Modal (Hennessy-Milner) logic: syntax

\[ \Phi ::= \text{tt} \mid \text{ff} \mid \Phi_1 \land \Phi_2 \mid \Phi_1 \lor \Phi_2 \mid [K]\Phi \mid \langle K \rangle \Phi \]

A formula can be

▶ the constant true formula \text{tt}
▶ the constant false formula \text{ff},
▶ a conjunction of formulas \(\Phi_1 \land \Phi_2\)
▶ a disjunction of formulas \(\Phi_1 \lor \Phi_2\),
▶ a formula \([K]\Phi\), where \(K\) is any set of actions, read as "box \(K\ \Phi\)", or "for all \(K\)-derivatives \(\Phi\),"
▶ a formula \(\langle K \rangle \Phi\), where \(K\) is any set of actions, read as "diamond \(K\ \Phi\)", or "for some \(K\)-derivative \(\Phi\)."
Modal (Hennessy-Milner) logic: syntax

\[ \Phi ::= \text{tt} \mid \text{ff} \mid \Phi_1 \land \Phi_2 \mid \Phi_1 \lor \Phi_2 \mid [K] \Phi \mid \langle K \rangle \Phi \]

A formula can be

- the constant true formula tt
- the constant false formula ff,
- a conjunction of formulas \( \Phi_1 \land \Phi_2 \)
- a disjunction of formulas \( \Phi_1 \lor \Phi_2 \),
- a formula \([K] \Phi\), where \( K \) is any set of actions, read as “box \( K \Phi\)”, or “for all \( K\)-derivatives \( \Phi\),”
- a formula \( \langle K \rangle \Phi\), where \( K \) is any set of actions, read as “diamond \( K \Phi\)”, or “for some \( K\)-derivative \( \Phi\).”

Modal (Hennessy-Milner) logic: semantics

We define when a process \( E \) satisfies a formula \( \Phi \). Either \( E \) satisfies \( \Phi \), denoted by \( E \models \Phi \), or it doesn’t, denoted by \( E \not\models \Phi \).

- \( E \models \text{tt} \) \( E \not\models \text{ff} \)
- \( E \models \Phi \) \iff \( E \not\models \Phi \) \iff \( E \models \Phi \land \Psi \) \iff \( E \models \Phi \) and \( E \models \Psi \)
- \( E \models \Phi \lor \Psi \) \iff \( E \models \Phi \) or \( E \models \Psi \)
We define when a process $E$ satisfies a formula $\Phi$. Either $E$ satisfies $\Phi$, denoted by $E \models \Phi$, or it doesn't, denoted by $E \not\models \Phi$.

- $E \models \top \quad E \not\models \bot$
- $E \models \Phi \land \Psi$ iff $E \models \Phi$ and $E \models \Psi$
- $E \models \Phi \lor \Psi$ iff $E \models \Phi$ or $E \models \Psi$
- $E \models [K]\Phi$ iff $\forall F \in \{E' : E \xrightarrow{a} E'\text{ and } a \in K\}. F \models \Phi$
- $E \models \langle K \rangle \Phi$ iff $\exists F \in \{E' : E \xrightarrow{a} E'\text{ and } a \in K\}. F \models \Phi$

A process $E$ has the property $[K]\Phi$ if every process which $E$ evolves to after carrying out any action in $K$ has the property $\Phi$.

A process $E$ has the property $\langle K \rangle \Phi$ if every process which $E$ evolves to after carrying out any action in $K$ has the property $\Phi$.

A process $E$ satisfies $[K]\Phi$ if $E$ can become a process that satisfies $\Phi$ by carrying out an action in $K$.
Examples

- \( E \models (\text{tick})tt \)
  - \( E \) can do a tick

- \( E \models (\text{tick})(\text{tock})tt \)
  - \( E \) can do a tick and then a tock

- \( E \models \{\text{tick}, \text{tock}\}tt \)
  - \( E \) can do a tick or a tock

- \( E \models [\text{tick}]tt \)
  - \( E \) cannot do a tick

This is equivalent to true!
Examples

- $E \models (\text{tick})tt$
  - $E$ can do a tick

- $E \models (\text{tick})(\text{tock})tt$
  - $E$ can do a tick and then a tock

- $E \models \{\text{tick}, \text{tock}\}tt$
  - $E$ can do a tick or a tock

- $E \models [\text{tick}]ff$
  - $E$ cannot do a tick

- $E \models (\text{tick})ff$
  - This is equivalent to $ff$!

