

Part I
Modal Logic

1 Basics

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, and to provide an introduction to the coalgebraic perspective on modal logic (this perspective will not play a role until Chapter ??).

Convention 1.1 Throughout this text we let \mathbf{P} be a set of *proposition letters*, whose elements are usually denoted as p, q, r, x, y, z, \dots , and let \mathbf{D} be a set of (atomic) *actions*, whose elements are usually denoted as d, e, c, \dots .

1.1 Basics

Structures

- Introduce LTSs as process graphs

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

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 $d \in \mathbf{D}$ and $s \in S$, and *deterministic* if each $R_d[s]$ is a singleton.

A *pointed* transition system or Kripke model is a pair (\mathbb{S}, s) consisting of a transition system \mathbb{S} and a designated state s in \mathbb{S} . ◁

In practice we will often suppress explicit reference to \mathbf{P} and \mathbf{D} .

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

Combining these two maps into one single function, we see that a transition system $\mathbb{S} = \langle S, V, R \rangle$ of type (\mathbf{P}, \mathbf{D}) can be seen as a pair $\langle S, \sigma \rangle$, where $\sigma : S \rightarrow \wp(\mathbf{P}) \times \wp(S)^{\mathbf{D}}$ 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 \mathbf{P} of proposition letters and a set \mathbf{D} of atomic actions. Given a set S , let $K_{\mathbf{D},\mathbf{P}}S$ denote the set

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

This operation will be called the *Kripke functor* associated with \mathbf{D} and \mathbf{P} .

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

When we take this perspective we will sometimes refer to Kripke models as $K_{\mathbf{D},\mathbf{P}}S$ -*coalgebras* or *Kripke coalgebras*. \triangleleft

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

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 set $\text{PML}_{\mathbf{D}}(\mathbf{P})$ of *Polymodal Logic* in \mathbf{D} and \mathbf{P} as follows:

$$\varphi ::= p \mid \neg p \mid \perp \mid \top \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \diamond_d \varphi \mid \square_d \varphi$$

where $p \in \mathbf{P}$, and $d \in \mathbf{D}$. Elements of $\text{PML}_{\mathbf{D}}(\mathbf{P})$ are called (*poly-*)*modal formulas*, or briefly, *formulas*. Formulas of the form p or $\neg p$ are called *literals*.

In case the set \mathbf{D} is a singleton, we speak of the language $\text{BML}(\mathbf{P})$ of *Basic Modal Logic*; in this case we will denote the modal operators by \diamond and \square , respectively. \triangleleft

Often the sets \mathbf{P} and \mathbf{D} are implicitly understood, and suppressed in the notation.

Remark 1.6 Generally it will suffice to treat examples, proofs, etc., from basic modal logic. \triangleleft

Remark 1.7 The *negation* $\sim\varphi$ of a formula φ can inductively be defined as follows:

$$\begin{array}{ll} \sim\perp & := \top & \sim\top & := \perp \\ \sim p & := \neg p & \sim\neg p & := p \\ \sim(\varphi \vee \psi) & := \sim\varphi \wedge \sim\psi & \sim(\varphi \wedge \psi) & := \sim\varphi \vee \sim\psi \\ \sim\square_d \varphi & := \diamond_d \sim\varphi & \sim\diamond_d \varphi & := \square_d \sim\varphi \end{array}$$

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

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 φ by ψ .

Semantics

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

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

Definition 1.8 Let $\mathbb{S} = \langle S, \sigma \rangle$ be a transition system of type (P, D) . Then the *satisfaction relation* \Vdash between states of \mathbb{S} and formulas of PML is defined by the following formula induction.

$$\begin{array}{ll}
\mathbb{S}, s \Vdash p & \text{if } s \in V(p), \\
\mathbb{S}, s \Vdash \neg p & \text{if } s \notin V(p), \\
\mathbb{S}, s \Vdash \perp & \text{never,} \\
\mathbb{S}, s \Vdash \top & \text{always,} \\
\mathbb{S}, s \Vdash \varphi \vee \psi & \text{if } \mathbb{S}, s \Vdash \varphi \text{ or } \mathbb{S}, s \Vdash \psi, \\
\mathbb{S}, s \Vdash \varphi \wedge \psi & \text{if } \mathbb{S}, s \Vdash \varphi \text{ and } \mathbb{S}, s \Vdash \psi, \\
\mathbb{S}, s \Vdash \diamond_d \varphi & \text{if } \mathbb{S}, t \Vdash \varphi \text{ for some } t \in R_d[s], \\
\mathbb{S}, s \Vdash \square_d \varphi & \text{if } \mathbb{S}, t \Vdash \varphi \text{ for all } t \in R_d[s].
\end{array}$$

