Duality in Probabilistic Automata

Prakash Panangaden[1]

[1]Computing Laboratory
Oxford University
on sabbatical leave from
McGill University

University of Bath: 19th May 2011

Outline

Outline

Outline

Outline

Outline

Outline

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.

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.

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.

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.

Joint work with Doina Precup, Joelle Pineau at the RL Lab at McGill and Chris Hundt now working for Google. More recently with Nick Bezhanishvili and Clemens Kupke.

Deterministic Automata

- $\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta, \gamma)$: a deterministic finite automaton. S is the set of **states**, $\mathcal{A}$ an **input alphabet** (actions), $\mathcal{O}$ is a set of **observations**.
- $\delta : S \times \mathcal{A} \longrightarrow S$ is the **state transition function.**
- $\gamma : S \longrightarrow \mathbf{2}^{\mathcal{O}}$ or $\gamma : S \times \mathcal{O} \longrightarrow \mathbf{2}$ is a labeling function.
- If $\mathcal{O} = \{\textbf{accept}\}$ we have ordinary deterministic finite automata.

Deterministic Automata

- $\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta, \gamma)$: a deterministic finite automaton. S is the set of **states**, $\mathcal{A}$ an **input alphabet** (actions), $\mathcal{O}$ is a set of **observations**.
- $\delta : S \times \mathcal{A} \longrightarrow S$ is the **state transition function**.
- $\gamma : S \longrightarrow \mathbf{2}^{\mathcal{O}}$ or $\gamma : S \times \mathcal{O} \longrightarrow \mathbf{2}$ is a labeling function.
- If $\mathcal{O} = \{\textbf{accept}\}$ we have ordinary deterministic finite automata.

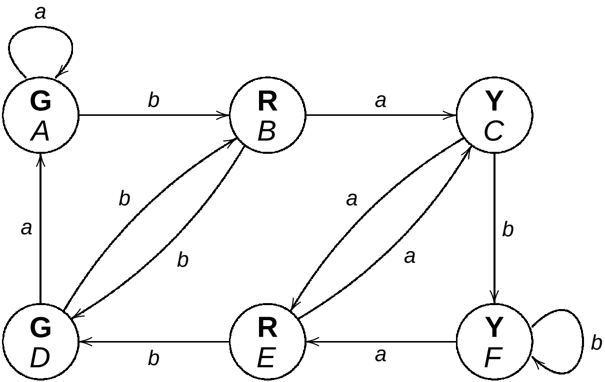
Deterministic Automata

- $\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta, \gamma)$: a deterministic finite automaton. S is the set of **states**, $\mathcal{A}$ an **input alphabet** (actions), $\mathcal{O}$ is a set of **observations**.
- $\delta : S \times \mathcal{A} \longrightarrow S$ is the **state transition function**.
- $\gamma : S \longrightarrow \mathbf{2}^{\mathcal{O}}$ or $\gamma : S \times \mathcal{O} \longrightarrow \mathbf{2}$ is a labeling function.
- If $\mathcal{O} = \{\mathbf{accept}\}$ we have ordinary deterministic finite automata.

Deterministic Automata

- $\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta, \gamma)$: a deterministic finite automaton. S is the set of **states**, $\mathcal{A}$ an **input alphabet** (actions), $\mathcal{O}$ is a set of **observations**.
- $\delta : S \times \mathcal{A} \longrightarrow S$ is the **state transition function**.
- $\gamma : S \longrightarrow \mathbf{2}^{\mathcal{O}}$ or $\gamma : S \times \mathcal{O} \longrightarrow \mathbf{2}$ is a labeling function.
- If $\mathcal{O} = \{\textbf{accept}\}$ we have ordinary deterministic finite automata.

An Example



States are $\{A, B, C, D, E, F\}$ and $\{\mathbf{G}, \mathbf{R}, \mathbf{Y}\}$ are lights.

Testing the Machine

- What can we do with this machine?
- We can ask if *in the present state* the red light is on.
- We can ask whether *after an a-transition* the yellow light is on.
- We can ask whether *after some fixed sequence of transitions* a particular light is on.

Testing the Machine

- What can we do with this machine?
- We can ask if *in the present state* the red light is on.
- We can ask whether *after an a-transition* the yellow light is on.
- We can ask whether *after some fixed sequence of transitions* a particular light is on.

Testing the Machine

- What can we do with this machine?
- We can ask if *in the present state* the red light is on.
- We can ask whether *after an a-transition* the yellow light is on.
- We can ask whether *after some fixed sequence of transitions* a particular light is on.

Testing the Machine

- What can we do with this machine?
- We can ask if *in the present state* the red light is on.
- We can ask whether *after an a-transition* the yellow light is on.
- We can ask whether *after some fixed sequence of transitions* a particular light is on.

States Satisfy Tests (Or Not)

- R is satisfied by states $\{B, E\}$
- After a, **red** is on, is satisfied by $\{C, F\}$.
- After ba, **yellow** is on is satisfied by $\{A, D\}$

States Satisfy Tests (Or Not)

- R is satisfied by states $\{B, E\}$
- After a, **red** is on, is satisfied by $\{C, F\}$.
- After ba, **yellow** is on is satisfied by $\{A, D\}$

States Satisfy Tests (Or Not)

- R is satisfied by states $\{B, E\}$
- After *a*, **red** is on, is satisfied by $\{C, F\}$.
- After *ba*, **yellow** is on is satisfied by $\{A, D\}$

A Simple Modal Logic

- Thinking of the elements of $\mathcal{O}$ as formulas we can use them to define a simple modal logic. We define a *formula* φ according to the following grammar:

$$\varphi ::== \omega \in \mathcal{O} \mid (a)\varphi$$

where $a \in \mathcal{A}$.

- We say $s \models \omega$, if $\omega \in \gamma(s)$ (or $\gamma(s, \omega) = T$). We say $s \models (a)\varphi$ if $\delta(s, a) \models \varphi$.
- Now we define $[\![\varphi]\!]_{\mathcal{M}} = \{s \in S | s \models \varphi\}$.

A Simple Modal Logic

- Thinking of the elements of $\mathcal{O}$ as formulas we can use them to define a simple modal logic. We define a *formula* φ according to the following grammar:

$$\varphi ::== \omega \in \mathcal{O} \mid (a)\varphi$$

  where $a \in \mathcal{A}$.

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

- Now we define $[\![\varphi]\!]_{\mathcal{M}} = \{s \in S | s \models \varphi\}$.

A Simple Modal Logic

- Thinking of the elements of $\mathcal{O}$ as formulas we can use them to define a simple modal logic. We define a *formula* φ according to the following grammar:

$$\varphi ::== \omega \in \mathcal{O} \mid (a)\varphi$$

  where $a \in \mathcal{A}$.

