Knowledge Representation

Lecture 3:
Reasoning about Actions III

dr Anna Maria Radzikowska
Wydział Matematyki i Nauk Informacyjnych
Pokój 504, gmach MiNI
E-mail: A.Radzikowska@mini.pw.edu.pl
Action Language $AR$ – recall...

Statements:

- **value statement:** $\alpha$ after $A_1, \ldots, A_n$
- **observation statement:** observable $\alpha$ after $A_1, \ldots, A_n$
- **effect statement:** $A$ causes $\alpha$ if $\pi$
- **release statement:** $A$ releases $f$ if $\pi$
- **constraint statement:** always $\alpha$. 
Action Language $\mathcal{AR}$ – recall...

Statements:

- **value statement**: $\alpha$ after $A_1, \ldots, A_n$
- **observation statement**: observable $\alpha$ after $A_1, \ldots, A_n$
- **effect statement**: $A$ causes $\alpha$ if $\pi$
- **release statement**: $A$ releases $f$ if $\pi$
- **constraint statement**: always $\alpha$.

A **structure** for a language $\mathcal{L}$ of the class $\mathcal{AR}$ is a triple $S = (\Sigma, \sigma_0, Res)$, where

- $\Sigma$ is a set of states
- $\sigma_0 \in \Sigma$ is the initial state
- $Res : Ac \times \Sigma \to 2^\Sigma$ is a transition function.
Action Language $\mathcal{AR}$ – model

A structure $S = (\Sigma, \sigma_0, \text{Res})$ is a **model of** an action domain $D$ iff

**(M.1)** $\Sigma$ is the set of all states for $D$

**(M.2)** every value statement and every observation statement is true in $S$

**(M.3)** for any $A \in \mathcal{A}c$ and for any $\sigma \in \Sigma$, $\text{Res}(A, \sigma)$ is the set of all states $\sigma' \in \Sigma$ for which the sets $\text{New}(A, \sigma, \sigma')$ are minimal (wrt set inclusion).

Recall that $\text{New}(A, \sigma, \sigma')$ is the set of literals $\overline{f}$ that hold in $\sigma'$ and

- $f$ is inertial and $\sigma(f) \neq \sigma'(f)$, or
- there is a statement $A$ releases $f$ if $\pi$ in $D$ such that $\sigma \models \pi$. 
Queries in $\mathcal{AR}$

By a **query of $\mathcal{AR}$** we mean any expression of the form:

- **value queries:**

  \[
  \text{necessary } \alpha \text{ after } A_1, \ldots, A_n \text{ from } \pi \\
  \text{possibly } \alpha \text{ after } A_1, \ldots, A_n \text{ from } \pi
  \]

The 1st query states that $\alpha$ **always** holds after performing the sequence $A_1, \ldots, A_n$ of actions from any state satisfying $\pi$.

The 2nd query says that $\alpha$ **sometimes** holds after executing $A_1, \ldots, A_n$ from any state satisfying $\pi$.

When the option **from** $\pi$ is omitted, these queries refer to the initial state.
Basic queries (cont.)

- **executability query:**

  \[
  \text{executable } A_1, \ldots, A_n \text{ from } \pi
  \]

  Intuitively, this query states that the sequence \((A_1, \ldots, A_n)\) is executable from any state satisfying \(\pi\).

  Again, if the phrase *from* \(\pi\) is omitted, then the initial state is to be taken into account.
accessibility query:

\[ \text{accessible } \gamma \text{ from } \pi. \]

This query states that the goal \( \gamma \) is achieved from any state satisfying \( \pi \).
If the phrase \textit{from} \( \pi \) is omitted, then it refers to the initial state.
Basic queries (cont.)

- **accessibility query:**

  \[ \text{accessible } \gamma \text{ from } \pi. \]

  This query states that the goal \( \gamma \) is achieved from any state satisfying \( \pi \).

  If the phrase *from* \( \pi \) is omitted, then it refers to the initial state.

**Underlying intuition:**

There exists a sequence \( (A_1, \ldots, A_n), n \geq 0 \), of actions, executable from any state satisfying \( \pi \) (resp. the initial state), which leads to states satisfying \( \gamma \). Various sequences of actions, starting in various initial states, may lead to various states, where \( \gamma \) holds.
goal query:

\[ \text{goal } \gamma \text{ from } \pi. \]

Intuitively, this query returns the shortest sequence \((A_1, \ldots, A_n)\) of actions, executable from any state satisfying \(\pi\) (if such a sequence exists at all), which leads to where the goal condition \(\gamma\) holds.

