* a minimal reduction

### 3. Iterative Merge

Next, states outside the anticlique selected, following a simple heuristic, and merged with compatible states on the anticlique (there must exist at least one).

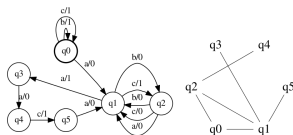


Figure 3: Reduced FSM after first iteration and its distinction graph

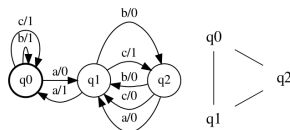


Figure 4: Reduced FSM after second iteration and its distinction graph

# Limitations

- Finding the maximal anticlique is still an NP-hard problem.
- Can do very badly on certain cases

## Example (Failure Case)

Consider the case when the distinction graph looks like

