Formalizing reachability, viability and avoidability in the context of sequential decision problems

Nicola Botta

Outline

- Why formalizing what?
- Minimal goals
- Sequential decision problems
- Reachability, viability and avoidability
- Decision procedures

Why formalizing what?

- *International emissions trading: Good or bad?*, Holtsmark & Sommervoll, 2012: "[. . .] we find that an agreement with international emissions trading leads to increased emissions and reduced efficiency."
- *The case for international emission trade in the absence of cooperative climate policy*, Carbone et al., 2009: "[. . .] we find that emission trade agreements can be effective [. . .]"

Why formalizing what?

- *International emissions trading: Good or bad?*, Holtsmark & Sommervoll, 2012: "[. . .] we find that an agreement with international emissions trading leads to increased emissions and reduced efficiency."
- *The case for international emission trade in the absence of cooperative climate policy*, Carbone et al., 2009: "[. . .] we find that emission trade agreements can be effective [. . .]"

Why formalizing what?

- *International emissions trading: Good or bad?*, Holtsmark & Sommervoll, 2012: "[. . .] we find that an agreement with international emissions trading leads to increased emissions and reduced efficiency."
- *The case for international emission trade in the absence of cooperative climate policy*, Carbone et al., 2009: "[. . .] we find that emission trade agreements can be effective [. . .]"

Why formalizing what?

- *International emissions trading: Good or bad?*, Holtsmark & Sommervoll, 2012: "[. . .] we find that an agreement with international emissions trading leads to increased emissions and reduced efficiency."
- *The case for international emission trade in the absence of cooperative climate policy*, Carbone et al., 2009: "[. . .] we find that emission trade agreements can be effective [. . .]"
- *Confronting Climate Change: Avoiding the Unmanageable and Managing the Unavoidable*, P. Raven, R. Bierbaum, J. Holdren, UN-Sigma Xi Climate Change Report, 2007.

Why formalizing what?

Some notion of avoidability is implicit in the WG3_IPCC_AR5_2014 definitions of

- *mitigation*: "A human intervention to reduce the sources or enhance the sinks of greenhouse gases"

Why formalizing what?

Some notion of avoidability is implicit in the WG3_IPCC_AR5_2014 definitions of

- *mitigation*: "A human intervention to reduce the sources or enhance the sinks of greenhouse gases" ⇒ avoid high atmospheric GHG concentrations

Why formalizing what?

Some notion of avoidability is implicit in the WG3_IPCC_AR5_2014 definitions of

- *mitigation*: "A human intervention to reduce the sources or enhance the sinks of greenhouse gases" ⇒ avoid high atmospheric GHG concentrations
- *adaptation*: "The process of adjustment to actual or expected climate and its effects . . . to moderate harm or exploit beneficial opportunities"

Why formalizing what?

Some notion of avoidability is implicit in the WG3_IPCC_AR5_2014 definitions of

- *mitigation*: "A human intervention to reduce the sources or enhance the sinks of greenhouse gases" ⇒ avoid high atmospheric GHG concentrations
- *adaptation*: "The process of adjustment to actual or expected climate and its effects ... to moderate harm or exploit beneficial opportunities" ⇒ avoid too much harm, realize beneficial opportunities

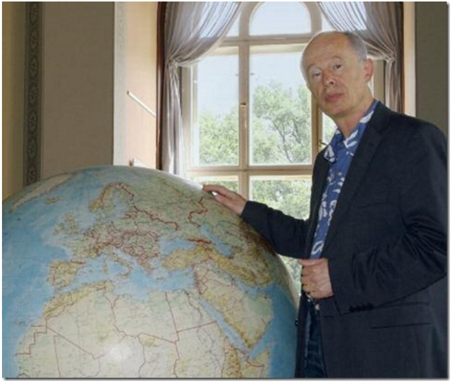
Why formalizing what?

Some notion of avoidability is implicit in the WG3_IPCC_AR5_2014 definitions of

- *mitigation*: "A human intervention to reduce the sources or enhance the sinks of greenhouse gases" ⇒ avoid high atmospheric GHG concentrations
- *adaptation*: "The process of adjustment to actual or expected climate and its effects . . . to moderate harm or exploit beneficial opportunities" ⇒ avoid too much harm, realize beneficial opportunities

But what does it mean (for atmospheric GHG concentrations) to be *avoidable*?

Why formalizing what?



"Die Rolle der Klimaforschung bleibt weiterhin, die Problemfakten auf den Tisch zu knallen und Optionen für geeignete Lösungswege zu identifizieren."

H.-J. Schellnhuber in *Frankfurter Allgemeine* from 2012-06-19

Why formalizing what?

But how can we produce "hard facts" if the notions used to phrase specific, concrete problems are ambiguous, devoid of precise, well established, meanings?

