Duality in Probabilistic Automata

Chris Hundt
Prakash Panangaden
Joelle Pineau
Doina Precup
Gavin Seal

McGill University
We have discovered an - apparently - new kind of duality for automata.
Overview

- We have discovered an - apparently - new kind of duality for automata.
- Special case of this construction known since 1962 to Brzozowski.
We have discovered an - apparently - new kind of duality for automata.

Special case of this construction known since 1962 to Brzozowski.

Works for probabilistic automata.
We have discovered an - apparently - new kind of duality for automata.

Special case of this construction known since 1962 to Brzozowski.

Works for probabilistic automata.

Seems interesting for learning and planning.
Overview

- We have discovered an - apparently - new kind of duality for automata.
- Special case of this construction known since 1962 to Brzozowski.
- Works for probabilistic automata.
- Seems interesting for learning and planning.
- Could be connected to duality in control theory, Pontryagin duality or general concrete dualities.
**Overview**

- We have discovered an - apparently - new kind of duality for automata.
- Special case of this construction known since 1962 to Brzozowski.
- Works for probabilistic automata.
- Seems interesting for learning and planning.
- Could be connected to duality in control theory, Pontryagin duality or general concrete dualities.
- We are not sure about the “right” categorical setting.
Deterministic Automata

- $A = (Q, \Sigma, P, \delta, \gamma)$: a deterministic finite automaton. $Q$ is the set of states, $\Sigma$ an input alphabet (actions), $P$ is a set of propositions.
Deterministic Automata

- \( A = (Q, \Sigma, P, \delta, \gamma) \): a deterministic finite automaton. \( Q \) is the set of states, \( \Sigma \) an input alphabet (actions), \( P \) is a set of propositions.

- \( \delta : Q \times \Sigma \rightarrow Q \) is the state transition function.
Deterministic Automata

- $A = (Q, \Sigma, P, \delta, \gamma)$: a deterministic finite automaton. $Q$ is the set of states, $\Sigma$ an input alphabet (actions), $P$ is a set of propositions.
- $\delta : Q \times \Sigma \rightarrow Q$ is the state transition function.
- $\gamma : Q \rightarrow 2^P$ or $\gamma : Q \times P \rightarrow 2$ is a labeling function.
Deterministic Automata

- \( A = (Q, \Sigma, P, \delta, \gamma) \): a deterministic finite automaton. \( Q \) is the set of states, \( \Sigma \) an input alphabet (actions), \( P \) is a set of propositions.
- \( \delta : Q \times \Sigma \rightarrow Q \) is the state transition function.
- \( \gamma : Q \rightarrow 2^P \) or \( \gamma : Q \times P \rightarrow 2 \) is a labeling function.
- If \( P = \{ \text{accept} \} \) we have ordinary deterministic finite automata.
Thinking of the elements of $P$ as formulas we can use them to define a simple modal logic. We define a formula $\varphi$ according to the following grammar:

$$\varphi ::= p \in P \mid (a)\varphi$$

where $a \in \Sigma$. 

A Simple Modal Logic
Thinking of the elements of $P$ as formulas we can use them to define a simple modal logic. We define a formula $\varphi$ according to the following grammar:

$$
\varphi ::= p \in P \mid (a)\varphi
$$

where $a \in \Sigma$.

We say $s \models p$, if $p \in \gamma(s)$ (or $\gamma(s, p) = T$).

We say $s \models (a)\varphi$ if $\delta(s, a) \models \varphi$. 
Thinking of the elements of $P$ as formulas we can use them to define a simple modal logic. We define a formula $\varphi$ according to the following grammar:

$$\varphi ::= p \in P \mid (a)\varphi$$

where $a \in \Sigma$.

We say $s \models p$, if $p \in \gamma(s)$ (or $\gamma(s, p) = T$).
We say $s \models (a)\varphi$ if $\delta(s, a) \models \varphi$.

Now we define $[\varphi]_A = \{ s \in Q \mid s \models \varphi \}$. 
An Equivalence Relation on Formulas

- We write $sa$ as shorthand for $\delta(s, a)$.
An Equivalence Relation on Formulas

- We write $sa$ as shorthand for $\delta(s, a)$.
- Define $\sim_A$ between formulas as $\varphi \sim_A \psi$ if $[\varphi]_A = [\psi]_A$. 

Note that this allows us to identify an equivalence class for $\varphi$ with the set of states $\left[\varphi\right]_A$ that satisfy $\varphi$. Note that another way of defining this equivalence relation is $\varphi \sim_0 \psi$ if

$$s \in \left[\varphi\right]_A \iff s \in \left[\psi\right]_A.$$
An Equivalence Relation on Formulas

- We write $sa$ as shorthand for $\delta(s, a)$.
- Define $\sim_A$ between formulas as $\varphi \sim_A \psi$ if $[\varphi]_A = [\psi]_A$.
- Note that this allows us to identify an equivalence class for $\varphi$ with the set of states $[\varphi]_A$ that satisfy $\varphi$. 
An Equivalence Relation on Formulas

- We write \( sa \) as shorthand for \( \delta(s, a) \).
- Define \( \sim_A \) between formulas as \( \varphi \sim_A \psi \) if
  \[
  \llbracket \varphi \rrbracket_A = \llbracket \psi \rrbracket_A.
  \]
- Note that this allows us to identify an equivalence class for \( \varphi \) with the set of states \( \llbracket \varphi \rrbracket_A \) that satisfy \( \varphi \).
- Note that another way of defining this equivalence relations is
  \[
  \varphi \sim_A \varphi' := \forall s \in Q. s \models \varphi \iff s \models \varphi'.
  \]
An Equivalence Relation on States

We also define an equivalence $\equiv$ between states in $A$ as $s_1 \equiv s_2$ if for all formulas $\varphi$ on $A$,

$$s_1 \models \varphi \iff s_2 \models \varphi.$$
We also define an equivalence \( \equiv \) between states in \( A \) as \( s_1 \equiv s_2 \) if for all formulas \( \varphi \) on \( A \),
\[
\models s_1 \varphi \iff \models s_2 \varphi.
\]
The equivalence relations \( \sim \) and \( \equiv \) are clearly closely related: they are the hinge of the duality between states and observations.
An Equivalence Relation on States

- We also define an equivalence \( \equiv \) between states in \( \mathcal{A} \) as \( s_1 \equiv s_2 \) if for all formulas \( \varphi \) on \( \mathcal{A} \),
  \[ s_1 \models \varphi \iff s_2 \models \varphi. \]

- The equivalence relations \( \sim \) and \( \equiv \) are clearly closely related: they are the hinge of the duality between states and observations.

- We say that \( \mathcal{A} \) is \textit{reduced} if the \( \equiv \)-equivalence classes are singletons.
An Equivalence Relation on States

We also define an equivalence $\equiv$ between states in $\mathcal{A}$ as $s_1 \equiv s_2$ if for all formulas $\varphi$ on $\mathcal{A}$,

$$s_1 \models \varphi \iff s_2 \models \varphi.$$ 

The equivalence relations $\sim$ and $\equiv$ are clearly closely related: they are the hinge of the duality between states and observations.