We say that φ is *true* or *holds* at s if $\mathbb{S}, s \Vdash \varphi$, and we let the set

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

denote the *meaning* or *extension* of φ in \mathbb{S} . ◁

Alternatively (but equivalently), one may define the semantics of modal formulas directly in terms of this meaning function $[[\varphi]]^{\mathbb{S}}$. This approach has some advantages in the context of fixpoint operators, since it brings out the role of the powerset algebra $\wp(S)$ more clearly.

Remark 1.9 Fix an LTS \mathbb{S} , then define $[[\varphi]]^{\mathbb{S}}$ by induction on the complexity of φ , where the operations $\langle R_d \rangle$ and $[R_d]$ are defined in Appendix A:

$$\begin{array}{ll}
[[p]]^{\mathbb{S}} & = V(p) \\
[[\neg p]]^{\mathbb{S}} & = S \setminus V(p) \\
[[\perp]]^{\mathbb{S}} & = \emptyset \\
[[\top]]^{\mathbb{S}} & = S \\
[[\varphi \vee \psi]]^{\mathbb{S}} & = [[\varphi]]^{\mathbb{S}} \cup [[\psi]]^{\mathbb{S}} \\
[[\varphi \wedge \psi]]^{\mathbb{S}} & = [[\varphi]]^{\mathbb{S}} \cap [[\psi]]^{\mathbb{S}} \\
[[\diamond_d \varphi]]^{\mathbb{S}} & = \langle R_d \rangle [[\varphi]]^{\mathbb{S}} \\
[[\square_d \varphi]]^{\mathbb{S}} & = [R_d] [[\varphi]]^{\mathbb{S}}
\end{array}$$

The satisfaction relation \Vdash may be recovered from this by putting $\mathbb{S}, s \Vdash \varphi$ iff $s \in \llbracket \varphi \rrbracket^{\mathbb{S}}$.
 \triangleleft

Definition 1.10 Let s and s' be two states in the transition systems \mathbb{S} and \mathbb{S}' of type (\mathbf{P}, \mathbf{D}) , respectively. Then we say that s and s' are *modally equivalent*, notation: $\mathbb{S}, s \rightsquigarrow_{(\mathbf{P}, \mathbf{D})} \mathbb{S}', s'$, if s and s' satisfy the same modal formulas, that is, $\mathbb{S}, s \Vdash \varphi$ iff $\mathbb{S}', s' \Vdash \varphi$, for all modal formulas $\varphi \in \text{PML}_{\mathbf{D}}(\mathbf{P})$. \triangleleft

Flows, trees and streams

In these notes we will devote a lot of attention to *deterministic* transition systems, and in particular, the ones that are trees. In case the set \mathbf{D} of actions is finite, we may just as well identify it with the set $k = \{0, \dots, k-1\}$, where k is the size of \mathbf{D} . We introduce some notation and terminology.

Definition 1.11 Let \mathbf{P} be a set of proposition letters, and let $k \geq 1$ be some natural number. A deterministic transition system \mathbb{S} of type (\mathbf{P}, k) is called a *k-ary flow over \mathbf{P}* , or, in case $k = 1$, simply a *flow over \mathbf{P}* . \triangleleft

As we shall see further on, of particular interest are the flows that have the shape of a *tree*. We first discuss the case $k = 1$, where we speak of *streams* rather than trees.