Minimal goals

Minimal goals

- Explain what it means for future (possibly harmful) states to be avoidable [reachable, viable, . . .]

Minimal goals

- Explain what it means for future (possibly harmful) states to be avoidable [reachable, viable, ...]
- Explain under which conditions is it *decidable* whether future states are avoidable [reachable, viable, ...] or not

Minimal goals

- Explain what it means for future (possibly harmful) states to be avoidable [reachable, viable, . . .]
- Explain under which conditions is it *decidable* whether future states are avoidable [reachable, viable, . . .] or not

Further questions, goals

Minimal goals

- Explain what it means for future (possibly harmful) states to be avoidable [reachable, viable, ...]
- Explain under which conditions is it *decidable* whether future states are avoidable [reachable, viable, ...] or not

Further questions, goals

- Can one exploit decidability to derive useful avoidability (levity?) measures?

Minimal goals

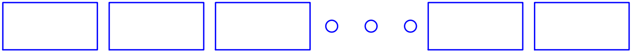
- Explain what it means for future (possibly harmful) states to be avoidable [reachable, viable, ...]
- Explain under which conditions is it *decidable* whether future states are avoidable [reachable, viable, ...] or not

Further questions, goals

- Can one exploit decidability to derive useful avoidability (levity?) measures?
- Can one refine decidable notions of viability, avoidability to derive operational notions (measures?) of sustainability, adaptability, resilience?

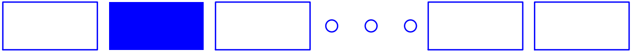
Sequential decision problems (intuition)

Sequential decision problems (intuition)



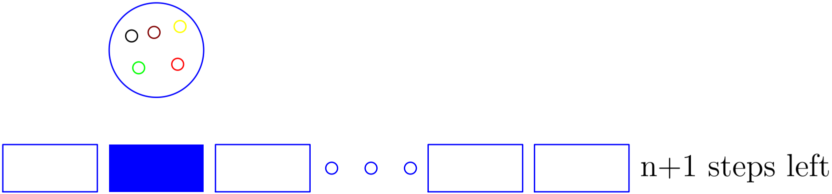
n+1 steps left

You are here. . .

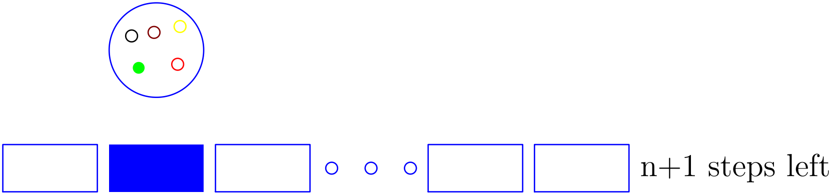


n+1 steps left

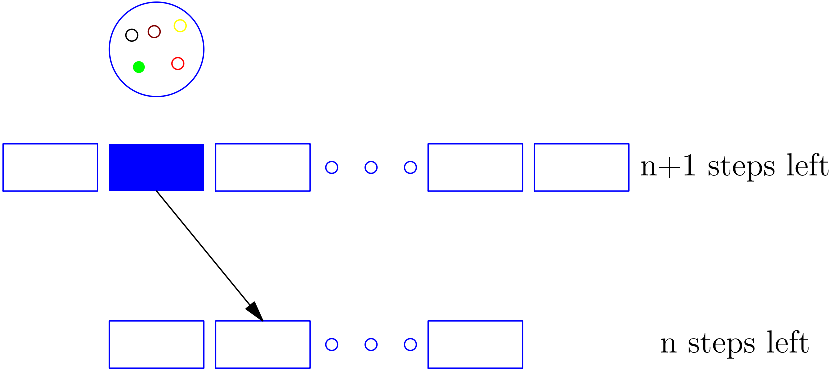
These are your options. . .



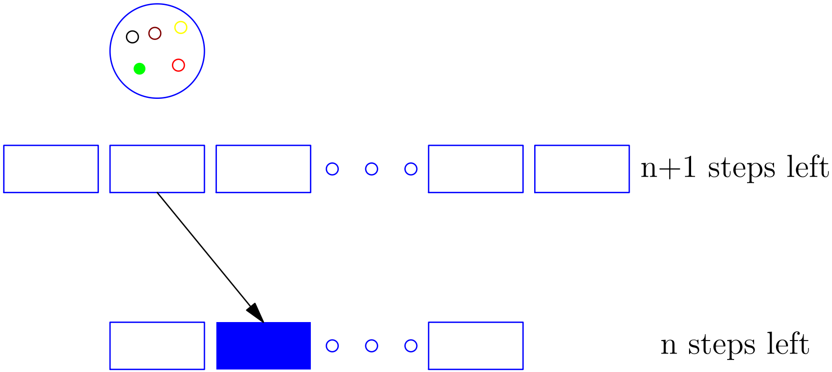
Pick one!