We say that $\mathcal{A}$ is reduced if the $\equiv$-equivalence classes are singletons.

Since there is more than just one proposition in general the relation $\equiv$ is finer than the usual equivalence of automata theory.
A Dual Automaton

Given a finite automaton $\mathcal{A} = (Q, \Sigma, P, \delta, \gamma)$. Let $T$ be the set of $\sim_{\mathcal{A}}$-equivalence classes of formulas on $\mathcal{A}$.
A Dual Automaton

- Given a finite automaton \( \mathcal{A} = (Q, \Sigma, P, \delta, \gamma) \).
  Let \( T \) be the set of \( \sim_{\mathcal{A}} \)-equivalence classes of formulas on \( \mathcal{A} \).
- We define \( \mathcal{A}' = (Q', \Sigma, P', \delta', \gamma') \) as follows:
A Dual Automaton

- Given a finite automaton \( A = (Q, \Sigma, P, \delta, \gamma) \). Let \( T \) be the set of \( \sim_A \)-equivalence classes of formulas on \( A \).
- We define \( A' = (Q', \Sigma, P', \delta', \gamma') \) as follows:
- \( Q' = T = \{ [\varphi]_A \} \)
A Dual Automaton

- Given a finite automaton $\mathcal{A} = (Q, \Sigma, P, \delta, \gamma)$. Let $T$ be the set of $\sim_\mathcal{A}$-equivalence classes of formulas on $\mathcal{A}$.
- We define $\mathcal{A}' = (Q', \Sigma, P', \delta', \gamma')$ as follows:
  - $Q' = T = \{ [\varphi]_\mathcal{A} \}$
  - $P' = Q$
A Dual Automaton

- Given a finite automaton $\mathcal{A} = (Q, \Sigma, P, \delta, \gamma)$. Let $T$ be the set of $\sim_{\mathcal{A}}$-equivalence classes of formulas on $\mathcal{A}$.
- We define $\mathcal{A}' = (Q', \Sigma, P', \delta', \gamma')$ as follows:
  - $Q' = T = \{[\varphi]_{\mathcal{A}}\}$
  - $P' = Q$
  - $\delta'([\varphi]_{\mathcal{A}}, a) = [(a)\varphi]_{\mathcal{A}}$
A Dual Automaton

- Given a finite automaton $\mathcal{A} = (Q, \Sigma, P, \delta, \gamma)$. Let $T$ be the set of $\sim_{\mathcal{A}}$-equivalence classes of formulas on $\mathcal{A}$.

- We define $\mathcal{A}' = (Q', \Sigma, P', \delta', \gamma')$ as follows:
  - $Q' = T = \{[\varphi]_A\}$
  - $P' = Q$
  - $\delta'( [\varphi]_A, a) = [(a)\varphi]_A$
  - $\gamma'( [\varphi]_A, p) = [\varphi]_A$
The intuition

We have interchanged the states and the observations or propositions; more precisely we have interchanged equivalence classes of formulas - based on the observations - with the states. We have made the states of the old machine the observations of the dual machine.
Now consider $A'' = (A')'$, the dual of the dual.
Now consider $A'' = (A')'$, the dual of the dual.

Its states are equivalence classes of $A'$-formulas.
Now consider $\mathcal{A}'' = (\mathcal{A}')'$, the dual of the dual.

Its states are equivalence classes of $\mathcal{A}'$-formulas.

Each such class is identified with a set $[\phi']_{\mathcal{A}'}$ of $\mathcal{A}'$-states by which formulas in that class are satisfied, and
The Double Dual

Now consider $A'' = (A')'$, the dual of the dual.

Its states are equivalence classes of $A'$-formulas.

Each such class is identified with a set $[\varphi']_{A'}$ of $A'$-states by which formulas in that class are satisfied, and

each $A'$-state is an equivalence class of $A$-formulas.
The Double Dual

- Now consider $\mathcal{A}'' = (\mathcal{A}')'$, the dual of the dual.
- Its states are equivalence classes of $\mathcal{A}'$-formulas.
- Each such class is identified with a set $[\varphi']_{\mathcal{A}'}$ of $\mathcal{A}'$-states by which formulas in that class are satisfied, and
- each $\mathcal{A}'$-state is an equivalence class of $\mathcal{A}$-formulas.
- Thus we can look at states in $\mathcal{A}''$ as collections of $S$-formula equivalence classes.
Let $\mathcal{A}''$ be the double dual, and for any state $s \in Q$ in the original automaton we define

$$Sat(s) = \{ [[\varphi]]_\mathcal{A} : s \models \varphi \}.$$
Let $A''$ be the double dual, and for any state $s \in Q$ in the original automaton we define

$$Sat(s) = \{ \llbracket \varphi \rrbracket_A : s \models \varphi \}.$$  

Lemma: For any $s \in Q$, $Sat(s)$ is a state in $A''$. 

The Double Dual 2

- Let $\mathcal{A}''$ be the double dual, and for any state $s \in Q$ in the original automaton we define

$$Sat(s) = \{ [[\varphi]]_{\mathcal{A}} : s \models \varphi \}.$$ 

- Lemma: For any $s \in Q$, $Sat(s)$ is a state in $\mathcal{A}''$.

- In fact, all the states of the double dual have this form.
Let $A''$ be the double dual, and for any state $s \in Q$ in the original automaton we define

$$Sat(s) = \{ [[\varphi]]_A : s \models \varphi \}.$$

**Lemma:** For any $s \in Q$, $Sat(s)$ is a state in $A''$.

In fact *all* the states of the double dual have this form.

**Lemma:** Let $s'' = [[\varphi]]_{A'} \in Q''$ be any state in $A''$. Then $s'' = Sat(s_\varphi)$ for some state $s_\varphi \in Q$. 


Let $A''$ be the double dual, and for any state $s \in Q$ in the original automaton we define

$$Sat(s) = \{ [[\varphi]]_A : s \models \varphi \}.$$ 

Lemma: For any $s \in Q$, $Sat(s)$ is a state in $A''$.

In fact all the states of the double dual have this form.

Lemma: Let $s'' = [[\varphi]]_{A'} \in Q''$ be any state in $A''$. Then $s'' = Sat(s_{\varphi})$ for some state $s_{\varphi} \in Q$.

