Duality for Transition Systems

Prakash Panangaden[1]

[1]School of Computer Science
McGill University
work done while on sabbatical leave at
Oxford University

Australian Category Seminar: 27th Feb 2013

Opening remarks

- Thanks to Annabelle Mciver for inviting me to Macquarie University

Opening remarks

- Thanks to Annabelle Mciver for inviting me to Macquarie University
- and to the Australian Category Seminar for hosting this talk.

Opening remarks

- Thanks to Annabelle Mciver for inviting me to Macquarie University
- and to the Australian Category Seminar for hosting this talk.
- Thanks to Prof. Mingsheng Ying for inviting me once again to wonderful Sydney, Australia.

Opening remarks

- Thanks to Annabelle Mciver for inviting me to Macquarie University
- and to the Australian Category Seminar for hosting this talk.
- Thanks to Prof. Mingsheng Ying for inviting me once again to wonderful Sydney, Australia.
- Cricket prediction for Australia v India: 1-1.

Overview

- 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.

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.

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. Now also with Helle Hvid Hansen, Alexandra Silva, Jan Rutten, Dexter Kozen, Marcello Bonsangue and Filippo Bonchi.

What is duality?

- Often in mathematics one has two types of structures: Shiv and Vish.

What is duality?

- Often in mathematics one has two types of structures: Shiv and Vish.
- It turns out that every Shiv has an associated Vish and vice versa.

What is duality?

- Often in mathematics one has two types of structures: Shiv and Vish.
- It turns out that every Shiv has an associated Vish and vice versa.
- If one starts with a Vish, construct the associated Shiv and come back one gets “practically” the same Vish that one started with.

What is duality?

- Often in mathematics one has two types of structures: Shiv and Vish.
- It turns out that every Shiv has an associated Vish and vice versa.
- If one starts with a Vish, construct the associated Shiv and come back one gets “practically” the same Vish that one started with.
- This means that these two – apparently – different structures are actually two different descriptions of the same thing.

What is duality?

- Often in mathematics one has two types of structures: Shiv and Vish.
- It turns out that every Shiv has an associated Vish and vice versa.
- If one starts with a Vish, construct the associated Shiv and come back one gets “practically” the same Vish that one started with.
- This means that these two – apparently – different structures are actually two different descriptions of the same thing.
- Thus, one has two completely different sets of theorems that one can use.

Examples of Duality

- Maximum and minimum principles for linear programming.

Examples of Duality

- Maximum and minimum principles for linear programming.
- Boolean algebras and Stone spaces.

Examples of Duality

- Maximum and minimum principles for linear programming.
- Boolean algebras and Stone spaces.
- Logics and Transition systems.

Examples of Duality

- Maximum and minimum principles for linear programming.
- Boolean algebras and Stone spaces.
- Logics and Transition systems.
- State transformer semantics and weakest precondition semantics.

Examples of Duality

- Maximum and minimum principles for linear programming.
- Boolean algebras and Stone spaces.
- Logics and Transition systems.
- State transformer semantics and weakest precondition semantics.
- Measures and random variables.

Examples of Duality

- Maximum and minimum principles for linear programming.
- Boolean algebras and Stone spaces.
- Logics and Transition systems.
- State transformer semantics and weakest precondition semantics.
- Measures and random variables.
- Compact groups and discrete groups.

Examples of Duality

- Maximum and minimum principles for linear programming.
- Boolean algebras and Stone spaces.
- Logics and Transition systems.
- State transformer semantics and weakest precondition semantics.
- Measures and random variables.
- Compact groups and discrete groups.
- C*-algebras and compact Hausdorff spaces.

Examples of Duality

- Maximum and minimum principles for linear programming.
- Boolean algebras and Stone spaces.
- Logics and Transition systems.
- State transformer semantics and weakest precondition semantics.
- Measures and random variables.
- Compact groups and discrete groups.
- C*-algebras and compact Hausdorff spaces.
- Vector spaces and vector spaces.

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**.

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**.

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.

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

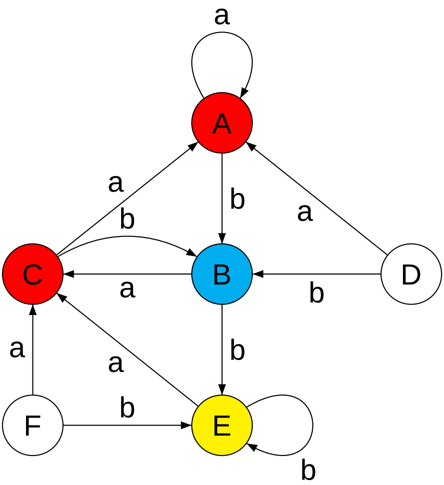
- $\mathcal{M} = (S, \mathcal{A}, \mathcal{O}, \delta, \gamma)$: a deterministic finite automaton. S is the set of **states**, $\mathcal{A}$ an **input alphabet** (actions), $\mathcal{O}$ is a set of **observations**.
- $\delta : S \times \mathcal{A} \longrightarrow S$ is the **state transition function**.
- $\gamma : S \longrightarrow \mathbf{2}^{\mathcal{O}}$ or $\gamma : S \times \mathcal{O} \longrightarrow \mathbf{2}$ is a labeling function.
- If $\mathcal{O} = \{\mathbf{accept}\}$ we have ordinary deterministic finite automata,
- *except* that we do not have a start state,

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,
- *except* that we do not have a start state,
- but this can be easily added to the framework.

An Example

States: $\{A, B, C, D, E, F\}$ Observations: $\{\textbf{Blue}, \textbf{Red}, \textbf{Yellow}\}$.



Testing the Machine

- What can we do with this machine?

Testing the Machine

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

States Satisfy Tests (Or Not)

- **red** is satisfied by $\{A, C\}$

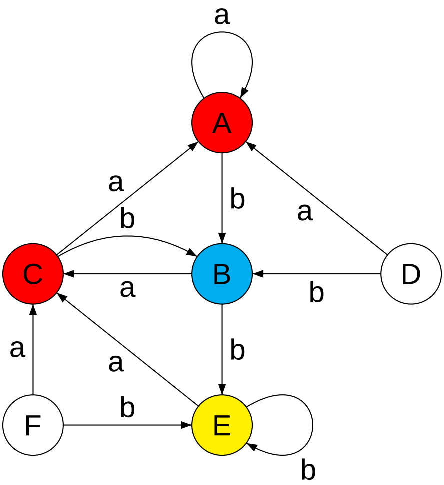
States Satisfy Tests (Or Not)

- **red** is satisfied by $\{A, C\}$
- After b, **yellow** is on, is satisfied by $\{B, E, F\}$ and no other states.

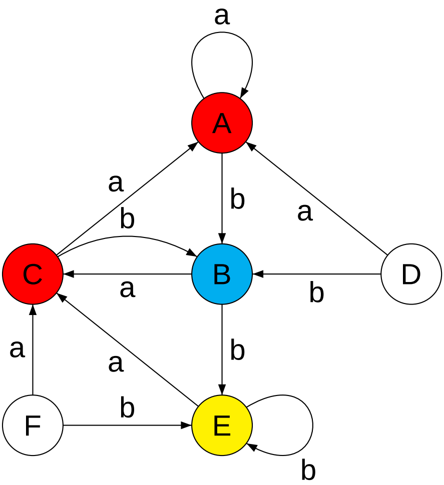
States Satisfy Tests (Or Not)

- **red** is satisfied by $\{A, C\}$
- After b, **yellow** is on, is satisfied by $\{B, E, F\}$ and no other states.
- After $abab$, **blue** is on is satisfied by $\{A, B, C, D, E, F\}$, i.e. by *all* states.

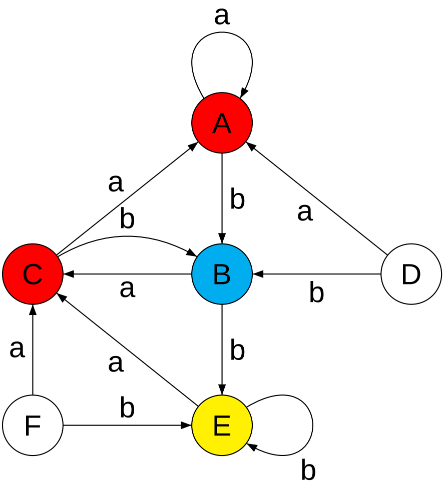
red is satisfied by $\{A, C\}$



b-Yellow is satisfied by $\{B, E, F\}$



abab-Blue is always satisfied



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 $a \in \mathcal{A}$.

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 $\delta(s, a) \models \varphi$.

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 $\delta(s, a)$.

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 $[\![\varphi]\!]_{\mathcal{M}} = [\![\psi]\!]_{\mathcal{M}}$.

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.

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 $bbb\mathbf{yellow}$ and $bbbbb\mathbf{yellow}$ are satisfied by all states. They are thus equivalent.

Examples of Equivalent Formulas

- The formulas $bbb\mathbf{yellow}$ and $bbbbb\mathbf{yellow}$ are satisfied by all states. They are thus equivalent.
- Other equivalent formulas are all formulas of the form $b^m\mathbf{yellow}$ for $m > 1$.

Examples of Equivalent Formulas

- The formulas $bbb\mathbf{yellow}$ and $bbbbb\mathbf{yellow}$ are satisfied by all states. They are thus equivalent.
- Other equivalent formulas are all formulas of the form $b^m\mathbf{yellow}$ for $m > 1$.
- There are a lot of formulas in this equivalence class!

Examples of Equivalent Formulas

- The formulas $bbb\mathbf{yellow}$ and $bbbbb\mathbf{yellow}$ are satisfied by all states. They are thus equivalent.
- Other equivalent formulas are all formulas of the form $b^m\mathbf{yellow}$ for $m > 1$.
- 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 $\mathcal{M}$, $s_1 \models \varphi \iff s_2 \models \varphi$.

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.

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.

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 $\mathcal{M}$.

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:

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:
- $S' = T = \{[\![\varphi]\!]_{\mathcal{M}}\}$

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:
- $S' = T = \{[\![\varphi]\!]_{\mathcal{M}}\}$
- $\mathcal{O}' = S$

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:
- $S' = T = \{[\![\varphi]\!]_{\mathcal{M}}\}$
- $\mathcal{O}' = S$
- $\delta'([\![\varphi]\!]_{\mathcal{M}}, a) = [\![(a)\varphi]\!]_{\mathcal{M}}$

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 $\gamma'([\![\varphi]\!]_{\mathcal{A}}, s) = (s \models \varphi)$.

Let's look at that last line again

- $$\gamma'(\llbracket\varphi\rrbracket_{\mathcal{M}}) = \llbracket\varphi\rrbracket_{\mathcal{M}}?$$

Let’s look at that last line again

- $$\gamma'([\![\varphi]\!]_{\mathcal{M}}) = [\![\varphi]\!]_{\mathcal{M}}?$$
- Does it make sense? Is γ' just the identity?

Let's look at that last line again

- $$\gamma'([\![\varphi]\!]_{\mathcal{M}}) = [\![\varphi]\!]_{\mathcal{M}}?$$
- Does it make sense? Is γ' just the identity?
- On the left-hand side $[\![\varphi]\!]$ is an equivalence class of formulas, hence a state of the dual machine;

Let's look at that last line again

- $$\gamma'([\![\varphi]\!]_{\mathcal{M}}) = [\![\varphi]\!]_{\mathcal{M}}?$$
- Does it make sense? Is γ' just the identity?
- On the left-hand side $[\![\varphi]\!]$ is an equivalence class of formulas, hence a state of the dual machine;
- so the right-hand side ought to have a set of observations of the dual machine,