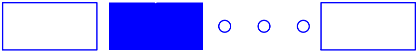
Advance one step...



. . . collect rewards . . .



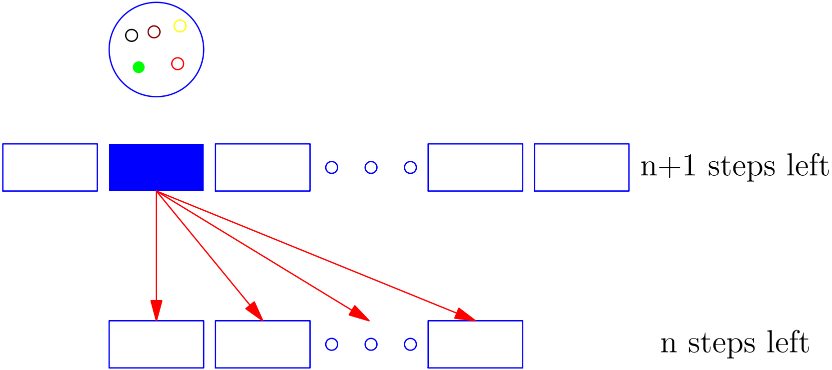
. . . and go!



n steps left

General sequential decision problems (intuition)

General sequential decision problems (intuition)

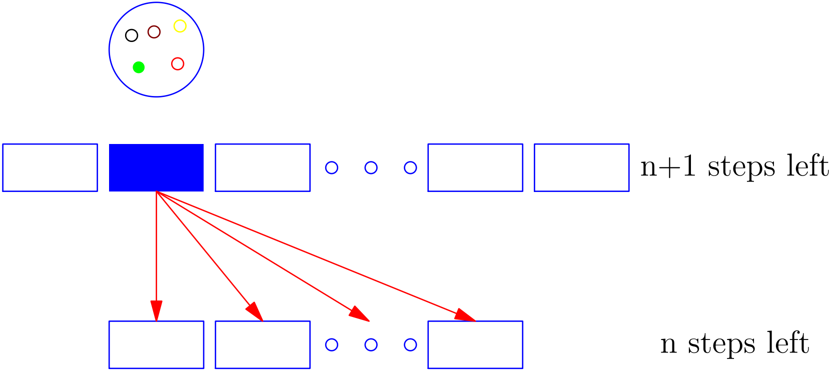


Sequential decision problems (notation)

Idris	Logic
A : $Type$	A is a set
a : A	$a \in A$
P : $Type$	P is a predicate
p : P	p is a proof of P
$FALSE$ (empty type)	False
non-empty type	True
$P \rightarrow Q$	P implies Q
$(a : A ** P\ a)$	there exists an $a \in A$ such that $P\ a$ holds
$(a : A) \rightarrow P\ a$	forall $a \in A$, $P\ a$ holds

Figure: Curry-Howard correspondence relating Idris and logic.

Sequential decision problems (basic ideas)



Sequential decision problems (basic ideas)

At each decision step, a set of possible states:

$$X : (t : \mathbb{N}) \rightarrow Type$$

Sequential decision problems (basic ideas)

At each decision step, a set of possible states:

$$X \;:\; (t \;:\; \mathbb{N}) \;\rightarrow\; \mathit{Type}$$

At each decision step and for each state, a set of options

$$Y \;:\; (t \;:\; \mathbb{N}) \;\rightarrow\; (x \;:\; X\ t) \;\rightarrow\; \mathit{Type}$$

Sequential decision problems (basic ideas)

At each decision step, a set of possible states:

$$X \; : \; (t \; : \; \mathbb{N}) \; \to \; Type$$

At each decision step and for each state, a set of options

$$Y \; : \; (t \; : \; \mathbb{N}) \; \to \; (x \; : \; X \; t) \; \to \; Type$$

A transition function

$$step \; : \; (t \; : \; \mathbb{N}) \; \to \; (x \; : \; X \; t) \; \to \; (y \; : \; Y \; t \; x) \; \to \; M \, (X \, (S \; t))$$

Sequential decision problems (basic ideas)

At each decision step, a set of possible states:

$$X \; : \; (t \; : \; \mathbb{N}) \; \rightarrow \; \mathit{Type}$$

At each decision step and for each state, a set of options

$$Y \; : \; (t \; : \; \mathbb{N}) \; \rightarrow \; (x \; : \; X \; t) \; \rightarrow \; \mathit{Type}$$

A transition function

$$\mathit{step} \; : \; (t \; : \; \mathbb{N}) \; \rightarrow \; (x \; : \; X \; t) \; \rightarrow \; (y \; : \; Y \; t \; x) \; \rightarrow \; M \, (X \, (S \; t))$$

What about rewards? What are M and S?

