1 Modal Logic

As mentioned in the preface, we assume familiarity with the basic definitions concerning the syntax and semantics of modal logic. The purpose of this first chapter is to briefly recall notation and terminology. We focus on some aspects of modal logic that feature prominently in its extensions with fixpoint operators.

Convention 1.1 Throughout this text we let $\mathsf{Prop}$ be a countably infinite set of propositional variables, whose elements are usually denoted as $p, q, r, x, y, z, \ldots$, and we let $D$ be a finite set of (atomic) actions, whose elements are usually denoted as $d, e, c, \ldots$. We will usually focus on a finite subset $P$ of $\mathsf{Prop}$, consisting of those propositional variables that occur freely in a particular formula. In practice we will often suppress explicit reference to $\mathsf{Prop}$, $P$ and $D$.

1.1 Basics

Structures

Introduction LTSs as process graphs

Definition 1.2 A (labelled) transition system, LTS, or Kripke model of type $(P, D)$ is a triple $S = \langle S, V, R \rangle$ such that $S$ is a set of objects called states or points, $V : P \to \wp(S)$ is a valuation, and $R = \{R_d \subseteq S \times S \mid d \in D\}$ is a family of binary accessibility relations.

Elements of the set $R_d[s] := \{t \in S \mid (s, t) \in R_d\}$ are called $d$-successors of $s$. A transition system is called image-finite or finitely branching if $R_d[s]$ is finite, for every $d \in D$ and $s \in S$.

A pointed transition system or Kripke model is a pair $(S, s)$ consisting of a transition system $S$ and a designated state $s$ in $S$.

Remark 1.3 It will be convenient to work with an alternative, coalgebraic presentation of transition systems. Intuitively, it should be clear that instead of having a valuation $V : P \to \wp(S)$, telling us at which states each proposition letter is true, we could just as well have a marking $\sigma_V : S \to \wp(P)$ informing us which proposition letters are true at each state. Also, a binary relation $R$ on a set $S$ can be represented as a map $R[\cdot] : S \to \wp(S)$ mapping a state $s$ to the collection $R[s]$ of its successors. In this line, a family $R = \{R_d \subseteq S \times S \mid d \in D\}$ of accessibility relations can be seen as a map $\sigma_R : S \to \wp(S)^D$, where $\wp(S)^D$ denotes the set of maps from $D$ to $\wp(S)$.

Combining these two maps into one single function, we see that a transition system $S = \langle S, V, R \rangle$ of type $(P, D)$ can be seen as a pair $\langle S, \sigma \rangle$, where $\sigma : S \to \wp(P) \times \wp(S)^D$ is the map given by $\sigma(s) := (\sigma_V(s), \sigma_R(s))$.

For future reference we define the notion of a Kripke functor.

Definition 1.4 Fix a set $P$ of proposition letters and a set $D$ of atomic actions. Given a set $S$, let $K_{D,P}S$ denote the set

$$K_{D,P}S := \wp(P) \times \wp(S)^D.$$  

This operation will be called the Kripke functor associated with $D$ and $P$. 

A typical element of $K_D,P,S$ will be denoted as $(\pi, X)$, with $\pi \subseteq P$ and $X = \{X_d \mid d \in D\}$ with $X_d \subseteq S$ for each $d \in D$.

When we take this perspective we will sometimes refer to Kripke models as $K_D,P,S$-coalgebras or Kripke coalgebras.

Given this definition we may summarize Remark 1.3 by saying that any transition system can be presented as a pair $S = (S, \sigma : S \to KS)$ where $K$ is the Kripke functor associated with $S$. In practice, we will usually write $K$ rather than $K_{D,P}$.

**Syntax**

Working with fixpoint operators, we may benefit from a set-up in which the use of the negation symbol may only be applied to atomic formulas. The price that one has to pay for this is an enlarged arsenal of primitive symbols. In the context of modal logic we then arrive at the following definition.

**Definition 1.5** The language $ML_D$ of polymodal logic in $D$ is defined as follows:

$$\varphi := p \mid \neg p \mid \bot \mid \top \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \Diamond_d \varphi \mid \Box_d \varphi$$

where $p \in \text{Prop}$, and $d \in D$. Elements of $ML_D$ are called (poly-)modal formulas, or briefly, formulas. Formulas of the form $p$ or $\neg p$ are called literals. In case the set $D$ is a singleton, we speak of the language $ML_D$ of basic modal logic or monomodal logic; in this case we will denote the modal operators by $\Box$ and $\Diamond$, respectively.

Given a finite set $P$ of propositional variables, we let $ML_D(P)$ denote the set of formulas in which only variables from $P$ occur.

Often the sets $P$ and $D$ are implicitly understood, and suppressed in the notation. Generally it will suffice to treat examples, proofs, etc., from monomodal logic.

**Remark 1.6** The negation $\neg \varphi$ of a formula $\varphi$ can inductively be defined as follows:

$$\neg \bot := \top \quad \neg \top := \bot \quad \neg \neg p := p \quad \neg (\varphi \lor \psi) := \neg \varphi \land \neg \psi \quad \neg (\varphi \land \psi) := \neg \varphi \lor \neg \psi \quad \neg \Diamond_d \varphi := \Diamond_d \neg \varphi \quad \neg \Box_d \varphi := \Box_d \neg \varphi$$

On the basis of this, we can also define the other standard abbreviated connectives, such as $\rightarrow$ and $\leftrightarrow$.