When the goal is to be achieved from the initial state, the phrase \textit{from }\(\pi\) will be omitted.
Let $D$ be an action domain and let $Q$ be a query of $\mathcal{AR}$. We say that $Q$ is a **consequence of $D$**, in symbols $D \simeq Q$, iff

- if $Q$ is a value query of the form

  \[
  \text{necessary } \alpha \text{ after } A_1, \ldots, A_n \text{ from } \pi,
  \]

  then $D \simeq Q$ iff for every model $S = (\Sigma, \sigma_0, \text{Res})$ of $D$, for every state $\sigma \in \Sigma$, and for every mapping $\Psi_S$, if $\sigma \models \pi$ then

  $\Psi_S((A_1, \ldots, A_n), \sigma) \models \alpha$
Satisfiability

Let $D$ be an action domain and let $Q$ be a query of $\mathcal{AR}$. We say that $Q$ is a consequence of $D$, in symbols $D \models Q$, iff

- if $Q$ is a value query of the form
  \[
  \text{necessary } \alpha \text{ after } A_1, \ldots, A_n \text{ from } \pi,
  \]
  then $D \models Q$ iff for every model $S = (\Sigma, \sigma_0, \text{Res})$ of $D$, for every state $\sigma \in \Sigma$, and for every mapping $\Psi_S$, if $\sigma \models \pi$ then $\Psi_S((A_1, \ldots, A_n), \sigma) \models \alpha$

- if $Q$ is a value query
  \[
  \text{necessary } \alpha \text{ after } A_1, \ldots, A_n,
  \]
  then $D \models Q$ iff for every model $S = (\Sigma, \sigma_0, \text{Res})$ of $D$, and for every mapping $\Psi_S$, it holds $\Psi_S((A_1, \ldots, A_n), \sigma_0) \models \alpha$. 
if $Q$ is a value query of the form

$$\textit{possibly } \alpha \textit{ after } A_1, \ldots, A_n \textit{ from } \pi,$$

then $D \models Q$ iff for every model $S = (\Sigma, \sigma_0, Res)$ of $D$, for every state $\sigma \in \Sigma$ and for some mapping $\Psi_S$, if $\sigma \models \pi$ then

$\Psi_S((A_1, \ldots, A_n), \sigma) \models \alpha$. 
Satisfiability (cont.)

- if $Q$ is a value query of the form
  
  \[ \text{possibly } \alpha \text{ after } A_1, \ldots, A_n \text{ from } \pi, \]

  then $D \models Q$ iff for every model $S = (\Sigma, \sigma_0, Res)$ of $D$, for every state $\sigma \in \Sigma$ and for some mapping $\Psi_S$, if $\sigma \models \pi$ then
  
  $\Psi_S((A_1, \ldots, A_n), \sigma) \models \alpha$.

- if $Q$ is a value query of the form

  \[ \text{possibly } \alpha \text{ after } A_1, \ldots, A_n, \]

  then $D \models Q$ iff for every model $S = (\Sigma, \sigma_0, Res)$ of $D$, and for some mapping $\Psi_S$, it holds $\Psi_S((A_1, \ldots, A_n), \sigma_0) \models \alpha$. 
Example: Tossing a coin

For example, let an action domain $D$ be as follows:

- Initially heads
- Toss releases heads.

Then

$$D \models \text{possibly } \neg \text{heads after } \text{Toss}$$

$$D \models \text{possibly } \text{heads after } \text{Toss}$$

$$D \not\models \text{necessary } ; \neg \text{head after } \text{Toss}$$

$$D \not\models \text{necessary } \text{head after } \text{Toss}. $$
if $Q$ is an executability query of the form

\[ \text{executable } A_1, \ldots, A_n \text{ from } \pi, \]

then $D \models Q$ iff for every model $S = (\Sigma, \sigma_0, Res)$ of $D$, for every state $\sigma \in \Sigma$, and for every mapping $\Psi_S$, if $\sigma \models \pi$ then $\Psi_S((A_1, \ldots, A_n), \sigma)$ is defined.
if $Q$ is an executability query of the form 

\[
\text{executable } A_1, \ldots, A_n \text{ from } \pi,
\]

then $D \models Q$ iff for every model $S = (\Sigma, \sigma_0, \text{Res})$ of $D$, for every state $\sigma \in \Sigma$, and for every mapping $\Psi_S$, if $\sigma \models \pi$ then $\Psi_S((A_1, \ldots, A_n), \sigma)$ is defined.

if $Q$ is an executability query of the form 

\[
\text{executable } A_1, \ldots, A_n,
\]

then $D \models Q$ iff for every model $S = (\Sigma, \sigma_0, \text{Res})$ of $D$ and for every mapping $\Psi_S$, $\Psi_S((A_1, \ldots, A_n), \sigma_0)$ is defined.
if $Q$ is an accessibility query of the form

$$\textit{accessible } \gamma \textit{ from } \pi,$$

then $D \models Q$ iff for every model $S = (\Sigma, \sigma_0, Res)$ of $D$, for every state $\sigma \in \Sigma$ and for every mapping $\Psi_S$, if $\sigma \models \pi$ then there exists a sequence $(A_1, \ldots, A_n)$ of actions such that