Sequential decision problems (uncertainties)

$S\ t$ is just the successor of t:

$$
\begin{array}{l}
\textbf{data}\ \mathbb{N}\ :\ \mathit{Type}\ \textbf{where} \\
\quad Z\ :\ \mathbb{N} \\
\quad S\ :\ \mathbb{N}\ \rightarrow\ \mathbb{N}
\end{array}
$$

Sequential decision problems (uncertainties)

$S\ t$ is just the successor of t:

$$
\begin{array}{l}
\textbf{data}\ \mathbb{N}\ :\ \mathit{Type}\ \textbf{where} \\
\quad Z\ :\ \mathbb{N} \\
\quad S\ :\ \mathbb{N}\ \rightarrow\ \mathbb{N}
\end{array}
$$

$M\ :\ \mathit{Type}\ \rightarrow\ \mathit{Type}$ represents the uncertainties of the problem:

- deterministic problems: $M = \mathit{Id}$
- non-deterministic problems: $M = \mathit{List}$
- stochastic problems: $M = \mathit{Prob}$

Sequential decision problems (uncertainties)

$S\ t$ is just the successor of t:

data $\mathbb{N}$: *Type* **where**
  Z : $\mathbb{N}$
  S : $\mathbb{N} \rightarrow \mathbb{N}$

M : $Type \rightarrow Type$ represents the uncertainties of the problem:

- deterministic problems: $M = Id$
- non-deterministic problems: $M = List$
- stochastic problems: $M = Prob$

data *Prob* : $Type \rightarrow Type$ **where**
  $mkProb$: $(as : Vect\ n\ a) \rightarrow (ps : Vect\ n\ Float) \rightarrow$
    $sum\ ps = 1.0 \rightarrow Prob\ a$

Sequential decision problems (container monad)

Formally, M is a container monad, that is M is a monad:

$$
\begin{array}{lll}
fmap & : & (a \rightarrow b) \rightarrow M\ a \rightarrow M\ b \\
ret & : & a \rightarrow M\ a \\
bind & : & M\ a \rightarrow (a \rightarrow M\ b) \rightarrow M\ b \\
join & : & M\ (M\ a) \rightarrow M\ a
\end{array}
$$

$$
\begin{array}{lll}
functorSpec1 & : & fmap \circ id = id \\
functorSpec2 & : & fmap\ (f \circ g) = (fmap\ f) \circ (fmap\ g)
\end{array}
$$

$$
\begin{array}{lll}
monadSpec1 & : & (fmap\ f) \circ ret = ret \circ f \\
monadSpec2 & : & bind\ (ret\ a)\ f = f\ a \\
monadSpec3 & : & bind\ ma\ ret = ma \\
monadSpec4 & : & \{f : a \rightarrow M\ b\} \rightarrow \{g : b \rightarrow M\ c\} \rightarrow \\
 & & bind\ (bind\ ma\ f)\ g = bind\ ma\ (\lambda x \Rightarrow bind\ (f\ x)\ g) \\
monadSpec5 & : & join\ mma = bind\ mma\ id
\end{array}
$$

Sequential decision problems (container monad)

and M is a container:

$$\begin{aligned} &Elem &&: a \rightarrow M\ a \rightarrow Type \\ &All &&: (a \rightarrow Type) \rightarrow M\ a \rightarrow Type \end{aligned}$$

$$\begin{aligned} &containerSpec1 &&: a\ `Elem`\ (ret\ a) \\ &containerSpec2 &&: a\ `Elem`\ ma \rightarrow ma\ `Elem`\ mma \rightarrow a\ `Elem`\ (join\ mma) \\ &containerSpec3 &&: All\ p\ ma \rightarrow a\ `Elem`\ ma \rightarrow p\ a \end{aligned}$$

Sequential decision problems (basic ideas)

Thus, a concrete sequential decision problem is defined (up to the rewards) in terms of 4 entities: X, Y, M and $step$

$$X : (t : \mathbb{N}) \rightarrow Type$$

$$Y : (t : \mathbb{N}) \rightarrow (x : X\ t) \rightarrow Type$$

$$step : (t : \mathbb{N}) \rightarrow (x : X\ t) \rightarrow (y : Y\ t\ x) \rightarrow M\ (X\ (S\ t))$$

Sequential decision problems (basic ideas)

Thus, a concrete sequential decision problem is defined (up to the rewards) in terms of 4 entities: X, Y, M and $step$

$$X \; : \; (t \; : \; \mathbb{N}) \; \to \; \mathit{Type}$$

$$Y \; : \; (t \; : \; \mathbb{N}) \; \to \; (x \; : \; X \; t) \; \to \; \mathit{Type}$$

$$\mathit{step} \; : \; (t \; : \; \mathbb{N}) \; \to \; (x \; : \; X \; t) \; \to \; (y \; : \; Y \; t \; x) \; \to \; M \, (X \, (S \; t))$$