We assume that the reader is familiar with standard syntactic notions such as those of a subformula or the construction tree of a formula, and with standard syntactic operations such as substitution. Concerning the latter, we let $\varphi[\psi/p]$ denote the formula that we obtain by substituting all occurrences of $p$ in $\varphi$ by $\psi$. 

\end{document}
Definition 1.7 We define the collection $Sfor(\xi)$ of subformulas of a modal formula $\varphi$ by the following induction on the complexity of $\varphi$:

$$
\begin{align*}
Sfor(\bot) & := \{\bot\} \\
Sfor(\top) & := \{\top\} \\
Sfor(p) & := \{p\} \\
Sfor(\neg p) & := \{p, \neg p\} \\
FV(\varphi \ast \psi) & := \{\varphi \ast \psi\} \cup Sfor(\varphi) \cup Sfor(\psi) \quad \text{where } \ast \in \{\lor, \land\} \\
Sfor(\Diamond \varphi) & := \{\Diamond \varphi\} \cup Sfor(\varphi) \quad \text{where } \Diamond \in \{\Diamond_d, \Box_d | d \in D\}
\end{align*}
$$

We write $\varphi \sqsubseteq \psi$ to denote that $\varphi$ is a subformula of $\psi$. The size of a formula $\xi$ is defined as the number of its subformulas, $|\xi| := |Sfor(\xi)|$.  

Semantics

The relational semantics of modal logic is well known. The basic idea is that the modal operators $\Diamond_d$ and $\Box_d$ are both interpreted using the accessibility relation $R_d$.

The notion of truth (or satisfaction) is defined as follows.

Definition 1.8 Let $S = (S, \sigma)$ be a transition system of type $(P, D)$. Then the satisfaction relation $\models$ between states of $S$ and formulas of $\text{ML}_D(P)$ is defined by the following formula induction.

$$
\begin{align*}
S, s \models p & \quad \text{if } s \in V(p), \\
S, s \models \neg p & \quad \text{if } s \notin V(p), \\
S, s \models \bot & \quad \text{never}, \\
S, s \models \top & \quad \text{always}, \\
S, s \models \varphi \lor \psi & \quad \text{if } S, s \models \varphi \text{ or } S, s \models \psi, \\
S, s \models \varphi \land \psi & \quad \text{if } S, s \models \varphi \text{ and } S, s \models \psi, \\
S, s \models \Diamond_d \varphi & \quad \text{if } S, t \models \varphi \text{ for some } t \in R_d[s], \\
S, s \models \Box_d \varphi & \quad \text{if } S, t \models \varphi \text{ for all } t \in R_d[s].
\end{align*}
$$

We say that $\varphi$ is true or holds at $s$ if $S, s \models \varphi$, and we let the set

$$
[\varphi]^S := \{s \in S \mid S, s \models \varphi\}.
$$

We may define the semantics of modal formulas directly in terms of this meaning function $[\varphi]^S$. This approach has some advantages in the context of fixpoint operators, since it brings out the role of the powerset algebra $\varphi(S)$ more clearly.
Remark 1.9 Fix an LTS $S$, then define $[\varphi]^S$ by induction on the complexity of $\varphi$:

\[
\begin{align*}
[p]^S &= V(p) \\
[\neg p]^S &= S \setminus V(p) \\
[\bot]^S &= \emptyset \\
[T]^S &= S \\
[\varphi \lor \psi]^S &= [\varphi]^S \cup [\psi]^S \\
[\varphi \land \psi]^S &= [\varphi]^S \cap [\psi]^S \\
[\Diamond d\varphi]^S &= [R_d][\varphi]^S \\
[\square d\varphi]^S &= [R_d][\varphi]^S
\end{align*}
\]

Here the operations $\langle R_d \rangle$ and $[R_d]$ on $\psi(S)$ are defined by putting

\[
\begin{align*}
\langle R \rangle(X) &= \{s \in S \mid R_d[s] \cap X \neq \emptyset\} \\
[R](X) &= \{s \in S \mid R_d[s] \subseteq X\}.
\end{align*}
\]

The satisfaction relation $\models$ may be recovered from this by putting $S, s \models \varphi$ iff $s \in [\varphi]^S$. $\Box$

Definition 1.10 Let $s$ and $s'$ be two states in the transition systems $S$ and $S'$ of type $(P, D)$, respectively. Then we say that $s$ and $s'$ are modally equivalent, notation: $S, s \equiv_{(P, D)} S', s'$, if $s$ and $s'$ satisfy the same modal formulas, that is, $S, s \models \varphi$ iff $S', s' \models \varphi$, for all modal formulas $\varphi \in \text{ML}_D(P)$. $\Box$

Flows, trees and streams

In some parts of these notes deterministic transition systems feature prominently.

Definition 1.11 A transition system $S = \langle S, V, R \rangle$ is called deterministic if each $R_d[s]$ is a singleton. $\Box$

Note that our definition of determinism does not allow $R_d = \emptyset$ for any point $s$. We first consider the monomodal case.

