Redoing the Foundations of Decision Theory

Joe Halpern

Cornell University

Joint work with Larry Blume and David Easley, Economics, Cornell

Decision Making: Savage's Approac

Savage's approach to decision making has dominated decision theory since the 1950's. It assumes that a decision maker (DM) is given/has

- a set S of states
- a set O of outcomes

A (Savage) *act* is a function from states to outcomes.

Decision Making: Savage's Approac

Savage's approach to decision making has dominated decision theory since the 1950's. It assumes that a decision maker (DM) is given/has

- a set S of states
- a set O of outcomes

A (Savage) *act* is a function from states to outcomes.

Example: Betting on a horse race.

- S = possible orders of finish
- O = how much you win
- act = bet

Savage's Theorem

Savage assumes that a DM has a preference order $\succeq$ on acts satisfying certain postulates:

- E.g. transitivity: if $a_1 \succeq a_2$ and $a_2 \succeq a_3$, then $a_1 \succeq a_3$.

He proves that if a DM's preference order satisfies these postulates, then the DM is acting as if

- he has a probability $\Pr$ on states
- he has a utility function u on outcomes
- he is maximizing expected utility:
  - $a \succeq b$ iff $E_{\Pr}[u_a] \geq E_{\Pr}[u_b]$.
  - $u_a(s) = u(a(s))$: the utility of act a in state s

Are Savage Acts Reasonable?

Many problems have been pointed out with Savage's framework. We focus on one:

- In a complex environment, can a DM completely specify the state space or the outcome space?
  - What are the states/outcomes if we're trying to decide whether to attack Iraq?
- What are the acts if we can't specify the state/outcome space?

Acts as Programs

Claim: people don’t think of acts as functions:

- We don’t think of the state space and the outcomes when we contemplate the act “Buy 100 shares of IBM”!

Acts as Programs

Claim: people don't think of acts as functions:

- We don't think of the state space and the outcomes when we contemplate the act "Buy 100 shares of IBM"!

An alternative: instead of taking acts to be functions from states to outcomes, we take acts to be syntactic objects

- essentially, acts are *programs* that the DM can run.
- Call the stock broker, place the order, . . .

Acts as Programs

Claim: people don't think of acts as functions:

- We don't think of the state space and the outcomes when we contemplate the act "Buy 100 shares of IBM"!

An alternative: instead of taking acts to be functions from states to outcomes, we take acts to be syntactic objects

- essentially, acts are *programs* that the DM can run.
- Call the stock broker, place the order, . . .

But if acts are programs:

- What is the programming language?
- What is the semantics of a program?

The Programming Language

In this talk, we consider only one programming construct:

- **if . . . then . . . else**
  - If a_1 and a_2 are programs, and t is a test, then

    if t **then** a_1 **else** a_2 is a program
  - **if** moon in seventh house **then** buy 100 shares IBM

In the full paper we also consider randomization:

- With probability r perform a_1; with probability $1 - r$, perform a_2

The Programming Language

In this talk, we consider only one programming construct:

- **if . . . then . . . else**
  - If a_1 and a_2 are programs, and t is a test, then

    if t **then** a_1 **else** a_2 is a program
  - **if** moon in seventh house **then** buy 100 shares IBM

In the full paper we also consider randomization:

- With probability r perform a_1; with probability $1 - r$, perform a_2

Key point:

- We need to specify the language of tests
- The modeler and the DM may use different languages

Programming Language: Syntax

Start with

- a set $\mathcal{A}_0$ of primitive acts
  - Buy 100 shares of IBM
  - Attack Iraq
- A set $\mathcal{T}_0$ of primitive tests (propositions)
  - The price/earnings ratio is at least 7
  - The moon is in the seventh house

Form the set $\mathcal{T}$ of tests by closing off under conjunction and negation:

- tests are just propositional formulas

Form the set $\mathcal{A}$ of acts by closing off under **if** . . . **then** . . . **else**

Programming Language Semantics

What should a program *mean*?