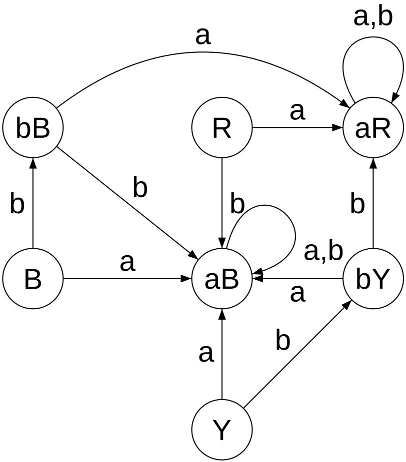
Let's look at that last line again

- $$\gamma'([\![\varphi]\!]_{\mathcal{M}}) = [\![\varphi]\!]_{\mathcal{M}}?$$
- Does it make sense? Is γ' just the identity?
- On the left-hand side $[\![\varphi]\!]$ is an equivalence class of formulas, hence a state of the dual machine;
- so the right-hand side ought to have a set of observations of the dual machine,
- but that is just a set of states of the original machine!

The intuition

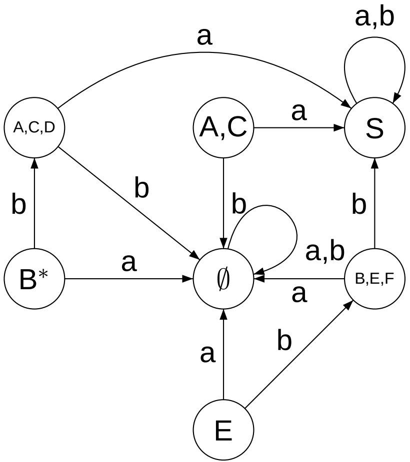
We have interchanged the states and the observations or propositions; more precisely we have interchanged equivalence classes of formulas - based on the observations - with the states. We have made the states of the old machine the observations of the dual machine.

The Dual Machine For Our Example



Note that $aB \sim abY \sim bR \sim$ **false** and that $aR \sim bbY \sim abB \sim$ **true**.

The Dual Machine Labelled with Observations (aka States)



$*$: This means the state B, not the colour Blue!
S stands for the set of all states.

Some Remarks

- The dual machine has more states than the primal machine.

Some Remarks

- The dual machine has more states than the primal machine.
- The dual machine could have at most $2^{|S|}$ states.

Some Remarks

- The dual machine has more states than the primal machine.
- The dual machine could have at most $2^{|S|}$ states.
- Not every possible set of states is the denotation of some formula.

Some Remarks

- The dual machine has more states than the primal machine.
- The dual machine could have at most $2^{|S|}$ states.
- Not every possible set of states is the denotation of some formula.
- If it were the case that every possible set of states is described by some formula then we would indeed have exponential blow up in the size.

Some Remarks

- The dual machine has more states than the primal machine.
- The dual machine could have at most $2^{|S|}$ states.
- Not every possible set of states is the denotation of some formula.
- If it were the case that every possible set of states is described by some formula then we would indeed have exponential blow up in the size.
- If we had a richer logic then more sets of states would be definable.

The Double Dual

- Now consider $\mathcal{M}'' = (\mathcal{M}')'$, the dual of the dual.

The Double Dual

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

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

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.

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

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

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 $\mathcal{M}''$.

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.

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 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 $\mathcal{M}$ is reduced then Sat is a bijection from S to S''.

Minimality Properties

- If $\mathcal{M}$ 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.

Minimality Properties

- If $\mathcal{M}$ 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.
- In general, the double dual is the minimal machine with the same behaviour!

Minimality Properties

- If $\mathcal{M}$ 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.
- In general, the double dual is the minimal machine with the same behaviour!
- For deterministic machines bisimulation is the same as trace equivalence so there is no question about what equivalence we have in mind.

Brzozowski's Algorithm 1962

- Take a NFA and just reverse all the transitions 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.

Brzozowski's Algorithm 1962

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

Brzozowski's Algorithm 1962

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

Brzozowski's Algorithm 1962

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

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

The τ_a are transition probability functions (matrices).

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.

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

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

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.

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.

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

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

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*

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 $(a)\varphi$.

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.

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:

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

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

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

- $S' = \{[\![t]\!]_{\mathcal{E}}\}$

Duality with e-tests

- $S' = \{[\![t]\!]_{\mathcal{E}}\}$
- $\mathcal{O}' = S$

Duality with e-tests

- $S' = \{[\![t]\!]_{\mathcal{E}}\}$
- $\mathcal{O}' = S$
- $\gamma'([\![t]\!]_{\mathcal{E}}, s) = [\![t]\!]_{\mathcal{E}}(s)$

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.

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.

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

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

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.
- $$[\![o_1 a_1 o_2 a_2 o_3]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}) =$$
  $$[\![o_1]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}''}) \cdot [\![a_1 o_2]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}) \cdot [\![a_1 a_2 o_3]\!]_{\mathcal{E}''}([\![s]\!]_{\mathcal{E}'}).$$

  This does not hold in the primal.

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

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

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 $n \geq 1$.

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.

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

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.

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 $\langle s|e|s'\rangle$.

Notation continued

- 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

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

Equivalence on Experiments

- For experiments e_1, e_2, we say

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

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

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

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 $\mathcal{M}' =$

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

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 $S' = \{[e]_{\mathcal{M}}\}$, $\mathcal{O}' = S$

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

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

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

- 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

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
- $S'' = \{[t]_{\mathcal{M}'}\}$, $\mathcal{O}'' = S'$,

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

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 can use experiments to construct the dual,

The Main Theorem

- One can use experiments to construct the dual,
- this gives a DASO. Now we can use e-tests to construct the double dual.

The Main Theorem

- One can use experiments to construct the dual,
- this gives a DASO. Now we can use e-tests to construct the double dual.
- 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.

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.

What is the right categorical description?

- Is this is some kind of familiar Stone-type duality?

What is the right categorical description?

- Is this is some kind of familiar Stone-type duality?
- We know that machines are co-algebras and logics are algebras but

What is the right categorical description?

- Is this is some 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:

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

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

Homomorphisms

A homomorphism for these coalgebras is a function $f : S \longrightarrow S'$ 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** ($\mathbf{FBAO}$) has as objects finite boolean algebras B with

The Dual Category

- The category of **finite boolean algebras with operators** ($\mathbf{FBAO}$) has as objects finite boolean algebras B with
- the usual operations $\wedge$, $\neg$ and constants T and $\perp$ and, in addition,

The Dual Category

- The category of **finite boolean algebras with operators** ($\mathbf{FBAO}$) 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 $\underline{\omega}$.

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.

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

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

- $\mathcal{D}(S)$ 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

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

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

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

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

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

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

The Secret of Minimization

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

A Simpler Logic

- Why did the minimization work with just the logic

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

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

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”

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 $(a)\omega_1 \wedge (a)(b)\omega_2$.

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 $E(S)$.

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.

Weighted Automata

- These are automata where the state space is a vector space and the transitions are given by matrices.

Weighted Automata

- These are automata where the state space is a vector space and the transitions are given by matrices.
- Stefan Kiefer came up with a beautiful backwards-and-forwards minimization algorithm after hearing my original talk on this last autumn.

Weighted Automata

- These are automata where the state space is a vector space and the transitions are given by matrices.
- Stefan Kiefer came up with a beautiful backwards-and-forwards minimization algorithm after hearing my original talk on this last autumn.
- Recently, Nick Bezhanishvili, Clemens Kupke and I showed that this construction is a beautiful example of our categorical picture

Weighted Automata

- These are automata where the state space is a vector space and the transitions are given by matrices.
- Stefan Kiefer came up with a beautiful backwards-and-forwards minimization algorithm after hearing my original talk on this last autumn.
- Recently, Nick Bezhanishvili, Clemens Kupke and I showed that this construction is a beautiful example of our categorical picture
- exploiting the fact that the category of vector spaces is self dual.

Probabilistic Automata

- With probabilistic automata one can define an associated deterimnistic automaton where the states are probability distributions (belief automata).

