1 Formalization

Let $\mathcal{Y}$ be a set of observations and $\mathcal{U}$ a set of control actions. The original formalization defines a decision table as a map $\mathcal{T} : \mathcal{C} \to \mathcal{U}$, where $\mathcal{C} = \bigcup_{i=1}^{n} \mathcal{Y}^i$ for some $n \in \mathbb{N}$. This definition is often stronger than necessary, as there may be contexts c in which certain observations $y \in \mathcal{Y}$ are never encountered. In such a context, any assignment $\mathcal{T}(c) \in \mathcal{U}$, produces the same policy, although the choice of $\mathcal{T}(c)$ can affect a policy's reducibility.

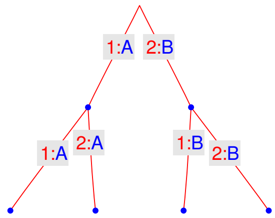
Example 1 (Equivalent Decision Tables) *Suppose* $\mathcal{Y} = \{1, 2\}$, $\mathcal{U} = \{A, B\}$, $\mathcal{C} = \bigcup_{i=1}^{2} \mathcal{Y}^i$ *and*

$$\mathcal{T}(c) = \begin{cases} A & c \in \{(1), (1, 2)\} \\ B & c \in \{(2), (2, 1)\} \\ \mathcal{T}'(c) & \textit{otherwise.} \end{cases},$$

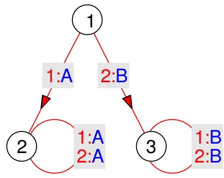
is an optimal policy, for arbitrary $\mathcal{T}' : \mathcal{C} \to \mathcal{U}$. *However, depending on the choice of* $\mathcal{T}'$ *(highlighted in the tables below), the completed policy can have differently-sized minimal representations:*

$\mathcal{T} : \mathcal{C} \to \mathcal{U}$	
(1)	$\mapsto A$
(2)	$\mapsto B$
$(1, 1)$	$\mapsto A$
$(1, 2)$	$\mapsto A$
$(2, 1)$	$\mapsto B$
$(2, 2)$	$\mapsto B$

(a) Decision Table



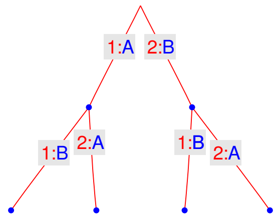
(b) Decision Tree



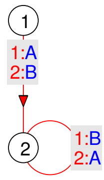
(c) Minimal Policy

$\mathcal{T} : \mathcal{C} \to \mathcal{U}$	
(1)	$\mapsto A$
(2)	$\mapsto B$
$(1, 1)$	$\mapsto B$
$(1, 2)$	$\mapsto A$
$(2, 1)$	$\mapsto B$
$(2, 2)$	$\mapsto A$

(d) Decision Table



(e) Decision Tree



(f) Minimal Policy

Instead, we propose an "incompletely-determined" formalization:

Definition 1 (Policies) *Given a set* $\mathcal{Y}$ *of observations, recursively construct*

$$\mathcal{C}_0 = \{\emptyset\} \quad \textit{and} \quad \mathcal{C}_{i+1} = \{(c, y_{i+1}) : c \in \mathcal{C}_i,\ y_{i+1} \in \mathcal{Y}_c\}, \tag{1}$$

where $\mathcal{Y}_c \subseteq \mathcal{Y}$ *are the observations that may be seen in context* c. *Let* $\mathcal{C} = \bigcup_{i=0}^{\infty} \mathcal{C}_i$. *A* ***policy*** P *is then a tuple* $\langle \mathcal{C}, \mathcal{U}, \mathcal{T}, \mathcal{Y} \rangle$, *where* $\mathcal{U}$ *is some decision set and* $\mathcal{T} : \mathcal{C} \setminus \{\emptyset\} \to \mathcal{U}$.