Definition 1.12 Let $P$ be a set of proposition letters. A deterministic monomodal Kripke model for this language is called a flow model for $P$, or a $\varphi(P)$-flow. In case such a structure is of the form $\langle \omega, V, \text{Succ} \rangle$, where $\text{Succ}$ is the standard successor relation on the set $\omega$ of natural numbers, we call the structure a stream model for $P$, or a $\varphi(P)$-stream. $\Box$

In case the set $D$ of actions is finite, we may just as well identify it with the set $k = \{0, \ldots, k-1\}$, where $k$ is the size of $D$. We usually restrict to the binary case, that is, $k = 2$. Our main interest will be in Kripke models that are based on the binary tree, i.e., a tree in which every node has exactly two successors, a left and a right one.

Definition 1.13 With $2 = \{0, 1\}$, we let $2^*$ denote the set of finite strings of 0s and 1s. We let $\varepsilon$ denote the empty string, while the left- and right successor of a node $s$ are denoted by $s \cdot 0$ and $s \cdot 1$, respectively. Written as a relation, we put

$$\text{Succ}_i = \{(s, s \cdot i) \mid s \in 2^*\}.$$
A binary tree over \( P \), or a binary \( \wp(P) \)-tree is a Kripke model of the form \( \langle 2^*, V, \text{Succ}_0, \text{Succ}_1 \rangle \).

**Remark 1.14** In the general case, the \( k \)-ary tree is the structure \( \langle k^*, \text{Succ}_0, \ldots, \text{Succ}_{k-1} \rangle \), where \( k^* \) is the set of finite sequences of natural numbers smaller than \( k \), and \( \text{Succ}_i \) is the \( i \)-th successor relation given by

\[
\text{Succ}_i = \{(s, s \cdot i) \mid s \in k^*\}.
\]

A \( k \)-flow model is a Kripke model \( S = \langle S, V, R \rangle \) with \( k \) many deterministic accessibility relations, and a \( k \)-ary tree model is a \( k \)-flow model which is based on the \( k \)-ary tree.

In deterministic transition systems, the distinction between boxes and diamonds evaporates. It is then convenient to use a single symbol \( \Box \) to denote either the box or the diamond.

**Definition 1.15** The set \( \text{MFL}_k(P) \) of formulas of \( k \)-ary Modal Flow Logic in \( P \) is given as follows:

\[
\varphi ::= p \mid \neg p \mid \bot \mid \top \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \Diamond_i \varphi
\]

where \( p \in P \), and \( i < k \). In case \( k = 1 \) we will also speak of modal stream logic, notation: \( \text{MSL}(P) \).

### 1.2 Game semantics

We will now describe the semantics defined above in game-theoretic terms. That is, we will define the evaluation game \( E(\xi, S) \) associated with a (fixed) formula \( \xi \) and a (fixed) LTS \( S \). This game is an example of a board game. In a nutshell, board games are games in which the players move a token along the edge relation of some graph, so that a match of the game corresponds to a (finite or infinite) path through the graph. Furthermore, the winning conditions of a match are determined by the nature of this path. We will meet many examples of board games in these notes, and in Chapter 5 we will study them in more detail.

The evaluation game \( E(\xi, S) \) is played by two players: Éloïse (\( \exists \) or 0) and Abélard (\( \forall \) or 1). Given a player \( \Pi \), we always denote the opponent of \( \Pi \) by \( \overline{\Pi} \). As mentioned, a match of the game consists of the two players moving a token from one position to another. Positions are of the form \( (\varphi, s) \), with \( \varphi \) a subformula of \( \xi \), and \( s \) a state of \( S \).

It is useful to assign goals to both players: in an arbitrary position \( (\varphi, s) \), think of \( \exists \) trying to show that \( \varphi \) is true at \( s \) in \( S \), and of \( \forall \) of trying to convince her that \( \varphi \) is false at \( s \).

Depending on the type of the position (more precisely, on the formula part of the position), one of the two players may move the token to a next position. For instance, in a position of the form \( (\Diamond_d \varphi, s) \), it is \( \exists \)'s turn to move, and she must choose an arbitrary \( d \)-successor \( t \) of \( s \), thus making \( (\varphi, t) \) the next position. Intuitively, the idea is that in order to show that \( \Diamond \varphi \) is true at \( s \), \( \exists \) has to come up with a successor of \( s \) where \( \varphi \) holds. Formally, we say that the set of (admissible) next positions that \( \exists \) may choose from is given as the set \( \{(\varphi, t) \mid t \in R_d[s]\} \).