Definition 1.12 Let \mathbf{P} be a set of proposition letters. A Kripke model of the form $\langle \omega, V, \text{Succ} \rangle$, where Succ is the standard successor relation on the set ω of natural numbers, will be called a *stream over \mathbf{P}* , or a $\wp(\mathbf{P})$ -*stream*. \triangleleft

In case $k = 2$ we will be very interested in Kripke models that are based on the *binary tree*, i.e., a trees 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 ϵ denotes the empty string, while the left- and right successor of a node s are denoted by $s0$ and $s1$, respectively. Written as a relation, we put

$$\text{Succ}_i = \{(s, si) \mid s \in 2^*\}.$$

A *binary tree over \mathbf{P}* , or a *binary \mathbf{P} -tree* is a Kripke model of the form $\langle 2^*, V, \text{Succ}_0, \text{Succ}_1 \rangle$.
 \triangleleft

For simplicity we will usually restrict to the binary case.

Remark 1.14 In the general case, the k -ary tree is the structure $(k^*, Succ_0, \dots, Succ_{k-1})$, where k^* is the set of finite sequences of natural numbers smaller than k , and $Succ_i$ is the i -th successor relation given by

$$Succ_i = \{(\pi, \pi i) \mid s \in k^*\}.$$

A k -ary flow $\mathbb{S} = \langle S, V, R \rangle$ is called a k -ary tree flow if (S, R) is the k -ary tree. \triangleleft

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

Definition 1.15 The set $MFL_k(\mathbf{P})$ of formulas of k -ary Modal Flow Logic in \mathbf{P} is given as follows:

$$\varphi ::= p \mid \neg p \mid \perp \mid \top \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \circ_i \varphi$$

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

1.2 Game semantics

We will now describe the semantics defined above in game-theoretic terms. That is, we will define the *evaluation game* $\mathcal{E}(\xi, \mathbb{S})$ associated with a (fixed) formula ξ and a (fixed) LTS \mathbb{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 ?? we will study them in more detail.

The evaluation game $\mathcal{E}(\xi, \mathbb{S})$ is played by two *players*: Éloise (\exists or 0) and Abélard (\forall or 1). Given a player σ , we always denote the *opponent* of σ by $\bar{\sigma}$. 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 (φ, s) , with φ a *subformula* of ξ , and s a state of \mathbb{S} .

It is useful to assign *goals* to both players: in an arbitrary position (φ, s) , think of \exists trying to show that φ is *true* at s in \mathbb{S} , and of \forall of trying to convince her that φ 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 (φ, 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 φ 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 (φ, s) has no admissible next positions, the player whose turn it is to play at (φ, s) 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 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 ξ and a transition system \mathbb{S} , the *evaluation game* $\mathcal{E}(\xi, \mathbb{S})$ is defined as the board game given by Table 1. The instantiation of this game with starting point (ξ, s) is denoted as $\mathcal{E}(\xi, \mathbb{S})@(\xi, s)$. \triangleleft

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

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

Theorem 1.17 *Let ξ be a modal formula, and let \mathbb{S} be an LTS. Then for any state s in \mathbb{S} it holds that*

$$(\xi, s) \in \text{Win}_\exists(\mathcal{E}(\xi, \mathbb{S})) \iff \mathbb{S}, s \Vdash \xi.$$

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.

Position	Player	Admissible moves
$(\varphi_1 \vee \varphi_2, s)$	\exists	$\{(\varphi_1, s), (\varphi_2, s)\}$
$(\varphi_1 \wedge \varphi_2, s)$	\forall	$\{(\varphi_1, s), (\varphi_2, s)\}$
$(\diamond_d \varphi, s)$	\exists	$\{(\varphi, t) \mid t \in R_d[s]\}$
$(\square_d \varphi, s)$	\forall	$\{(\varphi, t) \mid t \in R_d[s]\}$
(\perp, s)	\exists	\emptyset
(\top, s)	\forall	\emptyset
$(p, s), s \in V(p)$	\forall	\emptyset
$(p, s), s \notin V(p)$	\exists	\emptyset
$(\neg p, s), s \notin V(p)$	\forall	\emptyset
$(\neg p, s), s \in V(p)$	\exists	\emptyset

Table 1: Evaluation game for modal logic

► discuss bisimilarity as a notion of behavioral equivalence

Definition 1.18 Let \mathbb{S} and \mathbb{S}' be two transition systems of the same type (P, D) . Then a non-empty relation $Z \subseteq S \times S'$ is a *bisimulation* 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: $\mathbb{S}, s \Leftrightarrow \mathbb{S}', s'$ if there is some bisimulation Z with $(s, s') \in Z$.