- $E \models [\text{tick}]tt$
  - This is equivalent to $true$!

Checking satisfaction

$Cl \overset{\text{def}}{=} \text{tick.Cl}$

Does $Cl$ have the property: $[\text{tick}](\text{tick}tt \land [\text{tock}]ff)$?

- $Cl \models [\text{tock}]ff$
  - $iff \exists F \in \{Cl\} \land Cl \models [\text{tock}]ff$
  - $iff Cl \models [\text{tock}]ff$
  - $iff \{E : Cl \text{ tock} \rightarrow E\} = \emptyset$
  - $iff \emptyset = \emptyset$
Checking satisfaction

\[ C_l \overset{\text{def}}{=} \text{tick.Cl} \]

Does \( C_l \) have the property: \([\text{tick}](\langle \text{tick} \rangle \text{tt} \land [\text{tock}] \text{ff})\)?

\begin{align*}
& \quad \text{Cl} \models [\text{tick}](\langle \text{tick} \rangle \text{tt} \land [\text{tock}] \text{ff}) \\
\iff & \quad \exists F \in \{ C_l \overset{\text{tick}}{\to} E \}. F \models \langle \text{tick} \rangle \text{tt} \land [\text{tock}] \text{ff} \\
\iff & \quad \text{Cl} \models [\text{tock}] \text{ff} \\
\iff & \quad \{ E : C_l \overset{\text{tock}}{\to} E \} = \emptyset \\
\iff & \quad \emptyset = \emptyset
\end{align*}

Checking satisfaction

\[ C_l \overset{\text{def}}{=} \text{tick.Cl} \]

Does \( C_l \) have the property: \([\text{tick}](\langle \text{tick} \rangle \text{tt} \land [\text{tock}] \text{ff})\)?

\begin{align*}
& \quad \text{Cl} \models [\text{tick}](\langle \text{tick} \rangle \text{tt} \land [\text{tock}] \text{ff}) \\
\iff & \quad \exists F \in \{ C_l \overset{\text{tick}}{\to} E \}. F \models \langle \text{tick} \rangle \text{tt} \land [\text{tock}] \text{ff} \\
\iff & \quad \text{Cl} \models [\text{tock}] \text{ff} \\
\iff & \quad \{ E : C_l \overset{\text{tock}}{\to} E \} = \emptyset \\
\iff & \quad \emptyset = \emptyset
\end{align*}
Checking satisfaction

\( Cl \overset{\text{def}}{=} \text{tick.Cl} \)

Does \( Cl \) have the property: \([\text{tick}](\langle \text{tick} \rangle \text{tt} \land \langle \text{tock} \rangle \text{ff}) \) ?

\[ \begin{align*}
\text{Cl} & \models [\text{tick}](\langle \text{tick} \rangle \text{tt} \land \langle \text{tock} \rangle \text{ff}) \\
\text{iff} & \exists F \in \{ \text{Cl} \} \text{ and } \text{Cl} \models [\text{tock}]\text{ff} \\
\text{iff} & \text{Cl} \models [\text{tock}]\text{ff} \\
\text{iff} & \{ E : \text{Cl tock} \rightarrow E \} = \emptyset \\
\text{iff} & \emptyset = \emptyset
\end{align*} \]
Syntactic sugar for sets of actions

Let $A$ be a universal set of actions including $\tau$.
We write

- $a_1, \ldots, a_n$ for $\{a_1, \ldots, a_n\}$
- $\neg$ for the set $A$
- $\neg K$ for the set $A - K$
More examples

- $E \models \preceq \neg \mathbf{f}\mathbf{f}$
- $E \models \neg \mathbf{f}\mathbf{f}$
- $E$ is deadlocked, i.e., it cannot execute any action

- $E \models \preceq \neg \mathbf{t}\mathbf{t}$
- $E$ can execute some action

- $E \models \preceq \neg \mathbf{t}\mathbf{t} \land \neg \mathbf{a}\mathbf{f}\mathbf{f}$
- $E$ can execute some action

- $E \models \preceq \neg \mathbf{t}\mathbf{t} \land \neg \Phi$
- $\Phi$ holds after one step

- $E \models \preceq \neg \mathbf{t}\mathbf{t} \land \neg \left( \neg \mathbf{t}\mathbf{t} \land \neg \left( \neg \mathbf{t}\mathbf{t} \land \neg \mathbf{a}\mathbf{f}\mathbf{f} \right) \right)$
- $E$ can execute some action
More examples

- $E \models [\neg]ff$
- $E$ is deadlocked, i.e., it cannot execute any action
- $E \models \langle\neg\rangle tt$
- $E$ can execute some action
- $E \models \langle\neg\rangle tt \land [\neg a] ff$
- $a$ must happen next; something can happen, and nothing but $a$ can happen
- $E \models \langle\neg\rangle tt \land [\neg \Phi]$
- $\Phi$ holds after one step

More examples

- $E \models [\neg]ff$
- $E$ is deadlocked, i.e., it cannot execute any action
- $E \models \langle\neg\rangle tt$
- $E$ can execute some action
- $E \models \langle\neg\rangle tt \land [\neg a] ff$
- $a$ must happen next; something can happen, and nothing but $a$ can happen
- $E \models \langle\neg\rangle tt \land [\neg \Phi]$
- $\Phi$ holds after one step
More examples

- \( E \models [\neg]ff \)
- \( E \) is deadlocked, i.e., it cannot execute any action
- \( E \models (\neg)tt \)
- \( E \) can execute some action
- \( E \models (\neg)tt \land \neg a \)
- \( a \) must happen next; something can happen, and nothing but \( a \) can happen
- \( E \models (\neg)tt \land \neg \Phi \)
- \( \Phi \) holds after one step
- \( E \models (\neg)tt \land (\neg)tt \land (\neg)tt \land (\neg)a \)ff

Negation

HML can be extended with a negation operator \( \neg \) having the semantics: \( E \models \neg \Phi \) iff \( E \not\models \Phi \)

Exercise

<table>
<thead>
<tr>
<th>Process</th>
<th>Formula</th>
<th>Y/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a.0 + a.b.0 )</td>
<td>( \langle a \rangle \langle b \rangle tt )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \langle a \rangle \langle b \rangle ff )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \langle a \rangle tt )</td>
<td></td>
</tr>
<tr>
<td>( (a.0</td>
<td>\bar a.0) )</td>
<td>( \langle a \rangle tt )</td>
</tr>
<tr>
<td></td>
<td>( \langle a \rangle \tau tt )</td>
<td></td>
</tr>
<tr>
<td>( (a.0</td>
<td>\bar a.0) {a} )</td>
<td>( \langle a \rangle \tau tt )</td>
</tr>
</tbody>
</table>

Negation

HML can be extended with a negation operator \( \neg \) having the semantics: \( E \models \neg \Phi \) iff \( E \not\models \Phi \)

Negation is redundant in the following sense: For every formula \( \Phi \) of HML there is a formula \( \Phi^c \) such that for every process \( E \)

\[
E \models \Phi^c \quad \text{iff} \quad E \not\models \Phi
\]
Negation

HML can be extended with a negation operator \( \neg \) having the semantics:

\[
E \models \neg \Phi \iff E \not\models \Phi
\]

Negation is redundant in the following sense: For every formula \( \Phi \) of HML there is a formula \( \Phi^c \) such that for every process \( E \)

\[
E \models \Phi^c \iff E \not\models \Phi
\]

\( \Phi^c \) is inductively defined as follows:

\[
\begin{align*}
\text{tt}^c &= \text{ff} \\
\text{ff}^c &= \text{tt} \\
(\Phi_1 \land \Phi_2)^c &= \Phi_1^c \lor \Phi_2^c \\
(\Phi_1 \lor \Phi_2)^c &= \Phi_1^c \land \Phi_2^c \\
([K] \Phi)^c &= \langle K \rangle \Phi^c \\
(\langle K \rangle \Phi)^c &= [K] \Phi^c
\end{align*}
\]

Proposition: For every process \( F \) and HML-formula \( \Phi \):

\[
F \models \Phi^c \iff F \not\models \Phi
\]

Proof: By induction on the structure of \( \Phi \)

Basis: \( \Phi = \text{tt} \) and \( \Phi = \text{ff} \). Trivial.
Proposition: For every process $F$ and HML-formula $\Phi$:

\[ F \models \Phi^c \text{ iff } F \not\models \Phi. \]

Proof: By induction on the structure of $\Phi$

Basis: $\Phi = \tt$ and $\Phi = \ff$. Trivial.