We formalize reachability, viability and avoidability in terms of these notions

Reachability and viability (intuition)

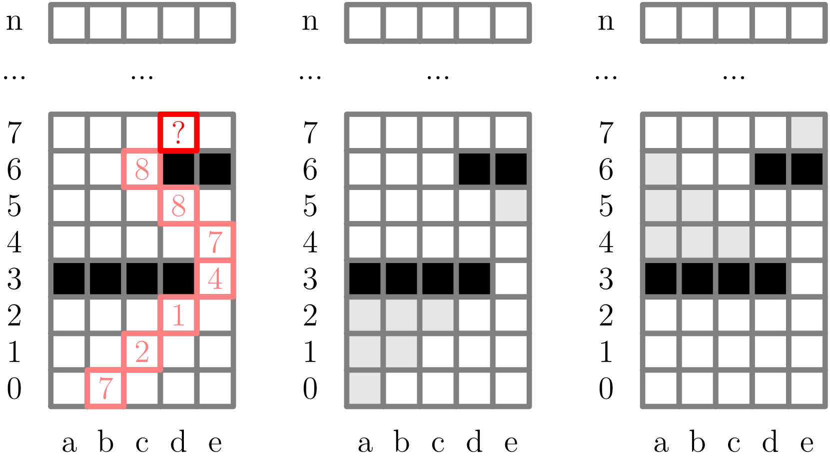


Figure: Possible evolution starting from *b* (left), states with limited viability (middle) and unreachable states (right).

Predecessor relation, reachability and viability

The (possible) predecessor relation:

$$
\begin{aligned}
&Pred\ :\ X\ t\ \to\ X\ (S\ t)\ \to\ Type\\
&Pred\ \{t\}\ x\ x' = (y\ :\ Y\ t\ x\ \ **\ x'\ \text{`}Elem\text{`}\ (step\ t\ x\ y))
\end{aligned}
$$

Predecessor relation, reachability and viability

The (possible) predecessor relation:

$$\begin{aligned}
&\mathit{Pred} \;:\; X\ t \;\to\; X\ (S\ t) \;\to\; \mathit{Type}\\
&\mathit{Pred}\ \{t\}\ x\ x' = (y \;:\; Y\ t\ x \;**\; x'\ \text{`}\mathit{Elem}\text{`}\ (\mathit{step}\ t\ x\ y))
\end{aligned}$$

reachability

$$\begin{aligned}
&\mathit{Reachable} \;:\; X\ t' \;\to\; \mathit{Type}\\
&\mathit{Reachable}\ \{t' = Z\}\ x' \quad = \mathit{Unit}\\
&\mathit{Reachable}\ \{t' = S\ t\}\ x' = (x \;:\; X\ t \;**\; (\mathit{Reachable}\ x, x\ \text{`}\mathit{Pred}\text{`}\ x'))
\end{aligned}$$

Predecessor relation, reachability and viability

The (possible) predecessor relation:

$$\begin{aligned}
&\mathit{Pred} \;:\; X\ t \;\to\; X\ (S\ t) \;\to\; \mathit{Type}\\
&\mathit{Pred}\ \{t\}\ x\ x' = (y\ :\ Y\ t\ x\ **\ x'\ \text{`}\mathit{Elem}\text{`}\ (\mathit{step}\ t\ x\ y))
\end{aligned}$$

reachability

$$\begin{aligned}
&\mathit{Reachable} \;:\; X\ t' \;\to\; \mathit{Type}\\
&\mathit{Reachable}\ \{t' = Z\}\ x' \quad = \mathit{Unit}\\
&\mathit{Reachable}\ \{t' = S\ t\}\ x' = (x\ :\ X\ t\ **\ (\mathit{Reachable}\ x, x\ \text{`}\mathit{Pred}\text{`}\ x'))
\end{aligned}$$

and viability

$$\begin{aligned}
&\mathit{Viable} \;:\; (n\ :\ \mathbb{N}) \;\to\; X\ t \;\to\; \mathit{Type}\\
&\mathit{Viable}\ \{t\}\ Z\ x \qquad = \mathit{Unit}\\
&\mathit{Viable}\ \{t\}\ (S\ m)\ x = (y\ :\ Y\ t\ x\ **\ \mathit{All}\ (\mathit{Viable}\ m)\ (\mathit{step}\ t\ x\ y))
\end{aligned}$$

Avoidability (intuition)

Avoidability (intuition)

- The notion of avoidability is necessarily a relative one: whether a future state is avoidable or not, depends in general on an (actual or hypothetical) current state.