- $\Psi_S((A_1, \ldots, A_n), \sigma)$ is defined
- $\Psi_S((A_1, \ldots, A_n), \sigma) \models \gamma$
if $Q$ is an accessibility query of the form

\[ \textit{accessible} \ \gamma, \]

then $D \models Q$ iff for every model $S = (\Sigma, \sigma_0, \text{Res})$ of $D$, and for every mapping $\Psi_S$, there exists a sequence $(A_1, \ldots, A_n)$ of actions, $n \geq 0$, such that

- $\Psi_S((A_1, \ldots, A_n), \sigma_0)$ is defined
- $\Psi_S((A_1, \ldots, A_n), \sigma_0) \models \gamma$. 
Consider an action domain:

A \textit{causes} \ p \lor q;
B \textit{causes} \ \neg r;
\textit{impossible} \ B \textit{ if} \ \neg q.
Consider an action domain:

A causes \( p \lor q \);
B causes \( \neg r \);

impossible B if \( \neg q \).
An example (cont.)

Query: \textit{accessible }\neg r \textit{ from } \neg p.
Query: $accessible \neg r \text{ from } \neg p$.

$D \not\models accessible \neg r \text{ from } \neg p$.

Why?
Query: *accessible* \( \neg r \) *from* \( \neg p \).

\( D \not\models \text{accessible} \ \neg r \text{ from } \neg p \).

**WHY?** Because performing the sequence \((A, B)\) in the state \((\neg p, \neg q, r)\) may (but not need) lead to the state \((\neg p, q, \neg r)\).
if $Q$ is a goal query of the form

$$\text{goal } \gamma \text{ from } \pi,$$

then $D \models Q$ iff there exists a sequence $(A_1, \ldots, A_n)$ of actions such that for every model $S = (\Sigma, \sigma_0, \text{Res})$ of $D$, for every state $\sigma \in \Sigma$, and for every mapping $\Psi_S$, if $\sigma \models \pi$ then

- $\Psi_S((A_1, \ldots, A_n), \sigma)$ is defined
- $\Psi_S((A_1, \ldots, A_n), \sigma) \models \gamma$.

The sequence $(A_1, \ldots, A_n)$, if it exists, is called an answer for the query $Q$ wrt $D$. 
if $Q$ is a goal query of the form

$$\text{goal} \; \gamma$$

then $D \vdash Q$ iff there exists a sequence $(A_1, \ldots, A_n)$ of actions such that for every model $S = (\Sigma, \sigma_0, Res)$ of $D$ and for every mapping $\Psi_S$, it holds

- $\Psi_S((A_1, \ldots, A_n), \sigma_0)$ is defined
- $\Psi_S((A_1, \ldots, A_n), \sigma_0) \models \gamma$.

As before, the sequence $(A_1, \ldots, A_n)$, if it exists, is called an answer for the query $Q$ wrt $D$. 
Actions with Qualifications
The qualification problem concerns conditions under which the performance of an action is possible, it proceeds successfully and leads to expected results. Since all preconditions of actions are usually immense, it is usually unreasonable (if ever possible) to explicitly enumerate and check all of possibilities (even most unlikely).
Consider an action domain:

*In an ancient kingdom there are two blocks, A and B. Either block may be painted yellow, but by order of the emperor at most one of the blocks is permitted to be yellow at a time. Initially the first block is yellow. A painter tried to paint the second one yellow.*
Consider an action domain:

*In an ancient kingdom there are two blocks, A and B. Either block may be painted yellow, but by order of the emperor at most one of the blocks is permitted to be yellow at a time. Initially the first block is yellow. A painter tried to paint the second one yellow.*

Representation in $\mathcal{AR}$:

- $\text{always } \neg \text{yellow}A \lor \neg \text{yellow}B$;
- $\text{initially } \text{yellow}A$;
- $\text{PAINT}A \text{ causes } \text{yellow}A$;
- $\text{PAINT}B \text{ causes } \text{yellow}B$. 
Here $\Sigma = \{\sigma_0, \sigma_1, \sigma_2\}$, where

$$\sigma_0 = \{\text{yellow} A, \neg \text{yellow} B\} \quad \sigma_1 = \{\neg \text{yellow} A, \neg \text{yellow} B\}.$$  
$$\sigma_2 = \{\neg \text{yellow} A, \text{yellow} B\}$$

By simple calculations we get:

$$Res_0(P\text{AINT}B, \sigma_0) = Res(P\text{AINT}B, \sigma_0) = \{\sigma_2\}.$$  

So painting the block B changes the color of the block A!
Consider an action domain:

*initially* \( \text{loaded} \land \text{walking} \);
*always* \( \text{walking} \rightarrow \text{alive} \);
\( \text{SHOOT} \) *causes* \( \neg \text{loaded} \);
\( \text{SHOOT} \) *causes* \( \neg \text{alive} \text{ if } \text{loaded} \);
\( \text{ENTICE} \) *causes* \( \text{walking} \).
Example 2: Enticing Fred

Consider an action domain:

- **initially** loaded \& walking;
- **always** walking → alive;
- **Shoot** causes ¬loaded;
- **Shoot** causes ¬alive if loaded;
- **Entice** causes walking.