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

- Now we define $[\![\varphi]\!]_{\mathcal{M}} = \{s \in S | s \models \varphi\}$.

An Equivalence Relation on Formulas

- We write sa as shorthand for $\delta(s, a)$.
- Define $\sim_{\mathcal{M}}$ between *formulas* as $\varphi \sim_{\mathcal{M}} \psi$ if $[\![\varphi]\!]_{\mathcal{M}} = [\![\psi]\!]_{\mathcal{M}}$.
- Note that this allows us to identify an equivalence class for φ with the set of states $[\![\varphi]\!]_{\mathcal{M}}$ that satisfy φ.
- Note that another way of defining this equivalence relations is

$$\varphi \sim_{\mathcal{M}} \varphi' := \forall s \in S. s \models \varphi \iff s \models \varphi'.$$

An Equivalence Relation on Formulas

- We write *sa* as shorthand for $\delta(s, a)$.
- Define $\sim_{\mathcal{M}}$ between *formulas* as $\varphi \sim_{\mathcal{M}} \psi$ if $[\![\varphi]\!]_{\mathcal{M}} = [\![\psi]\!]_{\mathcal{M}}$.
- Note that this allows us to identify an equivalence class for φ with the set of states $[\![\varphi]\!]_{\mathcal{M}}$ that satisfy φ.
- Note that another way of defining this equivalence relations is

$$\varphi \sim_{\mathcal{M}} \varphi' := \forall s \in S. s \models \varphi \iff s \models \varphi'.$$

An Equivalence Relation on Formulas

- We write *sa* as shorthand for $\delta(s, a)$.
- Define $\sim_{\mathcal{M}}$ between *formulas* as $\varphi \sim_{\mathcal{M}} \psi$ if $[\![\varphi]\!]_{\mathcal{M}} = [\![\psi]\!]_{\mathcal{M}}$.
- Note that this allows us to identify an equivalence class for φ with the set of states $[\![\varphi]\!]_{\mathcal{M}}$ that satisfy φ.
- Note that another way of defining this equivalence relations is

$$\varphi \sim_{\mathcal{M}} \varphi' := \forall s \in S. s \models \varphi \iff s \models \varphi'.$$

An Equivalence Relation on Formulas

- We write *sa* as shorthand for $\delta(s, a)$.
- Define $\sim_{\mathcal{M}}$ between *formulas* as $\varphi \sim_{\mathcal{M}} \psi$ if $[\![\varphi]\!]_{\mathcal{M}} = [\![\psi]\!]_{\mathcal{M}}$.
- Note that this allows us to identify an equivalence class for φ with the set of states $[\![\varphi]\!]_{\mathcal{M}}$ that satisfy φ.
- Note that another way of defining this equivalence relations is

$$\varphi \sim_{\mathcal{M}} \varphi' := \forall s \in S.s \models \varphi \iff s \models \varphi'.$$

Examples of Equivalent Formulas

- The formulas **G** and **aG** are only satisfied by A, D. They are thus equivalent.
- Other equivalent formulas are all formulas of the form $a^m G$.
- There are a lot of formulas in this equivalence class!
- But there are only finitely many equivalence classes.

Examples of Equivalent Formulas

- The formulas **G** and **aG** are only satisfied by A, D. They are thus equivalent.
- Other equivalent formulas are all formulas of the form $a^m G$.
- There are a lot of formulas in this equivalence class!
- But there are only finitely many equivalence classes.

Examples of Equivalent Formulas

- The formulas **G** and **aG** are only satisfied by A, D. They are thus equivalent.
- Other equivalent formulas are all formulas of the form $a^m G$.
- There are a lot of formulas in this equivalence class!
- But there are only finitely many equivalence classes.

Examples of Equivalent Formulas

- The formulas **G** and **aG** are only satisfied by A, D. They are thus equivalent.
- Other equivalent formulas are all formulas of the form $a^m G$.
- There are a lot of formulas in this equivalence class!
- But there are only finitely many equivalence classes.

An Equivalence Relation on States

- We also define an equivalence $\equiv$ between *states* in $\mathcal{M}$ as $s_1 \equiv s_2$ if for all formulas φ on $\mathcal{M}$, $s_1 \models \varphi \iff s_2 \models \varphi$.
- The equivalence relations $\sim$ and $\equiv$ are clearly closely related: they are the hinge of the duality between states and observations.
- We say that $\mathcal{M}$ 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.

An Equivalence Relation on States

- We also define an equivalence $\equiv$ between *states* in $\mathcal{M}$ as $s_1 \equiv s_2$ if for all formulas φ on $\mathcal{M}$, $s_1 \models \varphi \iff s_2 \models \varphi$.
- The equivalence relations $\sim$ and $\equiv$ are clearly closely related: they are the hinge of the duality between states and observations.
- We say that $\mathcal{M}$ 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.

An Equivalence Relation on States

- We also define an equivalence $\equiv$ between *states* in $\mathcal{M}$ as $s_1 \equiv s_2$ if for all formulas φ on $\mathcal{M}$, $s_1 \models \varphi \iff s_2 \models \varphi$.
- The equivalence relations $\sim$ and $\equiv$ are clearly closely related: they are the hinge of the duality between states and observations.
- We say that $\mathcal{M}$ 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.

An Equivalence Relation on States

- We also define an equivalence $\equiv$ between *states* in $\mathcal{M}$ as $s_1 \equiv s_2$ if for all formulas φ on $\mathcal{M}$, $s_1 \models \varphi \iff s_2 \models \varphi$.
- The equivalence relations $\sim$ and $\equiv$ are clearly closely related: they are the hinge of the duality between states and observations.
- We say that $\mathcal{M}$ 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 $\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta, \gamma)$. Let T be the set of $\sim_{\mathcal{M}}$-equivalence classes of formulas on $\mathcal{M}$.
- We define $\mathcal{M}' = (S', \mathcal{A}, \mathcal{O}', \delta', \gamma')$ as follows:
- $S' = T = \{[\![\varphi]\!]_{\mathcal{M}}\}$
- $\mathcal{O}' = S$
- $\delta'([\![\varphi]\!]_{\mathcal{M}}, a) = [\![(a)\varphi]\!]_{\mathcal{M}}$
- $\gamma'([\![\varphi]\!]_{\mathcal{M}}) = [\![\varphi]\!]_{\mathcal{M}}$ or $\gamma'([\![\varphi]\!]_{\mathcal{A}}, s) = (s \models \varphi)$.

A Dual Automaton

- Given a finite automaton $\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta, \gamma)$. Let T be the set of $\sim_{\mathcal{M}}$-equivalence classes of formulas on $\mathcal{M}$.
- We define $\mathcal{M}' = (S', \mathcal{A}, \mathcal{O}', \delta', \gamma')$ as follows:
- $S' = T = \{[\![\varphi]\!]_{\mathcal{M}}\}$
- $\mathcal{O}' = S$
- $\delta'([\![\varphi]\!]_{\mathcal{M}}, a) = [\![(a)\varphi]\!]_{\mathcal{M}}$
- $\gamma'([\![\varphi]\!]_{\mathcal{M}}) = [\![\varphi]\!]_{\mathcal{M}}$ or $\gamma'([\![\varphi]\!]_{\mathcal{A}}, s) = (s \models \varphi)$.