Avoidability (intuition)

- The notion of avoidability is necessarily a relative one: whether a future state is avoidable or not, depends in general on an (actual or hypothetical) current state.
- Thus, avoidability is a relation between states. More precisely, it is a relation between states at a given time and states at some later times.

Avoidability (intuition)

- The notion of avoidability is necessarily a relative one: whether a future state is avoidable or not, depends in general on an (actual or hypothetical) current state.
- Thus, avoidability is a relation between states. More precisely, it is a relation between states at a given time and states at some later times.
- We are interested in the avoidability of “possible” future states. Specifically, we are interested in the avoidability of states which are reachable from some given state.

Avoidability (intuition)

- The notion of avoidability is necessarily a relative one: whether a future state is avoidable or not, depends in general on an (actual or hypothetical) current state.
- Thus, avoidability is a relation between states. More precisely, it is a relation between states at a given time and states at some later times.
- We are interested in the avoidability of "possible" future states. Specifically, we are interested in the avoidability of states which are reachable from some given state.
- The notion of avoidability entails the notion of an alternative.

Avoidability

We are interested in the avoidability of states which are reachable from some given state:

$$
\begin{array}{l}
\mathit{ReachableFrom} \;:\; X\ t'' \;\rightarrow\; X\ t \;\rightarrow\; \mathit{Type} \\
\mathit{ReachableFrom}\ \{t'' = Z\}\ \{t\}\ x''\ x \quad = (t = Z, x = x'') \\
\mathit{ReachableFrom}\ \{t'' = S\ t'\}\ \{t\}\ x''\ x = \\
\quad \mathit{Either}\ (t = S\ t', x = x'') \\
\qquad\qquad (x' \;:\; X\ t' \;**\; (x'\ \text{`}\mathit{ReachableFrom}\text{`}\ x, x'\ \text{`}\mathit{Pred}\text{`}\ x''))
\end{array}
$$

where

$$
\textbf{data}\ \mathit{Either}\ a\ b = \mathit{Left}\ a \mid \mathit{Right}\ b
$$

Avoidability

We are interested in the avoidability of states which are reachable from some given state:

$$
\begin{aligned}
&\mathit{ReachableFrom} \;:\; X\ t'' \;\rightarrow\; X\ t \;\rightarrow\; \mathit{Type} \\
&\mathit{ReachableFrom}\ \{t'' = Z\}\ \{t\}\ x''\ x \quad = (t = Z, x = x'') \\
&\mathit{ReachableFrom}\ \{t'' = S\ t'\}\ \{t\}\ x''\ x = \\
&\quad \mathit{Either}\ (t = S\ t', x = x'') \\
&\qquad\qquad (x' \;:\; X\ t' \;**\; (x'\ \text{`}\mathit{ReachableFrom}\text{`}\ x, x'\ \text{`}\mathit{Pred}\text{`}\ x''))
\end{aligned}
$$

where

$$\textbf{data}\ \mathit{Either}\ a\ b = \mathit{Left}\ a \mid \mathit{Right}\ b$$

Proof of concept: show that

$$
\begin{aligned}
\mathit{reachableFromLemma} \;:\; & (x'' \;:\; X\ t'') \;\rightarrow\; (x \;:\; X\ t) \;\rightarrow \\
& x''\ \text{`}\mathit{ReachableFrom}\text{`}\ x \;\rightarrow\; t''\ \text{`}\mathit{GTE}\text{`}\ t
\end{aligned}
$$

Avoidability

The notion of avoidability entails the notion of an alternative state x''. This has to fulfill three conditions:

$$
\begin{aligned}
& \mathit{AvoidableFrom} \; : \; (x' \; : \; X\ t') \; \to \; (x \; : \; X\ t) \; \to \\
& \qquad\qquad\qquad\quad x' \,\text{`}\mathit{ReachableFrom}\text{`}\, x \; \to \; (m \; : \; \mathbb{N}) \; \to \; \mathit{Type} \\
& \mathit{AvoidableFrom} \; \{\, t' \,\} \; x' \; x \; r \; m = \\
& \quad (x'' \; : \; X\ t' \; ** \; (x'' \,\text{`}\mathit{ReachableFrom}\text{`}\, x, (\mathit{Viable}\ m\ x'', \mathit{Not}\ (x'' = x'))))
\end{aligned}
$$

Avoidability

The notion of avoidability entails the notion of an alternative state x''. This has to fulfill three conditions:

$$
\begin{aligned}
&\mathit{AvoidableFrom} \;:\; (x' : X\ t') \to (x : X\ t) \to \\
&\qquad\qquad\qquad x' \text{ `}\mathit{ReachableFrom}\text{` } x \to (m : \mathbb{N}) \to \mathit{Type} \\
&\mathit{AvoidableFrom}\ \{t'\}\ x'\ x\ r\ m = \\
&\quad (x'' : X\ t' \ {*}{*}\ (x'' \text{ `}\mathit{ReachableFrom}\text{` } x, (\mathit{Viable}\ m\ x'', \mathit{Not}\ (x'' = x'))))
\end{aligned}
$$

Back to the minimal goals: under which conditions are reachability, viability and avoidability decidable?