Let

\[ \sigma_0 = \{ \text{alive, loaded, walking} \} \]
\[ \sigma_1 = \{ \neg \text{alive, } \neg \text{loaded, } \neg \text{walking} \} . \]

It is easy to check that \[ \text{Res}(\text{Shoot}, \sigma_0) = \{ \sigma_1 \} . \]
Furthermore, let

\[ \sigma_2 = \{ \neg \text{alive}, \text{loaded}, \neg \text{walking} \} \]
\[ \sigma_3 = \{ \text{alive}, \neg \text{loaded}, \text{walking} \} \]
\[ \sigma_4 = \{ \text{alive}, \text{loaded}, \text{walking} \} \].

In AR we get:

\[ \text{Res}_0(\text{ENTICE}, \sigma_2) = \{ \sigma_3, \sigma_4 \} \]
\[ \text{New}(\text{ENTICE}, \sigma_2, \sigma_3) = \{ \text{alive}, \neg \text{loaded}, \text{walking} \} \]
\[ \text{New}(\text{ENTICE}, \sigma_2, \sigma_4) = \{ \text{alive}, \text{walking} \} \]
\[ \text{Res}(\text{ENTICE}, \sigma_2) = \{ \sigma_4 \} \].

In other words, **enticing a dead turkey makes him miraculously alive !!!**
Furthermore, let

\[ \sigma_2 = \{ \neg \text{alive}, \text{loaded}, \neg \text{walking} \} \]
\[ \sigma_3 = \{ \text{alive}, \neg \text{loaded}, \text{walking} \} \]
\[ \sigma_4 = \{ \text{alive}, \text{loaded}, \text{walking} \}. \]

In \( \mathcal{AR} \) we get:

\[ \text{Res}_0(\text{ENTICE}, \sigma_2) = \{ \sigma_3, \sigma_4 \} \]
\[ \text{New}(\text{ENTICE}, \sigma_2, \sigma_3) = \{ \text{alive}, \neg \text{loaded}, \text{walking} \} \]
\[ \ast \quad \text{New}(\text{ENTICE}, \sigma_2, \sigma_4) = \{ \text{alive}, \text{walking} \} \]
\[ \text{Res}(\text{ENTICE}, \sigma_2) = \{ \sigma_4 \}. \]

In other words, **enticing a dead turkey makes him miraculously alive !!!**

The reasoning methods of \( \mathcal{AR} \) \[\text{is not}\] adequate to represent this scenario.
Basic assumptions

- Inertia law
- Complete information about all actions and all fluents
- Nondeterminism is allowed.
- Whenever any indirect precondition of an action fails, the action is unexecutable
- Action can be unexecutable in some states.
Basic assumptions

- Inertia law
- Complete information about all actions and all fluents
- Nondeterminism is allowed.
- Whenever any indirect precondition of an action fails, the action is unexecutable
- Action can be unexecutable is some states.

To represent this class of dynamic systems we will use action languages of the class $AQ$. 
Syntax is identical as in $\mathcal{AR}$.

**Semantics of $\mathcal{AQ}$**

**Definition 3.1.** Let $D$ be an action domain. A *structure* for a language $\mathcal{L}$ of the class $\mathcal{AQ}$ is a triple $S = (\Sigma, \sigma_0, \text{Res})$ such that

- $\Sigma$ is a set of states
- $\sigma_0 \in \Sigma$ is the initial state
- $\text{Res} : \mathcal{Ac} \times \Sigma \rightarrow 2^{\Sigma}$ is a transition function.
Denote

\[ D^- \] – the action domain obtained from \( D \) by removing all constraint statements.
Denote

- $D^–$ – the action domain obtained from $D$ by removing all constraint statements.
- $\text{Mod}(D^–)$ – the set of all models $S^– = (\Sigma^–, \sigma^–_0, \text{Res}^–)$ of $D^–$ in the sense of $\mathcal{AR}$. 


Denote

- $D^{-}$ – the action domain obtained from $D$ by removing all constraint statements.

- $\text{Mod}(D^{-})$ – the set of all models $S^{-} = (\Sigma^{-}, \sigma_0^{-}, \text{Res}^{-})$ of $D^{-}$ in the sense of $\mathcal{AR}$.

- $\text{Mod}^*(D^{-})$ – the set of all models $S = (\Sigma, \sigma_0, \text{Res}) \in \text{Mod}(D^{-})$ such that $\sigma_0$ satisfies all constraint statements in $D$. 
Let $S^* = (\Sigma^*, \sigma_0^*, Res^*) \in Mod^*(D^-)$. Put

- $\Sigma \subseteq \Sigma^*$ be the set of all states satisfying all constraint statements in $D$

- for any action $A \in \mathcal{A}c$ and for any state $\sigma \in \Sigma$,
  
  put $Res(A, \sigma) = Res^*(A, \sigma) \cap \Sigma$. 
Let \( S^* = (\Sigma^*, \sigma_0^*, Res^*) \in Mod^*(D^-) \). Put

- \( \Sigma \subseteq \Sigma^* \) be the set of all states satisfying all constraint statements in \( D \)
- for any action \( A \in Ac \) and for any state \( \sigma \in \Sigma \),
  put \( Res(A, \sigma) = Res^*(A, \sigma) \cap \Sigma \).

**Definition 3.2.** The structure \( S = (\Sigma, \sigma_0, Res) \) defined as above is a *model of* \( D \).
Introductory example (cont.)

always \neg \text{yellow}A \lor \neg \text{yellow}B;
initially \text{yellow}A;
\text{\texttt{Paint}}A \text{ causes } \text{yellow}A;
\text{\texttt{Paint}}B \text{ causes } \text{yellow}B.

Now \Sigma = \{\sigma_1, \sigma_1, \sigma_2, \sigma_3\}, where

\begin{align*}
\sigma_0 &= \{\text{yellow}A, \neg \text{yellow}B\} & \sigma_1 &= \{\neg \text{yellow}A, \neg \text{yellow}B\} \\
\sigma_2 &= \{\neg \text{yellow}A, \text{yellow}B\} & \sigma_3 &= \{\text{yellow}A, \text{yellow}B\}.
\end{align*}
σ₀ = \{yellowA, ¬yellowB\} \quad σ₁ = \{¬yellowA, ¬yellowB\}
σ₂ = \{¬yellowA, yellowB\} \quad σ₃ = \{yellowA, yellowB\}.

Res₀\(\neg\)(PAINTB, σ₀) = \{σ₂, σ₃\}
New(PAINTB, σ₀, σ₂) = \{¬yellowA, yellowB\}
\ast \quad New(PAINTB, σ₀, σ₃) = \{yellowB\}
Res\(\neg\)(PAINTB, σ₀) = \{σ₃\}

Res(PAINTB, σ₀) = \emptyset.
σ₀ = \{yellowA, ¬yellowB\} \quad σ₁ = \{¬yellowA, ¬yellowB\}
σ₂ = \{¬yellowA, yellowB\} \quad σ₃ = \{yellowA, yellowB\}.

Analogously:

\[ Res₀^−(PAINTA, σ₂) = \{σ₀, σ₃\} \]
\[ New(PAINTA, σ₂, σ₀) = \{yellowA, ¬yellowB\} \]
\[ * \quad New(PAINTA, σ₂, σ₃) = \{yellowA\} \]
\[ Res^−(PAINTA, σ₂) = \{σ₃\} \]
\[ Res(PAINTA, σ₂) = \emptyset. \]
Introductory example (cont.)

Fig. 3.1: Model of $D^-$
Fig. 3.2: Model of D
Example 2: Enticing Fred (cont.)

\[ \sigma_0 = \{ \text{alive, loaded, walking} \} \]
\[ \sigma_1 = \{ \text{alive, } \neg \text{loaded, walking} \} \]
\[ \sigma_2 = \{ \neg \text{alive, loaded, walking} \} \]
\[ \sigma_3 = \{ \neg \text{alive, } \neg \text{loaded, walking} \} \]
\[ \sigma_4 = \{ \text{alive, loaded, } \neg \text{walking} \} \]
\[ \sigma_5 = \{ \text{alive, } \neg \text{loaded, } \neg \text{walking} \} \]
\[ \sigma_6 = \{ \neg \text{alive, loaded, } \neg \text{walking} \} \]
\[ \sigma_7 = \{ \neg \text{alive, } \neg \text{loaded, } \neg \text{walking} \} \].

\[ Res_{0}^{-}(\text{ENTICE}, \sigma_6) = \{ \sigma_0, \sigma_1, \sigma_2, \sigma_3 \} \]
\[ New(\text{ENTICE}, \sigma_6, \sigma_0) = \{ \text{alive, walking} \} \]
\[ New(\text{ENTICE}, \sigma_6, \sigma_1) = \{ \text{alive, } \neg \text{loaded, walking} \} \]
\[ * \quad New(\text{ENTICE}, \sigma_6, \sigma_2) = \{ \text{walking} \} \]
\[ New(\text{ENTICE}, \sigma_6, \sigma_3) = \{ \neg \text{loaded, walking} \} \]
\[ Res^{-}(\text{ENTICE}, \sigma_6, ) = \{ \sigma_2 \} \]
\[ Res(\text{ENTICE}, \sigma_6) = \emptyset. \]
Example 2: Enticing Fred (cont.)