In this paper, we consider *input-output* semantics:

- A program defines a function from states to outcomes
  - once we are given a state space and an outcome space, a program determines a Savage act
- The state and outcome spaces are now subjective.
  - Different agents can model them differently

Semantics: Formal Details

Given a state space S and an outcome space O, we want to view a program as a function from S to O. We first need

- a *program interpretation* ρ_{SO} that associates with each primitive program in $\mathcal{A}_0$ a function from S to O
- a *test interpretation* π_S that associates with each primitive proposition in T_0 an event (a subset of S)

Then can extend ρ_{SO} to a function that associates with each program in $\mathcal{A}$ a function from S to O:

$$\rho_{SO}(\textbf{if } t \textbf{ then } a_1 \textbf{ else } a_2)(s) = \begin{cases} \rho_{SO}(a_1)(s) \text{ if } s \in \pi_S(t) \\ \rho_{SO}(a_2)(s) \text{ if } s \notin \pi_S(t) \end{cases}$$

Where We're Headed

We prove the following type of theorem:

If a DM has a preference order on programs satisfying appropriate postulates, then there exist

- a state space S,
- a probability $\Pr$ on S,
- an outcome space O,
- a utility function u on O,
- a program interpretation ρ_{SO},
- a test interpretation π_S

such that $a \succeq b$ iff $E_{\Pr}[u_{\rho_{SO}(a)}] \geq E_{\Pr}[u_{\rho_{SO}(b)}]$.

- This is a Savage-like result
  - The postulates are variants of standard postulates
  - The DM has to put a preference order only on “reasonable” acts

But now S and O are subjective, just like $\Pr$ and u!

- S, O, $\Pr$, u, ρ_{SO}, and π_S are all in the DM’s head
- S and O are not part of the description of the problem

The Benefits of the Approach

We have replaced Savage acts by programs and prove Savage-type theorems. So what have we gained?

The Benefits of the Approach

We have replaced Savage acts by programs and prove Savage-type theorems. So what have we gained?

- Acts are easier for a DM to contemplate
  - No need to construct a state space/outcome space
  - Just think about what you can do

The Benefits of the Approach

We have replaced Savage acts by programs and prove Savage-type theorems. So what have we gained?

- Acts are easier for a DM to contemplate
  - No need to construct a state space/outcome space
  - Just think about what you can do
- Different agents can have different conceptions of the world
  - You might make decision on stock trading based on price/earnings ratio
  - I might use astrology (and might not even understand the notion of p/e ratio)

- “Agreeing to disagree” results [Aumann] (which assume a common state space) disappear

- “Agreeing to disagree” results [Aumann] (which assume a common state space) disappear
- (Un)awareness becomes particularly important

- “Agreeing to disagree” results [Aumann] (which assume a common state space) disappear
- (Un)awareness becomes particularly important
- Can deal with unanticipated events, novel concepts:
  - Updating $\neq$ conditioning

- "Agreeing to disagree" results [Aumann] (which assume a common state space) disappear
- (Un)awareness becomes particularly important
- Can deal with unanticipated events, novel concepts:
  - Updating $\neq$ conditioning
- We do not have to identify two acts that act the same as functions
  - Can capture framing effects: With 600 people, can treat "200 people die" and "400 people live" as different outcomes
  - Can capture resource-bounded reasoning (agent can't tell two acts are equivalent)

The Cancellation Postulate

Back to the Savage framework:

Cancellation Postulate: Given two sequences $\langle a_1, \ldots, a_n \rangle$ and $\langle b_1, \ldots, b_n \rangle$ of acts, suppose that for each state $s \in S$

$$\{\{a_1(s), \ldots, a_n(s)\}\} = \{\{b_1(s), \ldots, b_n(s)\}\}.$$

- $\{\{o, o, o, o', o'\}\}$ is a *multiset*

If $a_i \succeq b_i$ for $i = 1, \ldots, n-1$, then $b_n \succeq a_n$.

Cancellation is surprising powerful. It implies

- Reflexivity
- Transitivity:
  - Suppose $a \succeq b$ and $b \succeq c$. Take $\langle a_1, a_2, a_3 \rangle = \langle a, b, c \rangle$ and $\langle b_1, b_2, b_3 \rangle = \langle b, c, a \rangle$.
- Event independence:
  - Suppose that $T \subseteq S$ and $f_T g \succeq f'_T g$
    - $f_T g$ is the act that agrees with f on T and g on $S - T$.
  - Take $\langle a_1, a_2 \rangle = \langle f_T g, f'_T g' \rangle$ and $\langle b_1, b_2 \rangle = \langle f'_T g, f_T g' \rangle$.
  - Conclusion: $f_T g' \succeq f'_T g'$

Cancellation in Our Framework

A program maps truth assignments to primitive programs:

- E.g., consider **if** t **then** a_1 **else** (**if** t' **then** a_2 **else** a_3):
  - $t \wedge t' \longrightarrow a_1$
  - $t \wedge \neg t' \longrightarrow a_1$
  - $\neg t \wedge t' \longrightarrow a_2$
  - $\neg t \wedge \neg t' \longrightarrow a_3$

Similarly for every program.