The proof is by an easy induction on $\varphi$. 
If $A$ is reduced then $Sat$ is a bijection from $Q$ to $Q''$.
Minimality Properties

- If $\mathcal{A}$ is reduced then $Sat$ is a bijection from $Q$ to $Q''$.
- The statement above can be strengthened to show that we actually have an isomorphism of automata.
Minimality Properties

- If $A$ is reduced then $Sat$ is a bijection from $Q$ to $Q''$.
- The statement above can be strengthened to show that we actually have an isomorphism of automata.
- If we define a notion of bisimulation we can show that a machine and its double dual are bisimilar.
Minimality Properties

- If $A$ is reduced then $Sat$ is a bijection from $Q$ to $Q''$.
- The statement above can be strengthened to show that we actually have an isomorphism of automata.
- If we define a notion of bisimulation we can show that a machine and its double dual are bisimilar.
- The minimality is, of course, due to the use of the equivalence relations in the duality.
The Nondeterministic Case

Here we consider automata of the type

\[ A = (Q, \Sigma, P, \delta: Q \times \Sigma \rightarrow 2^Q, \gamma: Q \rightarrow 2^P) \].
The Nondeterministic Case

- Here we consider automata of the type

\[ A = (Q, \Sigma, P, \delta : Q \times \Sigma \rightarrow 2^Q, \gamma : Q \rightarrow 2^P). \]

- We use the same formulas but we have a different notion of satisfaction: \( S \subseteq Q \)

\[ S \models p \iff \exists s \in S : p \in \gamma(s) \]

\[ S \models (a)\varphi \iff \delta(S, a) \models \varphi. \]
The Nondeterministic Case

Here we consider automata of the type

\[ A = (Q, \Sigma, P, \delta : Q \times \Sigma \to 2^Q, \gamma : Q \to 2^P). \]

We use the same formulas but we have a different notion of satisfaction: \( S \subseteq Q \)

\[ S \models p \iff \exists s \in S : p \in \gamma(s) \]

\[ S \models (a)\varphi \iff \delta(S, a) \models \varphi. \]

We define an appropriate notion of simulation and prove: \( A \) is simulated by \( A'' \).
The Nondeterministic Case

- Here we consider automata of the type

\[ A = (Q, \Sigma, P, \delta : Q \times \Sigma \rightarrow 2^Q, \gamma : Q \rightarrow 2^P). \]

- We use the same formulas but we have a different notion of satisfaction: \( S \subseteq Q \)

\[ S \models p \iff \exists s \in S : p \in \gamma(s) \]

\[ S \models (a)\varphi \iff \delta(S, a) \models \varphi. \]

- We define an appropriate notion of simulation and prove: \( A \) is simulated by \( A'' \).

- The double dual is always deterministic.
Brzozowski’s Algorithm 1962

- Take a NFA and just reverse all the transitions and interchange initial and final states.

Determinize the result.

Reverse all the transitions again and interchange initial and final states.

Determinize the result.

This gives the minimal DFA recognizing the same language. The intermediate step can blow up the size of the automaton exponentially before minimizing it.
Brzozowski’s Algorithm 1962

- Take a NFA and just reverse all the transitions and interchange initial and final states.
- Determinize the result.
Brzozowski’s Algorithm 1962

- Take a NFA and just reverse all the transitions and interchange initial and final states.
- Determinize the result.
- Reverse all the transitions again and interchange initial and final states.
Brzozowski’s Algorithm 1962

- Take a NFA and just reverse all the transitions and interchange initial and final states.
- Determinize the result.
- Reverse all the transitions again and interchange initial and final states.
- Determinize the result.

This gives the minimal DFA recognizing the same language. The intermediate step can blow up the size of the automaton exponentially before minimizing it.
Brzozowski’s Algorithm 1962

- Take a NFA and just reverse all the transitions and interchange initial and final states.
- Determinize the result.
- Reverse all the transitions again and interchange initial and final states.
- Determinize the result.
- This gives the minimal DFA recognizing the same language. The intermediate step can blow up the size of the automaton exponentially before minimizing it.
Probabilistic systems

- Everything is discrete.
Probabilistic systems

- Everything is discrete.
- Markov Decision Processes aka Labelled Markov Processes:

\[ \mathcal{M} = (S, A, \forall a \in A \tau_a : S \times S \rightarrow [0, 1]). \]

The \( \tau_a \) are transition probability functions (matrices).
Probabilistic systems

- Everything is discrete.
- Markov Decision Processes aka Labelled Markov Processes:

  \[ M = (S, A, \forall a \in A \tau_a : S \times S \rightarrow [0, 1]). \]

  The \( \tau_a \) are transition probability functions (matrices).
- Usually MDPs have rewards but I will not consider them for now.
Probabilistic systems

- Everything is discrete.
- Markov Decision Processes aka Labelled Markov Processes:

\[
\mathcal{M} = (S, A, \forall a \in A \tau_a : S \times S \rightarrow [0, 1]).
\]

The \( \tau_a \) are transition probability functions (matrices).

- Usually MDPs have rewards but I will not consider them for now.
- We could make things continuous but that is orthogonal.
Partial Observations

- Partially Observable Markov Decision Processes (POMDPs). We cannot see the entire state but we can see something.
Partial Observations

- Partially Observable Markov Decision Processes (POMDPs). We cannot see the entire state but we can see something.
- In process algebra we typically take actions as not always being enabled and we observe whether actions are accepted or rejected.
Partial Observations

- Partially Observable Markov Decision Processes (POMDPs). We cannot see the entire state but we can see something.

- In process algebra we typically take actions as not always being enabled and we observe whether actions are accepted or rejected.

- In POMDPs we assume actions are always accepted but with each transition some propositions are true, or some boolean observables are “on.”
Partial Observations

- Partially Observable Markov Decision Processes (POMDPs). We cannot see the entire state but we can see something.

- In process algebra we typically take actions as not always being enabled and we observe whether actions are accepted or rejected.

- In POMDPs we assume actions are always accepted but with each transition some propositions are true, or some boolean observables are “on.”

- Note that the observations can depend probabilistically on the action taken and the final state. Many variations are possible.
Formal Definition of a POMDP

\[ \mathcal{M} = (S, \Sigma, O, \delta : S \times \Sigma \times S \rightarrow [0, 1], \gamma : S \times \Sigma \times O \rightarrow [0, 1]), \]
Formal Definition of a POMDP

\[ \mathcal{M} = (S, \Sigma, \mathcal{O}, \delta : S \times \Sigma \times S \rightarrow [0, 1], \gamma : S \times \Sigma \times \mathcal{O} \rightarrow [0, 1]), \]

where \( S \) is a set of states, \( \mathcal{O} \) is a set of observations, \( \Sigma \) is a set of actions, \( \delta \) is the transition probability function and \( \gamma \) gives the observation probabilities.
Formal Definition of a POMDP

\[ M = (S, \Sigma, \mathcal{O}, \delta : S \times \Sigma \times S \rightarrow [0, 1], \gamma : S \times \Sigma \times \mathcal{O} \rightarrow [0, 1]), \]

where \( S \) is a set of states, \( \mathcal{O} \) is a set of observations, \( \Sigma \) is a set of actions, \( \delta \) is the transition probability function and \( \gamma \) gives the observation probabilities.
An **automaton with stochastic observations** is a quintuple

\[ E = (S, \Sigma, O, \delta : S \times \Sigma \rightarrow S, \gamma : S \times O \rightarrow [0, 1]) . \]

Note that this has deterministic transitions and stochastic observations.
Automata with State-based Observations

- An automaton with stochastic observations is a quintuple

\[ E = (S, \Sigma, O, \delta : S \times \Sigma \rightarrow S, \gamma : S \times O \rightarrow [0, 1]). \]

Note that this has deterministic transitions and stochastic observations.