Definition 2 (Completely-Determined Policies) *If* $\mathcal{Y} = \bigcup_{\mathcal{C}} \mathcal{Y}_c$ *and* $\mathcal{C} = \mathcal{Y}^{\leq n}$ *for some* $n \in \mathbb{N}$, *then* $P = \langle \mathcal{C}, \mathcal{U}, \mathcal{T}, \mathcal{Y} \rangle$ *is* ***completely determined****. A* ***completion*** *of* P *is a policy* $P' = \langle \mathcal{C}', \mathcal{U}', \mathcal{T}', \mathcal{Y} \rangle$ *such that*

$$\mathcal{Y} \subseteq \mathcal{Y}', \quad \mathcal{C}' = \bigcup_{i=0}^{\infty} (\mathcal{Y}')^i, \quad \mathcal{U} \subseteq \mathcal{U}', \quad \textit{and} \quad \mathcal{T}'|_{\mathcal{C}} = \mathcal{T}. \tag{2}$$

Let $\mathrm{Comp}(P)$ *be the set of completions of the policy* P.

Definition 3 (FSM Representations) *An* ***FSM representation*** *(or just* ***representation****) is a tuple* $\langle \mathcal{C}, \mathcal{R}, \mathcal{U}, \mathcal{S}, \mathcal{T}, \mathcal{Y} \rangle$ *(abbreviated to* $\langle \mathcal{R}, \mathcal{S} \rangle$ *when* $P = \langle \mathcal{C}, \mathcal{U}, \mathcal{T}, \mathcal{Y} \rangle$ *is given), with "states"* $\mathcal{S} \subseteq \mathbb{N}$ *and state assignments* $\mathcal{R} : \mathcal{C} \to \mathcal{S}$, *such that*

$$\mathcal{R}(c) = \mathcal{R}(c') \quad \textit{and} \quad y \in \mathcal{Y}_c \cap \mathcal{Y}_{c'} \implies \mathcal{T}(c, y) = \mathcal{T}(c', y). \tag{3}$$

Let $\mathrm{Rep}(P)$ *be the set of representations of the policy* P.

Definition 4 (Minimal Representations) *The **size** of an FSM representation is the cardinality of its state set. A representation* $\langle \mathcal{R}, \mathcal{S} \rangle$ *of* P *is **minimal** if* $|\mathcal{S}| = \min\{|\mathcal{S}'| : \langle \mathcal{R}', \mathcal{S}' \rangle \in \mathrm{Rep}(P)\}$*. A representation* $\langle \mathcal{R}', \mathcal{S}' \rangle$ *is a **reduction** of the representation* $\langle \mathcal{R}, \mathcal{S} \rangle$ *if there is a surjection* $\phi : \mathcal{S} \to \mathcal{S}'$ *such that* $\mathcal{R}' = \phi(\mathcal{R})$*.*

Example 2 *If* $\mathcal{C} = \{c_1, c_2, \ldots\}$*, then we have a canonical representation* $\langle \mathcal{R}, \mathcal{S} \rangle$*, where*

$$\mathcal{S} = \{1, \ldots, |\mathcal{C}|\} \quad \textit{and} \quad \mathcal{R} : c_k \mapsto k. \tag{4}$$

It can be shown that the size of a minimal representation of a policy P is equal to the minimum size of the minimal representations of its completions, i.e.

$$\min\{|\mathcal{S}'| : \langle \mathcal{R}', \mathcal{S}' \rangle \in \mathrm{Rep}(P)\} = \min\{|\mathcal{S}'| : \langle \mathcal{R}', \mathcal{S}' \rangle \in \mathrm{Rep}(P'),\ P' \in \mathrm{Comp}(P)\} \tag{5}$$

Incompletely-determined policies allow more freedom in representation reduction, as shown in the next example.

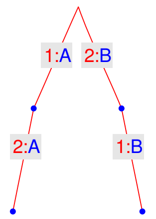
Example 3 (Incompletely-Determined Policies) *Let* $\mathcal{C} = \{\emptyset, (1), (2), (1,2), (2,1)\}$*,* $\mathcal{U} = \{A, B\}$*, and*

$$\mathcal{T}(c, y) = \begin{cases} A & c \in \{(1), (1,2)\} \\ B & c \in \{(2), (2,1)\} \end{cases}.$$

Observe that the minimal policy is the same as that of the completely-determined policy in Example 1(f).

$\mathcal{T} : \mathcal{C} \setminus \{\emptyset\} \to \mathcal{U}$	
(1)	$\mapsto A$
(2)	$\mapsto B$
$(1,2)$	$\mapsto A$
$(2,1)$	$\mapsto B$

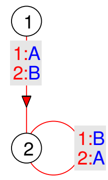
(a) Decision Table



(b) Decision Tree

$\mathcal{R} : \mathcal{C} \to \mathcal{S}$	
$\emptyset$	$\mapsto 1$
(1)	$\mapsto 2$
(2)	$\mapsto 2$
$(1,2)$	$\mapsto 2$
$(2,1)$	$\mapsto 2$

(c) Representation



(d) Minimal Policy

2 Algorithm

To find a minimum representation of a given policy, we first compute a graph of reducibility relations, then compute a minimal clique-covering. For practical computation of reducibility, we'll start with the weaker condition of compatibility.

2.1 Reducibility Relations

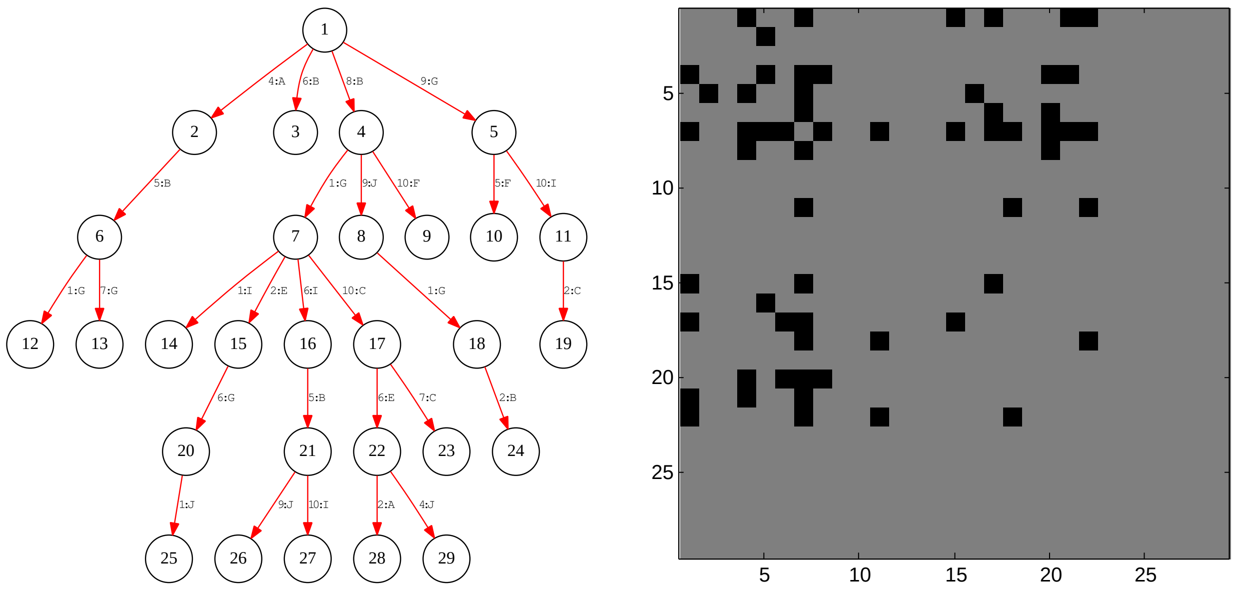
Definition 5 (Reducibility) *For a given policy* $P = \langle \mathcal{C}, \mathcal{U}, \mathcal{T}, \mathcal{Y} \rangle$*, two contexts* $c_1, c_2 \in \mathcal{C}$ *are **reducible** (write* $c_1 \sim c_2$*) if there exists a representation* $\langle \mathcal{R}, \mathcal{S} \rangle$ *of* P *such that* $\mathcal{R}(c_1) = \mathcal{R}(c_2)$*. Likewise, for a given representation* $R = \langle \mathcal{R}, \mathcal{S} \rangle$*, two states* $s_1, s_2 \in \mathcal{S}$ *are **reducible** if there exists a reduction* $(\phi, \langle \mathcal{R}', \mathcal{S}' \rangle)$ *of* R *such that* $\phi(s_1) = \phi(s_2)$*.*