Relations satisfying the back and forth clauses, but the (prop) clause only for a subset $Q \subseteq P$ are called *Q-bisimulations*, and the corresponding notion of bisimilarity is denoted by \Leftrightarrow_Q . ◁

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 \rightsquigarrow denotes the relation of modal equivalence.

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

$$\mathbb{S}, s \Leftrightarrow \mathbb{S}', s' \Rightarrow \mathbb{S}, s \rightsquigarrow \mathbb{S}', s'$$

for every pair of states s in \mathbb{S} and s' in \mathbb{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 *Hennessey-Milner* property. As an example we mention the class of finitely branching transition systems.

Theorem 1.20 (Hennessey-Milner Property) *Let \mathbb{S} and \mathbb{S}' be two finitely branching transition systems of the same type. Then*

$$\mathbb{S}, s \leftrightarrow \mathbb{S}', s' \iff \mathbb{S}, s \rightsquigarrow \mathbb{S}', s'$$

for every pair of states s in \mathbb{S} and s' in \mathbb{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 \rightsquigarrow of modal equivalence itself is a bisimulation. Details are left to the reader. 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. For the definition of a path through a relational structure, and that of a tree, we refer to the appendix.

Definition 1.21 A transition system \mathbb{S} of type (P, D) is called *tree-like* if the structure $\langle S, \bigcup_{d \in D} R_d \rangle$ 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 ‘unravalled’ 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 $\mathbb{S} = \langle S, V, R \rangle$ be a transition system of type (P, D) . A *path* through \mathbb{S} is a nonempty sequence of the form $(s_0, d_1, s_1, \dots, d_n, s_n)$ such that $R_{d_i} s_{i-1} s_i$ for all $i \leq n$. The set of paths through \mathbb{S} is denoted as $Paths(\mathbb{S})$; we use the notation $Paths_s(\mathbb{S})$ for the set of paths starting at s .

The *unravelling* of \mathbb{S} around a state s is the transition system $\vec{\mathbb{S}}_s$ which is coalgebraically defined as the structure $\langle Paths_s(\mathbb{S}), \vec{\sigma} \rangle$, where the coalgebra map $\vec{\sigma} = (\sigma_V, (\sigma_d \mid d \in D))$ is defined by putting

$$\begin{aligned} \vec{\sigma}_V(s_0, d_1, s_1, \dots, d_n, s_n) &:= \sigma_V(s_n), \\ \vec{\sigma}_d(s_0, d_1, s_1, \dots, d_n, s_n) &:= \{(s_0, d_1, s_1, \dots, d_n, s_n, d, t) \in Paths_s(\mathbb{S}) \mid R_d s_n t\}. \end{aligned}$$

Finally, the unravelling of a pointed transition system (\mathbb{S}, s) is the pointed structure $(\vec{\mathbb{S}}_s, (s))$, where (s) denotes the empty path starting and finishing at s . \triangleleft

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 φ be a satisfiable modal formula. Then φ 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. A match of the *bisimilarity game* between two Kripke models \mathbb{S} and \mathbb{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 (see below) for s and s' . \forall then challenges her by picking a pair $(t, t') \in Z$ as the next basic position.

Definition 1.24 Let \mathbb{S} and \mathbb{S}' be two transition systems of the same type (P, D) . Then a non-empty 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' . \triangleleft

Note that if s and s' disagree about the truth of some proposition letter, then there is *no* local bisimulation for s and s' . Also observe that a bisimulation is a relation which is a local bisimulation for each of its members.

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.

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 the corresponding

position 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* $\mathcal{B}(\mathbb{S}, \mathbb{S}')$ between two Kripke models \mathbb{S} and \mathbb{S}' is the board game given by Table 2, where the winning condition for infinite matches is simply that \exists wins all infinite matches. \triangleleft

Position	Player	Admissible moves
$(s, s') \in S \times S'$	\exists	$\{Z \in \wp(S \times S') \mid Z \text{ is a local bisimulation for } s \text{ and } s'\}$
$Z \in \wp(S \times S')$	\forall	$Z = \{(t, t') \mid (t, t') \in Z\}$

Table 2: Bisimilarity game for Kripke models

The following theorem states that the collection of basic winning positions for \exists forms the *largest bisimulation* between \mathbb{S} and \mathbb{S}' .