In the case there is no successor of \( s \) to choose, she immediately loses the game. This is a convenient way to formulate the rules for winning and losing this game: if a position \( (\varphi, s) \)
<table>
<thead>
<tr>
<th>Position</th>
<th>Player</th>
<th>Admissible moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\varphi_1 \lor \varphi_2, s)$</td>
<td>$\exists$</td>
<td>${(\varphi_1, s), (\varphi_2, s)}$</td>
</tr>
<tr>
<td>$(\varphi_1 \land \varphi_2, s)$</td>
<td>$\forall$</td>
<td>${(\varphi_1, s), (\varphi_2, s)}$</td>
</tr>
<tr>
<td>$(\Box_d \varphi, s)$</td>
<td>$\exists$</td>
<td>${(\varphi, t) \mid t \in R_d[s]}$</td>
</tr>
<tr>
<td>$(\square_d \varphi, s)$</td>
<td>$\forall$</td>
<td>${(\varphi, t) \mid t \in R_d[s]}$</td>
</tr>
<tr>
<td>$(\perp, s)$</td>
<td>$\forall$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$(\top, s)$</td>
<td>$\forall$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$(p, s), s \in V(p)$</td>
<td>$\forall$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$(p, s), s \notin V(p)$</td>
<td>$\exists$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$(\neg p, s), s \notin V(p)$</td>
<td>$\forall$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$(\neg p, s), s \in V(p)$</td>
<td>$\exists$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

Table 1: Evaluation game for modal logic

has no admissible next positions, the player whose turn it is to play at $(\varphi, s)$ gets stuck and immediately loses the game.

This convention gives us a nice handle on positions of the form $(p, s)$ where $p$ is a proposition letter: we always assign to such a position an empty set of admissible moves, but we make $\exists$ responsible for $(p, s)$ in case $p$ is false at $s$, and $\forall$ in case $p$ is true at $s$. In this way, $\exists$ immediately wins if $p$ is true at $s$, and $\forall$ if it is otherwise. The rules for the negative literals $(\neg p)$ and the constants, $\perp$ and $\top$, follow a similar pattern.

The full set of rules of the game is given in Table 1. Observe that all matches of this game are finite, since at each move of the game the active formula is reduced in size. (From the general perspective of board games, this means that we need not worry about winning conditions for matches of infinite length.) We may now summarize the game as follows.

**Definition 1.16** Given a modal formula $\xi$ and a transition system $S$, the evaluation game $E(\xi, S)$ is defined as the board game given by Table 1, with the set $Sfor(\xi) \times S$ providing the positions of the game; that is, a position is a pair consisting of a subformula of $\xi$ and a point in $S$. The instantiation of this game with starting point $(\xi, s)$ is denoted as $E(\xi, S)@((\xi, s))$.

An instance of an evaluation game is a pair consisting of an evaluation game and a starting position of the game. Such an instance will also be called an initialized game, or sometimes, if no confusion is likely, simply a game.

A strategy for a player $\Pi$ in an initialized game is a method that $\Pi$ uses to select his moves during the play. Such a strategy is winning for $\Pi$ if every match of the game (starting at the given position) is won by $\Pi$, provided $\Pi$ plays according to this strategy. A position $(\varphi, s)$ is winning for $\Pi$ if $\Pi$ has a winning strategy for the game initialized in that position. (This is independent of whether it is $\Pi$’s turn to move at the position.) The set of winning positions in $E(\xi, S)$ for $\Pi$ is denoted as $\text{Win}_{\Pi}(E(\xi, S))$.

The main result concerning these games is that they provide an alternative, but equivalent, semantics for modal logic.

- Example to be added
Theorem 1.17 Let $\xi$ be a modal formula, and let $\mathcal{S}$ be an LTS. Then for any state $s$ in $\mathcal{S}$ it holds that

$$(\xi, s) \in \text{Win}_3(\mathcal{F}(\xi, \mathcal{S})) \iff \mathcal{S}, s \models \xi.$$ 

The proof of this Theorem is left to the reader.

1.3 Bisimulations and bisimilarity

One of the most fundamental notions in the model theory of modal logic is that of a bisimulation between two transition systems.

> discuss bisimilarity as a notion of behavioral equivalence

Definition 1.18 Let $\mathcal{S}$ and $\mathcal{S}'$ be two transition systems of the same type $(P, D)$. Then a relation $Z \subseteq S \times S'$ is a bisimulation of type $(P, D)$ if the following hold, for every $(s, s') \in Z$.

- **(prop)** $s \in V(p)$ iff $s' \in V'(p)$, for all $p \in P$;
- **(forth)** for all actions $d$, and for all $t \in R_d[s]$ there is a $t' \in R'_d[s']$ with $(t, t') \in Z$;
- **(back)** for all actions $d$, and for all $t' \in R'_d[s']$ there is a $t \in R_d[s]$ with $(t, t') \in Z$.

Two states $s$ and $s'$ are called bisimilar, notation: $\mathcal{S}, s \equiv_{P, D} \mathcal{S}', s'$ if there is some bisimulation $Z$ of type $(P, D)$ with $(s, s') \in Z$. If no confusion is likely to arise, we generally drop the subscripts, writing ‘$\equiv$’ rather than ‘$\equiv_{P, D}$’.

Bisimilarity and modal equivalence

In order to understand the importance of this notion for modal logic, the starting point should be the observation that the truth of modal formulas is invariant under bisimilarity. Recall that $\equiv$ denotes the relation of modal equivalence.

Theorem 1.19 (Bisimulation Invariance) Let $\mathcal{S}$ and $\mathcal{S}'$ be two transition systems of the same type. Then

$$\mathcal{S}, s \equiv \mathcal{S}', s' \Rightarrow \mathcal{S}, s \equiv \mathcal{S}', s'$$

for every pair of states $s$ in $\mathcal{S}$ and $s'$ in $\mathcal{S}'$.

Proof. By a straightforward induction on the complexity of modal formulas one proves that bisimilar states satisfy the same formulas. QED

But there is much more to say about the relation between modal logic and bisimilarity than Theorem 1.19. In particular, for some classes of models, one may prove a converse statement, which amounts to saying that the notions of bisimilarity and modal equivalence coincide. Such classes are said to have the Hennessy-Milner property. As an example we mention the class of finitely branching transition systems.
Theorem 1.20 (Hennessy-Milner Property) Let $\mathcal{S}$ and $\mathcal{S}'$ be two finitely branching transition systems of the same type. Then
\[ \mathcal{S}, s \leftrightarrow \mathcal{S}', s' \iff \mathcal{S}, s \overset{\sim}{\leftrightarrow} \mathcal{S}', s' \]
for every pair of states $s$ in $\mathcal{S}$ and $s'$ in $\mathcal{S}'$.

Proof. The direction from left to right follows from Theorem 1.19. In order to prove the opposite direction, one may show that the relation $\overset{\sim}{\leftrightarrow}$ of modal equivalence itself is a bisimulation. Details are left to the reader. \(\text{QED}\)

This theorem can be read as indication of the expressiveness of modal logic: any difference in behaviour between two states in finitely branching transition systems can in fact be witnessed by a concrete modal formula. As another witness to this expressivity, in section 1.5 we will see that modal logic is sufficiently rich to express all bisimulation-invariant first-order properties. Obviously, this result also adds considerable strength to the link between modal logic and bisimilarity.

As a corollary of the bisimulation invariance theorem, modal logic has the tree model property, that is, every satisfiable modal formula is satisfiable on a structure that has the shape of a tree.

Definition 1.21 A transition system $\mathcal{S}$ of type $(P,D)$ is called tree-like if the structure $\langle \mathcal{S}, \bigcup_{d \in D} R_d \rangle$ is a tree.

The key step in the proof of the tree model property of modal logic is the observation that every transition system can be ‘unravelled’ into a bisimilar tree-like model. The basic idea of such an unravelling is the new states encode (part of) the history of the old states. Technically, the new states are the paths through the old system.

Definition 1.22 Let $\mathcal{S} = \langle S, V, R \rangle$ be a transition system of type $(P,D)$. A (finite) path through $\mathcal{S}$ is a nonempty sequence of the form $(s_0, d_1, s_1, \ldots, d_n, s_n)$ such that $R_{d_i} s_{i-1} s_i$ for all $i$ with $0 < i \leq n$. The set of paths through $\mathcal{S}$ is denoted as $\text{Paths}(\mathcal{S})$; we use the notation $\text{Paths}_s(\mathcal{S})$ for the set of paths starting at $s$.

The unravelling of $\mathcal{S}$ around a state $s$ is the transition system $\mathcal{\tilde{S}}_s$ which is coalgebraically defined as the structure $\langle \text{Paths}_s(\mathcal{S}), \tilde{\sigma} \rangle$, where the coalgebra map $\tilde{\sigma} = (\tilde{\sigma}_V, (\tilde{\sigma}_d | d \in D))$ is given by putting
\[ \tilde{\sigma}_V(s_0, d_1, s_1, \ldots, d_n, s_n) := \sigma_V(s_n), \]
\[ \tilde{\sigma}_d(s_0, d_1, s_1, \ldots, d_n, s_n) := \{(s_0, d_1, s_1, \ldots, d_n, s_n, d, t) \in \text{Paths}_s(\mathcal{S}) | R_d s_n t\}. \]

Finally, the unravelling of a pointed transition system $(\mathcal{S}, s)$ is the pointed structure $(\mathcal{\tilde{S}}_s, s)$, where (with some abuse of notation) we let $s$ denote the path of length zero that starts and finishes at $s$.

Clearly, unravellings are tree-like structures, and any pointed transition system is bisimilar to its unravelling. But then the following theorem is immediate by Theorem 1.19.

Theorem 1.23 (Tree Model Property) Let $\varphi$ be a satisfiable modal formula. Then $\varphi$ is satisfiable at the root of a tree-like model.
Bisimilarity game

We may also give a game-theoretic characterization of the notion of bisimilarity. We first give an informal description of the game that we will employ. A match of the bisimilarity game between two Kripke models $S$ and $S'$ is played by two players, $\exists$ and $\forall$. As in the evaluation game, these players move a token around from one position of the game to the next one. In the game there are two kinds of positions: pairs of the form $(s, s') \in S \times S'$ are called basic positions and belong to $\exists$. The other positions are of the form $Z \subseteq S \times S'$ and belong to $\forall$.