Probabilistic Automata

- With probabilistic automata one can define an associated deterimnistic automaton where the states are probability distributions (belief automata).
- These are compact Hausdorff spaces with *transitions* and *observations*.

Probabilistic Automata

- With probabilistic automata one can define an associated deterimnistic automaton where the states are probability distributions (belief automata).
- These are compact Hausdorff spaces with *transitions* and *observations*.
- The dual is a C* algebra with operations.

Probabilistic Automata

- With probabilistic automata one can define an associated deterimnistic automaton where the states are probability distributions (belief automata).
- These are compact Hausdorff spaces with *transitions* and *observations*.
- The dual is a C* algebra with operations.
- The same picture applies.

Diagram of an automaton

$$\begin{tikzcd} 1 \arrow[dr, "i"] & & 2 \\ & S \arrow[ur, "f"] \arrow[d, "\delta"] & \\ & S^A & \end{tikzcd}$$

S is the state space
δ is the transition function
f defines the final states.

The butterfly

$$
\begin{tikzcd}
1 \arrow[d, "\varepsilon"'] \arrow[dr, "i"] & & 2 \\
A^* \arrow[r, dashed, "r"] \arrow[d, "\alpha"'] & S \arrow[ur, "f"] \arrow[r, dashed, "o"] \arrow[d, "\delta"] & 2^{A^*} \arrow[u, "\varepsilon?"'] \arrow[d, "\beta"] \\
(A^*)^A \arrow[r, dashed, "r^A"'] & S^A \arrow[r, dashed, "o^A"'] & (2^{A^*})^A
\end{tikzcd}
$$

Left: Automaton of words (initial)

$$\alpha\colon A^* \longrightarrow (A^*)^A \qquad \alpha(w)(a) = w \cdot a$$

Right: Automaton of languages (terminal)

$$\beta\colon 2^{A^*} \longrightarrow (2^{A^*})^A \qquad \beta(L)(a) = \{w \in A^* \mid a \cdot w \in L\}$$

r defines reachability; o defines observability.

Reachability and observability

A deterministic automaton (S, δ, i, f) is *reachable* if r is surjective, it is *observable* if o is injective, and it is *minimal* if it is both reachable and observable.

Contravariant power set functor

$$2^{(-)} \; : \quad \begin{array}{c} V \\ {\scriptstyle f}\big\downarrow \\ W \end{array} \quad \mapsto \quad \begin{array}{c} 2^V \\ \big\uparrow{\scriptstyle 2^f} \\ 2^W \end{array}$$

which is defined, for a set V, by $2^V = \{S \mid S \subseteq V\}$ and, for $f : V \longrightarrow W$ and $S \subseteq W$, by

$$2^f \; : \; 2^W \longrightarrow 2^V \qquad 2^f(S) = \{v \in V \mid f(v) \in S\}$$

Reversing

$$
\begin{array}{c||c|c||c}
S & A \times A & 2^{S\times A} & (2^S)^A \\
\Big\downarrow{\scriptstyle \delta} & \Big\downarrow & \Big\uparrow & \Big\uparrow{\scriptstyle 2^{\delta}} \\
S^A & S & 2^S & 2^S
\end{array}
$$

Initial becomes final

Applying the operation $2^{(-)}$ to the initial state (function) of our automaton gives

$$\begin{array}{c|c} 1 & 2 \\ \downarrow{\scriptstyle i} & \uparrow{\scriptstyle 2^i} \\ X & 2^S \end{array}$$

Reachable becomes observable - I

Apply the powerset functor:

$$\begin{tikzcd}
 & 2 \\
2^S \arrow[ur, "2^i"] \arrow[r, "2^r"] \arrow[d, "2^\delta"'] & 2^{A^*} \arrow[u, "2^\varepsilon"'] \arrow[d, "2^\alpha"] \\
(2^S)^A \arrow[r, "2^{r^A}"'] & (2^{A^*})^A
\end{tikzcd}$$

For any $L \in 2^{A^*}$, we have $2^\varepsilon(L) = \varepsilon?(L)$ and, for any $a \in A$,

$$2^\alpha(L)(a) = \{w \in A^* \mid w \cdot a \in L\}$$

Like $\beta(L)(a)$ but it uses $w \cdot a$ instead of $a \cdot w$.