\[\sigma_0 = \{\text{alive}, \text{loaded}, \text{walking}\}\]
\[\sigma_1 = \{\text{alive}, \neg\text{loaded}, \text{walking}\}\]
\[\sigma_2 = \{\neg\text{alive}, \text{loaded}, \text{walking}\}\]
\[\sigma_3 = \{\neg\text{alive}, \neg\text{loaded}, \text{walking}\}\]
\[\sigma_4 = \{\text{alive}, \text{loaded}, \neg\text{walking}\}\]
\[\sigma_5 = \{\text{alive}, \neg\text{loaded}, \neg\text{walking}\}\]
\[\sigma_6 = \{\neg\text{alive}, \text{loaded}, \neg\text{walking}\}\]
\[\sigma_7 = \{\neg\text{alive}, \neg\text{loaded}, \neg\text{walking}\}\].

Now consider the shooting action in the initial state \(\sigma_0\).
Example 2: Enticing Fred (cont.)

\[\sigma_0 = \{\text{alive, loaded, walking}\} \quad \sigma_4 = \{\text{alive, loaded, } \neg \text{walking}\}\]
\[\sigma_1 = \{\text{alive, } \neg \text{loaded, walking}\} \quad \sigma_5 = \{\text{alive, } \neg \text{loaded, } \neg \text{walking}\}\]
\[\sigma_2 = \{\neg \text{alive, loaded, walking}\} \quad \sigma_6 = \{\neg \text{alive, loaded, } \neg \text{walking}\}\]
\[\sigma_3 = \{\neg \text{alive, } \neg \text{loaded, walking}\} \quad \sigma_7 = \{\neg \text{alive, } \neg \text{loaded, } \neg \text{walking}\}\].

Now consider the shooting action in the initial state \(\sigma_0\).

\[\text{Res}^-_0(\text{SHOOT}, \sigma_0) = \{\sigma_3, \sigma_7\}\]
\[\text{New(\text{SHOOT}, } \sigma_0, \sigma_3) = \{\neg \text{alive, } \neg \text{loaded}\}\]
\[\text{New(\text{SHOOT, } } \sigma_0, \sigma_7) = \{\neg \text{alive, } \neg \text{loaded, } \neg \text{walking}\}\]
\[\text{Res}^-_0(\text{SHOOT, } \sigma_0) = \{\sigma_3\}\]
\[\text{Res(\text{SHOOT, } } \sigma_0) = \emptyset\].
Example 2: Enticing Fred (cont.)

\(\sigma_0 = \{\text{alive, loaded, walking}\} \quad \sigma_4 = \{\text{alive, loaded, } \neg \text{walking}\}\)
\(\sigma_1 = \{\text{alive, } \neg \text{loaded, walking}\} \quad \sigma_5 = \{\text{alive, } \neg \text{loaded, } \neg \text{walking}\}\)
\(\sigma_2 = \{\neg \text{alive, loaded, walking}\} \quad \sigma_6 = \{\neg \text{alive, loaded, } \neg \text{walking}\}\)
\(\sigma_3 = \{\neg \text{alive, } \neg \text{loaded, walking}\} \quad \sigma_7 = \{\neg \text{alive, } \neg \text{loaded, } \neg \text{walking}\}\).

Now consider the shooting action in the initial state \(\sigma_0\).

\[
Res_0^- (\text{SHOOT}, \sigma_0) = \{\sigma_3, \sigma_7\}
\]
\[
\star \quad \text{New}(\text{SHOOT}, \sigma_0, \sigma_3) = \{\neg \text{alive, } \neg \text{loaded}\}
\]
\[
\text{New}(\text{SHOOT}, \sigma_0, \sigma_7) = \{\neg \text{alive, } \neg \text{loaded, } \neg \text{walking}\}
\]
\[
Res^- (\text{SHOOT}, \sigma_0) = \{\sigma_3\}
\]
\[
Res(\text{SHOOT}, \sigma_0) = \emptyset.
\]

So it is impossible to shoot a walking turkey!
Therefore the following question arises:

*Should we reject inadmissible states before or after minimization of changes?*
Therefore the following question arises:

*Should we reject inadmissible states *before* or *after* minimization of changes?*

Using some simplification we can give the following answer:

- If constraints influence only on actions’ *ramifications* and do not influence on actions’ qualifications, then inadmissible states are to be rejected *before* minimization of changes.
Therefore the following question arises:

*Should we reject inadmissible states before or after minimization of changes?*

Using some simplification we can give the following answer:

- If constraints influence only on actions’ *ramifications* and do not influence on actions’ qualifications, then inadmissible states are to be rejected *before* minimization of changes.