Decision procedures

For every type (predicate) $P\ :\ Type$, $Not\ P$ is just a synonym for $P\ \rightarrow\ Void$:

$$Not\ :\ Type\ \rightarrow\ Type$$
$$Not\ P = P\ \rightarrow\ Void$$

Decision procedures

For every type (predicate) $P : Type$, $Not\ P$ is just a synonym for $P \rightarrow Void$:

$$
\begin{aligned}
&Not : Type \rightarrow Type \\
&Not\ P = P \rightarrow Void
\end{aligned}
$$

A predicate $P : Type$ is decidable if one can compute either a value $p : P$ or a value of type $Not\ P$:

$$
\begin{aligned}
&Decidable : Type \rightarrow Type \\
&Decidable\ P = Either\ P\ (Not\ P)
\end{aligned}
$$

Decision procedures

Thus, the question is under which conditions one can implement

$$decReachable : (x : X\ t) \rightarrow Decidable\ (Reachable\ x)$$

$$decViable : (n : \mathbb{N}) \rightarrow (x : X\ t) \rightarrow Decidable\ (Viable\ n\ x)$$

$$\begin{aligned} decAvoidableFrom : \; & \{t' : \mathbb{N}\} \rightarrow \{t : \mathbb{N}\} \rightarrow \\ & (x' : X\ t') \rightarrow (x : X\ t) \rightarrow \\ & (r : x'\ \text{`}ReachableFrom\text{`}\ x) \rightarrow (n : \mathbb{N}) \rightarrow \\ & Decidable\ (AvoidableFrom\ \{t'\}\ \{t\}\ x'\ x\ r\ n) \end{aligned}$$

Decision procedures

As one would expect, the conditions

$$fX : (t : \mathbb{N}) \rightarrow Finite\,(X\ t)$$

$$fY : (t : \mathbb{N}) \rightarrow (x : X\ t) \rightarrow Finite\,(Y\ t\ x)$$

$$decElem : \{t : \mathbb{N}\} \rightarrow (x : X\ t) \rightarrow (mx : M\,(X\ t)) \rightarrow Decidable\,(x\ \text{`}Elem\text{`}\ mx)$$

$$decAll : \{t : \mathbb{N}\} \rightarrow (P : X\ t \rightarrow Type) \rightarrow ((x : X\ t) \rightarrow Decidable\,(P\ x)) \rightarrow (mx : M\,(X\ t)) \rightarrow Decidable\,(All\ P\ mx)$$

are sufficient for decidability.

Decision procedures

The key lemma for implementing decision procedures for *Reachable*, *Viable* and *AvoidableFrom* is intuitively obvious

$$
\begin{aligned}
\mathit{finiteDecidableLemma} : \; & \{A : \mathit{Type}\} \to \{P : A \to \mathit{Type}\} \to \\
& \mathit{Finite}\ A \to ((a : A) \to \mathit{Decidable}\,(P\ a)) \to \\
& \mathit{Decidable}\,(a : A \ast\ast\ P\ a)
\end{aligned}
$$

Decision procedures

The key lemma for implementing decision procedures for *Reachable*, *Viable* and *AvoidableFrom* is intuitively obvious

$$\begin{aligned} \mathit{finiteDecidableLemma} \ : \ & \{A : \mathit{Type}\} \rightarrow \{P : A \rightarrow \mathit{Type}\} \rightarrow \\ & \mathit{Finite}\ A \rightarrow ((a : A) \rightarrow \mathit{Decidable}\ (P\ a)) \rightarrow \\ & \mathit{Decidable}\ (a : A \ast\ast P\ a) \end{aligned}$$

but implementing *finiteDecidableLemma* is not trivial!

Decision procedures

With *finiteDecidableLemma*, *fY* and decidability of *Elem* one immediately has decidability of *Pred*

$$\begin{aligned}
&decPred : \{t : \mathbb{N}\} \to (x : X\ t) \to (x' : X\ (S\ t)) \to \\
&\qquad\qquad Decidable\ (x\ `Pred`\ x') \\
&decPred\ \{t\}\ x\ x' = finiteDecidableLemma\ (fY\ t\ x)\ prf\ \mathbf{where} \\
&\quad prf : (y : Y\ t\ x) \to Decidable\ (x'\ `Elem`\ (step\ t\ x\ y)) \\
&\quad prf\ y = decElem\ x'\ (step\ t\ x\ y)
\end{aligned}$$