Theorem 1.26 *Let (\mathbb{S}, s) and (\mathbb{S}', s') be two pointed Kripke models. Then $\mathbb{S}, s \Leftrightarrow \mathbb{S}', s'$ iff $(s, s') \in \text{Win}_{\exists}(\mathcal{B}(\mathbb{S}, \mathbb{S}'))$.*

Proof. For the direction from left to right: suppose that Z is a bisimulation between \mathbb{S} and \mathbb{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 $\text{Win}_{\exists}(\mathcal{B}(\mathbb{S}, \mathbb{S}'))$ itself is in fact a bisimulation. We leave the details for the reader. QED

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* $\bar{\wp}Z$ of the bisimulation Z . In fact, the notion of a bisimulation can be completely defined in terms of *relation lifting*.

Definition 1.27 Given a relation $Z \subseteq A \times A'$, define the relation $\bar{\wp}Z \subseteq \wp A \times \wp A'$ as follows:

$$\bar{\wp}Z := \{(X, X') \mid \begin{array}{l} \text{for all } x \in X \text{ there is an } x' \in X' \text{ with } (x, x') \in Z \\ \& \text{for all } x' \in X' \text{ there is an } x \in X \text{ with } (x, x') \in Z \end{array}\}.$$

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

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

The relations $\bar{\wp}Z$ and $\bar{K}Z$ 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 $B' \in \wp A'$, notation: $Z \in B \bowtie B'$, if $(B, B') \in \bar{\wp}Z$. \triangleleft

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

Proposition 1.28 *Let \mathbb{S} and \mathbb{S}' be two Kripke coalgebras for some Kripke functor K , and let $Z \subseteq S \times S'$ be some relation. Then*

$$Z \text{ is a bisimulation iff } (\sigma(s), \sigma'(s')) \in \bar{K}Z \text{ for all } (s, s') \in Z. \quad (1)$$

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 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 1.29 Let Φ 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 Φ holds at *some* successor of s , while at the same time, *every* successor of s makes *some* formula in Φ true. The operator ∇ is called the *cover modality*. \triangleleft

Observe that this definition involves the $\forall\exists\&\forall\exists$ pattern that we know from the notion of *relation lifting* $\bar{\varphi}$ 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 \Vdash simply as a binary relation between states and formulas.

Proposition 1.30 Let s be some state in a Kripke model \mathbb{S} , and let Φ be a set of formulas. Then

$$\mathbb{S}, s \Vdash \nabla\Phi \text{ iff } (\sigma_R(s), \Phi) \in \bar{\varphi}(\Vdash).$$

Proof. Immediate by unravelling the definitions. QED

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

$$\nabla\Phi \equiv \Box \bigvee \Phi \wedge \bigwedge \Diamond\Phi, \quad (2)$$

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):

$$\begin{aligned} \Diamond\varphi &\equiv \nabla\{\varphi, \top\}, \\ \Box\varphi &\equiv \nabla\emptyset \vee \nabla\{\varphi\}. \end{aligned} \quad (3)$$

Here, as always, we use the convention that $\bigvee\emptyset = \perp$ and $\bigwedge\emptyset = \top$.

Making the above observations more precise, we arrive at the following definition and proposition.

Definition 1.31 Formulas of the language BML_{∇} are given by the following recursive definition:

$$\varphi ::= p \mid \neg p \mid \perp \mid \top \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \nabla\Phi$$

where Φ denotes a finite set of formulas. \triangleleft

Proposition 1.32 The languages BML and BML_{∇} are equally expressive.

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

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 distributive law. Recall from Definition 1.27 that we write $Z \in A \bowtie A'$ if a relation $Z \subseteq A \times A'$ is *full on* A and A' , that is, if $(A, A') \in \overline{\text{p}}Z$.