- If constraints influence only on actions’ *qualifications* and do not influence on actions’ ramifications, then inadmissible states should be rejected *after* minimization of changes.
However,

*What should be done if a constraint works for ramifications wrt one action, and at the same time for qualifications wrt another one?*
However,

*What should be done if a constraint works for ramifications wrt one action, and at the same time for qualifications wrt another one?*

**Syntax of ARQ**

The set of statements of $\mathcal{AR}$ is extended by an *fluent preserving statement*:

$$A \text{ preserves } f \text{ if } \pi$$

Intuitively, in any state satisfying $\pi$, the action $A$, when performed, does not change the value of the fluent $f$. 
A **structure** for a language $\mathcal{L}$ of the class $\mathcal{ARQ}$ is a triple $S = (\Sigma, \sigma_0, Res)$ such that

- $\Sigma$ is a set of states
- $\sigma_0 \in \Sigma$ is the initial state, and
- $Res : \Sigma \times Ac \rightarrow 2^\Sigma$ is a transition function.
Semantics of $\mathcal{ARQ}$

A **structure** for a language $\mathcal{L}$ of the class $\mathcal{ARQ}$ is a triple $S = (\Sigma, \sigma_0, Res)$ such that

- $\Sigma$ is a set of states
- $\sigma_0 \in \Sigma$ is the initial state, and
- $Res : \Sigma \times \mathcal{Ac} \rightarrow 2^\Sigma$ is a transition function.

Let $D^-$ be the action domain obtained from $D$ by removing all fluent preserving statements and let $\text{Mod}(D^-)$ be the set of models of $D^-$ in the sense of $\mathcal{AR}$. 
Definition 3.3. A structure $S = (\Sigma, \sigma_0, \text{Res}) \in \text{Mod}(D^-)$ is a model of $D$ iff for any action $A \in \mathcal{A}c$ and for any state $\sigma \in \Sigma$,

$$\text{Res}(A, \sigma) = \{ \sigma' \in \Sigma : (A \text{ preserves } f \text{ if } \pi) \in D \land \sigma \models \pi \implies (\sigma(f) = \sigma'(f)) \}.$$
Example 2: Enticing Fred (cont.)

Initially $\text{loaded} \land \text{walking}$;
Always $\text{walking} \rightarrow \text{alive}$;
Shoot causes $\neg\text{loaded}$;
Shoot causes $\neg\text{alive} \text{ if } \text{loaded}$;
Entice causes $\text{walking}$
Entice preserves $\text{alive}$.

All admissible states:

- $\sigma_0 = \{\text{alive, loaded, walking}\}$
- $\sigma_1 = \{\text{alive, loaded, } \neg\text{walking}\}$
- $\sigma_2 = \{\neg\text{alive, loaded, } \neg\text{walking}\}$
- $\sigma_3 = \{\text{alive, } \neg\text{loaded, walking}\}$
- $\sigma_4 = \{\text{alive, } \neg\text{loaded, } \neg\text{walking}\}$
- $\sigma_5 = \{\neg\text{alive, } \neg\text{loaded, } \neg\text{walking}\}$. 
Example 2: Enticing Fred (cont.)

\[ \sigma_0 = \{\text{alive, loaded, walking} \} \]
\[ \sigma_1 = \{\text{alive, loaded, } \neg \text{walking} \} \]
\[ \sigma_2 = \{\neg \text{alive, loaded, } \neg \text{walking} \} \]
\[ \sigma_3 = \{\text{alive, } \neg \text{loaded, walking} \} \]
\[ \sigma_4 = \{\text{alive, } \neg \text{loaded, } \neg \text{walking} \} \]
\[ \sigma_5 = \{\neg \text{alive, } \neg \text{loaded, } \neg \text{walking} \} \]

\[ \text{Res}^-_0(E\text{NTICE, } \sigma_2) = \{\sigma_0, \sigma_3\} \]
\[ \text{New}(E\text{NTICE, } \sigma_2, \sigma_0) = \{\text{alive, walking}\} \]
\[ \text{New}(E\text{NTICE, } \sigma_2, \sigma_3) = \{\text{alive, } \neg \text{loaded, walking}\} \]
\[ \text{Res}^-_0(E\text{NTICE, } \sigma_2) = \{\sigma_0\} \].
Example 2: Enticing Fred (cont.)

\[ \sigma_0 = \{ \text{alive, loaded, walking} \} \quad \sigma_3 = \{ \text{alive, } \neg \text{loaded, walking} \} \]
\[ \sigma_1 = \{ \text{alive, loaded, } \neg \text{walking} \} \quad \sigma_4 = \{ \text{alive, } \neg \text{loaded, } \neg \text{walking} \} \]
\[ \sigma_2 = \{ \neg \text{alive, loaded, } \neg \text{walking} \} \quad \sigma_5 = \{ \neg \text{alive, } \neg \text{loaded, } \neg \text{walking} \} . \]

\[ \text{Res}^-(\text{ENTICE}, \sigma_2) = \{ \sigma_0, \sigma_3 \} \]
\[ \star \quad \text{New}(\text{ENTICE}, \sigma_2, \sigma_0) = \{ \text{alive, walking} \} \]
\[ \text{New}(\text{ENTICE}, \sigma_2, \sigma_3) = \{ \text{alive, } \neg \text{loaded, walking, } \} \]
\[ \text{Res}^-(\text{ENTICE}, \sigma_2) = \{ \sigma_0 \} . \]