The idea of the game is that at a position $(s, s')$, $\exists$ claims that $s$ and $s'$ are bisimilar, and to substantiate this claim she proposes a local bisimulation $Z \subseteq S \times S'$ (see below) for $s$ and $s'$. This relation $Z$ can be seen as providing a set of witnesses for $\exists$’s claim that $s$ and $s'$ are bisimilar. Implicitly, $\exists$’s claim at a position $Z \subseteq S \times S'$ is that all pairs in $Z$ are bisimilar, so $\forall$ can pick an arbitrary pair $(t, t') \in Z$ and challenge $\exists$ to show that these $t$ and $t'$ are bisimilar.

Definition 1.24 Let $S$ and $S'$ be two transition systems of the same type $(P, D)$. Then a relation $Z \subseteq S \times S'$ is a local bisimulation for two points $s \in S$ and $s' \in S'$, if it satisfies the properties (prop), (back) and (forth) of Definition 1.18 for this specific $s$ and $s'$.

If a player gets stuck in a match of this game, then the opponent wins the match. For instance, if $s$ and $s'$ disagree about some proposition letter, then there is no local bisimulation for $s$ and $s'$, and so the corresponding position $(s, s')$ is an immediate loss for $\exists$. Or, if neither $s$ nor $s'$ has successors, and agree on the truth of all proposition letters, then $\exists$ could choose the empty relation as a local bisimulation, so that $\forall$ would lose the match at his next move.

A new option arises if neither player gets stuck: this game may also have matches that last forever. Nevertheless, we can still declare a winner for such matches, and the agreement is that $\exists$ is the winner of any infinite match. Formally, we put the following.

Definition 1.25 The bisimilarity game $B(S, S')$ between two Kripke models $S$ and $S'$ is the board game given by Table 2, with the winning condition that finite matches are lost by the player who got stuck, while all infinite matches are won by $\exists$.

A position $(s, s')$ is winning for $\Pi$ if $\Pi$ has a winning strategy for the game initialized in that position. The set of these positions is denoted as $\text{Win}_\Pi(B(S, S'))$.

<table>
<thead>
<tr>
<th>Position</th>
<th>Player</th>
<th>Admissible moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(s, s') \in S \times S'$</td>
<td>$\exists$</td>
<td>${Z \in \wp(S \times S') \mid Z$ is a local bisimulation for $s$ and $s'}$</td>
</tr>
<tr>
<td>$Z \in \wp(S \times S')$</td>
<td>$\forall$</td>
<td>$Z = {(t, t') \mid (t, t') \in Z}$</td>
</tr>
</tbody>
</table>

Table 2: Bisimilarity game for Kripke models

Also observe that a bisimulation is a relation which is a local bisimulation for each of its members. The following theorem states that the collection of basic winning positions for $\exists$ forms the largest bisimulation between $S$ and $S'$.
Theorem 1.26 Let $(S, s)$ and $(S', s')$ be two pointed Kripke models. Then $S, s \leftrightarrow S', s'$ iff $(s, s') \in \text{Win}_\exists (B(S, S'))$.

Proof. For the direction from left to right: suppose that $Z$ is a bisimulation between $S$ and $S'$ linking $s$ and $s'$. Suppose that $\exists$, starting from position $(s, s')$, always chooses the relation $Z$ itself as the local bisimulation. A straightforward verification, by induction on the length of the match, shows that this strategy always provides her with a legitimate move, and that it keeps her alive forever. This proves that it is a winning strategy.

For the converse direction, it suffices to show that the relation \{(t, t') \in S \times S' \mid (t, t') \in \text{Win}_\exists (B(S, S'))\} itself is in fact a bisimulation. We leave the details for the reader. QED

Remark 1.27 The bisimilarity game should not be confused with the bisimulation game.

Bisimulations via relation lifting

Together, the back- and forth clause of the definition of a bisimulation express that the pair of respective successor sets of two bisimilar states must belong to the so-called Egli-Milner lifting $\overline{\pi}Z$ of the bisimulation $Z$. In fact, the notion of a bisimulation can be completely defined in terms of relation lifting.

Definition 1.28 Given a relation $Z \subseteq A \times A'$, define the relation $\overline{\pi}Z \subseteq \wp A \times \wp A'$ as follows:

$$\overline{\pi}Z := \{(X, X') \mid \text{for all } x \in X \text{ there is an } x' \in X' \text{ with } (x, x') \in Z$$
& $$\text{for all } x' \in X' \text{ there is an } x \in X \text{ with } (x, x') \in Z\}.$$  

Similarly, define, for a Kripke functor $K = K_{D, p}$, the relation $\overline{K}Z \subseteq KA \times KA'$ as follows:

$$\overline{K}Z := \{((\pi, X), (\pi', X')) \mid \pi = \pi' \text{ and } (X_d, X_d') \in \overline{\pi}Z \text{ for each } d \in D\}.$$  

The relations $\overline{\pi}Z$ and $\overline{K}Z$ are called the lifting of $Z$ with respect to $\wp$ and $K$, respectively. We say that $Z \subseteq A \times A'$ is full on $B \in \wp A$ and $B' \in \wp A'$ if $(B, B') \in \overline{\pi}Z$. QED

It is completely straightforward to check that a nonempty relation $Z$ linking two transition systems $S$ and $S'$ is a local bisimulation for two states $s$ and $s'$ iff $(\sigma(s), \sigma'(s')) \in \overline{K}Z$. In particular, $\exists$’s move in the bisimilarity game at a position $(s, s')$ consists of choosing a binary relation $Z$ such that $(\sigma(s), \sigma'(s')) \in \overline{K}Z$. The following characterization of bisimulations is also an immediate consequence.

Proposition 1.29 Let $S$ and $S'$ be two Kripke coalgebras for some Kripke functor $K$, and let $Z \subseteq S \times S'$ be some relation. Then

$$Z \text{ is a bisimulation iff } (\sigma(s), \sigma'(s')) \in \overline{K}Z \text{ for all } (s, s') \in Z.$$  

(1)
1.4 Finite models and computational aspects

- complexity of model checking
- filtration & polysize model property
- complexity of satisfiability
- complexity of global consequence

1.5 Modal logic and first-order logic

- modal logic is the bisimulation invariant fragment of first-order logic

1.6 Completeness

1.7 The cover modality

As we will see now, there is an interesting alternative for the standard formulation of basic modal logic in terms of boxes and diamonds. This alternative set-up is based on a connective which turns sets of formulas into formulas.

Definition 1.30 Let \( \Phi \) be a finite set of formulas. Then \( \nabla \Phi \) is a formula, which holds at a state \( s \) in a Kripke model if every formula in \( \Phi \) holds at some successor of \( s \), while at the same time, every successor of \( s \) makes some formula in \( \Phi \) true. The operator \( \nabla \) is called the cover modality.

It is not so hard to see that the cover modality can be defined in the standard modal language:

\[
\nabla \Phi \equiv \Box \bigvee \Phi \wedge \bigwedge \Diamond \Phi,
\]

where \( \Diamond \Phi \) denotes the set \( \{ \Diamond \varphi \mid \varphi \in \Phi \} \). Things start to get interesting once we realize that both the ordinary diamond \( \Diamond \) and the ordinary box \( \Box \) can be expressed in terms of the cover modality (and the disjunction):

\[
\begin{align*}
\Diamond \varphi &\equiv \nabla \{ \varphi, \top \}, \\
\Box \varphi &\equiv \nabla \emptyset \vee \nabla \{ \varphi \}.
\end{align*}
\]

Remark 1.31 Observe that this definition involves the \( \forall \exists \& \forall \exists \) pattern that we know from the notion of relation lifting \( \nabla \Phi \) defined in the previous section. In other words, the semantics of the cover modality can be expressed in terms of relation lifting. For that purpose, observe that we may think of the forcing or satisfaction relation \( \models \) simply as a binary relation between states and formulas. The we find that

\[
S, s \models \nabla \Phi \iff (\sigma_R(s), \Phi) \in \nabla(\models).
\]

for any pointed Kripke model \((S, s)\) and any finite set \( \Phi \) of formulas.
Given that $\nabla$ and $\{\Diamond, \Box\}$ are mutually expressible, we arrive at the following definition and proposition.

**Definition 1.32** Formulas of the language $\text{ML}_\nabla$ are given by the following recursive definition:

$$\varphi ::= \ p \mid \neg p \mid \bot \mid T \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \nabla \Phi$$

where $\Phi$ denotes a finite set of formulas.

**Proposition 1.33** The languages $\text{ML}$ and $\text{ML}_\nabla$ are equally expressive.

**Proof.** Immediate by (2) and (3).

The real importance of the cover modality is that it allows us to almost completely eliminate the Boolean conjunction. This remarkable fact is based on the following modal distributive law. Recall from Definition 1.28 that a relation $Z \subseteq A \times A'$ is full on $A$ and $A'$ if $(A, A') \in \nabla Z$.

**Proposition 1.34** Let $\Phi$ and $\Phi'$ be two sets of formulas. Then the following two formulas are equivalent:

$$\nabla \varphi \land \nabla \varphi' \equiv \bigvee \{\nabla \Gamma_Z \mid Z \text{ is full on } \Phi \text{ and } \Phi'\}, \quad (4)$$

where, given a relation $Z \subseteq \Phi \times \Phi'$, we define

$$\Gamma_Z := \{\varphi \land \varphi' \mid (\varphi, \varphi') \in Z\}.$$

**Proof.** For the direction from left to right, suppose that $S, s \models \nabla \varphi \land \nabla \varphi'$. Let $Z \subseteq \Phi \times \Phi'$ consist of those pairs $(\varphi, \varphi')$ such that the conjunction $\varphi \land \varphi'$ is true at some successor $t$ of $s$. It is then straightforward to verify that $Z$ is full on $\Phi$ and $\Phi'$, and that $S, s \models \nabla \Gamma_Z$.

The converse direction follows fairly directly from the definitions.

As a corollary of Proposition 1.34 we can restrict the use of conjunction in modal logic to that of a special conjunction connective $\bullet$ which may only be applied to a propositional formula and a certain conjunction of $\nabla$-formula.

**Definition 1.35** Fix finite sets $P$ of proposition letters and $D$ of atomic actions, respectively. We first define the set $\text{CL}(P)$ of literal conjunctions by the following grammar:

$$\pi ::= p \mid \neg p \mid \bot \mid T \mid \pi \land \pi.$$

Next, let $\Phi = \{\Phi_d \mid d \in D\}$ be a $D$-indexed family of formulas, and write $\nabla_D \Phi := \bigwedge_{d \in D} \nabla_d \Phi_d$, where $\nabla_d$ is the cover modality associated with the accessibility relation $R_d$ of $d$.