A Dual Automaton

- Given a finite automaton $\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta, \gamma)$. Let T be the set of $\sim_{\mathcal{M}}$-equivalence classes of formulas on $\mathcal{M}$.
- We define $\mathcal{M}' = (S', \mathcal{A}, \mathcal{O}', \delta', \gamma')$ as follows:
- $S' = T = \{[\![\varphi]\!]_{\mathcal{M}}\}$
- $\mathcal{O}' = S$
- $\delta'([\![\varphi]\!]_{\mathcal{M}}, a) = [\![(a)\varphi]\!]_{\mathcal{M}}$
- $\gamma'([\![\varphi]\!]_{\mathcal{M}}) = [\![\varphi]\!]_{\mathcal{M}}$ or $\gamma'([\![\varphi]\!]_{\mathcal{A}}, s) = (s \models \varphi)$.

A Dual Automaton

- Given a finite automaton $\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta, \gamma)$. Let T be the set of $\sim_{\mathcal{M}}$-equivalence classes of formulas on $\mathcal{M}$.
- We define $\mathcal{M}' = (S', \mathcal{A}, \mathcal{O}', \delta', \gamma')$ as follows:
- $S' = T = \{[\![\varphi]\!]_{\mathcal{M}}\}$
- $\mathcal{O}' = S$
- $\delta'([\![\varphi]\!]_{\mathcal{M}}, a) = [\![(a)\varphi]\!]_{\mathcal{M}}$
- $\gamma'([\![\varphi]\!]_{\mathcal{M}}) = [\![\varphi]\!]_{\mathcal{M}}$ or $\gamma'([\![\varphi]\!]_{\mathcal{A}}, s) = (s \models \varphi)$.

A Dual Automaton

- Given a finite automaton $\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta, \gamma)$.
  Let T be the set of $\sim_{\mathcal{M}}$-equivalence classes of formulas on $\mathcal{M}$.
- We define $\mathcal{M}' = (S', \mathcal{A}, \mathcal{O}', \delta', \gamma')$ as follows:
- $S' = T = \{[\![\varphi]\!]_{\mathcal{M}}\}$
- $\mathcal{O}' = S$
- $\delta'([\![\varphi]\!]_{\mathcal{M}}, a) = [\![(a)\varphi]\!]_{\mathcal{M}}$
- $\gamma'([\![\varphi]\!]_{\mathcal{M}}) = [\![\varphi]\!]_{\mathcal{M}}$ or $\gamma'([\![\varphi]\!]_{\mathcal{A}}, s) = (s \models \varphi)$.

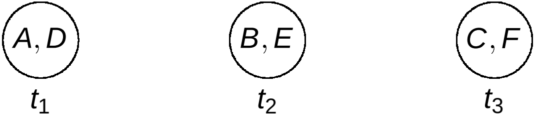
A Dual Automaton

- Given a finite automaton $\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta, \gamma)$.
  Let T be the set of $\sim_{\mathcal{M}}$-equivalence classes of formulas on $\mathcal{M}$.
- We define $\mathcal{M}' = (S', \mathcal{A}, \mathcal{O}', \delta', \gamma')$ as follows:
- $S' = T = \{[\![\varphi]\!]_{\mathcal{M}}\}$
- $\mathcal{O}' = S$
- $\delta'([\![\varphi]\!]_{\mathcal{M}}, a) = [\![(a)\varphi]\!]_{\mathcal{M}}$
- $\gamma'([\![\varphi]\!]_{\mathcal{M}}) = [\![\varphi]\!]_{\mathcal{M}}$ or $\gamma'([\![\varphi]\!]_{\mathcal{A}}, s) = (s \models \varphi)$.

The intuition

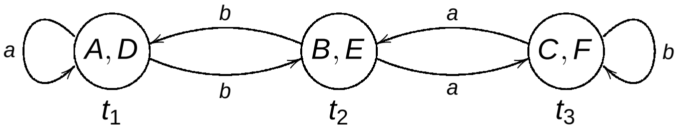
We have interchanged the states and the observations or propositions; more precisely we have interchanged equivalence classes of formulas - based on the observations - with the states. We have made the states of the old machine the observations of the dual machine.

The States of the Dual: Our Example



The states are equivalence classes of formulas but we have labelled them with the *set of states of the original machine* that satisfies the tests.

The Dual Machine in Full: Our Example



The Double Dual

- Now consider $\mathcal{M}'' = (\mathcal{M}')'$, the dual of the dual.
- Its states are equivalence classes of $\mathcal{M}'$-formulas.
- Each such class is identified with a set $[\![\varphi']\!]_{\mathcal{M}'}$ of $\mathcal{M}'$-states by which formulas in that class are satisfied, and
- each $\mathcal{M}'$-state is an equivalence class of $\mathcal{M}$-formulas.
- Thus we can look at states in $\mathcal{M}''$ as collections of $\mathcal{M}$-formula equivalence classes.

The Double Dual

- Now consider $\mathcal{M}'' = (\mathcal{M}')'$, the dual of the dual.
- Its states are equivalence classes of $\mathcal{M}'$-formulas.
- Each such class is identified with a set $[\![\varphi']\!]_{\mathcal{M}'}$ of $\mathcal{M}'$-states by which formulas in that class are satisfied, and
- each $\mathcal{M}'$-state is an equivalence class of $\mathcal{M}$-formulas.
- Thus we can look at states in $\mathcal{M}''$ as collections of $\mathcal{M}$-formula equivalence classes.

The Double Dual

- Now consider $\mathcal{M}'' = (\mathcal{M}')'$, the dual of the dual.
- Its states are equivalence classes of $\mathcal{M}'$-formulas.
- Each such class is identified with a set $[\![\varphi']\!]_{\mathcal{M}'}$ of $\mathcal{M}'$-states by which formulas in that class are satisfied, and
- each $\mathcal{M}'$-state is an equivalence class of $\mathcal{M}$-formulas.
- Thus we can look at states in $\mathcal{M}''$ as collections of $\mathcal{M}$-formula equivalence classes.

The Double Dual

- Now consider $\mathcal{M}'' = (\mathcal{M}')'$, the dual of the dual.
- Its states are equivalence classes of $\mathcal{M}'$-formulas.
- Each such class is identified with a set $[\![\varphi']\!]_{\mathcal{M}'}$ of $\mathcal{M}'$-states by which formulas in that class are satisfied, and
- each $\mathcal{M}'$-state is an equivalence class of $\mathcal{M}$-formulas.
- Thus we can look at states in $\mathcal{M}''$ as collections of $\mathcal{M}$-formula equivalence classes.