Observe that for any representation $\langle \mathcal{C}, \mathcal{R}, \mathcal{U}, \mathcal{S}, \mathcal{T}, \mathcal{Y} \rangle$, the contexts $c_1, c_2 \in \mathcal{C}$ are reducible if and only if the states $\mathcal{R}(c_1)$ and $\mathcal{R}(c_2)$ are reducible. Observe also that for incompletely-determined policies, reducibility is a symmetric but not-necessarily-transitive relation

Example 4 (Non-Transitive Reducibility) *Suppose* $\mathcal{Y} = \{1, 2, 3\}$*,* $\mathcal{C} = \{\emptyset, (1), (2), (1,3), (2,3)\}$*,* $\mathcal{U} = \{A, B\}$*, and*

$$\mathcal{T}(c) = \begin{cases} A & c \in \{(1), (1,3)\} \\ B & c \in \{(2), (2,3)\} \end{cases}. \tag{6}$$

Observe that, under this policy, $\emptyset \sim (1)$ *and* $\emptyset \sim (2)$*, but* $(1) \not\sim (2)$*, since* $\mathcal{T}(1,3) \neq \mathcal{T}(2,3)$*.*



(a) Canonical Representation (b) Compatibility Matrix

However, it can be shown that, under a completely-determined policy, reducibility induces an equivalence relation. In either case, we compute reducibility using the following criterion:

Lemma 1 *Two contexts $c_1, c_2 \in \mathcal{C}$ are reducible iff*

$$\mathcal{T}(c_1, s) = \mathcal{T}(c_2, s) \quad \textit{for all} \quad s \in \mathcal{Y}^* \quad \textit{such that} \quad (c_1, s), (c_2, s) \in \mathcal{C} \tag{7}$$

This informs the following algorithm

Algorithm 1: Compute Reducibility Relations

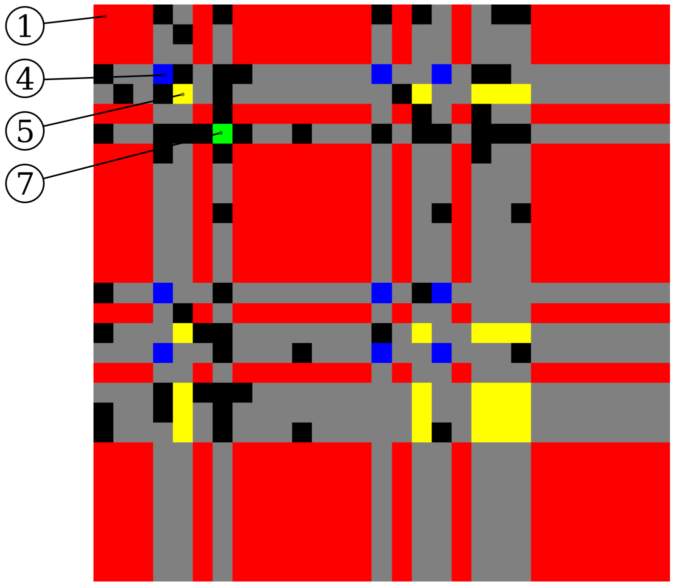
Input: A representation $\langle \mathcal{C}, \mathcal{R}, \mathcal{U}, \mathcal{S}, \mathcal{T}, \mathcal{Y} \rangle$
Output: A reducibility matrix $A : \mathcal{S} \times \mathcal{S} \to \{\mathbf{true}, \mathbf{false}\}$.

```
A(s_1, s_2) ⇐ true   for all s_1, s_2 ∈ S.
repeat
  isChanged ⇐ false
  for s_1 < s_2 ∈ S do
    if A(s_1, s_2) = true then
      for c_1 ∈ R^{-1}(s_1), c_2 ∈ R^{-1}(s_2) do
        for y ∈ Y_{c_1} ∩ Y_{c_2} do
          if T(c_1, y) ≠ T(c_2, y) or ~A(R(c_1, y), R(c_2, y)) then
            A(s_1, s_2) ⇐ false.
            isChanged ⇐ true.
          end if
        end for
      end for
    end if
  end for
until ~isChanged
```