Finally, the following grammar:

$$\varphi ::= \bot \mid T \mid \varphi \lor \varphi \mid \pi \cdot \nabla_D \Phi.$$

defines the set $\text{DML}_D(P)$ of disjunctive polymodal formulas in $D$ and $P$. 😡
The following theorem states that every modal formula can be rewritten into an equivalent disjunctive normal form.

**Theorem 1.36** For any \( P \) and \( D \), the languages \( \text{ML}_D(P) \) and \( \text{DML}_D(P) \) are expressively equivalent.

We leave the proof of this result as an exercise to the reader.

**Notes**

Modal logic has a long history in philosophy and mathematics, for an overview we refer to Blackburn, de Rijke and Venema [3]. The use of modal formalisms as specification languages in process theory goes back at least to the 1970s, with Pratt [26] and Pnueli [25] being two influential early papers.

The notion of bisimulation, which plays an important role in modal logic and process theory alike, was first introduced in a modal logic context by van Benthem [2], who proved that modal logic is the bisimulation invariant fragment of first-order logic. The notion was later, but independently, introduced in a process theory setting by Park [24]. At the time of writing we do not know who first took a game-theoretical perspective on the semantics of modal logic. The cover modality \( \nabla \) was introduced independently by Moss [17] and Janin & Walukiewicz [10].

Readers who want to study modal logic in more detail are referred to Blackburn, de Rijke and Venema [3] or Chagrov & Zakharyaschev [5].

**Exercises**

**Exercise 1.1** Prove Theorem 1.17.

**Exercise 1.2 (bisimilarity game)** Consider the following version \( \mathcal{B}_\omega(S, S') \) of the bisimilarity game between two transition systems \( S \) and \( S' \). Positions of this game are of the form either \((s, s', \forall, \alpha)\), \((s, s', \exists, \alpha)\) or \((Z, \alpha)\), with \( s \in S, s' \in S', Z \subseteq S \times S' \) and \( \alpha \) either a natural number or \( \omega \). The admissible moves for \( \exists \) and \( \forall \) are displayed in the following table:

<table>
<thead>
<tr>
<th>Position</th>
<th>Player</th>
<th>Admissible moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>((s, s', \forall, \alpha))</td>
<td>(\forall)</td>
<td>({(s, s', \exists, \beta) \mid \beta &lt; \alpha})</td>
</tr>
<tr>
<td>((s, s', \exists, \alpha))</td>
<td>(\exists)</td>
<td>({(Z, \alpha) \mid Z \text{ is a local bisimulation for } s \text{ and } s'})</td>
</tr>
<tr>
<td>((Z, \alpha))</td>
<td>(\forall)</td>
<td>({(s, s', \forall, \alpha) \mid (s, s') \in Z})</td>
</tr>
</tbody>
</table>

Note that all matches of this game have finite length.

We write \( S, s \leftrightarrow_\omega S', s' \) to denote that \( \exists \) has a winning strategy in the game \( \mathcal{B}_\omega(S, S') \) starting at position \((s, s', \forall, \alpha)\). It is not hard to see that \( S, s \leftrightarrow_\omega S', s' \) iff \( S, s \leftrightarrow_k S', s' \) for all \( k < \omega \).

(a) Give concrete examples such that \( S, s \leftrightarrow_\omega S', s' \) but not \( S, s \leftrightarrow S', s' \).

(Hint: think of two modally equivalent but not bisimilar states.)
(b) Let $k \geq 0$ be a natural number. Prove that, for all $S, s$ and $S', s'$:

$$S, s \equiv_k S', s' \Rightarrow S, s \equiv_{\sim k} S', s'.$$

Here $\equiv_k$ denotes the modal equivalence relation with respect to formulas of modal depth at most $k$. Here we use a slightly nonstandard notion of modal depth, defined as follows:

$$d(\bot) = 0, d(\top) := 0, d(p) := 1 \text{ for } p \in \mathbb{P}, d(\varphi \wedge \psi), d(\varphi \lor \psi) := \max(d(\varphi), d(\psi)), \text{ and } d(\Diamond \varphi), d(\Box \varphi) := 1 + d(\varphi).$$

(c) Let $S$ and $S'$ be finitely branching transition systems. Prove directly (i.e., without using part (b)) that (i) $\Rightarrow$ (ii), for all $s \in S$ and $s' \in S'$:

(i) $S, s \equiv_\omega S', s'$

(ii) $S, s \equiv S', s'$.

(d) Does the implication in (c) hold in the case that only one of the two transition systems is finitely branching?

**Exercise 1.3** Let $\Phi$ and $\Theta$ be finite sets of formulas. Prove that

$$\nabla(\Phi \cup \{\lor \Theta\}) \equiv \bigvee \{\nabla(\Phi \cup \Theta') \mid \emptyset \neq \Theta' \subseteq \Theta\}.$$

**Exercise 1.4** Prove Theorem 1.36.