The Double Dual

- Now consider $\mathcal{M}'' = (\mathcal{M}')'$, the dual of the dual.
- Its states are equivalence classes of $\mathcal{M}'$-formulas.
- Each such class is identified with a set $[\![\varphi']\!]_{\mathcal{M}'}$ of $\mathcal{M}'$-states by which formulas in that class are satisfied, and
- each $\mathcal{M}'$-state is an equivalence class of $\mathcal{M}$-formulas.
- Thus we can look at states in $\mathcal{M}''$ as collections of $\mathcal{M}$-formula equivalence classes.

The Double Dual 2

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

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

- Lemma: For any $s \in S$, $Sat(s)$ is a state in $\mathcal{M}''$.
- In fact *all* the states of the double dual have this form.
- Lemma: Let $s'' = [\![\varphi]\!]_{\mathcal{M}'} \in S''$ be any state in $\mathcal{M}''$. Then $s'' = Sat(s_\varphi)$ for some state $s_\varphi \in S$.
- The proof is by an easy induction on φ.

The Double Dual 2

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

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

- Lemma: For any $s \in S$, $Sat(s)$ is a state in $\mathcal{M}''$.
- In fact *all* the states of the double dual have this form.
- Lemma: Let $s'' = [\![\varphi]\!]_{\mathcal{M}'} \in S''$ be any state in $\mathcal{M}''$. Then $s'' = Sat(s_\varphi)$ for some state $s_\varphi \in S$.
- The proof is by an easy induction on φ.

The Double Dual 2

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

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

- Lemma: For any $s \in S$, $Sat(s)$ is a state in $\mathcal{M}''$.
- In fact *all* the states of the double dual have this form.
- Lemma: Let $s'' = [\![\varphi]\!]_{\mathcal{M}'} \in S''$ be any state in $\mathcal{M}''$. Then $s'' = Sat(s_\varphi)$ for some state $s_\varphi \in S$.
- The proof is by an easy induction on φ.

The Double Dual 2

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

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

- Lemma: For any $s \in S$, $Sat(s)$ is a state in $\mathcal{M}''$.
- In fact *all* the states of the double dual have this form.
- Lemma: Let $s'' = [\![\varphi]\!]_{\mathcal{M}'} \in S''$ be any state in $\mathcal{M}''$. Then $s'' = Sat(s_\varphi)$ for some state $s_\varphi \in S$.
- The proof is by an easy induction on φ.

The Double Dual 2

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

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

- Lemma: For any $s \in S$, $Sat(s)$ is a state in $\mathcal{M}''$.
- In fact *all* the states of the double dual have this form.
- Lemma: Let $s'' = [\![\varphi]\!]_{\mathcal{M}'} \in S''$ be any state in $\mathcal{M}''$. Then $s'' = Sat(s_\varphi)$ for some state $s_\varphi \in S$.
- The proof is by an easy induction on φ.

Minimality Properties

- If $\mathcal{A}$ is reduced then *Sat* is a bijection from S to S''.
- 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.

Minimality Properties

- If $\mathcal{A}$ is reduced then *Sat* is a bijection from S to S''.
- 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.

Minimality Properties

- If $\mathcal{A}$ is reduced then *Sat* is a bijection from S to S''.
- 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.

Minimality Properties

- If $\mathcal{A}$ is reduced then *Sat* is a bijection from S to S''.
- 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

$$\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta : S \times \mathcal{A} \longrightarrow 2^S, \gamma : S \longrightarrow 2^{\mathcal{O}}).$$

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

$$Q \models \omega \iff \exists s \in Q : \omega \in \gamma(s)$$

$$Q \models (a)\varphi \iff \delta(Q, a) \models \varphi.$$

- We define an appropriate notion of simulation and prove: $\mathcal{M}$ is simulated by $\mathcal{M}''$.
- The double dual is always deterministic; we have sneaked in the notion of determinization into the satisfaction relation.

The Nondeterministic Case

- Here we consider automata of the type
$$\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta : S \times \mathcal{A} \longrightarrow 2^S, \gamma : S \longrightarrow 2^{\mathcal{O}}).$$
- We use the same formulas but we have a different notion of satisfaction: $Q \subseteq S$
$$Q \models \omega \iff \exists s \in Q : \omega \in \gamma(s)$$
$$Q \models (a)\varphi \iff \delta(Q, a) \models \varphi.$$
- We define an appropriate notion of simulation and prove: $\mathcal{M}$ is simulated by $\mathcal{M}''$.
- The double dual is always deterministic; we have sneaked in the notion of determinization into the satisfaction relation.

The Nondeterministic Case

- Here we consider automata of the type
$$\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta : S \times \mathcal{A} \longrightarrow 2^S, \gamma : S \longrightarrow 2^{\mathcal{O}}).$$
- We use the same formulas but we have a different notion of satisfaction: $Q \subseteq S$
$$Q \models \omega \iff \exists s \in Q : \omega \in \gamma(s)$$
$$Q \models (a)\varphi \iff \delta(Q, a) \models \varphi.$$
- We define an appropriate notion of simulation and prove: $\mathcal{M}$ is simulated by $\mathcal{M}''$.
- The double dual is always deterministic; we have sneaked in the notion of determinization into the satisfaction relation.

The Nondeterministic Case

- Here we consider automata of the type
$$\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta : S \times \mathcal{A} \to 2^S, \gamma : S \to 2^{\mathcal{O}}).$$
- We use the same formulas but we have a different notion of satisfaction: $Q \subseteq S$
$$Q \models \omega \iff \exists s \in Q : \omega \in \gamma(s)$$
$$Q \models (a)\varphi \iff \delta(Q, a) \models \varphi.$$
- We define an appropriate notion of simulation and prove: $\mathcal{M}$ is simulated by $\mathcal{M}''$.
- The double dual is always deterministic; we have sneaked in the notion of determinization into the satisfaction relation.

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.

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.

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.
- Markov Decision Processes aka Labelled Markov Processes:

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

The τ_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.

Probabilistic systems

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

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

The τ_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.

Probabilistic systems

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

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

  The τ_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.

Probabilistic systems

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

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

  The τ_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.
- 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.

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.

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.

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

- $\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta : S \times \mathcal{A} \times S \longrightarrow [0, 1], \gamma : S \times \mathcal{A} \times \mathcal{O} \longrightarrow [0, 1])$,
- where S is the set of states, $\mathcal{O}$ is the set of observations, $\mathcal{A}$ is the set of actions, δ is the transition probability function and γ gives the observation probabilities.