Reachable becomes observable - II

By finality of $(2^{A^*}, \beta, \varepsilon?)$, there exists a unique homomorphism $rev\colon 2^{A^*} \to 2^{A^*}$

$$\begin{tikzcd}
 & & 2 \\
2^{A^*} \arrow[rru, "2^{\varepsilon}"] \arrow[rr, dashed, "rev"] \arrow[d, "2^{\alpha}"'] & & 2^{A^*} \arrow[u, "\varepsilon?"'] \arrow[d, "\beta"] \\
(2^{A^*})^A \arrow[rr, dashed, "rev^A"'] & & (2^{A^*})^A
\end{tikzcd}$$

which sends a language L to its reverse

$$rev(L) = \{w \in A^* \mid w^R \in L\}$$

where w^R is the reverse of w.

Reachable becomes observable - III

Combining diagrams yields:

$$\begin{tikzcd}
 & & 2 \\
2^S \arrow[rru, "2^i"] \arrow[r, "2^r"'] \arrow[d, "2^\delta"'] & 2^{A^*} \arrow[ru, "2^\varepsilon"] \arrow[r, "rev"'] \arrow[d, "2^\alpha"] & 2^{A^*} \arrow[u, "\varepsilon?"'] \arrow[d, "\beta"] \\
(2^S)^A \arrow[r, "2^{r^A}"'] & (2^{A^*})^A \arrow[r, "rev^A"'] & (2^{A^*})^A
\end{tikzcd}$$

Reachable becomes observable - IV

Thus the composition of rev and 2^r is the unique function that makes the following diagram commute:

$$\begin{tikzcd} & 2 \\ 2^S \arrow[ur, "2^i"] \arrow[r, dashed, "O"] \arrow[d, "2^\delta"'] & 2^{A^*} \arrow[u, "\varepsilon?"'] \arrow[d, "\beta"] \\ (2^S)^A \arrow[r, dashed, "O^A"'] & (2^{A^*})^A \end{tikzcd}$$

$$O = rev \circ 2^r$$

One can easily show that it satisfies, for any $X \subseteq S$,

$$O(X) = \{w^R \in A^* \mid i_w \in X\}$$

Final becomes initial

$$\begin{array}{c||c} 2 & 1 \\ f \big\uparrow & \big\downarrow f \\ S & 2^S \end{array}$$

Putting everything together

We have obtained the following, new deterministic automaton:

$$
\begin{tikzcd}
1 \arrow[d, "\varepsilon"'] \arrow[dr, "f"] & & 2 \\
A^* \arrow[r, dashed, "R"'] \arrow[d, "\alpha"'] & 2^X \arrow[r, dashed, "O"'] \arrow[d, "2^t"] \arrow[ur, "2^i"] & 2^{A^*} \arrow[u, "\varepsilon?"'] \arrow[d, "\beta"] \\
(A^*)^A \arrow[r, dashed, "R^A"'] & (2^X)^A \arrow[r, dashed, "O^A"'] & (2^{A^*})^A
\end{tikzcd}
$$

The main theorem

Let (S, δ, i, f) be a deterministic automaton and let $(2^S, 2^\delta, f, 2^i)$ be the reversed deterministic automaton constructed as above.

1. If (S, δ, i, f) is reachable, then $(2^S, 2^\delta, f, 2^i)$ is observable.

The main theorem

Let (S, δ, i, f) be a deterministic automaton and let $(2^S, 2^\delta, f, 2^i)$ be the reversed deterministic automaton constructed as above.

1. If (S, δ, i, f) is reachable, then $(2^S, 2^\delta, f, 2^i)$ is observable.
2. If (S, δ, i, f) accepts the language L, then $(2^S, 2^\delta, f, 2^i)$ accepts $rev(L)$.

Conclusions

- 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.

Conclusions

- 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.

Conclusions

- 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?