- A probabilistic automaton with stochastic observations is

\[ F = (S, \Sigma, O, \delta : S \times \Sigma \times S \rightarrow [0, 1], \gamma : S \times O \rightarrow [0, 1]). \]
Simple Tests

- Rather than thinking of propositions and formulas we will think of observations and tests. I will look at state-based notions of observations.

Recall probabilistic automata $E = (S; O; \tau)$; where $S \subseteq S \times \{0, 1\}$ is the transition function and $O : S \to \{0, 1\}$ is the observation function.
Simple Tests

- Rather than thinking of propositions and formulas we will think of observations and tests. I will look at state-based notions of observations.

- Recall probabilistic automata

\[ E = (S, \Sigma, O, \delta, \gamma), \]
Rather than thinking of propositions and formulas we will think of observations and tests. I will look at state-based notions of observations.

Recall probabilistic automata

$$\mathcal{E} = (S, \Sigma, O, \delta, \gamma),$$

where $\delta : S \times \Sigma \times S \to [0, 1]$ is the transition function.
Simple Tests

- Rather than thinking of propositions and formulas we will think of observations and tests. I will look at state-based notions of observations.

- Recall probabilistic automata

\[ \mathcal{E} = (S, \Sigma, O, \delta, \gamma), \]

- where \( \delta : S \times \Sigma \times S \rightarrow [0, 1] \) is the transition function

- and \( \gamma : S \times O \rightarrow [0, 1] \) is the observation function.
Simple Tests 2

- We use the same logic as before except that we give a probabilistic semantics and call the formulas “tests.” I will \( a.t \) or \( at \) rather than \( (a)\varphi \).
Simple Tests 2

- We use the same logic as before except that we give a probabilistic semantics and call the formulas “tests.” I will \( a.t \) or \( at \) rather than \( (a)\varphi \).

- Tests define functions from states to \([0, 1]\). If they define the same function they are equivalent.
Simple Tests 2

- We use the same logic as before except that we give a probabilistic semantics and call the formulas “tests.” I will $a.t$ or $at$ rather than $(a)\varphi$.

- Tests define functions from states to $[0, 1]$. If they define the same function they are equivalent.

- The explicit definition of these functions are:

$$[o]e(s) = \gamma(s, o)$$

$$[at]e(s) = \sum_{s'} \delta(s, a, s')[t]e(s').$$
We use the same logic as before except that we give a probabilistic semantics and call the formulas “tests.” I will $a.t$ or $at$ rather than $(a)\varphi$.

Tests define functions from states to $[0, 1]$. If they define the same function they are equivalent.

The explicit definition of these functions are:

$$[o] \varepsilon(s) = \gamma(s, o)$$

$$[at] \varepsilon(s) = \sum_{s'} \delta(s, a, s')[t] \varepsilon(s').$$

In AI these are called “e-tests.”
Duality with e-tests

- \( S' = \{ [t] \varepsilon \} \)
Duality with e-tests

- \( S' = \{ [t] \varepsilon \} \)
- \( \mathcal{O}' = S \)
Duality with e-tests

- $S' = \{ [t] \epsilon \}$
- $\mathcal{O}' = S$
- $\gamma'( [t] \epsilon, s) = [t] \epsilon(s)$
Duality with e-tests

- $S' = \{[t]\varepsilon\}$
- $\mathcal{O}' = S$
- $\gamma'([t]\varepsilon, s) = [t]\varepsilon(s)$
- $\delta'([t]\varepsilon, a, [at]\varepsilon) = 1; 0$ otherwise.
Duality with e-tests

- \( S' = \{ [t] \varepsilon \} \)
- \( \mathcal{O}' = S \)
- \( \gamma'( [t] \varepsilon, s) = [t] \varepsilon(s) \)
- \( \delta'( [t] \varepsilon, a, [at] \varepsilon) = 1; 0 \) otherwise.
- This machine has deterministic transitions and \( \gamma' \) is just the transpose of \( \gamma \).
Duality with e-tests

- \( S' = \{ [t]_e \} \)
- \( \mathcal{O}' = S \)
- \( \gamma'([t]_e, s) = [t]_e(s) \)
- \( \delta'([t]_e, a, [at]_e) = 1; 0 \text{ otherwise.} \)
- This machine has deterministic transitions and \( \gamma' \) is just the transpose of \( \gamma \).
The Double Dual

If $\mathcal{E}$ is the primal and $\mathcal{E}'$ is the dual then the states of the double dual, $\mathcal{E}''$ are $\mathcal{E}'$-equivalence classes of tests.

An "atomic" test is just an observation of $\mathcal{E}_0$, which is just a state of $\mathcal{E}$ so it has the form $[s]_{\mathcal{E}_0}$ for some $s$.

We see that $\mathcal{E}_0([s]_{\mathcal{E}_0};[o]_{\mathcal{E}}) = [s]_{\mathcal{E}_0}(o) = \mathcal{E}_0([o]_{\mathcal{E}};s) = [o]_{\mathcal{E}}(s)$:

An easy calculation shows:

$$[a_1 a_2 \ldots a_k o]_{\mathcal{E}_0}(s)_{\mathcal{E}_0} = [a_1 a_2 \ldots a_k o]_{\mathcal{E}}(s)$$
The Double Dual

- If $\mathcal{E}$ is the primal and $\mathcal{E}'$ is the dual then the states of the double dual, $\mathcal{E}''$ are $\mathcal{E}'$-equivalence classes of tests.

- An “atomic” test is just an observation of $\mathcal{E}'$, which is just a state of $\mathcal{E}$ so it has the form $[s]_{\mathcal{E}'}$ for some $s$. 
The Double Dual

- If $\mathcal{E}$ is the primal and $\mathcal{E}'$ is the dual then the states of the double dual, $\mathcal{E}''$ are $\mathcal{E}'$-equivalence classes of tests.

- An “atomic” test is just an observation of $\mathcal{E}'$, which is just a state of $\mathcal{E}$ so it has the form $[s]_{\mathcal{E}'}$ for some $s$.