Formal Definition of a POMDP

- $\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta : S \times \mathcal{A} \times S \longrightarrow [0,1], \gamma : S \times \mathcal{A} \times \mathcal{O} \longrightarrow [0,1]),$
- where S is the set of states, $\mathcal{O}$ is the set of observations, $\mathcal{A}$ is the set of actions, δ is the transition probability function and γ gives the observation probabilities.

Automata with State-based Observations

- A **deterministic automaton with stochastic observations** is a quintuple

$$\mathcal{E} = (S, \mathcal{A}, \mathcal{O}, \delta : S \times \mathcal{A} \longrightarrow S, \gamma : S \times \mathcal{O} \longrightarrow [0, 1]).$$

  Note that this has deterministic transitions and stochastic observations.

- A **probabilistic automaton with stochastic observations** is

$$\mathcal{F} = (S, \mathcal{A}, \mathcal{O}, \delta : S \times \mathcal{A} \times S \longrightarrow [0, 1], \gamma : S \times \mathcal{O} \longrightarrow [0, 1]).$$

Automata with State-based Observations

- A **deterministic automaton with stochastic observations** is a quintuple

$$\mathcal{E} = (S, \mathcal{A}, \mathcal{O}, \delta : S \times \mathcal{A} \longrightarrow S, \gamma : S \times \mathcal{O} \longrightarrow [0,1]).$$

  Note that this has deterministic transitions and stochastic observations.
- A **probabilistic automaton with stochastic observations** is

$$\mathcal{F} = (S, \mathcal{A}, \mathcal{O}, \delta : S \times \mathcal{A} \times S \longrightarrow [0,1], \gamma : S \times \mathcal{O} \longrightarrow [0,1]).$$

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

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

- where $\delta : S \times \mathcal{A} \times S \longrightarrow [0, 1]$ is the *transition function*
- and $\gamma : S \times \mathcal{O} \longrightarrow [0, 1]$ 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

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

- where $\delta : S \times \mathcal{A} \times S \longrightarrow [0, 1]$ is the *transition function*
- and $\gamma : S \times \mathcal{O} \longrightarrow [0, 1]$ 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

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

- where $\delta : S \times \mathcal{A} \times S \longrightarrow [0, 1]$ is the *transition function*
- and $\gamma : S \times \mathcal{O} \longrightarrow [0, 1]$ 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

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

- where $\delta : S \times \mathcal{A} \times S \longrightarrow [0, 1]$ is the *transition function*
- and $\gamma : S \times \mathcal{O} \longrightarrow [0, 1]$ 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 write $a.t$ 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:

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

$$[\![at]\!]_{\mathcal{E}}(s) = \sum_{s'} \delta(s, a, s')[\![t]\!]_{\mathcal{E}}(s').$$

- In AI these are called “e-tests.”

Simple Tests 2

- We use the same logic as before except that we give a probabilistic semantics and call the formulas "tests." I write $a.t$ 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:

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

$$[\![at]\!]_{\mathcal{E}}(s) = \sum_{s'} \delta(s,a,s')[\![t]\!]_{\mathcal{E}}(s').$$

- In AI these are called "e-tests."

Simple Tests 2

- We use the same logic as before except that we give a probabilistic semantics and call the formulas “tests.” I write $a.t$ 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:

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

$$[\![at]\!]_{\mathcal{E}}(s) = \sum_{s'} \delta(s, a, s')[\![t]\!]_{\mathcal{E}}(s').$$

- In AI these are called “e-tests.”

Simple Tests 2

- We use the same logic as before except that we give a probabilistic semantics and call the formulas "tests." I write $a.t$ 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:

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

$$[\![at]\!]_{\mathcal{E}}(s) = \sum_{s'} \delta(s, a, s')[\![t]\!]_{\mathcal{E}}(s').$$

- In AI these are called "e-tests."

Duality with e-tests

- $S' = \{[\![t]\!]_{\mathcal{E}}\}$
- $\mathcal{O}' = S$
- $\gamma'([\![t]\!]_{\mathcal{E}}, s) = [\![t]\!]_{\mathcal{E}}(s)$
- $\delta'([\![t]\!]_{\mathcal{E}}, a, [\![at]\!]_{\mathcal{E}}) = 1$; 0 otherwise.
- This machine has deterministic transitions and γ' is just the transpose of γ.

Duality with e-tests

- $S' = \{[\![t]\!]_{\mathcal{E}}\}$
- $\mathcal{O}' = S$
- $\gamma'([\![t]\!]_{\mathcal{E}}, s) = [\![t]\!]_{\mathcal{E}}(s)$
- $\delta'([\![t]\!]_{\mathcal{E}}, a, [\![at]\!]_{\mathcal{E}}) = 1$; 0 otherwise.
- This machine has deterministic transitions and γ' is just the transpose of γ.

Duality with e-tests

- $S' = \{[\![t]\!]_{\mathcal{E}}\}$
- $\mathcal{O}' = S$
- $\gamma'([\![t]\!]_{\mathcal{E}}, s) = [\![t]\!]_{\mathcal{E}}(s)$
- $\delta'([\![t]\!]_{\mathcal{E}}, a, [\![at]\!]_{\mathcal{E}}) = 1$; 0 otherwise.
- This machine has deterministic transitions and γ' is just the transpose of γ.

Duality with e-tests

- $S' = \{[\![t]\!]_{\mathcal{E}}\}$
- $\mathcal{O}' = S$
- $\gamma'([\![t]\!]_{\mathcal{E}}, s) = [\![t]\!]_{\mathcal{E}}(s)$
- $\delta'([\![t]\!]_{\mathcal{E}}, a, [\![at]\!]_{\mathcal{E}}) = 1$; 0 otherwise.
- This machine has deterministic transitions and γ' is just the transpose of γ.

Duality with e-tests

- $S' = \{[\![t]\!]_{\mathcal{E}}\}$
- $\mathcal{O}' = S$
- $\gamma'([\![t]\!]_{\mathcal{E}}, s) = [\![t]\!]_{\mathcal{E}}(s)$
- $\delta'([\![t]\!]_{\mathcal{E}}, a, [\![at]\!]_{\mathcal{E}}) = 1$; 0 otherwise.
- This machine has deterministic transitions and γ' is just the transpose of γ.