Proposition 1.33 *For all pairs Φ, Φ' of sets of formulas, the following two formulas are equivalent:*

$$\nabla\Phi \wedge \nabla\Phi' \equiv \bigvee_{Z \in \Phi \bowtie \Phi'} \nabla\{\varphi \wedge \varphi' \mid (\varphi, \varphi') \in Z\}. \quad (4)$$

Proof. For the direction from left to right, suppose that $\mathbb{S}, s \Vdash \nabla\Phi \wedge \nabla\Phi'$. Let $Z \subseteq \Phi \times \Phi'$ consist of those pairs (φ, φ') such that the conjunction $\varphi \wedge \varphi'$ is true at some successor t of s . It is then straightforward to verify that Z is full on Φ and Φ' , and that $\mathbb{S}, s \Vdash \nabla\{\varphi \wedge \varphi' \mid (\varphi, \varphi') \in Z\}$.

The converse direction follows fairly directly from the definitions. QED

1.7 Coalgebraic modal logic

Using the cover modality introduced in the previous section, we can show that we can restrict the use of conjunction in modal logic to that of the *special conjunction* connective \bullet . First however, we take care of the proposition letters.

Definition 1.34 Fix a finite set P of proposition letters. Given a subset $\pi \subseteq P$, we let $\odot\pi$ denote the formula with semantics given by

$$\mathbb{S}, s \Vdash \odot\pi \text{ iff } \sigma_V(s) = \pi$$

for any $K_{D,P}$ -coalgebra $\mathbb{S} = \langle S, \sigma \rangle$. \triangleleft

In words, the formula $\odot\pi$ holds at a state s iff π consists precisely of those proposition letters in P that are true at s , or equivalently,

$$\odot\pi := \bigwedge_{p \in \pi} p \wedge \bigwedge_{p \notin \pi} \neg p.$$

It is not difficult to see that every propositional formula with proposition letters from P can be expressed as disjunctions of formulas of the form $\odot\pi$. In particular, it is straightforward to verify that

$$q \equiv \bigvee_{q \in \pi} \odot\pi$$

for every $q \in P$.

We are now ready for the introduction of the coalgebraic modal connective \bullet .

Definition 1.35 Fix finite sets P of proposition letters and D of atomic actions, respectively. Given a subset $\pi \subseteq P$, and a D -indexed family $\Phi = \{\Phi_d \mid d \in D\}$ of formulas, then $\pi \bullet \Phi$ is a formula, of which the semantics is defined by the following equivalence:

$$\pi \bullet \Phi \equiv \odot \pi \wedge \bigwedge_{d \in D} \nabla_d \Phi_d.$$

Here ∇_d is the cover modality associated with the accessibility relation R_d of d .

The set $\text{CML}_D(P)$ of *coalgebraic modal formulas* is given as follows:

$$\varphi ::= \perp \mid \top \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \pi \bullet \Phi. \quad \triangleleft$$

In words, $\pi \bullet \Phi$ is the conjunction of (i) a complete description of the *local* situation in terms of the proposition letters being true or false, and (ii) for each action d , a description of the d -successor set of the current state, using the cover modality for R_d .

► Explain why \bullet and the language CML are called ‘coalgebraic’.

Proposition 1.36 Fix sets P of proposition letters and D of atomic actions, respectively. Then we have

$$\mathbb{S}, s \Vdash \pi \bullet \Phi \text{ iff } (\sigma(s), (\pi, \Phi)) \in \overline{K}(\Vdash) \quad (5)$$

for any $\pi \in \wp(P)$ any D -indexed family $\Phi = \{\Phi_d \mid d \in D\}$, and for any pointed transition system (\mathbb{S}, s) of type (P, D) , the following equivalence holds.

Proof. Simply spell out the definitions. QED

In fact, we could have taken (5) below as the *definition* of the semantics of the bullet modality.

The following result is not very hard to prove.

Theorem 1.37 For any P and D , the languages $\text{PML}_D(P)$ and $\text{CML}_D(P)$ are expressively equivalent.

Proof. There is a straightforward translation from CML-formulas to ordinary modal formulas, so we focus on the other direction.