Cancellation in Our Framework

A program maps truth assignments to primitive programs:

- E.g., consider **if** t **then** a_1 **else** (**if** t' **then** a_2 **else** a_3):
  - $t \wedge t' \longrightarrow a_1$
  - $t \wedge \neg t' \longrightarrow a_1$
  - $\neg t \wedge t' \longrightarrow a_2$
  - $\neg t \wedge \neg t' \longrightarrow a_3$

Similarly for every program.

Can rewrite the cancellation postulate using programs:

- replace “outcomes” by “primitive programs”
- replace “states” by “truth assignments”

The Main Result

Theorem: Given a preference orders $\succeq$ on acts satisfying Cancellation, there exist

- a set S of states and a set $\mathcal{P}$ of probability measures on S,
- a set O of outcomes and a utility function u on O,
- a program interpretation ρ_{SO},
- a test interpretation π_S

such that

$$a \succeq b \text{ iff } E_{\Pr}[u_a] \geq E_{\Pr}[u_b] \text{ for all } \Pr \in \mathcal{P}.$$

Moreover, if $\succeq$ is totally ordered, then $\mathcal{P}$ can be taken to be a singleton.

Uniqueness

Savage gets uniqueness; we don't:

- In the totally ordered case, S can be taken to be a subset of the set of truth assignments.
- Not in the partially ordered case:
  - Even with no primitive propositions, if primitive programs a and b are incomparable, need two states, two outcomes, and two probability measures to represent this.
- Can't hope to have a unique probability measure on S, even in the totally ordered case: there aren't enough acts.
  - Savage's postulates force uncountably many acts

Program Equivalence

When are two programs *equivalent*?

- That depends on the choice of semantics
- With input-output semantics, two programs are equivalent if they determine the same functions *no matter what* S, O, π_S, ρ_{SO} are.

Program Equivalence

When are two programs *equivalent*?

- That depends on the choice of semantics
- With input-output semantics, two programs are equivalent if they determine the same functions *no matter what* S, O, π_S, ρ_{SO} are.

Example 1: (**if** t **then** a **else** b) $\equiv$ (**if** $\neg t$ **then** b **else** a).

- These programs determine the same functions, no matter how t, a, and b are interpreted.

Program Equivalence

When are two programs *equivalent*?

- That depends on the choice of semantics
- With input-output semantics, two programs are equivalent if they determine the same functions *no matter what* S, O, π_S, ρ_{SO} are.

Example 1: (**if** t **then** a **else** b) $\equiv$ (**if** $\neg t$ **then** b **else** a).

- These programs determine the same functions, no matter how t, a, and b are interpreted.

Example 2: If $t \equiv t'$, then

(**if** t **then** a **else** b) $\equiv$ (**if** t' **then** a **else** b).

Cancellation and Equivalence

Testing equivalence of propositional formulas is hard

- co-NP complete, even for this simple programming language
- Have to check propositional equivalence

Cancellation implies a DM is indifferent beteween equivalent programs.

Lemma: Cancellation $\Rightarrow$ if $a \equiv b$, then $a \sim b$.

Cancellation and Equivalence

Testing equivalence of propositional formulas is hard

- co-NP complete, even for this simple programming language
- Have to check propositional equivalence

Cancellation implies a DM is indifferent beteween equivalent programs.

Lemma: Cancellation $\Rightarrow$ if $a \equiv b$, then $a \sim b$.

- Cancellation requires smart decision makers!
- We don't have to require cancellation
  - Can consider more resource-bounded DM's

Conclusions

The theorems we have proved show only that this approach generalizes the classic Savage approach.

- The really interesting steps are now to use the approach to deal with issues that the classical approach can't deal with
  - conditioning on unanticipated events
  - (un)awareness
    - See other KR talk; AAMAS paper
  - . . .