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:

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

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:

$$\begin{aligned} &[\![a_1 a_2 \cdots a_k o]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}) \\ &= [\![a_1 a_2 \cdots a_k o]\!]_{\mathcal{E}}(s). \end{aligned}$$

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:

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

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:

$$\begin{aligned} &[\![a_1 a_2 \cdots a_k o]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}) \\ &= [\![a_1 a_2 \cdots a_k o]\!]_{\mathcal{E}}(s). \end{aligned}$$

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.
- $$[\![o_1 a_1 o_2 a_2 o_3]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}) =$$
  $$[\![o_1]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}''}) \cdot [\![a_1 o_2]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}) \cdot [\![a_1 a_2 o_3]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}).$$
  This does not hold in the primal.
- The double dual does not conditionalize with respect to intermediate 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.
- $$[\![o_1 a_1 o_2 a_2 o_3]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}) =$$
  $$[\![o_1]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}''}) \cdot [\![a_1 o_2]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}) \cdot [\![a_1 a_2 o_3]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}).$$
  This does not hold in the primal.
- The double dual does not conditionalize with respect to intermediate 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.
- $$[\![o_1 a_1 o_2 a_2 o_3]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}) =$$
  $$[\![o_1]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}''}) \cdot [\![a_1 o_2]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}) \cdot [\![a_1 a_2 o_3]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}).$$
  This does not hold in the primal.
- The double dual does not conditionalize with respect to intermediate 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.
- $$[\![o_1 a_1 o_2 a_2 o_3]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}) =$$
  $$[\![o_1]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}''}) \cdot [\![a_1 o_2]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}) \cdot [\![a_1 a_2 o_3]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}).$$
  This does not hold in the primal.
- The double dual does not conditionalize with respect to intermediate observations.

More General Tests

- Recall the definition of a POMDP

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

- 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

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

- 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

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

- 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.

$$\delta_\epsilon(s, s') = \mathbf{1}_{s=s'} \qquad \forall s, s' \in S$$
$$\delta_{a\alpha}(s, s') = \sum_{s''} \delta_a(s, s'')\delta_\alpha(s'', s') \qquad \forall s, s' \in S.$$

- We have written $\mathbf{1}_{s=s'}$ 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 $\langle s|e|s'\rangle$.

Some Notation

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

$$\delta_\epsilon(s, s') = \mathbf{1}_{s=s'} \qquad \forall s, s' \in S$$
$$\delta_{a\alpha}(s, s') = \sum_{s''} \delta_a(s, s'')\delta_\alpha(s'', s') \qquad \forall s, s' \in S.$$

- We have written $\mathbf{1}_{s=s'}$ 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 $\langle s|e|s'\rangle$.

Some Notation

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

$$
\begin{aligned}
\delta_\epsilon(s,s') &= \mathbf{1}_{s=s'} && \forall s,s' \in S \\
\delta_{a\alpha}(s,s') &= \sum_{s''} \delta_a(s,s'')\delta_\alpha(s'',s') && \forall s,s' \in S.
\end{aligned}
$$

- We have written $\mathbf{1}_{s=s'}$ 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 $\langle s|e|s'\rangle$.

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
$$\langle s|e\rangle = \sum_{s'} \langle s|e|s'\rangle.$$

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
$$e_1 \sim_{\mathcal{M}} e_2 \Leftrightarrow \langle s|e_1\rangle = \langle s|e_2\rangle \forall s \in S.$$
- Then $[e]_{\mathcal{M}}$ is the $\sim_{\mathcal{M}}$-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.

Equivalence on Experiments

- For experiments e_1, e_2, we say

$$e_1 \sim_{\mathcal{M}} e_2 \Leftrightarrow \langle s|e_1\rangle = \langle s|e_2\rangle \forall s \in S.$$

- Then $[e]_{\mathcal{M}}$ is the $\sim_{\mathcal{M}}$-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.

Equivalence on Experiments

- For experiments e_1, e_2, we say

$$e_1 \sim_{\mathcal{M}} e_2 \Leftrightarrow \langle s|e_1\rangle = \langle s|e_2\rangle \forall s \in S.$$

- Then $[e]_{\mathcal{M}}$ is the $\sim_{\mathcal{M}}$-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.

Equivalence on Experiments

- For experiments e_1, e_2, we say

$$e_1 \sim_{\mathcal{M}} e_2 \Leftrightarrow \langle s|e_1\rangle = \langle s|e_2\rangle \forall s \in S.$$

- Then $[e]_{\mathcal{M}}$ is the $\sim_{\mathcal{M}}$-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 $\mathcal{M}' =$

$$(S', \mathcal{A}, \mathcal{O}', \delta' : S' \times \mathcal{A} \to S', \gamma' : S' \times \mathcal{O}' \to [0, 1]),$$

- where $S' = \{[e]_{\mathcal{M}}\}$, $\mathcal{O}' = S$
- $\delta'([e]_{\mathcal{M}}, a_0) = [a_0 e]_{\mathcal{M}}$ and
- $\gamma'([e]_{\mathcal{M}}, s) = \langle s|e\rangle$.
- We get a deterministic transition system with stochastic observations.

The Dual Machine

- We define the dual as $\mathcal{M}' =$

$$(S', \mathcal{A}, \mathcal{O}', \delta' : S' \times \mathcal{A} \to S', \gamma' : S' \times \mathcal{O}' \to [0,1]),$$

- where $S' = \{[e]_{\mathcal{M}}\}$, $\mathcal{O}' = S$
- $\delta'([e]_{\mathcal{M}}, a_0) = [a_0 e]_{\mathcal{M}}$ and
- $\gamma'([e]_{\mathcal{M}}, s) = \langle s | e \rangle$.
- We get a deterministic transition system with stochastic observations.

The Dual Machine

- We define the dual as $\mathcal{M}' =$

$$(S', \mathcal{A}, \mathcal{O}', \delta' : S' \times \mathcal{A} \to S', \gamma' : S' \times \mathcal{O}' \to [0,1]),$$

- where $S' = \{[e]_{\mathcal{M}}\}$, $\mathcal{O}' = S$
- $\delta'([e]_{\mathcal{M}}, a_0) = [a_0 e]_{\mathcal{M}}$ and
- $\gamma'([e]_{\mathcal{M}}, s) = \langle s | e \rangle$.
- We get a deterministic transition system with stochastic observations.

The Dual Machine

- We define the dual as $\mathcal{M}' =$

$$(S', \mathcal{A}, \mathcal{O}', \delta' : S' \times \mathcal{A} \longrightarrow S', \gamma' : S' \times \mathcal{O}' \longrightarrow [0,1]),$$

- where $S' = \{[e]_{\mathcal{M}}\}$, $\mathcal{O}' = S$
- $\delta'([e]_{\mathcal{M}}, a_0) = [a_0 e]_{\mathcal{M}}$ and
- $\gamma'([e]_{\mathcal{M}}, s) = \langle s|e\rangle$.
- We get a deterministic transition system with stochastic observations.

The Dual Machine

- We define the dual as $\mathcal{M}' =$

$$(S', \mathcal{A}, \mathcal{O}', \delta' : S' \times \mathcal{A} \longrightarrow S', \gamma' : S' \times \mathcal{O}' \longrightarrow [0, 1]),$$

- where $S' = \{[e]_{\mathcal{M}}\}$, $\mathcal{O}' = S$
- $\delta'([e]_{\mathcal{M}}, a_0) = [a_0 e]_{\mathcal{M}}$ and
- $\gamma'([e]_{\mathcal{M}}, s) = \langle s | e \rangle$.
- We get a deterministic transition system with stochastic observations.