Since \( \sigma_0(\text{walking}) \neq \sigma_2(\text{walking}) \), we get \( \sigma_0 \notin \text{Res}(\text{ENTICE}, \sigma_2) \).
Hence \( \text{Res}(\text{ENTICE}, \sigma_2) = \emptyset \).
Example 2: Enticing Fred (cont.)

\[ \sigma_0 = \{\text{alive, loaded, walking}\} \quad \sigma_3 = \{\text{alive, \neg loaded, walking}\} \]
\[ \sigma_1 = \{\text{alive, loaded, \neg walking}\} \quad \sigma_4 = \{\text{alive, \neg loaded, \neg walking}\} \]
\[ \sigma_2 = \{\neg\text{alive, loaded, \neg walking}\} \quad \sigma_5 = \{\neg\text{alive, \neg loaded, \neg walking}\}. \]

\[ \text{Res}^{-}_0(\text{SHOOT}, \sigma_0) = \text{Res}^{-}(\text{SHOOT}, \sigma_0) = \{\sigma_5\}. \]
Example 2: Enticing Fred (cont.)

\[
\sigma_0 = \{ \text{alive, loaded, walking} \} \quad \sigma_3 = \{ \text{alive, \neg loaded, walking} \} \\
\sigma_1 = \{ \text{alive, loaded, \neg walking} \} \quad \sigma_4 = \{ \text{alive, \neg loaded, \neg walking} \} \\
\sigma_2 = \{ \neg \text{alive, loaded, \neg walking} \} \quad \sigma_5 = \{ \neg \text{alive, \neg loaded, \neg walking} \}.
\]

\[
Res^{-0}_0(\text{SHOT}, \sigma_0) = Res^{-}(\text{SHOT}, \sigma_0) = \{ \sigma_5 \}.
\]

Since there is no fluent preserving statement wrt the action \text{SHOT} and the fluent \text{walking}, we finally get

\[
Res(\text{SHOT}, \sigma_0) = \{ \sigma_5 \}.
\]
We can apply the reasoning method from AQ with modifications of domain descriptions. Specifically:

whenever constraints work for actions ramifications, the respective \textit{releases} statements are to be added.
Alternative solution

We can apply the reasoning method from AQ with modifications of domain descriptions. Specifically:

*whenever constraints work for actions ramifications, the respective releases statements are to be added.*

Consider again the *Enticing Fred* example.

\[
\begin{align*}
\text{always } &\text{ walking } \rightarrow \text{ alive;} \\
\text{initially } &\text{ loaded } \land \text{ walking} \\
\text{SHOOT } &\text{ causes } \neg \text{loaded;} \\
\text{SHOOT } &\text{ causes } \neg \text{alive if } \text{loaded;} \\
\text{SHOOT } &\text{ releases walking;} \\
\text{ENTICE } &\text{ causes walking.}
\end{align*}
\]
Example 2: Enticing Fred (cont.)

\[ \sigma_0 = \{\text{alive, loaded, walking}\} \quad \sigma_4 = \{\text{alive, loaded, } \neg \text{walking}\} \]
\[ \sigma_1 = \{\text{alive, } \neg \text{loaded, walking}\} \quad \sigma_5 = \{\text{alive, } \neg \text{loaded, } \neg \text{walking}\} \]
\[ \sigma_2 = \{\neg \text{alive, loaded, walking}\} \quad \sigma_6 = \{\neg \text{alive, loaded, } \neg \text{walking}\} \]
\[ \sigma_3 = \{\neg \text{alive, } \neg \text{loaded, walking}\} \quad \sigma_7 = \{\neg \text{alive, } \neg \text{loaded, } \neg \text{walking}\} \]

\[ \text{Res}^{-0}(\text{SHOOT, } \sigma_0) = \{\sigma_3, \sigma_7\} \]
\[ \quad \text{New(} \text{SHOOT, } \sigma_0, \sigma_3\) = \{\neg \text{alive, } \neg \text{loaded, walking}\} \]
\[ \quad \text{New(} \text{SHOOT, } \sigma_0, \sigma_7\) = \{\neg \text{alive, } \neg \text{loaded, } \neg \text{walking}\} \]
\[ \quad \text{Res}^{-}(\text{SHOOT, } \sigma_0) = \{\sigma_3, \sigma_7\} \]
\[ \quad \text{Res}(\text{SHOOT, } \sigma_0) = \{\sigma_7\}. \]
Thank you for your attention!

Any questions are welcome.