- We see that

$$
\gamma''([s]_{\mathcal{E}'}, [o]_{\mathcal{E}}) = [s]_{\mathcal{E}'}([o]_{\mathcal{E}}) = \gamma'([o]_{\mathcal{E}}, s) = [o]_{\mathcal{E}}(s) = \gamma(s, o).
$$

An easy calculation shows:

$$
[[a_1 a_2 \ldots a_k o]]_{\mathcal{E}''}([s]_{\mathcal{E}'}) = [[[a_1 a_2 \ldots a_k o]]_{\mathcal{E}}]_{\mathcal{E}}(s) = \gamma(s, o).
$$
The Double Dual

- If $\mathcal{E}$ is the primal and $\mathcal{E}'$ is the dual then the states of the double dual, $\mathcal{E}''$ are $\mathcal{E}'$-equivalence classes of tests.

- An “atomic” test is just an observation of $\mathcal{E}'$, which is just a state of $\mathcal{E}$ so it has the form $[s]_{\mathcal{E}'}$ for some $s$.

- We see that

$$\gamma''([s]_{\mathcal{E}'}, [o]_{\mathcal{E}}) = [s]_{\mathcal{E}'}([o]_{\mathcal{E}}) = \gamma'([o]_{\mathcal{E}}, s) = [o]_{\mathcal{E}}(s) = \gamma(s, o).$$

- An easy calculation shows:

$$[a_1 a_2 \cdots a_k o]_{\mathcal{E}''}([s]_{\mathcal{E}'})$$

$$= [a_1 a_2 \cdots a_k o]_{\mathcal{E}}(s).$$
Inadequacy of e-tests

- There is a loss of information in the previous construction.
Inadequacy of e-tests

- There is a loss of information in the previous construction.
- The double dual behaves just like the primal with respect to “e-tests” but not with respect to more refined kinds of observations.
Inadequacy of e-tests

- There is a loss of information in the previous construction.
- The double dual behaves just like the primal with respect to “e-tests” but not with respect to more refined kinds of observations.

\[
\begin{align*}
[0_1 a_1 o_2 a_2 o_3] \psi''(\{s\} \psi') &= \\
[0_1] \psi''(\{s\} \psi'') \cdot [a_1 o_2] \psi''(\{s\} \psi') \cdot [a_1 a_2 o_3] \psi''(\{s\} \psi').
\end{align*}
\]

This does not hold in the primal.
Inadequacy of e-tests

- There is a loss of information in the previous construction.
- The double dual behaves just like the primal with respect to “e-tests” but not with respect to more refined kinds of observations.

\[
\begin{align*}
\left[ o_1 a_1 o_2 a_2 o_3 \right] & \mathcal{E}''\left( \left[ s \right] \mathcal{E}' \right) = \\
\left[ o_1 \right] & \mathcal{E}''\left( \left[ s \right] \mathcal{E}'' \right) \cdot \left[ a_1 o_2 \right] \mathcal{E}''\left( \left[ s \right] \mathcal{E}' \right) \cdot \left[ a_1 a_2 o_3 \right] \mathcal{E}''\left( \left[ s \right] \mathcal{E}' \right).
\end{align*}
\]

This does not hold in the primal.
- The double dual does not conditionalize with respect to intermediate observations.
Recall the definition of a POMDP

\[ \mathcal{M} = (S, \Sigma, \mathcal{O}, \delta_a : S \times S \rightarrow [0, 1], \gamma_a : S \times \mathcal{O} \rightarrow [0, 1]) \].
More General Tests

- Recall the definition of a POMDP

\[ M = (S, \Sigma, O, \delta_a : S \times S \rightarrow [0, 1], \gamma_a : S \times O \rightarrow [0, 1]). \]

- A **test** \( t \) is a non-empty sequence of actions followed by an observation, i.e. \( t = a_1 \cdots a_n o \), with \( n \geq 1 \).
More General Tests

- Recall the definition of a POMDP

\[ \mathcal{M} = (S, \Sigma, \mathcal{O}, \delta_a : S \times S \to [0, 1], \gamma_a : S \times \mathcal{O} \to [0, 1]). \]

- A **test** \( t \) is a non-empty sequence of actions followed by an observation, i.e. \( t = a_1 \cdots a_n o \), with \( n \geq 1 \).

- An **experiment** is a non-empty sequence of tests \( e = t_1 \cdots t_m \) with \( m \geq 1 \).
More General Tests

- Recall the definition of a POMDP

  \[ \mathcal{M} = (S, \Sigma, \mathcal{O}, \delta_a : S \times S \rightarrow [0, 1], \gamma_a : S \times \mathcal{O} \rightarrow [0, 1]). \]

- A test \( t \) is a non-empty sequence of actions followed by an observation, i.e. \( t = a_1 \cdots a_n o \), with \( n \geq 1 \).

- An experiment is a non-empty sequence of tests \( e = t_1 \cdots t_m \) with \( m \geq 1 \).
Some Notation

- We need to generalize the transition function to keep track of the final state.

\[
\delta_\epsilon(s, s') = 1_{s=s'} \
\forall s, s' \in S \\
\delta_{a\alpha}(s, s') = \sum_{s''} \delta_a(s, s'') \delta_{\alpha}(s'', s') \
\forall s, s' \in S.
\]
Some Notation

- We need to generalize the transition function to keep track of the final state.

\[ \delta_{\epsilon}(s, s') = 1_{s=s'} \quad \forall s, s' \in S \]

\[ \delta_{a\alpha}(s, s') = \sum_{s''} \delta_a(s, s'')\delta_{\alpha}(s'', s') \quad \forall s, s' \in S. \]

- We have written \(1_{s=s'}\) for the indicator function.
Some Notation

- We need to generalize the transition function to keep track of the final state.

\[
\delta_{e}(s, s') = 1_{s=s'} \quad \forall s, s' \in S
\]

\[
\delta_{a\alpha}(s, s') = \sum_{s''} \delta_{a}(s, s'')\delta_{\alpha}(s'', s') \quad \forall s, s' \in S.
\]

- We have written \(1_{s=s'}\) for the indicator function.

- We define the symbol \(\langle s | t | s' \rangle\) which gives the probability that the system starts in \(s\), is subjected to the test \(t\) and ends up in the state \(s'\); similarly \(\langle s | e | s' \rangle\).
We have

$$\langle s|a_1 \cdots a_n o|s'\rangle = \delta_\alpha(s, s') \gamma(a_n(s', o)).$$
Notation continued

- We have

\[ \langle s | a_1 \cdots a_n o | s' \rangle = \delta_\alpha(s, s') \gamma_{a_n}(s', o). \]

- We define

\[ \langle s | e \rangle = \sum_{s'} \langle s | e | s' \rangle. \]
Equivalence on Experiments

For experiments $e_1, e_2$, we say

$$e_1 \sim_M e_2 \iff \langle s | e_1 \rangle = \langle s | e_2 \rangle \forall s \in S.$$
Equivalence on Experiments

- For experiments $e_1, e_2$, we say

$$e_1 \sim_M e_2 \iff \langle s|e_1 \rangle = \langle s|e_2 \rangle \forall s \in S.$$ 

- Then $[e]_M$ is the $\sim_M$-equivalence class of $e$. 
For experiments $e_1, e_2$, we say

$$e_1 \sim_M e_2 \iff \langle s | e_1 \rangle = \langle s | e_2 \rangle \forall s \in S.$$ 

Then $[e]_M$ is the $\sim_M$-equivalence class of $e$.