The Double Dual

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

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

where

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

The Double Dual

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

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

where

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

The Double Dual

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

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

where

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

The Double Dual

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

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

where

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

The Double Dual

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

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

where

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

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. $\langle s|e\rangle$ is given by $\langle [s]_{\mathcal{M}'}|[e]_{\mathcal{M}}\rangle = \gamma''([s]_{\mathcal{M}'})|[e]_{\mathcal{M}}\rangle$, 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.

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. $\langle s|e\rangle$ is given by $\langle [s]_{\mathcal{M}'}|[e]_{\mathcal{M}}\rangle = \gamma''([s]_{\mathcal{M}'})|[e]_{\mathcal{M}}\rangle$, 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.
- 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.

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.

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.

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.

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.

What is the right categorical description?

- Is this is any kind of familiar Stone-type duality?
- We know that machines are co-algebras and logics are algebras but
- why is the dual another automaton?

What is the right categorical description?

- Is this is any kind of familiar Stone-type duality?
- We know that machines are co-algebras and logics are algebras but
- why is the dual another automaton?

What is the right categorical description?

- Is this is any kind of familiar Stone-type duality?
- We know that machines are co-algebras and logics are algebras but
- why is the dual another automaton?

Automata as Coalgebras

Our automata are coalgebras of the following functor:

$$F(S) = S^{\mathcal{A}} \times \mathbf{2}^{\mathcal{O}}, \quad F(f : S \to S') = \lambda(\alpha : \mathcal{A} \to S, \ O \subset \mathcal{O}).(f \circ \alpha, \ O).$$

The category of these coalgebras is called **PODFA.**

Homomorphisms

A homomorphism for these coalgebras is a function $f : S \to S'$ such that the following diagram commutes:

$$\begin{tikzcd} S \arrow[r, "f"] \arrow[d, "{(\delta,\ \gamma)}"'] & S' \arrow[d, "{(\delta',\ \gamma')}"] \\ S^{\mathcal{A}} \times \mathbf{2}^{\mathcal{O}} \arrow[r, "f^{\mathcal{A}} \times \mathrm{id}"'] & S'^{\mathcal{A}} \times \mathbf{2}^{\mathcal{O}} \end{tikzcd}$$

where $f^{\mathcal{A}}(\alpha) = f \circ \alpha$.

This translates to the following conditions:

$$\forall s \in S, \omega \in \mathcal{O}, \ \omega \in \gamma(s) \iff \omega \in \gamma'(f(s)) \tag{1}$$

and

$$\forall s \in S, a \in \mathcal{A}, \ f(\delta(s,a)) = \delta'(f(s),a). \tag{2}$$

The Dual Category

- The category of **finite boolean algebras with operators** (**FBAO**) has as objects finite boolean algebras B with
- the usual operations $\wedge$, $\neg$ and constants $\top$ and $\bot$ and, in addition,
- together with unary operators (a) and constants $\underline{\omega}$.
- We denote an object by $\mathcal{B} = (B, \{(a)|a \in \mathcal{A}\}, \{\underline{\omega}|\omega \in \mathcal{O}\}, \top, \wedge, \neg)$.

The Dual Category

- The category of **finite boolean algebras with operators** (**FBAO**) has as objects finite boolean algebras B with
- the usual operations $\wedge$, $\neg$ and constants T and $\perp$ and, in addition,
- together with unary operators (a) and constants $\underline{\omega}$.
- We denote an object by $\mathcal{B} = (B, \{(a) | a \in \mathcal{A}\}, \{\underline{\omega} | \omega \in \mathcal{O}\}, \mathrm{T}, \wedge, \neg)$.

The Dual Category

- The category of **finite boolean algebras with operators** (**FBAO**) has as objects finite boolean algebras B with
- the usual operations $\wedge$, $\neg$ and constants T and $\perp$ and, in addition,
- together with unary operators (a) and constants $\underline{\omega}$.
- We denote an object by $\mathcal{B} = (B, \{(a)|a \in \mathcal{A}\}, \{\underline{\omega}|\omega \in \mathcal{O}\}, \mathrm{T}, \wedge, \neg)$.

The Dual Category

- The category of **finite boolean algebras with operators** (**FBAO**) has as objects finite boolean algebras B with
- the usual operations $\wedge$, $\neg$ and constants T and $\perp$ and, in addition,
- together with unary operators (a) and constants $\underline{\omega}$.
- We denote an object by $\mathcal{B} = (B, \{(a) | a \in \mathcal{A}\}, \{\underline{\omega} | \omega \in \mathcal{O}\}, \mathsf{T}, \wedge, \neg)$.

Morphisms

The morphisms are the usual boolean homomorphisms preserving, in addition, the constants and the unary operators. The following three equations hold:

$$(a)(b_1 \wedge b_2) = (a)b_1 \wedge (a)b_2, \tag{3a}$$

$$(a)\mathsf{T} = \mathsf{T}, \tag{3b}$$

$$\neg(a)\neg b = (a)b. \tag{3c}$$

Duality Theorem

There is a dual equivalence of categories

$$\mathbf{PODFA}^{op} \cong \mathbf{FBAO}.$$

One functor $\mathcal{P}$ is just the contravariant power set functor and the other one $\mathcal{H}$ maps a boolean algebra to its set of atoms.

Minimization?

- Obviously, if we have an equivalence of categories we get the same machine back when we go back and forth.
- So how do we explain the minimization?

Minimization?

- Obviously, if we have an equivalence of categories we get the same machine back when we go back and forth.
- So how do we explain the minimization?

Definable Subsets

Define a logic $\mathcal{L}$ by

$$\phi ::== \mathsf{T} | \bot | \phi_1 \wedge \phi_2 | \neg\phi | (a)\phi | \underline{\omega}$$

and define the **definable subsets** $\mathcal{D}(S)$ of a machine $\mathcal{M} = (S, \delta, \gamma)$ as sets of the form $[\![\phi]\!]$.

- $\mathcal{D}(S)$ is a subobject of $\mathcal{P}(\mathcal{M})$
- in fact it is the *smallest* possible subalgebra and
- any other subalgebra must contain $\mathcal{D}(S)$.

- $\mathcal{D}(S)$ is a subobject of $\mathcal{P}(\mathcal{M})$
- in fact it is the *smallest* possible subalgebra and
- any other subalgebra must contain $\mathcal{D}(S)$.

- $\mathcal{D}(S)$ is a subobject of $\mathcal{P}(\mathcal{M})$
- in fact it is the *smallest* possible subalgebra and
- any other subalgebra must contain $\mathcal{D}(S)$.