It is not hard to verify that every polymodal formula can be rewritten to an equivalent formula using the connectives $\top, \perp, \wedge, \vee, \odot$ and ∇_d . But then the proof of the theorem is straightforward by the observation that both $\odot \pi$ and $\nabla_d \Phi$ can be rewritten to formulas using the bullet connective, using the equivalences below:

$$\begin{aligned} \top &\equiv \nabla_d \emptyset \vee \nabla_d \{\top\} \\ \top &\equiv \bigvee_{\pi \subseteq P} \odot \pi. \end{aligned}$$

For instance, this allows us to write

$$\begin{aligned}
\odot\pi &\equiv \odot\pi \wedge \bigwedge_{d \in \mathbf{D}} \top \\
&\equiv \odot\pi \wedge \bigwedge_d (\nabla_d \emptyset \vee \nabla_d \{\top\}) \\
&\equiv \bigvee_{\Phi: \mathbf{D} \rightarrow \{\emptyset, \{\top\}\}} \pi \bullet \{\Phi(d) \mid d \in \mathbf{D}\}.
\end{aligned}$$

QED

It may come as a surprise to the reader that the bullet operator is in fact the *only* form of conjunction that we need! More precisely, Theorem 1.39 below states that every formula of CML can be rewritten into an equivalent version that does not use the ordinary Boolean conjunction, but only the special ‘bullet conjunction’.

Definition 1.38 Formulas of the language $\text{CML}_{\overline{\mathbf{D}}}(\mathbf{P})$ are given by the following recursive definition:

$$\varphi ::= \top \mid \perp \mid \varphi \vee \psi \mid \pi \bullet \Phi$$

where π denotes a subset of \mathbf{P} , and Φ a \mathbf{D} -indexed set of $\text{CML}_{\overline{\mathbf{D}}}(\mathbf{P})$ -formulas. \triangleleft

Theorem 1.39 For any \mathbf{P} and \mathbf{D} , the languages $\text{PML}_{\mathbf{D}}(\mathbf{P})$ and $\text{CML}_{\overline{\mathbf{D}}}(\mathbf{P})$ are expressively equivalent.

Proof. Obviously it suffices to prove that every CML-formula φ has an equivalent formula $\overline{\varphi}$ that does not use the conjunction symbol. We will prove this result by induction on the length of a formula, confining ourselves to the case of basic modal logic (with one action).

In the base step of this induction there is nothing to prove. In the inductive step, the clauses for the disjunction and the cover modality speak for themselves:

$$\begin{aligned}
\overline{\varphi \vee \psi} &:= \overline{\varphi} \vee \overline{\psi}, \\
\overline{\pi \bullet \Phi} &:= \pi \bullet \{\overline{\varphi} \mid \varphi \in \Phi\}.
\end{aligned}$$

This leaves the case of a conjunction $\varphi \wedge \varphi'$, where we make a further case distinction. If either of the formulas is of the form \top or \perp it is obvious how to proceed: $\overline{\perp \wedge \varphi} := \perp$, $\overline{\top \wedge \varphi} := \overline{\varphi}$, etc. Also, in case either of the two conjuncts is a disjunction, say $\varphi = \varphi_0 \vee \varphi_1$, using induction loading we may correctly define $\overline{\varphi \wedge \varphi'} := \overline{\varphi_0 \wedge \varphi'} \vee \overline{\varphi_1 \wedge \varphi'}$.

The heart of the proof lies in the one remaining inductive case, namely, where $\varphi = \pi \bullet \Phi$ and $\varphi' = \pi' \bullet \Phi'$. Here we put

$$\overline{\varphi \wedge \varphi'} := \begin{cases} \perp & \text{if } \pi \neq \pi', \\ \bigvee_{Z \in \Phi \times \Phi'} (\pi \bullet \{\overline{\varphi \wedge \varphi'} \mid (\varphi, \varphi') \in Z\}) & \text{if } \pi = \pi'. \end{cases}$$

It then follows immediately from the inductive assumptions that $\overline{\varphi \wedge \varphi'}$ is a $\text{BML}_{\overline{\mathbf{D}}}$ -formula, and from Proposition 1.33 that $\overline{\varphi \wedge \varphi'}$ is equivalent to $\varphi \wedge \varphi'$. QED

Exercises

Exercise 1.1 Prove Theorem 1.17.