The construction of the dual proceeds as before by making equivalence classes of experiments the states of the dual machine and
Equivalence on Experiments

- For experiments $e_1, e_2$, we say

$$e_1 \sim_M e_2 \iff \langle s|e_1 \rangle = \langle s|e_2 \rangle \forall s \in S.$$ 

- Then $[e]_M$ is the $\sim_M$-equivalence class of $e$.

- The construction of the dual proceeds as before by making equivalence classes of experiments the states of the dual machine and

- the states of the primal machine become the observations of the dual machine.
The Dual Machine

- We define the dual as \( \mathcal{M}' = (S', \Sigma, \mathcal{O}', \delta' : S' \times \Sigma \rightarrow S', \gamma' : S' \times \mathcal{O}' \rightarrow [0, 1]) \),
The Dual Machine

- We define the dual as $\mathcal{M}' =\n\left(S', \Sigma, \mathcal{O}', \delta' : S' \times \Sigma \rightarrow S', \gamma' : S' \times \mathcal{O}' \rightarrow [0, 1]\right)$,

- where $S' = \{[e]_\mathcal{M}\}$, $\mathcal{O}' = S$
The Dual Machine

- We define the dual as $\mathcal{M}' = (S', \Sigma, \mathcal{O}', \delta' : S' \times \Sigma \rightarrow S', \gamma' : S' \times \mathcal{O}' \rightarrow [0, 1])$,
- where $S' = \{[e]_\mathcal{M}\}$, $\mathcal{O}' = S$
- $\delta'(\langle e, a_0 \rangle) = [a_0e]_\mathcal{M}$ and
The Dual Machine

- We define the dual as $M' = (S', \Sigma, O', \delta' : S' \times \Sigma \rightarrow S', \gamma' : S' \times O' \rightarrow [0, 1])$,
- where $S' = \{[e]_M\}$, $O' = S$
- $\delta'([e]_M, a_0) = [a_0e]_M$ and
- $\gamma'([e]_M, s) = \langle s | e \rangle$. 
The Double Dual

- We use the e-test construction to go from the dual to the double dual.
The Double Dual

- We use the e-test construction to go from the dual to the double dual.
- The double dual is

\[ M'' = (S'', A', O'', \delta'', \gamma'') \]

where
The Double Dual

- We use the e-test construction to go from the dual to the double dual.
- The double dual is

\[ \mathcal{M}'' = (S'', A', O'', \delta'', \gamma''), \]

where

- \( S'' = \{ [t] \mathcal{M}' \} \), \( O'' = S' \),
The Double Dual

- We use the e-test construction to go from the dual to the double dual.
- The double dual is

\[ \mathcal{M}'' = (S'', A', O'', \delta'', \gamma'') , \]

where

- \( S'' = \{ [t] \mathcal{M}' \} \), \( O'' = S' \),
- \( \delta''([t] \mathcal{M}', a_0) = [a_0 e] \mathcal{M} \) and
The Double Dual

- We use the e-test construction to go from the dual to the double dual.
- The double dual is

$$\mathcal{M}'' = (S'', A', O'', \delta'', \gamma''),$$

where

- $S'' = \{[t]\mathcal{M}'\}$, $O'' = S'$,
- $\delta''([t]\mathcal{M}', a_0) = [a_0e]\mathcal{M}$ and
- $\gamma''([t]\mathcal{M}', [t]\mathcal{M}) = \langle [t]\mathcal{M}|e \rangle = \langle s|\alpha^R t\rangle$ (e = $\alpha s$).
The Main Theorem

- One has to check that everything is well defined.
The Main Theorem

- One has to check that everything is well defined.
- The main result is: The probability of a state $s$ in the primal satisfying a experiment $e$, i.e. $\langle s \mid e \rangle$ is given by $\langle [s] \mathcal{M} \mid [e] \mathcal{M} \rangle = \gamma''([s] \mathcal{M}')\mid [e] \mathcal{M} \rangle$, where $[s]$ indicates the equivalence class of the e-test on the dual which has $s$ as an observation and an empty sequence of actions.
AI Motivation

- One can plan when one has the model: value iteration etc., but quite often one does not have the model.
AI Motivation

- One can plan when one has the model: value iteration etc., but quite often one does not have the model.
- In the absence of a model, one is forced to learn from data.
AI Motivation

- One can plan when one has the model: value iteration etc., but quite often one does not have the model.
- In the absence of a model, one is forced to learn from data.
- Learning is hopeless when one has no idea what the state space is.
AI Motivation

- One can plan when one has the model: value iteration etc., but quite often one does not have the model.
- In the absence of a model, one is forced to learn from data.
- Learning is hopeless when one has no idea what the state space is.
- There should be no such thing as absolute state! State is just a summary of past observations that can be used to make predictions.
AI Motivation

- One can plan when one has the model: value iteration etc., but quite often one does not have the model.
- In the absence of a model, one is forced to learn from data.
- Learning is hopeless when one has no idea what the state space is.
- There should be no such thing as absolute state! State is just a summary of past observations that can be used to make predictions.
- The double dual shows that the state can be regarded as just the summary of the outcomes of experiments.
We have a paper in the upcoming AAAI conference showing how to use the double-dual to represent systems with hidden state.
$A$ a set and $T : \text{Set} \to \text{Set}$ is the functor $TS = S \times A$. 
A set and $T : \text{Set} \to \text{Set}$ is the functor $TS = S \times A$.

A machine $\mathcal{M}$ is a pair $(\delta, \gamma)$ where $\delta : S \times A \to S$ is a $T$-algebra and $\gamma : S \times P \to 2$ is a relation in $\text{Set}$.
Machines Categorically

- A a set and \( T : \text{Set} \to \text{Set} \) is the functor \( TS = S \times A \).
- A \textit{machine} \( M \) is a pair \( (\delta, \gamma) \) where \( \delta : S \times A \to S \) is a \( T \)-algebra and \( \gamma : S \times P \to 2 \) is a relation in \( \text{Set} \).
- \( S \) is the set of states, \( A \) the actions and \( P \) the propositions.
A morphism $m$ from
$\mathcal{M}_1 = (\delta_1 : S_1 \times A \rightarrow S_1, \gamma_1 : S_1 \times P_1 \rightarrow 2)$
to $\mathcal{M}_2 = (\delta_{21} : S_2 \times A \rightarrow S_2, \gamma_2 : S_2 \times P_2 \rightarrow 2)$
is a pair $m = (f : S_1 \rightarrow S_2, g : P_2 \rightarrow P_1)$ making the following diagrams commute.
Morphisms of Machines