In Pictures

$$\mathcal{M} \dashrightarrow \mathcal{P}(\mathcal{M})$$

$$\begin{tikzcd} \mathcal{M} \arrow[r, dotted] & \mathcal{P}(\mathcal{M}) \\ & D(S) \arrow[u, hook, "1"'] \end{tikzcd}$$

$$
\begin{tikzcd}
\mathcal{M} \arrow[r, dotted] \arrow[d, two heads, "2"'] & \mathcal{P}(\mathcal{M}) \\
\mathcal{H}(D(S)) & D(S) \arrow[l, dotted] \arrow[u, hook, "1"']
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & \mathcal{M} \arrow[r, dotted] \arrow[d, "2"'] \arrow[dl, "3"'] & \mathcal{P}(\mathcal{M}) & \\
 & \mathcal{H}(D(S)) & D(S) \arrow[l, dotted] \arrow[u, hook, "1"'] & \\
\mathcal{S}' \arrow[rrr, dotted, "4"'] & & & \mathcal{P}(\mathcal{S}')
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & \mathcal{M} \arrow[r, dotted] \arrow[d, "2"] \arrow[dl, "3"'] & \mathcal{P}(\mathcal{M}) & \\
 & \mathcal{H}(D(S)) & D(S) \arrow[l, dotted] \arrow[u, hook, "1"'] & \\
\mathcal{S}' \arrow[rrr, dotted, "4"'] & & & \mathcal{P}(\mathcal{S}') \arrow[uul, hook, "5"']
\end{tikzcd}
$$

$$
\begin{tikzcd}
 & \mathcal{M} \arrow[r, dashed] \arrow[d, "2"'] \arrow[ddl, "3"'] & \mathcal{P}(\mathcal{M}) & \\
 & \mathcal{H}(D(S)) & D(S) \arrow[l, dashed] \arrow[u, hook, "1"'] \arrow[dr, hook, "6"'] & \\
\mathcal{S}' \arrow[rrr, dashed, "4"'] & & & \mathcal{P}(\mathcal{S}') \arrow[uul, hook, "5"']
\end{tikzcd}
$$

The Secret of Minimization

$$\begin{tikzcd}
 & \mathcal{M} \arrow[r, dotted] \arrow[d, two heads, "2"'] \arrow[dl, two heads, "3"'] & \mathcal{P}(\mathcal{M}) & \\
 & \mathcal{H}(D(S)) & D(S) \arrow[l, dotted] \arrow[u, hook, "1"'] \arrow[dr, hook, "6"'] & \\
\mathcal{S}' \arrow[rrr, dotted, "4"'] \arrow[ur, two heads, "7"'] & & & \mathcal{P}(\mathcal{S}') \arrow[uul, hook, "5"']
\end{tikzcd}$$

A Simpler Logic

- Why did the minimization work with just the logic

$$\phi ::== \underline{\omega} | (a)\phi?$$

- With this logic the definable subsets $E(S)$ do not form a boolean algebra
- it is just a "set with operations"
- in other words it can be viewed as an automaton!

A Simpler Logic

- Why did the minimization work with just the logic

$$\phi ::== \underline{\omega} | (a)\phi ?$$

- With this logic the definable subsets $E(S)$ do not form a boolean algebra
- it is just a “set with operations”
- in other words it can be viewed as an automaton!

A Simpler Logic

- Why did the minimization work with just the logic

$$\phi ::== \underline{\omega} | (a)\phi?$$

- With this logic the definable subsets $E(S)$ do not form a boolean algebra
- it is just a “set with operations”
- in other words it can be viewed as an automaton!

A Simpler Logic

- Why did the minimization work with just the logic

$$\phi ::== \underline{\omega} | (a)\phi?$$

- With this logic the definable subsets $E(S)$ do not form a boolean algebra
- it is just a “set with operations”
- in other words it can be viewed as an automaton!

Deterministic vs Nondeterministic Automata

- For deterministic automata we can flatten formulas like $(a)(\omega_1 \wedge (b)\omega_2)$ to $(a)\omega_1 \wedge (a)(b)\omega_2$.
- Thus for **deterministic** automata the boolean algebra generated by $E(S)$ is just the same as $D(S)$ so the minimization picture works with boolean algebra generated by $E(S)$.
- For nondeterministic automata the story is different.

Deterministic vs Nondeterministic Automata

- For deterministic automata we can flatten formulas like $(a)(\omega_1 \wedge (b)\omega_2)$ to $(a)\omega_1 \wedge (a)(b)\omega_2$.
- Thus for **deterministic** automata the boolean algebra generated by $E(S)$ is just the same as $D(S)$ so the minimization picture works with boolean algebra generated by $E(S)$.
- For nondeterministic automata the story is different.

Deterministic vs Nondeterministic Automata

- For deterministic automata we can flatten formulas like $(a)(\omega_1 \wedge (b)\omega_2)$ to $(a)\omega_1 \wedge (a)(b)\omega_2$.
- Thus for **deterministic** automata the boolean algebra generated by $E(S)$ is just the same as $D(S)$ so the minimization picture works with boolean algebra generated by $E(S)$.
- For nondeterministic automata the story is different.

Conclusions

- We need to understand the categorical story for nondeterministic automata, probabilistic automata (weighted automata) and perhaps other things as well.
- We are experimenting with these ideas for use in *approximation* in the RL Lab at McGill; joint with Doina Precup and Joelle Pineau and their students.
- 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?
- Connect with the theory of derivatives and perhaps to on-the-fly algorithms for minimization.

Conclusions

- We need to understand the categorical story for nondeterministic automata, probabilistic automata (weighted automata) and perhaps other things as well.
- We are experimenting with these ideas for use in *approximation* in the RL Lab at McGill; joint with Doina Precup and Joelle Pineau and their students.
- 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?
- Connect with the theory of derivatives and perhaps to on-the-fly algorithms for minimization.

Conclusions

- We need to understand the categorical story for nondeterministic automata, probabilistic automata (weighted automata) and perhaps other things as well.
- We are experimenting with these ideas for use in *approximation* in the RL Lab at McGill; joint with Doina Precup and Joelle Pineau and their students.
- 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?
- Connect with the theory of derivatives and perhaps to on-the-fly algorithms for minimization.

Conclusions

- We need to understand the categorical story for nondeterministic automata, probabilistic automata (weighted automata) and perhaps other things as well.
- We are experimenting with these ideas for use in *approximation* in the RL Lab at McGill; joint with Doina Precup and Joelle Pineau and their students.
- 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?
- Connect with the theory of derivatives and perhaps to on-the-fly algorithms for minimization.

Conclusions

- We need to understand the categorical story for nondeterministic automata, probabilistic automata (weighted automata) and perhaps other things as well.
- We are experimenting with these ideas for use in *approximation* in the RL Lab at McGill; joint with Doina Precup and Joelle Pineau and their students.
- 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?
- Connect with the theory of derivatives and perhaps to on-the-fly algorithms for minimization.