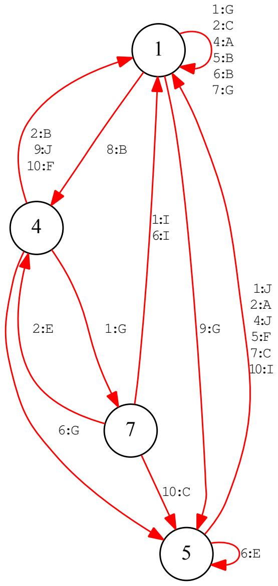
Now, although reducibility is not an equivalence relation, any reduction $\phi : \mathcal{S} \to \mathcal{S}'$ induces an equivalence relation, partitioning $\mathcal{S}$ into cliques of mutually-reducible states, i.e.

$$\mathcal{S} = \bigsqcup_{s' \in \mathcal{S}'} \phi^{-1}(s'), \quad \text{where} \quad \phi(s_1) = \phi(s_2) \implies A(s_1, s_2) \tag{8}$$

Thus, a minimal reduction induces a minimal clique partition of the reducible states of a representation.



(c) Greedy Clique Covering of 4(b)



(d) Reduced Representation

2.2 Assembling Cliques

Notation 1 (Arrow notation) *For a policy $P = \langle \mathcal{C}, \mathcal{U}, \mathcal{T}, \mathcal{Y} \rangle$, write $c \to c'$ if $c = (c_1, \ldots, c_i) \in \mathcal{C}_i \subseteq \mathcal{C}$ and $c' = (c_1, \ldots, c_i, y) \in \mathcal{C}_{i+1} \subseteq \mathcal{C}$, for some i. For a representation $\langle \mathcal{R}, \mathcal{S} \rangle$ of P, write $s_1 \to s_2$ if there are $c_1 \in f^{-1}(s_1)$ and $c_2 \in f^{-1}(s_2)$ such that $c_1 \to c_2$.*

We propose the following, greedy, approximate algorithm Although we have no proof that this algorithm

---

Algorithm 2: Compute Clique Covering

Input: A representation $\langle \mathcal{C}, \mathcal{R}, \mathcal{U}, \mathcal{S}, \mathcal{T}, \mathcal{Y} \rangle$ with $s_1 < s_2$ only if $s_2 \not\to s_1$.
Input: A reducibility matrix $A : \mathcal{S} \times \mathcal{S} \to \{\mathbf{true}, \mathbf{false}\}$ as computed by Algorithm 1.
Output: A partition function $\phi : \mathcal{S} \to \mathcal{S}'$ with $\phi(s_1) = \phi(s_2)$ only if $A(s_1, s_2)$.

$\mathcal{S}' \Leftarrow \mathcal{S}$
$\phi \Leftarrow id_{\mathcal{S}}$
$unused \Leftarrow \mathcal{S}$
while $|unused| > 0$ **do**
  $s_1 \Leftarrow \min(unused)$
  $unused \Leftarrow unused \setminus \{s_1\}$
  for $s_2 \in unused$ **do**
    if $A(s_1, s_2)$ **then**
      $\phi(s_2) \Leftarrow s_1$
      $unused \Leftarrow unused \setminus \{s_2\}$
    end if
  end for
end while

---

produces minimal representations of a given policy, it is not inconceivable that this or another greedy policy could work. In general, the Minimal Clique Covering problem is NP-Complete, but the tree structure of the decision policy is an a constraint that may simplify the problem.

3 Code

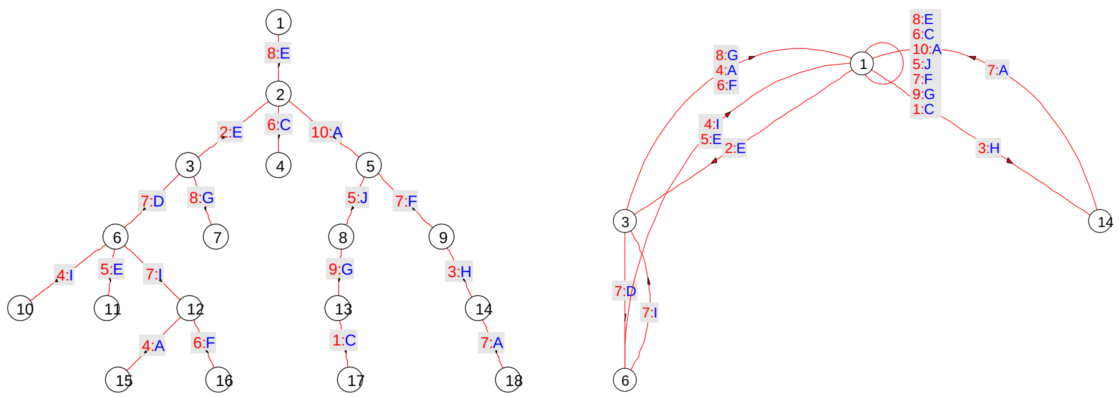
The main MATLAB routine is in `decision_script.m`. By default it generates a random decision tree, whose branching is governed by a Poisson distribution.