Decision procedures

and, with

$$decPair \;:\; Decidable\; p \;\to\; Decidable\; q \;\to\; Decidable\; (p, q)$$

decidability of *Reachable*:

$$
\begin{array}{l}
decReachable \;:\; \{t' \;:\; \mathbb{N}\} \;\to\; (x' \;:\; X\; t') \;\to\; Decidable\; (Reachable\; x') \\
decReachable\; \{t' = Z\}\; x' = Left\; () \\
decReachable\; \{t' = S\; t\}\; x' = s1\; \textbf{where} \\
\quad s1 \;:\; Decidable\; (x \;:\; X\; t \; {*}{*}\; (Reachable\; x, x\; \text{`}Pred\text{`}\; x')) \\
\quad s1 = finiteDecidableLemma \\
\qquad\quad (fX\; t) \\
\qquad\quad (\lambda x \Rightarrow decPair\; (decReachable\; x)\; (decPred\; x\; x'))
\end{array}
$$

Decision procedures

Similarly, one implement (prove) decidability of *Viable*:

$$
\begin{aligned}
&\mathit{decViable} : \{t : \mathbb{N}\} \to (n : \mathbb{N}) \to (x : X\ t) \to \mathit{Decidable}\,(\mathit{Viable}\ n\ x)\\
&\mathit{decViable}\,\{t\}\ Z\ x = \mathit{Left}\ ()\\
&\mathit{decViable}\,\{t\}\ (S\ m)\ x = \mathit{s3}\ \textbf{where}\\
&\quad \mathit{s1} \quad : (y : Y\ t\ x) \to \mathit{Decidable}\,(\mathit{All}\,(\mathit{Viable}\ m)\,(\mathit{step}\ t\ x\ y))\\
&\quad \mathit{s1}\ y = \mathit{decAll}\,(\mathit{Viable}\ m)\,(\mathit{decViable}\ m)\,(\mathit{step}\ t\ x\ y)\\
&\quad \mathit{s2} \quad : \mathit{Decidable}\,(y : Y\ t\ x \ast\ast \mathit{All}\,(\mathit{Viable}\ m)\,(\mathit{step}\ t\ x\ y))\\
&\quad \mathit{s2} \quad = \mathit{finiteDecidableLemma}\,(\mathit{fY}\ t\ x)\ \mathit{s1}\\
&\quad \mathit{s3} \quad : \mathit{Decidable}\,(\mathit{Viable}\,(S\ m)\ x)\\
&\quad \mathit{s3} \quad = \mathit{s2}
\end{aligned}
$$

Decision procedures

Similarly, one implement (prove) decidability of *Viable*:

$$
\begin{array}{l}
decViable : \{t : \mathbb{N}\} \to (n : \mathbb{N}) \to (x : X\ t) \to Decidable\ (Viable\ n\ x) \\
decViable\ \{t\}\ Z\ x = Left\ () \\
decViable\ \{t\}\ (S\ m)\ x = s3\ \textbf{where} \\
\quad s1 \quad : (y : Y\ t\ x) \to Decidable\ (All\ (Viable\ m)\ (step\ t\ x\ y)) \\
\quad s1\ y = decAll\ (Viable\ m)\ (decViable\ m)\ (step\ t\ x\ y) \\
\quad s2 \quad : Decidable\ (y : Y\ t\ x\ {*}{*}\ All\ (Viable\ m)\ (step\ t\ x\ y)) \\
\quad s2 \quad = finiteDecidableLemma\ (fY\ t\ x)\ s1 \\
\quad s3 \quad : Decidable\ (Viable\ (S\ m)\ x) \\
\quad s3 \quad = s2
\end{array}
$$

Implementing a decidion procedure for *AvoidableFrom* is a bit more complicated but conceptually equivalent.

Acknowledgments

Contributors:
Patrik Jansson (Chalmers Univ. of Technology), Cezar Ionescu (Chalmers Univ. of Technology), David Christiansen (IT University of Copenhagen), Edwin Brady (University of St. Andrews), Matteo Acerbi, members of the Cartesian Seminar at the Univ. of Potsdam

Acknowledgments

Contributors:
Patrik Jansson (Chalmers Univ. of Technology), Cezar Ionescu (Chalmers Univ. of Technology), David Christiansen (IT University of Copenhagen), Edwin Brady (University of St. Andrews), Matteo Acerbi, members of the Cartesian Seminar at the Univ. of Potsdam
These slides:

Acknowledgments

Contributors:
Patrik Jansson (Chalmers Univ. of Technology), Cezar Ionescu (Chalmers Univ. of Technology), David Christiansen (IT University of Copenhagen), Edwin Brady (University of St. Andrews), Matteo Acerbi, members of the Cartesian Seminar at the Univ. of Potsdam
These slides:

The code shown: