Knowledge Representation

Lecture 6:

Part 1: Reasoning about Scenarios
Part 2: Propositional Calculus

dr Anna Maria Radzikowska
Wydział Matematyki i Nauk Informacyjnych
Pokój 504, gmach MiNI
E–mail: A.Radzikowska@mini.pw.edu.pl
Basic assumptions

In contrast to previous approaches with *branching time* model, now we will consider action domains with *linear time*. 
In contrast to previous approaches with *branching time* model, now we will consider action domains with *linear time*.

- Inertia law.
- Linear model of time (discrete time).
- Actions with duration; during performance of the action, values of fluents changed by the actions are unknown.
- Dynamic effects of actions – one action can invoke another one(s).
- Situation can trigger actions — some states may cause executing some actions.
Basic assumptions

In contrast to previous approaches with *branching time* model, now we will consider action domains with *linear time*.

- Inertia law.
- Linear model of time (discrete time).
- Actions with duration; during performance of the action, values of fluents changed by the actions are unknown.
- Dynamic effects of actions – one action can invoke another one(s).
- Situation can trigger actions — some states may cause executing some actions.

To represent these dynamic systems we will use action languages of the class $\mathcal{AL}$. For simplicity we assume that each action is performed in 1 unit of time.
As before, a *signature* is a pair \((\mathcal{F}, \mathcal{A}_c)\), \(\mathcal{F} \cap \mathcal{A}_c = \emptyset\).
Action Language $\mathcal{AL}$ – syntax

As before, a *signature* is a pair $(\mathcal{F}, \mathcal{A}c)$, $\mathcal{F} \cap \mathcal{A}c = \emptyset$.

We have 5 types of statements:

- **fluent effect statement:**

$$A \text{ causes } \alpha \text{ if } \pi$$
As before, a *signature* is a pair \((\mathcal{F}, \mathcal{Ac}), \mathcal{F} \cap \mathcal{Ac} = \emptyset\).

We have 5 types of statements:

- **fluent effect statement:**

  \[ A \text{ causes } \alpha \text{ if } \pi \]

- **action effect statement:**

  \[ A \text{ invokes } B \text{ after } d \text{ if } \pi \]

where \(A, B \in \mathcal{Ac}, \pi \in Forms(\mathcal{F}), \text{ and } d \in \mathbb{N}\).

Intuitively, this statement says that the action \(B\) starts after \(d\) timepoints since the action \(A\) is completed, provided that the condition \(\pi\) holds when \(A\) starts.
release statement:

\[ A \text{ releases } f \text{ if } \pi \]
**Action Language $\mathcal{AL}$ (cont.)**

- **release statement:**

  \[ A \text{ releases } f \text{ if } \pi \]

- **trigger statement:**

  \[ \pi \text{ triggers } A \]

Intuitively, this statement says that the action $A$ starts at any timepoint when the condition $\pi$ holds.
Action Language $\mathcal{AL}$ (cont.)

- **release statement:**

  $A$ releases $f$ if $\pi$

- **trigger statement:**

  $\pi$ triggers $A$

Intuitively, this statement says that the action $A$ starts at any timepoint when the condition $\pi$ holds.
Action Language $\mathcal{AL}$ (cont.)

- **release statement:**

  \[ A \text{ releases } f \text{ if } \pi \]

- **trigger statement:**

  \[ \pi \text{ triggers } A \]

Intuitively, this statement says that the action $A$ starts at any timepoint when the condition $\pi$ holds.

A set of statements is called a *domain description*. 
By an action scenario \( (\text{scenario}) \), for short) we mean a pair \( Sc = (OBS, ACS) \), where

- \( OBS \) is the set of observations: \( OBS = \{ (\alpha_1, t_1), \ldots, (\alpha_n, t_n) \} \), where \( \alpha_i \in Forms(F) \) and \( t_i \in \mathbb{N} \), \( i = 1, \ldots, n \)

- \( ACS \) is the set of action occurrences: \( ACS = \{ (A_1, t_1), \ldots, (A_n, t_k) \} \), where \( A_i \in Ac \) and \( t_i \in \mathbb{N} \), \( i = 1, \ldots, k \).
A query in $\mathcal{AL}$ is an expression either of the form:

\[
\begin{align*}
\alpha & \text{ at } t \text{ when } Sc \\
A & \text{ at } t \text{ when } Sc.
\end{align*}
\]

Intuitively, the $1^{st}$ query states that the condition $\alpha$ holds at timepoint $t$ when the scenario is carrying out, whereas the $2^{nd}$ query says that the action $A$ is executed at timepoint $t$ when the scenario $Sc$ is carrying out.
Definition 5.1 A structure for $\mathcal{AL}$ is a system $S = (H, O, E)$ such that

$H : \mathcal{F} \times \mathbb{N} \rightarrow \{0, 1\}$ is a history function
Definition 5.1 A structure for $\mathcal{AL}$ is a system $S = (H, O, E)$ such that

- $H : \mathcal{F} \times \mathbb{N} \rightarrow \{0, 1\}$ is a history function
- $O : \mathcal{Ac} \times \mathbb{N} \rightarrow 2^\mathcal{F}$ is an occlusion function; for any action $A \in \mathcal{Ac}$ and for any timepoint $t \in \mathbb{N}$, $O(A, t)$ is the set of fluents under influence of the performance of $A$ when executed from timepoint $t - 1$ to $t$
Definition 5.1 A **structure** for \( \mathcal{AL} \) is a system \( S = (H, O, E) \) such that

- \( H : \mathcal{F} \times \mathbb{N} \rightarrow \{0, 1\} \) is a **history function**

- \( O : \mathcal{Ac} \times \mathbb{N} \rightarrow 2^{\mathcal{F}} \) is an **occlusion function**; for any action \( A \in \mathcal{Ac} \) and for any timepoint \( t \in \mathbb{N} \), \( O(A, t) \) is the set of fluents under influence of the performance of \( A \) when executed from timepoint \( t - 1 \) to \( t \)

- \( E \subseteq \mathcal{Ac} \times \mathbb{N} \) is an **actions occurrences relation**; if \( (A, t) \in E \) then \( A \) occurs at timepoint \( t \). We assume that for all \( A, B \in \mathcal{Ac} \) and every \( t \in \mathbb{N} \),

\[
(A, t) \in E \land (B, t) \in E \implies A = B. \quad (1)
\]
Definition 5.1 A **structure** for $\mathcal{AL}$ is a system $S = (H, O, E)$ such that

- $H : \mathcal{F} \times \mathbb{N} \rightarrow \{0, 1\}$ is a **history function**
- $O : \mathcal{Ac} \times \mathbb{N} \rightarrow 2^\mathcal{F}$ is an **occlusion function**; for any action $A \in \mathcal{Ac}$ and for any timepoint $t \in \mathbb{N}$, $O(A, t)$ is the set of fluents under influence of the performance of $A$ when executed from timepoint $t - 1$ to $t$
- $E \subseteq \mathcal{Ac} \times \mathbb{N}$ is an **actions occurrences relation**; if $(A, t) \in E$ then $A$ occurs at timepoint $t$. We assume that for all $A, B \in \mathcal{Ac}$ and every $t \in \mathbb{N}$,

$$ (A, t) \in E \& (B, t) \in E \implies A = B. $$

(1)

Note that the condition (1) guarantees that at most one action is executed at a time.
For any structure $S = (H, O, E)$ for $\mathcal{AL}$, the history function $H$ are extended for the set of all formulas according to rules well–known in propositional logic, i.e. for every timepoint $t \in \mathbb{N}$,

$$H^*(f, t) = H(f, t) \text{ for any } f \in \mathcal{F}$$

$$H^*(\neg \alpha, t) = 1 - H^*(\alpha, t)$$

$$H^*(\alpha \land \beta, t) = \min(H^*(\alpha, t), H^*(\beta, t))$$

$$H^*(\alpha \lor \beta, t) = \max(H^*(\alpha, t), H^*(\beta, t))$$

$$H^*(\alpha \rightarrow \beta, t) = \begin{cases} 0 & \text{iff } H^*(\alpha, t) = 1 \& H^*(\beta, t) = 0 \\ 1 & \text{otherwise} \end{cases}$$

$$H^*(\alpha \equiv \beta, t) = \begin{cases} 1 & \text{iff } H^*(\alpha, t) = H^*(\beta, t) \\ 0 & \text{otherwise} \end{cases}$$
Let $S = (H, O, E)$ be a structure for $\mathcal{AL}$, let $S_c = (OBS, ACS)$ be a scenario, and let $D$ be a domain description. We say that $S$ is a *structure for $S_c$ wrt $D$* iff

\[ \text{for each observation } (\alpha, t) \in OBS, \ H(\alpha, t) = 1 \]
Let $S = (H, O, E)$ be a structure for $\mathcal{AL}$, let $Sc = (OBS, ACS)$ be a scenario, and let $D$ be a domain description. We say that $S$ is a structure for $Sc$ wrt $D$ iff

- for each observation $(\alpha, t) \in OBS$, $H(\alpha, t) = 1$
- $ACS \subseteq E$
Denote: $\text{fl}(\alpha)$ – the set of fluents occurring in $\alpha$.

- for each statement $(A \text{ causes } \alpha \text{ if } \pi) \in D$ and for each timepoint $t \in \mathbb{N}$, if $H(\pi, t) = 1$ and $(A, t) \in E$, then $H(\alpha, t + 1) = 1$ and $\text{fl}(\alpha) \subseteq O(A, t + 1)$
Semantics of $\mathcal{AL}$ (cont.)

Denote: $fl(\alpha)$ – the set of fluents occurring in $\alpha$.

- for each statement $(A \text{ causes } \alpha \text{ if } \pi) \in D$ and for each timepoint $t \in \mathbb{N}$, if $H(\pi, t) = 1$ and $(A, t) \in E$, then $H(\alpha, t + 1) = 1$ and $fl(\alpha) \subseteq O(A, t + 1)$

- for each statement $(A \text{ releases } f \text{ if } \pi) \in D$ and for each timepoint $t \in \mathbb{N}$, if $H(\pi, t) = 1$ and $(A, t) \in E$, then $f \in O(A, t + 1)$
Denote: \( fl(\alpha) \) – the set of fluents occurring in \( \alpha \).

- for each statement \( (A \text{ causes } \alpha \text{ if } \pi) \in D \) and for each timepoint \( t \in \mathbb{N} \), if \( H(\pi, t) = 1 \) and \( (A, t) \in E \), then \( H(\alpha, t + 1) = 1 \) and \( fl(\alpha) \subseteq O(A, t + 1) \)

- for each statement \( (A \text{ releases } f \text{ if } \pi) \in D \) and for each timepoint \( t \in \mathbb{N} \), if \( H(\pi, t) = 1 \) and \( (A, t) \in E \), then \( f \in O(A, t + 1) \)

- for each statement \( (\pi \text{ triggers } A) \in D \) and for each timepoint \( t \in \mathbb{N} \), if \( H(\pi, t) = 1 \), then \( (A, t) \in E \)
Denote: \( fl(\alpha) \) – the set of fluents occurring in \( \alpha \).

- for each statement \((A \textit{ causes } \alpha \textit{ if } \pi) \in D\) and for each timepoint \(t \in \mathbb{N}\), if \(H(\pi, t) = 1\) and \((A, t) \in E\), then \(H(\alpha, t + 1) = 1\) and \(fl(\alpha) \subseteq O(A, t + 1)\)

- for each statement \((A \textit{ releases } f \textit{ if } \pi) \in D\) and for each timepoint \(t \in \mathbb{N}\), if \(H(\pi, t) = 1\) and \((A, t) \in E\), then \(f \in O(A, t + 1)\)

- for each statement \((\pi \textit{ triggers } A) \in D\) and for each timepoint \(t \in \mathbb{N}\), if \(H(\pi, t) = 1\), then \((A, t) \in E\)

- for each statement \((A \textit{ invokes } B \textit{ after } d \textit{ if } \pi) \in D\) and for each timepoint \(t \in \mathbb{N}\), if \(H(\pi, t) = 1\) and \((A, t) \in A\), then \((B, t + d + 1) \in E\).
Semantics of $AL$ (cont.)

Observe:

- Any change (in fluents’ values) are allowed *only* in occlusion regions.
Observe:

- Any change (in fluents’ values) are allowed only in occlusion regions.
- Consequently, we will be interested in structures $S = (H, O, E)$ for $Sc = (OBS, ACS)$ wrt $D$, which occlusion functions $O$ determine the smallest occlusion regions.
Denote

Let $O_1, O_2 : X \rightarrow 2^Y$. We write

$O_1 \preceq O_2$ iff $O_1(x) \subseteq O_2(x)$ for every $x \in X$. 
Denote

Let $O_1, O_2 : X \to 2^Y$. We write

- $O_1 \preceq O_2$ iff $O_1(x) \subseteq O_2(x)$ for every $x \in X$.
- $O_1 \prec O_2$ iff $O_1 \preceq O_2$ and $O_1 \neq O_2$. 
Denote

Let $O_1, O_2 : X \rightarrow 2^Y$. We write

- $O_1 \preceq O_2$ iff $O_1(x) \subseteq O_2(x)$ for every $x \in X$.
- $O_1 \prec O_2$ iff $O_1 \preceq O_2$ and $O_1 \neq O_2$.

**Definition 5.2** Let $S = (H, O, E)$ be a structure for a scenario $Sc = (OBS, ACS)$ wrt a domain description $D$. We say that $S$ is $O$–minimal iff there is no structure $S' = (H', O', E')$ for $Sc$ wrt $D$ such that $O' \prec O$. 
Definition 5.3 Let $S = (H, O, E)$ be a structure for a scenario $Sc = (OBS, ACS)$ wrt a domain description $D$. We say that $S$ is a model of $Sc$ wrt $D$ iff

(M.1) $S$ is $O$–minimal
**Definition 5.3** Let $S = (H, O, E)$ be a structure for a scenario $Sc = (OBS, ACS)$ wrt a domain description $D$. We say that $S$ is a model of $Sc$ wrt $D$ iff

(M.1) $S$ is $O$–minimal

(M.2) for every timepoint $t \in \mathbb{N}$,

$$\{ f \in F : H(f, t) \neq H(f, t + 1) \} \subseteq O(A, t + 1)$$

for some action $A \in Ac$
Definition 5.3 Let $S = (H, O, E)$ be a structure for a scenario $Sc = (OBS, ACS)$ wrt a domain description $D$. We say that $S$ is a model of $Sc$ wrt $D$ iff

(M.1) $S$ is $O$–minimal

(M.2) for every timepoint $t \in \mathbb{N}$,

$$\{ f \in \mathcal{F} : H(f, t) \neq H(f, t + 1) \} \subseteq O(A, t + 1) \}$$

for some action $A \in Ac$

(M.3) there is no structure $S' = (H', O', E')$ for $Sc$ wrt $D$ satisfying (M.1)–(M.2) such that $E' \subset E$. 
**Definition 5.3** Let $S = (H, O, E)$ be a structure for a scenario $Sc = (OBS, ACS)$ wrt a domain description $D$. We say that $S$ is a model of $Sc$ wrt $D$ iff

(M.1) $S$ is $O$–minimal

(M.2) for every timepoint $t \in \mathbb{N}$,

$$\{ f \in \mathcal{F} : H(f, t) \neq H(f, t + 1) \} \subseteq O(A, t + 1)$$

for some action $A \in Ac$

(M.3) there is no structure $S' = (H', O', E')$ for $Sc$ wrt $D$ satisfying (M.1)–(M.2) such that $E' \subset E$.

We say that a scenario $Sc$ is **consistent** wrt a domain description $D$ iff there exists a model $S$ of $Sc$ wrt $D$; otherwise it is called **inconsistent**.
Example 1: Inconsistency

Let $D$ be a domain description with two actions, $A$ and $B$, and let a scenario $Sc = (OBS, ACS)$ be given as

- $OBS = \emptyset$
- $ACS = \{(A, 1), (B, 1)\}$. 
Example 1: Inconsistency

Let $D$ be a domain description with two actions, $A$ and $B$, and let a scenario $Sc = (OBS, ACS)$ be given as

- $OBS = \emptyset$
- $ACS = \{(A, 1), (B, 1)\}$.

Since $A$ and $B$ are to be executed parallel, $Sc$ is inconsistent wrt any $D$ (with the actions $A$ and $B$) – there is no structure for $Sc$ wrt any $D$. 
Example 2: (In)consistency

Consider the following domain description $D$ and two scenarios: $Sc_1 = (OBS_1, ACS_1)$ and $Sc_2 = (OBS_2, ACS_2)$, given by

- $A$ causes $f$;
- $B$ causes $\neg f$;
- $C$ causes $g$;
- $A$ invokes $C$ after 1.

- $OBS_1 = OBS_2 = \emptyset$
- $ACS_1 = \{(A, 1), (B, 3)\}$
- $ACS_2 = \{(A, 1), (B, 2)\}$. 
Example 2: (In)consistency

Consider the following domain description $D$ and two scenarios:
$Sc_1 = (OBS_1, ACS_1)$ and $Sc_2 = (OBS_2, ACS_2)$, given by

- $A$ causes $f$;
- $B$ causes $\neg f$;
- $C$ causes $g$;
- $A$ invokes $C$ after 1.

- $OBS_1 = OBS_2 = \emptyset$
- $ACS_1 = \{(A, 1), (B, 3)\}$
- $ACS_2 = \{(A, 1), (B, 2)\}$.

For any $S = (H, O, E)$ for $Sc_1$ wrt $D$, we have $(A, 1), (B, 3) \in E$. Since $(C, 3) \in E$, $Sc_1$ is inconsistent wrt $D$. 
Example 2: (In)consistency

Consider the following domain description $D$ and two scenarios:
$S_{c1} = (OBS_1, ACS_1)$ and $S_{c2} = (OBS_2, ACS_2)$, given by

- $A$ causes $f$;
- $B$ causes $\neg f$;
- $C$ causes $g$;
- $A$ invokes $C$ after 1.

- $OBS_1 = OBS_2 = \emptyset$
- $ACS_1 = \{(A, 1), (B, 3)\}$
- $ACS_2 = \{(A, 1), (B, 2)\}$.

For any $S = (H, O, E)$ for $S_{c1}$ wrt $D$, we have $(A, 1), (B, 3) \in E$. Since $(C, 3) \in E$, $S_{c1}$ is inconsistent wrt $D$.

For any $S = (H, O, E)$ for $S_{c2}$ wrt $D$, $(A, 1), (B, 2), (C, 3) \in E$, so no inconsistency occurs.
Example 3: Modification of YSP

Consider the following domain description $D$:

- **LOAD** causes $\text{loaded}$;
- **SHOOT** causes $\neg\text{loaded}$;
- **SHOOT** causes $\neg\text{alive}$ if $\text{loaded} \land \neg\text{hidden}$;
- **LOAD** invokes **ESCAPE**;
- **ESCAPE** releases $\text{hidden}$.

and a scenario $Sc = (OBS, ACS)$, where

\[
OBS = \{(\text{alive} \land \neg\text{loaded} \land \neg\text{hidden}, 0)\}
\]
\[
ACS = \{(\text{LOAD}, 1), (\text{SHOOT}, 3)\}.
\]
There are two main classes of structures for $Sc$ wrt $D$. Namely,

**Class 1:**

\[
\begin{array}{ccccccc}
0 & 1 & 2 & 3 & 4 \\
\text{Load} & \text{Escape} & \text{Shoot} \\
a & a? & a? & a? & a? \\
\neg l & l? & l* & l? & \neg l* \\
\neg h & h? & h? & h* & h? \\
\end{array}
\]

Occlusion regions:
\[
\{l\} \subseteq Occlude(\text{LOAD}, 2) \\
\{h\} \subseteq Occlude(\text{ESCAPE}, 3) \\
\{l\} \subseteq Occlude(\text{SHOOT}, 4).
\]

Occurrences of actions: \[\{(\text{LOAD}, 1), (\text{ESCAPE}, 2), (\text{SHOOT}, 3)\} \subseteq E.\]
Class 2:

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 \\
\text{Load} & \text{Escape} & \text{Shoot} \\
\{a\} & \{a?\} & \{a?\} & \{a?\} & \{a?\} \\
\{\neg l\} & \{l?\} & \{l?\} & \{\neg l^*\} & \{\neg l^*\} \\
\{\neg h\} & \{h?\} & \{h?\} & \{\neg h^*\} & \{h?\} \\
\end{array}
\]

\[
\{l\} \subseteq \text{Occlude}(\text{LOAD}, 2) \\
\{h\} \subseteq \text{Occlude}(\text{ESCAPE}, 3) \\
\{l\} \subseteq \text{Occlude}(\text{SHOOT}, 4).
\]

\[\text{Occurrences of actions: } \{(\text{LOAD}, 1), (\text{ESCAPE}, 2), (\text{SHOOT}, 3)\} \subseteq E.\]
Class 2 has two subclasses:

**Subclass 1:**

\[
\begin{array}{ccccccc}
0 & 1 & 2 & 3 & 4 \\
\hline
a & a? & a? & a? & \neg a^* \\
\neg l & l? & l^* & l & \neg l^* \\
\neg h & h? & h? & \neg h^* & h?
\end{array}
\]

Occlusion regions:
\[\{l\} \subseteq Occlude(\text{LOAD}, 2)\]

\[\{h\} \subseteq Occlude(\text{ESCAPE}, 3)\]

\[\{a, l\} \subseteq Occlude(\text{SHOOT}, 4)\]

Occurrences of actions:
\[\{(\text{LOAD}, 1), (\text{ESCAPE}, 2), (\text{SHOOT}, 3)\} \subseteq E\]
Subclass 2:

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 & 4 \\
a & a? & a? & a? & a? \\
\neg l & l? & l* & \neg l & \neg l* \\
\neg h & h? & h? & \neg h* & h? \\
\end{array}
\]

Occlusion regions:
\[
\{l\} \subseteq Oclude(LOAD, 2) \\
\{h\} \subseteq Oclude(ESCAPE, 3) \\
\{l\} \subseteq Oclude(SHOOT, 4).
\]

Occurrences of actions:
\[
\{(LOAD, 1), (ESCAPE, 2), (SHOOT, 3)\} \subseteq E.
\]
Two main classes of $O$–minimal structures for $Sc \text{ wrt } D$:

**Class 1:**

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 \\
\text{Load} & \text{Escape} & \text{Shoot} \\
\ \quad a \quad & \quad a? \quad & \quad a? \quad & \quad a? \quad & \quad a? \\
\neg l & l? & l* & l? & \neg l* \\
\neg h & h? & h? & h* & h? \\
\end{array}
\]

Oclusion regions:

\[
\text{Occlude} (\text{LOAD}, 2) = \{l\} \\
\text{Occlude} (\text{ESCAPE}, 3) = \{h\} \\
\text{Occlude} (\text{SHOOT}, 4) = \{l\}.
\]

Occurrences of actions: \{ (\text{LOAD}, 1), (\text{ESCAPE}, 2), (\text{SHOOT}, 3) \} \subseteq E.
And two subclasses of the 2\textsuperscript{nd} class:

\textbf{1\textsuperscript{st} subclass:}

\begin{itemize}
  \item \textit{Load}:
    \begin{itemize}
      \item $t=0$: $a$
      \item $t=1$: $a^?$
    \end{itemize}
  \item \textit{Escape}:
    \begin{itemize}
      \item $t=2$: $a^?$
      \item $t=3$: $a^?
      \item $t=4$: $\neg a^*$
    \end{itemize}
  \item \textit{Shoot}:
    \begin{itemize}
      \item $t=0$: $\neg l$
      \item $t=1$: $l^*$
      \item $t=3$: $l$
      \item $t=4$: $\neg l^*$
    \end{itemize}
  \item \textit{Occlusion regions:}
    \begin{itemize}
      \item $\text{Occlude}(\text{LOAD}, 2) = \{l\}$
      \item $\text{Occlude}(\text{ESCAPE}, 3) = \{h\}$
      \item $\text{Occlude}(\text{SHOOT}, 4) = \{a, l\}$
    \end{itemize}
\end{itemize}

\textbf{Occurrences of actions:} $\{(\text{LOAD}, 1), (\text{ESCAPE}, 2), (\text{SHOOT}, 3)\} \subseteq E$. 
Structures for $S_c \text{ wrt } D$ (cont.)

And the $2^{nd}$ subclass of the class 1:

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 & 4 \\
\text{Load} & \text{Escape} & \text{Shoot} \\
\hline
a & a? & a? & a? & a? \\
\neg l & l? & l^* & \neg l & \neg l^* \\
\neg h & h? & h? & \neg h^* & h? \\
\end{array}
\]

\[\text{Occlude}(\text{Load}, 2) = \{l\} \]

\[\text{Occlusion regions: } \text{Occlude}(\text{Escape}, 3) = \{h\} \]

\[\text{Occlude}(\text{Shoot}, 4) = \{l\}.\]

\[\text{Occurrences of actions: } \{(\text{Load}, 1), (\text{Escape}, 2), (\text{Shoot}, 3)\} \subseteq E.\]
There are two models of $Sc$ wrt $D$:

$1^{st}$ model $S_1$:

\[ \text{Occurrences of actions: } \{ (\text{LOAD}, 1), (\text{ESCAPE}, 2), (\text{SHOOT}, 3) \} = E. \]

\[ \text{Occlusion regions: } \text{Occlude}(\text{LOAD}, 2) = \{ l \} \]

\[ \text{Occlude}(\text{ESCAPE}, 3) = \{ h \} \]

\[ \text{Occlude}(\text{SHOOT}, 4) = \{ l \}. \]
$2^{nd}$ model $S_2$:

\[\begin{align*}
\text{Load} & : a, a, a, a, \neg a^* \\
\text{Escape} & : \neg l, a, l^*, a, \neg a^* \\
\text{Shoot} & : \neg h, \neg h, \neg h, l, \neg l^* \\
\end{align*}\]

Occclusion regions:
- $\text{Occlude}(\text{LOAD}, 2) = \{l\}$
- $\text{Occlude}(\text{ESCAPE}, 3) = \{h\}$
- $\text{Occlude}(\text{SHOOT}, 4) = \{a, l\}$.

Occurrences of actions: $\{(\text{LOAD}, 1), (\text{ESCAPE}, 2), (\text{SHOOT}, 3)\} = E.$
Let $Sc$ be a scenario and let $D$ be a domain description. We say that a query $Q$ is a consequence of $Sc \text{ wrt } D$, in symbols $D, Sc \models Q$, iff

if $Q$ is of the form $\alpha \text{ at } t \text{ when } Sc$, then for every model $S = (H, O, E)$ of $Sc \text{ wrt } D$, it holds $H(\alpha, t) = 1$
Let $Sc$ be a scenario and let $D$ be a domain description. We say that a query $Q$ is a consequence of $Sc$ wrt $D$, in symbols $D, Sc \models Q$, iff

- if $Q$ is of the form $\alpha \text{ at } t \text{ when } Sc$, then for every model $S = (H, O, E)$ of $Sc$ wrt $D$, it holds $H(\alpha, t) = 1$.

- if $Q$ is of the form $A \text{ at } t \text{ when } Sc$, then for every model $S = (H, O, E)$ of $Sc$ wrt $D$, it holds $(A, t) \in E$. 
Example 3 (cont.)

Load | Escape | Shoot
-----|--------|--------
0    | 1      | 2      | 3      | 4      

\[
\begin{array}{c}
\text{Load} \\
\text{Escape} \\
\text{Shoot}
\end{array}
\]

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 & 4 \\
\text{a} & \text{a} & \text{a} & \text{a} & \text{a} \\
\neg\text{l} & \neg\text{l} & \text{l} & \text{l} & \neg\text{l} \\
\neg\text{h} & \neg\text{h} & \neg\text{h} & \text{h} & \neg\text{h}
\end{array}
\]
Example 3 (cont.)

\[ S_c, D \models \neg \text{loaded at } t \text{ when } S_c \text{ for } t \geq 4 \]
$Sc, \bar{D} \models \neg \text{loaded at } t \text{ when } Sc \text{ for } t \geq 4$

$Sc, \bar{D} \models \text{ESCAPE at } 2 \text{ when } Sc$
Example 3 (cont.)

\[ \text{Load} \quad \text{Escape} \quad \text{Shoot} \]

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 & 4 \\
\text{a} & \text{a} & \text{a} & \text{a} & \text{a} \\
\neg \text{l} & \neg \text{l} & \text{l} & \text{l} & \neg \text{l} \\
\neg \text{h} & \neg \text{h} & \neg \text{h} & \text{h} & \text{h} \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 & 4 \\
\text{a} & \text{a} & \text{a} & \text{a} & \neg \text{a} \\
\neg \text{l} & \neg \text{l} & \text{l} & \text{l} & \neg \text{l} \\
\neg \text{h} & \neg \text{h} & \neg \text{h} & \neg \text{h} & \text{h} \\
\end{array}
\]

\[ \text{Sc}, \text{D} \models \neg \text{loaded at } t \text{ when } \text{Sc for } t \geq 4 \]

\[ \text{Sc}, \text{D} \models \text{ESCAPE at } 2 \text{ when } \text{Sc} \]

\[ \text{Sc}, \text{D} \not\models \text{alive at } 4 \text{ when } \text{Sc} \]
Example 3 (cont.)

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 \\
\hline
\text{Load} & \text{Escape} & \text{Shoot} \\
\hline
a & a & a & a & a \\
\neg l & \neg l & l^* & l & \neg l^* \\
\neg h & \neg h & \neg h & h^* & h \\
\end{array}
\]

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 \\
\hline
\text{Load} & \text{Escape} & \text{Shoot} \\
\hline
a & a & a & a & \neg a^* \\
\neg l & \neg l & l^* & l & \neg l^* \\
\neg h & \neg h & \neg h & \neg h^* & \neg h \\
\end{array}
\]

\[
\begin{align*}
\text{Sc, } D & \models \neg \text{loaded at } t \text{ when } \text{Sc for } t \geq 4 \\
\text{Sc, } D & \models \text{Escape at } 2 \text{ when } \text{Sc} \\
\text{Sc, } D & \not\models \text{alive at } 4 \text{ when } \text{Sc} \\
\text{Sc, } D & \not\models \neg \text{alive at } 4 \text{ when } \text{Sc.}
\end{align*}
\]
Foundations to Classical Logic
The notion of logic

By a *logic* we mean a triple $\text{Log} = (\mathcal{L}, \Sigma, \models)$, where

- $\mathcal{L}$ is a *language* of $\text{Log}$ (i.e. the set of all formulas in $\text{Log}$)
The notion of logic

By a *logic* we mean a triple \( \text{Log} = (\mathcal{L}, \Sigma, \models) \), where

- \( \mathcal{L} \) is a *language* of \( \text{Log} \) (i.e. the set of all formulas in \( \text{Log} \))
- \( \Sigma \) is the class of all frames used for interpretation of formulas
The notion of logic

By a *logic* we mean a triple $Log = (\mathcal{L}, \Sigma, |=)$, where

- $\mathcal{L}$ is a *language* of $Log$ (i.e. the set of all formulas in $Log$)
- $\Sigma$ is the class of all frames used for interpretation of formulas
- $|= : 2^\Sigma \rightarrow 2^\mathcal{L}$ is the consequence mapping which for each set $\mathcal{M}$ of frames determines the set of formulas *satisfied* in every frame from $\mathcal{M}$. 
The notion of logic

By a logic we mean a triple $Log = (\mathcal{L}, \Sigma, |=)$, where

- $\mathcal{L}$ is a language of $Log$ (i.e. the set of all formulas in $Log$)
- $\Sigma$ is the class of all frames used for interpretation of formulas
- $|= : 2^\Sigma \to 2^\mathcal{L}$ is the consequence mapping which for each set $\mathcal{M}$ of frames determines the set of formulas satisfied in every frame from $\mathcal{M}$.

Classical logic:

- Propositional logic
- First order propositional calculus.
A language of a propositional logic (PC) is determined by the following disjoint sets of symbols:

- a set $Var$ of propositional variables
- the truth constant $\top$
- logical connectives $\neg$ and $\rightarrow$
- parentheses ( and ).
Propositional Calculus

A language of a propositional logic (PC) is determined by the following disjoint sets of symbols:

- a set $\text{Var}$ of propositional variables
- the truth constant $\top$
- logical connectives $\neg$ and $\rightarrow$
- parentheses ( and ).

The set of all propositional formulas $\mathcal{L}$ ($\text{language}$) is the smallest set of the following expressions:

- $\text{Var} \subseteq \mathcal{L}$
- $\top \in \mathcal{L}$
- if $\alpha$, $\beta$ are formulas, then so are $\neg \alpha$ and $\alpha \rightarrow \beta$. 
Propositional Calculus (cont.)

The remaining symbols are defined as:

- the truth constant $\bot : \bot \overset{def}{=} \neg T$
The remaining symbols are defined as:

- The truth constant $\bot$: $\bot \overset{def}{=} \neg \top$

- Logical connectives:
  - Disjunction: $\alpha \lor \beta \overset{def}{=} \neg \alpha \rightarrow \beta$
  - Conjunction: $\alpha \land \beta \overset{def}{=} \neg (\neg \alpha \lor \neg \beta)$
  - Equivalence: $\alpha \equiv \beta \overset{def}{=} (\alpha \rightarrow \beta) \land (\beta \rightarrow \alpha)$. 
The remaining symbols are defines as:

- the truth constant $\bot$: $\bot \overset{\text{def}}{=} \neg \top$

- logical connectives:
  - disjunction: $\alpha \lor \beta \overset{\text{def}}{=} \neg \alpha \to \beta$
  - conjunction: $\alpha \land \beta \overset{\text{def}}{=} \neg (\neg \alpha \lor \neg \beta)$
  - equivalence: $\alpha \equiv \beta \overset{\text{def}}{=} (\alpha \to \beta) \land (\beta \to \alpha)$. 
The remaining symbols are defined as:

- the truth constant \( \bot \): \( \bot \overset{\text{def}}{=} \neg \top \)

- logical connectives:
  - disjunction: \( \alpha \lor \beta \overset{\text{def}}{=} \neg \alpha \rightarrow \beta \)
  - conjunction: \( \alpha \land \beta \overset{\text{def}}{=} \neg (\neg \alpha \lor \neg \beta) \)
  - equivalence: \( \alpha \equiv \beta \overset{\text{def}}{=} (\alpha \rightarrow \beta) \land (\beta \rightarrow \alpha) \).
Propositional Calculus (cont.)

The remaining symbols are defined as:

- the truth constant $\bot$ : $\bot \overset{def}{=} \neg \top$

- logical connectives:
  - disjunction : $\alpha \lor \beta \overset{def}{=} \neg \alpha \rightarrow \beta$
  - conjunction : $\alpha \land \beta \overset{def}{=} \neg (\neg \alpha \lor \neg \beta)$
  - equivalence : $\alpha \equiv \beta \overset{def}{=} (\alpha \rightarrow \beta) \land (\beta \rightarrow \alpha)$. 
Let \( \mathcal{L} \) be a language of propositional logic. An interpretation of \( \mathcal{L} \) is a mapping

\[
m : \text{Var} \rightarrow \{0, 1\}
\]

The mapping \( m \) is easily extended for the set \( \mathcal{L} \) of all formulas.
Semantics of $PC$

Let $\mathcal{L}$ be a language of propositional logic. An *interpretation* of $\mathcal{L}$ is a mapping

$$m : Var \rightarrow \{0, 1\}$$

The mapping $m$ is easily extended for the set $\mathcal{L}$ of all formulas.

A formula $\alpha \in \mathcal{L}$ is **true in** $m$ ($m$ is a **model** of $\alpha$), in symbols $m \models \alpha$, iff $m(\alpha) = 1$. 
Let \( \mathcal{L} \) be a language of propositional logic. An \textit{interpretation} of \( \mathcal{L} \) is a mapping

\[
m : \text{Var} \rightarrow \{0, 1\}
\]

The mapping \( m \) is easily extended for the set \( \mathcal{L} \) of all formulas.

A formula \( \alpha \in \mathcal{L} \) is \textit{true in} \( m \) (\( m \) is a \textit{model} of \( \alpha \)), in symbols \( m \models \alpha \), iff \( m(\alpha) = 1 \).

Let \( \alpha \in \mathcal{L} \) be a formula. We say that \( \alpha \) is

- \textit{satisfiable} iff it has a model
- \textit{tautology}, written \( \models \alpha \), iff every interpretation of \( \mathcal{L} \) is a model of \( \alpha \)
- \textit{unsatisfiable} if it has no model.
Most famous tautologies

| |= \( \alpha \lor \neg \alpha \) | Excluded Middle Law |
| |= \( \neg (\alpha \land \beta) \equiv \neg \alpha \lor \neg \beta \) | De Morgan Law |
| |= \( \neg (\alpha \lor \beta) \equiv \neg \alpha \land \neg \beta \) | De Morgan Law |
| |= \( \neg \neg \alpha \equiv \alpha \) | Double Negation Law |
| |= \( \alpha \lor (\beta \land \gamma) \equiv (\alpha \lor \beta) \land (\alpha \lor \gamma) \) | Distributive Law |
| |= \( \alpha \land (\beta \lor \gamma) \equiv (\alpha \land \beta) \lor (\alpha \land \gamma) \) | Distributive Law |
| |= \( \alpha \land \beta \equiv \beta \land \alpha \) | Commutative Law |
| |= \( \alpha \lor \beta \equiv \beta \lor \alpha \) | Commutative Law |
| |= \( \alpha \to \beta \equiv \neg \beta \to \neg \alpha \) | Contraposition Law |
The validity problem is the task to determine whether or not a given formula is a tautology.
The validity problem is the task to determine whether or not a given formula is a tautology.

In virtue of the truth–table method we have the following:

**Theorem 5.1** The validity problem for classical propositional calculus is decidable.
Let $\mathcal{L}$ be a language of (classical) propositional logic. Any subset $T \subseteq \mathcal{L}$ is called a *theory*. 
Let $\mathcal{L}$ be a language of (classical) propositional logic. Any subset $T \subseteq \mathcal{L}$ is called a \textit{theory}.

For a set $T \subseteq \mathcal{L}$ of formulas and the set $\mathcal{M}$ of interpretations of $\mathcal{L}$, $\mathcal{M} \models T$, means that every formula $\alpha \in T$ is true in every interpretation $m \in \mathcal{M}$. 
Let $\mathcal{L}$ be a language of (classical) propositional logic. Any subset $T \subseteq \mathcal{L}$ is called a **theory**.

For a set $T \subseteq \mathcal{L}$ of formulas and the set $\mathcal{M}$ of interpretations of $\mathcal{L}$, $\mathcal{M} \models T$, means that every formula $\alpha \in T$ is true in every interpretation $m \in \mathcal{M}$.

Two theories $T_1, T_1 \subseteq \mathcal{L}$ are called **equivalent**, written $T_1 \iff T_2$, iff $\text{Mod}(T_1) = \text{Mod}(T_2)$, where $\text{Mod}(T)$ is a set of all models of $T$. 
A reasoning rule

\[ r = \frac{\alpha_1, \ldots, \alpha_n}{\gamma} \]

is a partial mapping \( r : \mathcal{L}^n \rightarrow \mathcal{L} \). For \( \alpha_1, \ldots, \alpha_n \) from the domain of \( r \), \( \alpha_1, \ldots, \alpha_n \) are premises of \( r \) and \( \gamma = r(\alpha_1, \ldots, \alpha_n) \) is the consequence of \( r \).
A reasoning rule

\[
 r = \frac{\alpha_1, \ldots, \alpha_n}{\gamma}
\]

is a partial mapping \( r : \mathcal{L}^n \to \mathcal{L} \). For \( \alpha_1, \ldots, \alpha_n \) from the domain of \( r \), \( \alpha_1, \ldots, \alpha_n \) are \textit{premises} of \( r \) and \( \gamma = r(\alpha_1, \ldots, \alpha_n) \) is the \textit{consequence} of \( r \).

A reasoning rule \( \frac{\alpha_1, \ldots, \alpha_n}{\gamma} \) is called \textit{sound} iff \( \{\alpha_1, \ldots, \alpha_n\} \models \gamma \).
A *deduction system* (*axiomatization*) is a triple $DS = (\mathcal{L}, A, R)$, where

- $\mathcal{L}$ is a language of propositional logic,
- $A \subseteq \mathcal{L}$ is the set of *logical axioms*, and
- $R$ is the set of reasoning rules.
A **deduction system (axiomatization)** is a triple $DS = (\mathcal{L}, \mathcal{A}, \mathcal{R})$, where

- $\mathcal{L}$ is a language of propositional logic,
- $\mathcal{A} \subseteq \mathcal{L}$ is the set of **logical axioms**, and
- $\mathcal{R}$ is the set of reasoning rules.

$DS$ is called **sound** iff

- each formula $\alpha \in \mathcal{A}$ is a tautology
- each reasoning rule $r \in \mathcal{R}$ is sound.
Let $DS = (\mathcal{L}, \mathcal{A}, \mathcal{R})$ be an axiomatization, $T \subseteq \mathcal{L}$, and let $\alpha \in \mathcal{L}$.

A **formal proof** of $\alpha$ in $DS$ from $T$ is a sequence $(\alpha_0, \ldots, \alpha_k)$ of formulas such that

- $\alpha_0 \in \mathcal{A} \cup T$
- $\alpha_n = \alpha$
- for every $i = 1, \ldots, k$, either $\alpha_i \in \mathcal{A} \cup T$ or $\alpha_i$ is a direct consequence of $\alpha_0, \ldots, \alpha_{i-1}$ wrt some reasoning rule $r \in \mathcal{R}$. 
Let $DS = (L, A, R)$ be an axiomatization, $T \subseteq L$, and let $\alpha \in L$.

A **formal proof** of $\alpha$ in $DS$ from $T$ is a sequence $(\alpha_0, \ldots, \alpha_k)$ of formulas such that

- $\alpha_0 \in A \cup T$
- $\alpha_k = \alpha$
- for every $i = 1, \ldots, k$, either $\alpha_i \in A \cup T$ or $\alpha_i$ is a direct consequence of $\alpha_0, \ldots, \alpha_{i-1}$ wrt some reasoning rule $r \in R$.

$\alpha$ is called **derivable** from $T$ wrt $DS$, written $T \vdash_{DS} \alpha$, iff there exists a formal proof of $\alpha$ from $T$ in $DS$. 

Derivability
Derivability

Let $DS = (\mathcal{L}, \mathcal{A}, \mathcal{R})$ be an axiomatization, $T \subseteq \mathcal{L}$, and let $\alpha \in \mathcal{L}$.

- A **formal proof** of $\alpha$ in $DS$ from $T$ is a sequence $(\alpha_0, \ldots, \alpha_k)$ of formulas such that
  1. $\alpha_0 \in \mathcal{A} \cup T$
  2. $\alpha_n = \alpha$
  3. for every $i = 1, \ldots, k$, either $\alpha_i \in \mathcal{A} \cup T$ or $\alpha_i$ is a direct consequence of $\alpha_0, \ldots, \alpha_{i-1}$ wrt some reasoning rule $r \in \mathcal{R}$.

- $\alpha$ is called **derivable** from $T$ wrt $DS$, written $T \vdash_{DS} \alpha$, iff there exists a formal proof of $\alpha$ from $T$ in $DS$.

- Derivability operator $Th$: for any $T \subseteq \mathcal{L}$, $Th(T) = \{ \alpha : T \vdash_{DS} \alpha \}$.
Derivability

Let $DS = (\mathcal{L}, A, R)$ be an axiomatization, $T \subseteq \mathcal{L}$, and let $\alpha \in \mathcal{L}$.

- A **formal proof** of $\alpha$ in $DS$ from $T$ is a sequence $(\alpha_0, \ldots, \alpha_k)$ of formulas such that
  - $\alpha_0 \in A \cup T$
  - $\alpha_n = \alpha$
  - for every $i = 1, \ldots, k$, either $\alpha_i \in A \cup T$ or $\alpha_i$ is a direct consequence of $\alpha_0, \ldots, \alpha_{i-1}$ wrt some reasoning rule $r \in R$.

- $\alpha$ is called **derivable** from $T$ wrt $DS$, written $T \vdash_{DS} \alpha$, iff there exists a formal proof of $\alpha$ from $T$ in $DS$.

- Derivability operator $Th$: for any $T \subseteq \mathcal{L}$, $Th(T) = \{\alpha : T \vdash_{DS} \alpha\}$.

- $T$ is **consistent** iff $T \not\vdash \alpha$ for some formula $\alpha$. 

dr Anna M. Radzikowska, Knowledge Representation 6, – p. 38/41
Let $DS = (\mathcal{L}, \mathcal{A}, \mathcal{R})$ be an axiomatization, $T \subseteq \mathcal{L}$, and let $\alpha \in \mathcal{L}$.

A **formal proof** of $\alpha$ in $DS$ from $T$ is a sequence $(\alpha_0, \ldots, \alpha_k)$ of formulas such that

- $\alpha_0 \in \mathcal{A} \cup T$
- $\alpha_k = \alpha$
- for every $i = 1, \ldots, k$, either $\alpha_i \in \mathcal{A} \cup T$ or $\alpha_i$ is a direct consequence of $\alpha_0, \ldots, \alpha_{i-1}$ wrt some reasoning rule $r \in \mathcal{R}$.

$\alpha$ is called **derivable** from $T$ wrt $DS$, written $T \vdash_{DS} \alpha$, iff there exists a formal proof of $\alpha$ from $T$ in $DS$.

Derivability operator $Th$: for any $T \subseteq \mathcal{L}$, $Th(T) = \{ \alpha : T \vdash_{DS} \alpha \}$.

$T$ is **consistent** iff $T \not\vdash \alpha$ for some formula $\alpha$. 
Let $DS = (L, A, R)$ be an axiomatization, $T \subseteq L$, and let $\alpha \in L$.

- A **formal proof** of $\alpha$ in $DS$ from $T$ is a sequence $(\alpha_0, \ldots, \alpha_k)$ of formulas such that
  - $\alpha_0 \in A \cup T$
  - $\alpha_n = \alpha$
  - for every $i = 1, \ldots, k$, either $\alpha_i \in A \cup T$ or $\alpha_i$ is a direct consequence of $\alpha_0, \ldots, \alpha_{i-1}$ wrt some reasoning rule $r \in R$.

- $\alpha$ is called **derivable** from $T$ wrt $DS$, written $T \vdash_{DS} \alpha$, iff there exists a formal proof of $\alpha$ from $T$ in $DS$.

- Derivability operator $Th$: for any $T \subseteq L$, $Th(T) = \{ \alpha : T \vdash_{DS} \alpha \}$.

- $T$ is **consistent** iff $T \not\models \alpha$ for some formula $\alpha$. 
Derivability

Let $DS = (\mathcal{L}, \mathcal{A}, \mathcal{R})$ be an axiomatization, $T \subseteq \mathcal{L}$, and let $\alpha \in \mathcal{L}$.

- A formal proof of $\alpha$ in $DS$ from $T$ is a sequence $(\alpha_0, \ldots, \alpha_k)$ of formulas such that
  - $\alpha_0 \in \mathcal{A} \cup T$
  - $\alpha_n = \alpha$
  - for every $i = 1, \ldots, k$, either $\alpha_i \in \mathcal{A} \cup T$ or $\alpha_i$ is a direct consequence of $\alpha_0, \ldots, \alpha_{i-1}$ wrt some reasoning rule $r \in \mathcal{R}$.

- $\alpha$ is called derivable from $T$ wrt $DS$, written $T \vdash_{DS} \alpha$, iff there exists a formal proof of $\alpha$ from $T$ in $DS$.

- Derivability operator $Th$: for any $T \subseteq \mathcal{L}$, $Th(T) = \{ \alpha : T \vdash_{DS} \alpha \}$.

- $T$ is consistent iff $T \not\vdash \alpha$ for some formula $\alpha$. 
Let $DS = (\mathcal{L}, A, R)$ be an axiomatization.

$DS$ is **sound** iff for every theory $T \subseteq \mathcal{L}$ and for every $\alpha \in \mathcal{L}$,

\[ T \vdash_{DS} \alpha \implies T \models \alpha \]
Soundness and Completeness

Let $DS = (\mathcal{L}, A, \mathcal{R})$ be an axiomatization.

- $DS$ is **sound** iff for every theory $T \subseteq \mathcal{L}$ and for every $\alpha \in \mathcal{L}$,
  \[ T \vdash_{DS} \alpha \implies T \models \alpha \]

- $DS$ is called **complete** iff for every theory $T \subseteq \mathcal{L}$ and for every formula $\alpha \in \mathcal{L}$,
  \[ T \models \alpha \implies T \vdash_{DS} \alpha. \]
The most popular (sound and complete) axiomatization of propositional logic:

- logical axioms:
  - $\top$
  - $\alpha \rightarrow (\beta \rightarrow \alpha)$
  - $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$
  - $(\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow \neg \beta) \rightarrow \neg \alpha)$
Axiomatization (cont.)

The most popular (sound and complete) axiomatization of propositional logic:

- logical axioms:
  - $\top$
  - $\alpha \rightarrow (\beta \rightarrow \alpha)$
  - $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$
  - $(\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow \neg \beta) \rightarrow \neg \alpha)$

- Reasoning rule: *Modus Ponens* $\frac{\alpha, \alpha \rightarrow \beta}{\beta}$. 
The most popular (sound and complete) axiomatization of propositional logic:

- logical axioms:
  - \( \top \)
  - \( \alpha \rightarrow (\beta \rightarrow \alpha) \)
  - \( (\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma)) \)
  - \( (\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow \neg \beta) \rightarrow \neg \alpha) \)

- Reasoning rule: *Modus Ponens* \( \alpha, \alpha \rightarrow \beta \rightarrow \beta \).
The most popular (sound and complete) axiomatization of propositional logic:

- Logical axioms:
  - $\top$
  - $\alpha \rightarrow (\beta \rightarrow \alpha)$
  - $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$
  - $(\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow \neg \beta) \rightarrow \neg \alpha)$

- Reasoning rule: **Modus Ponens**
  $$\frac{\alpha, \alpha \rightarrow \beta}{\beta}.$$
Axiomatization (cont.)

The most popular (sound and complete) axiomatization of propositional logic:

- logical axioms:
  - $\top$
  - $\alpha \rightarrow (\beta \rightarrow \alpha)$
  - $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$
  - $(\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow \neg \beta) \rightarrow \neg \alpha)$

- Reasoning rule: **Modus Ponens**  

\[
\frac{\alpha, \alpha \rightarrow \beta}{\beta}
\]
The most popular (sound and complete) axiomatization of propositional logic:

- **logical axioms:**
  - $\top$
  - $\alpha \rightarrow (\beta \rightarrow \alpha)$
  - $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$
  - $(\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow \neg \beta) \rightarrow \neg \alpha)$

- **Reasoning rule:** *Modus Ponens* $\frac{\alpha, \alpha \rightarrow \beta}{\beta}$. 
Thank you for your attention!

Any questions are welcome.