Induction step:

Case $\Phi = \Phi_1 \land \Phi_2$

\[ F \models (\Phi_1 \land \Phi_2)^c \]
\[ \text{iff } F \models \Phi_1^c \lor \Phi_2^c \]
\[ \text{iff } F \not\models \Phi_1 \text{ or } F \not\models \Phi_2 \quad (\text{by clause for } \lor) \]
\[ \text{iff } \exists G. \exists a \in \mathcal{K}. F \xrightarrow{a} G \text{ and } G \models \Phi_1 \]
\[ \text{iff } \exists G. \exists a \in \mathcal{K}. F \xrightarrow{a} G \text{ and } G \not\models \Phi_1 \quad (\text{by i.h.}) \]
\[ \text{iff } F \not\models \mathcal{K}\Phi_1 \quad (\text{by clause for } \land). \]

Case $\Phi = [\mathcal{K}\Phi_1]$

\[ F \models ([\mathcal{K}\Phi_1])^c \]
\[ \text{iff } F \models \langle \mathcal{K} \rangle \Phi_1^c \]
\[ \text{iff } \exists G. \exists a \in \mathcal{K}. F \xrightarrow{a} G \text{ and } G \models \Phi_1^c \]
\[ \text{iff } \exists G. \exists a \in \mathcal{K}. F \xrightarrow{a} G \text{ and } G \not\models \Phi_1 \quad (\text{by i.h.}) \]
\[ \text{iff } F \not\models [\mathcal{K}\Phi_1] \]

Realisability, validity, equivalence

- A formula is satisfiable (or realisable) if some process satisfies it.
A formula is satisfiable (or realisable) if some process satisfies it.

A formula is unsatisfiable if no process satisfies it.

A formula is valid if all processes satisfy it.

Two formulas are equivalent if they are satisfied by exactly the same processes.

Exercise

Are the following statements true?

<table>
<thead>
<tr>
<th>Statement</th>
<th>Y/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>If $\Phi$ valid then $\Phi$ satisfiable</td>
<td></td>
</tr>
<tr>
<td>If $\Phi$ satisfiable then $\Phi^c$ unsatisfiable</td>
<td></td>
</tr>
<tr>
<td>If $\Phi$ valid then $\Phi^c$ unsatisfiable</td>
<td></td>
</tr>
<tr>
<td>If $\Phi$ unsatisfiable then $\Phi^c$ valid</td>
<td></td>
</tr>
</tbody>
</table>
Exercise

Let → be the implies connective whose definition is

\[ \Phi \rightarrow \Psi \overset{\text{def}}{=} \Phi^c \lor \Psi. \]

Are the following statements true?

<table>
<thead>
<tr>
<th></th>
<th>Y/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>If (\Phi \rightarrow \Psi) valid and \Phi valid then \Psi valid</td>
<td></td>
</tr>
<tr>
<td>If (\Phi \rightarrow \Psi) satisfiable and \Phi satisfiable then \Psi satisfiable</td>
<td></td>
</tr>
<tr>
<td>If (\Phi \rightarrow \Psi) valid and \Phi satisfiable then \Psi satisfiable</td>
<td></td>
</tr>
</tbody>
</table>

Exercise

Which of the following are valid, V, unsatisfiable, U, or neither, N?

<table>
<thead>
<tr>
<th></th>
<th>V</th>
<th>U</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>\Phi \rightarrow \neg \Phi</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\neg \Phi \rightarrow \Phi</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\Phi \rightarrow (\Psi \rightarrow \Phi)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\Phi \rightarrow (\Phi \rightarrow \Psi)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\langle a \rangle \text{tt} \land \langle a \rangle \text{ff}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\langle a \rangle \text{tt} \land \langle a \rangle \text{ff}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\langle a \rangle \text{tt} \land \langle a \rangle \text{ff} \land \neg \langle b \rangle \text{tt}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\langle a \rangle \text{tt} \land \langle a \rangle \text{ff} \land \neg \langle a \rangle \text{tt}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\langle a \rangle (\Phi \lor \Psi) \rightarrow (\langle a \rangle \Phi \lor \langle a \rangle \Psi)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\langle a \rangle \Phi \land \langle a \rangle \Psi) \rightarrow \langle a \rangle (\Phi \land \Psi)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\langle a \rangle (\Phi \rightarrow \Psi) \rightarrow ([a] \Phi \rightarrow [a] \Psi)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>