3.1 Console Output

```
Initial FSM:
----------------------------------------------------
01:  08-->(05, 02),
02:  02-->(05, 03),  06-->(03, 04),  10-->(01, 05),
03:  07-->(04, 06),  08-->(07, 07),
04:
05:  05-->(10, 08),  07-->(06, 09),
06:  04-->(09, 10),  05-->(05, 11),  07-->(09, 12),
07:
08:  09-->(07, 13),
09:  03-->(08, 14),
10:
11:
12:  04-->(01, 15),  06-->(06, 16),
13:  01-->(03, 17),
14:  07-->(01, 18),
15:
16:
17:
18:

Final clique covering: {1, 2, 4, 5, 7, 8, 9, 10, 11, 13, 15, 16, 17, 18}  {3, 12}  {6}  {14}
```

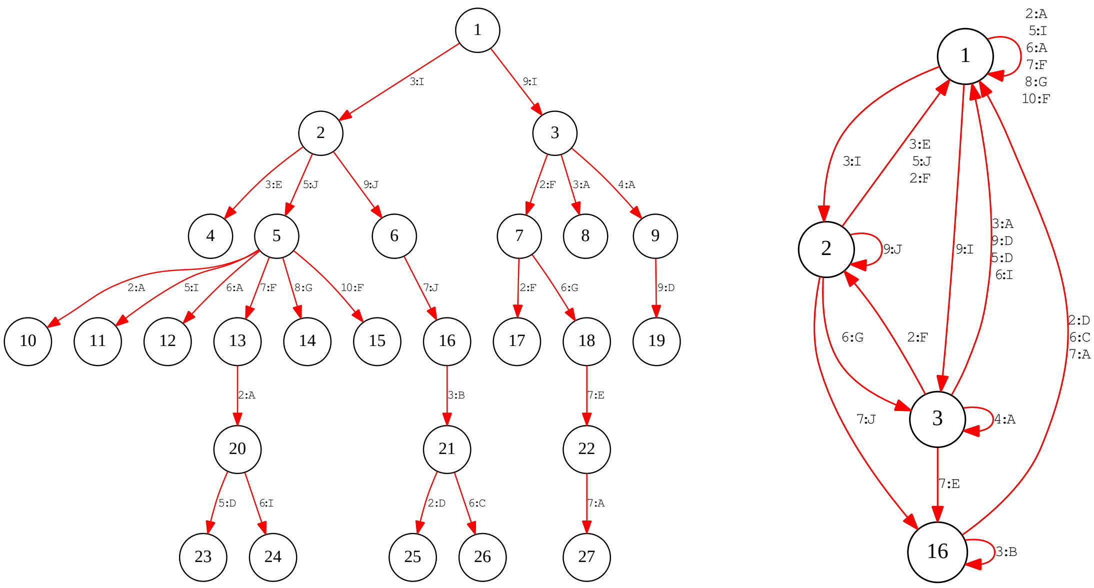
3.2 Visualizations



(a) Original Policy (b) Reduced Policy

Figure 4: MATLAB Visualizations

An optional section of this script uses Graphviz to produce high-quality FSM visualizations.



(a) Original Policy (b) Reduced Policy

Figure 5: Graphviz Visualizations

This makes a system call to the script `drawgraph.sh`, which runs in the bash shell, and uses a few non-standard graphics utilities. In Ubuntu, these dependencies should be resolved by the command

```
$ sudo apt-get install ghostscript texlive-extra-utils graphviz evince
```