- A morphism $m$ from
  $\mathcal{M}_1 = (\delta_1 : S_1 \times A \to S_1, \gamma_1 : S_1 \times P_1 \to 2)$
to $\mathcal{M}_2 = (\delta_2 : S_2 \times A \to S_2, \gamma_2 : S_2 \times P_2 \to 2)$
is a pair $m = (f : S_1 \to S_2, g : P_2 \to P_1)$ making the following diagrams commute

- $S_1 \times A \xrightarrow{f \times \text{id}_A} S_2 \times A$ and $S_1 \times P_2 \xrightarrow{f \times \text{id}_{P_2}} S_2 \times P_2$

  \[
  \begin{array}{c}
  \delta_1 \\
  \downarrow \delta_1 \\
  S_1 \xrightarrow{f} S_2
  \end{array}
  \quad
  \begin{array}{c}
  \delta_2 \\
  \downarrow \delta_2 \\
  S_1 \xrightarrow{f} S_2
  \end{array}
  \quad
  \begin{array}{c}
  \text{id}_{S_1} \times g \\
  \downarrow \text{id}_{S_1} \times g \\
  S_1 \xrightarrow{\gamma_1} S_2
  \end{array}
  \quad
  \begin{array}{c}
  \gamma_2 \\
  \downarrow \gamma_2 \\
  S_1 \xrightarrow{\gamma_1} S_2
  \end{array}
  \]
The category of machines is written $\mathbf{Mch}$.
The category of machines is written $\text{Mch}$.

Given a machine $\mathcal{M}$ we define the formulas of $\mathcal{M}$, $\mathcal{F}_\mathcal{M}$, to be the set $A^* \times P$. If $\phi = (w, p)$ we will write $a\phi$ for $(aw, p)$. 
The Dual Machine

- The category of machines is written $\text{Mch}$.
- Given a machine $\mathcal{M}$ we define the formulas of $\mathcal{M}$, $\mathcal{F}_\mathcal{M}$, to be the set $A^* \times P$. If $\phi = (w, p)$ we will write $a\phi$ for $(aw, p)$.
- We define satisfaction by

$$ s \models (w, p) \iff \delta^*(s, w) \gamma p. $$

The contravariant functor sends $\mathcal{M}$ to $\mathcal{M}^0$, the dual defined before, and the morphism $(f;g) : \mathcal{M}_1 \to \mathcal{M}_2$ to $(g^0;f)$ where $g^0((w;p))_{\mathcal{M}_2} = [(w;g(p))]_{\mathcal{M}_1}$.
The category of machines is written $\mathbf{Mch}$.

Given a machine $\mathcal{M}$ we define the formulas of $\mathcal{M}$, $\mathcal{F}_\mathcal{M}$, to be the set $A^* \times P$. If $\phi = (w, p)$ we will write $a\phi$ for $(aw, p)$.

We define satisfaction by

$$s \models (w, p) \iff \delta^* (s, w) \gamma p.$$

The contravariant functor $'$ sends $\mathcal{M}$ to $\mathcal{M}'$, the dual defined before, and the morphism $(f, g) : \mathcal{M}_1 \to \mathcal{M}_2$ to $(g', f)$ where

$$g'(\llbracket (w, p) \rrbracket_{\mathcal{M}_2}) = \llbracket (w, g(p)) \rrbracket_{\mathcal{M}_1}.$$
The Reduction Functor

- A machine is *state* reduced if

\[ s_1 \neq s_2 \Rightarrow \exists \phi \text{ such that } s_1 \not\models \phi \text{ and } s_2 \models \phi \text{ or vice versa.} \]
The Reduction Functor

- A machine is *state* reduced if
  \[ s_1 \neq s_2 \Rightarrow \exists \phi \text{ such that } s_1 \not\models \phi \text{ and } s_2 \models \phi \text{ or vice versa.} \]

- A machine is *proposition* reduced if
  \[ \forall p_1, p_2 (\forall w_1, w_2 \in A^* [(w_1, p_1)]_M = [(w_2, p_2)]_M) \Rightarrow p_1 = p_2. \]
The Reduction Functor

- A machine is *state* reduced if

\[ s_1 \neq s_2 \Rightarrow \exists \phi \text{ such that } s_1 \not\models \phi \text{ and } s_2 \models \phi \text{ or vice versa.} \]

- A machine is *proposition* reduced if

\[
\forall p_1, p_2 (\forall w_1, w_2 \in A^* [(w_1, p_1)]_M = [(w_2, p_2)]_M) \Rightarrow p_1 = p_2.
\]

- We define the *reduction* functor to be ’ composed with itself i.e. ”.
The Reduction Functor

- A machine is state reduced if
  \[ s_1 \neq s_2 \Rightarrow \exists \phi \text{ such that } s_1 \not\models \phi \text{ and } s_2 \models \phi \text{ or vice versa.} \]

- A machine is proposition reduced if
  \[ \forall p_1, p_2 (\forall w_1, w_2 \in A^*[(w_1, p_1)]_M = [(w_2, p_2)]_M) \Rightarrow p_1 = p_2. \]

- We define the reduction functor to be \( \cdot \) composed with itself i.e. "\( \cdot \)".

- If \( M = M'' \) we say that it is completely reduced.
It would be very pleasant if we took $Q : \text{Mch} \to \text{Mch}^{\text{op}}$ and $R : \text{Mch}^{\text{op}} \to \text{Mch}$ to be the two (covariant) functors that represent $\bot$ and get $Q \dashv R$. But this is not possible the way we have set things up! The unit of the adjunction would have to be a morphism $\eta : \text{M} \to \text{M}$ which would then require a map $g : \left[ F \text{M} \quad \text{P} \right] \to \text{2}$. Unless $\text{M}$ is proposition reduced there is no reason at all for such a thing to exist.
The Disappointment

- It would be very pleasant if we took $Q : \text{Mch} \to \text{Mch}^{\text{op}}$ and $R : \text{Mch}^{\text{op}} \to \text{Mch}$ to be the two (covariant) functors that represent $'$ and get $Q \vdash R$.

- But this is not possible the way we have set things up!
The Disappointment

- It would be very pleasant if we took $Q : \text{Mch} \to \text{Mch}^{\text{op}}$ and $R : \text{Mch}^{\text{op}} \to \text{Mch}$ to be the two (covariant) functors that represent $'$ and get $Q \dashv R$.

- But this is not possible the way we have set things up!

- The unit of the adjunction would have to be a morphism $\eta_M : M \to M''$ which would then require

...
The Disappointment

- It would be very pleasant if we took $Q : \text{Mch} \to \text{Mch}^{op}$ and $R : \text{Mch}^{op} \to \text{Mch}$ to be the two (covariant) functors that represent $'$ and get $Q \dashv R$.
- But this is not possible the way we have set things up!
- The unit of the adjunction would have to be a morphism $\eta_\mathcal{M} : \mathcal{M} \to \mathcal{M}''$ which would then require
- a map $g : [\mathcal{F}_\mathcal{M}] \times P \to 2$. 
The Disappointment

- It would be very pleasant if we took \( Q : \text{Mch} \rightarrow \text{Mch}^{\text{op}} \) and \( R : \text{Mch}^{\text{op}} \rightarrow \text{Mch} \) to be the two (covariant) functors that represent \( ' \) and get \( Q \dashv R \).

- But this is not possible the way we have set things up!

- The unit of the adjunction would have to be a morphism \( \eta_{\mathcal{M}} : \mathcal{M} \rightarrow \mathcal{M}'' \) which would then require

- a map \( g : [\mathcal{F}_\mathcal{M}] \times P \rightarrow 2 \).

- Unless \( \mathcal{M} \) is proposition reduced there is no reason at all for such a thing to exist.
But what did we prove before?

- We did not quite use the construction of the last two slides.
But what did we prove before?

- We did not quite use the construction of the last two slides.
- \( \tilde{\mathcal{M}} = (\delta''', \tilde{\gamma} : [\mathcal{F}]\mathcal{M}' \times P \rightarrow 2) \).
But what did we prove before?

- We did not quite use the construction of the last two slides.
- $\tilde{\mathcal{M}} = (\tilde{\delta}', \tilde{\gamma} : [\mathcal{F}]\mathcal{M}' \times P \rightarrow 2)$.
- We proved that this machine was state reduced.
But what did we prove before?

- We did not quite use the construction of the last two slides.
  \[ \tilde{M} = (\delta'', \tilde{\gamma} : [F]_{M'} \times P \rightarrow 2) \].

- We proved that this machine was state reduced.

- We quietly ignored the extra propositions in the double dual.
There is another way of decomposing "into a pair of (covariant) functors $F$ and $G$. $F$ modifies only the propositions and $G$ modifies only the states.
There is another way of decomposing " into a pair of (covariant) functors $F$ and $G$. $F$ modifies only the propositions and $G$ modifies only the states.

$F M = (\delta : S \times A \to S, \tilde{\gamma} : S \times [F]_M \to 2)$ where

$s \tilde{\gamma}[\phi]_M \iff [\phi]_M \gamma' s \iff s \in [\phi]_M$. 
There is another way of decomposing " into a pair of (covariant) functors $F$ and $G$. $F$ modifies only the propositions and $G$ modifies only the states.

$F\mathcal{M} = (\delta : S \times A \to S, \tilde{\gamma} : S \times [\mathcal{F}]_{\mathcal{M}} \to 2)$ where $s\tilde{\gamma}[\phi]_{\mathcal{M}} \iff [\phi]_{\mathcal{M}}\gamma's \iff s \in [\phi]_{\mathcal{M}}$.

$G\mathcal{M} = (\overline{\delta} : [S]_{\mathcal{M}} \times A \to [S]_{\mathcal{M}}, \overline{\gamma} : [S]_{\mathcal{M}} \times P \to 2)$; where
There is another way of decomposing "into a pair of (covariant) functors $F$ and $G$. $F$ modifies only the propositions and $G$ modifies only the states.

$F \mathcal{M} = (\delta : S \times A \rightarrow S, \tilde{\gamma} : S \times [\mathcal{F}]_{\mathcal{M}} \rightarrow 2)$ where $s \tilde{\gamma}[\phi]_{\mathcal{M}} \iff [\phi]_{\mathcal{M}} \gamma' s \iff s \in [\phi]_{\mathcal{M}}$.

$G \mathcal{M} = (\overline{\delta} : [S]_{\mathcal{M}} \times A \rightarrow [S]_{\mathcal{M}}, \overline{\gamma} : [S]_{\mathcal{M}} \times P \rightarrow 2)$; where

$[s]_{\mathcal{M}} := \{ s' \in S | \forall \phi \in \mathcal{F}, s' \models \phi \iff s \models \phi \}$ and

$\overline{\delta}([s]_{\mathcal{M}}, a) := [\delta(s, a)]_{\mathcal{M}}$ and $[s]_{\mathcal{M}} \overline{\gamma} p \iff s \gamma p$. 
The following natural isos hold:

\[ F^2 = F, \quad G^2 \cong G, \quad QF \cong Q, \quad \text{and} \quad GF = FG \cong RQ. \]
The following natural isos hold:

\[ F^2 = F, \quad G^2 \cong G, \quad QF \cong Q, \quad \text{and} \quad GF = FG \cong RQ. \]

For any machine \( \mathcal{M} \) the following diagram is a pullback.
The following natural isos hold:

\[ F^2 = F, \quad G^2 \cong G, \quad QF \cong Q, \quad \text{and} \quad GF = FG \cong RQ. \]

For any machine \( \mathcal{M} \) the following diagram is a pullback

and a pushout at the same time

\[
\begin{array}{ccc}
\mathcal{M} & \xrightarrow{(\pi_S, id_P)} & \mathcal{M} \\
\downarrow{id_S \times \pi_P} & & \downarrow{(id_S[\mathcal{F}], \pi_P)} \\
FM & \xrightarrow{(\pi_S, id_{[\mathcal{F}]})} & FGM \\
&& \downarrow{(id_{[\mathcal{F}]}, \pi_P)} \\
&& GM
\end{array}
\]
The following natural isos hold:

\[ F^2 = F, \quad G^2 \cong G, \quad QF \cong Q, \quad \text{and} \quad GF = FG \cong RQ. \]

For any machine \( \mathcal{M} \) the following diagram is a pullback and a pushout at the same time:

\[
\begin{array}{ccc}
F\mathcal{M} & \xrightarrow{(\pi_S, id_{\mathcal{F}})} & FGM \\
\downarrow id_S \times \pi_P & & \downarrow (id_S, \pi_P) \\
\mathcal{M} & \xrightarrow{(\pi_S, id_P)} & GM
\end{array}
\]
Conclusions

- We need to understand the general framework in which this fits. How is it related to dualities in control theory? [Alexander Kurz, Jan Rutten]
Conclusions

- We need to understand the general framework in which this fits. How is it related to dualities in control theory? [Alexander Kurz, Jan Rutten]
- We are experimenting with these ideas for practical problems in the RL Lab at McGill; joint with Doina Precup and Joelle Pineau.
Conclusions

- We need to understand the general framework in which this fits. How is it related to dualities in control theory? [Alexander Kurz, Jan Rutten]

- We are experimenting with these ideas for practical problems in the RL Lab at McGill; joint with Doina Precup and Joelle Pineau.

- Extension to continuous observation and continuous state spaces.
Conclusions

- We need to understand the general framework in which this fits. How is it related to dualities in control theory? [Alexander Kurz, Jan Rutten]

- We are experimenting with these ideas for practical problems in the RL Lab at McGill; joint with Doina Precup and Joelle Pineau.

- Extension to continuous observation and continuous state spaces.

- It is possible to eliminate state completely in favour of histories; when can this representation be compressed and made tractable?