Course Script

IN 5110: Specification and Verification of Parallel Systems

IN5110, autumn 2019

Martin Steffen, Volker Stolz

Contents

Chapter 1 Logics

Learning Targets of this Chapter

The chapter gives some basic information about "standard" logics, namely propositional logics and (classical) first-order logics.

Contents

What is it about?

1.1 Introduction

1.1.0.1 Logics

What's logic?

As discussed in the introductory part, we are concerned about formal methods, verification and nalysis of systems etc., and that is done relative to a *specification* of a system. The specification lays down (the) desired properties of a system and can be used to judge whether a system is correct or not. The requirements or properties can be given in many different forms, including informal ones. We are dealing with *formal* specifications. Formal for us means, it has not just a precise meaning, that meaning is also fixed in a mathematical form for instance a "model"[1] We will not deal with informal specifications nor with formal speificiations that are unrelated to the *behavior* in a broad sense of a system.

For example, a specification like

the system should cost 100 000$ or less, incl. VAT

could be seens as being formal and precise. In practice, such a statement is probably not precise enough for a legally binding contract (what's the exchange rate, if it's for Norwegian usage? Which date is taken to fix the exchange rate, the contract, the scheduled delivery date, the actual delivery date? What's the "system" anyway, the installation? The binary?

[1] The notion of model will be variously discussed later resp. given a precise meaning in the lection. Actually, it will be given different precise mathematical meaning, depending on which framework, logic etc we are facing; the rough idea remains the "same" though.

Training? etc.) All of that would be "formalized" in a legal contract readable not even for mathematicians, but only for lawyers, but that's not the kind of formalization we are dealing with.

For us, properties are expressed in "logics". That is a very broad term as well, and we will encounter various different logics and "classes" of logics.

This course is not about fundamentals of logics, like "is logic as discipline a subfield of math, or is it the other way around", like "math is about drawing conclusions about some abstract notions and proving things about those, and in order to draw conclusions in a rigourous manner, one should use logical systems (as opposed to hand-waving ...)". We are also mostly not much concerned with fundamental questions of *meta-theory*. If one has agreed on a logic (including notation and meaning), one can use that to fix some "theory" which is expressed inside the logic. For example, if one is interested in formally deriving things about natural numbers, one could first choose first-order logic as general framework, then select symbols proper for the task at hand (getting some grip on the natural numbers), and then try to axiomatize them and formally derive theorems inside the chosen logical system. As the name implies meta-theory is *not* about things like that, it's about what can be said *about* the chosen logic itself (is the logic decidable, how efficient can it be used for making arguments, how does it's expressivity compares to other logics ...). Such questions will pop up from time to time, but are not at the center of the course. For us, logic is more of a tool for validating programs, and for different kind of properties or system, we will see what kind of logics fits.

Still, we will talk about the basic vocabulary and terminology needed when talking *about* a logic (on the meta-level, so to say). That will include notions like formulas, satisfaction, validity, correctness, completeness, consistency, substitution ..., or at least a subset of those notions.

When talking about "math" and "logics" and what there relationship is: some may have the impression that math as discipline is a formal enterprise and formal methods is kind of like an invasion of math into computer science or programming. It's probably fair to say, however, that for the working academic mathematician, math is *not* a formal discipline in the sense the formal methods or conputer scientists do their business. Sure, math is about drawing conclusions and doing proofs. But most mathematician would balk at the question "what's the logical axioms you use in your aguments" or "what exact syntax you use". That's only bothers mathematicans (to some extent) who prove things *about* logical systems, i.e., who take logics as object of their study. But even those will probably not write their arguments *about* a formally defined logic *inside* a(nother?) logical system. That formal-method people are more obsessed with such nit-picking questions has perhaps two reason. For one is that they want not just clear, elegant and convincing arguments, they want that the computer makes the argument or at least assist in the argument. To have a computer program do that, one needs to be 100% explicit what the syntax of a formal system means, how to draw arguments or check satisfaction of a formula etc. Another reason is that the objects of study for formal-method people are, mathematically seen, "dirty stuff". One tries to argue for the corrrectness of a program, an algorithm, maybe even an implementation. As a fact of life, often that means one does not deal with any elegant mathematical structure but some specific artifact. It's not about "in principle, the idea of the algorithm is correct", whether there it's correct or not not depends on

special cases, uncovered conditions, or other circumstances. There is no such argument as "the remaining cases work analogously...": A mathematician would get away with that, but a formalistic argument covering really all cases would not. (additonally, in making proves about software, it's often not about "the remaining 5 cases". Especially, in concurrent program or algorithms, one has to cover a huge about of possible *interleavings* (combinations of orderings of executions), and a incorrectness (like a race condition) may occur only in some very seldom specific interleaving. Proving some correct (or test some) will not do the job.

1.1.0.2 General aspects of logics

- truth vs. provability
  - when does a formula *hold*, is *true*, is satisfied
  - valid
  - satisfiable
- syntax vs. semantics/models
- model theory vs. proof theory

We will encounter different logics. They differ in their syntax and their semantics (i.e., the way particular formulas are given meaning), but they share some commonalities. Actually, the fact that one distinguishes between the *syntax* of a logics and a way to fix the meaning of formulas is common to all the encountered approaches. The term "formula" refers in in general to a syntactic logical expression (details depend on the particular logic, of course, and sometimes there are alternative or more finegrained terminology, like *proposition*, or *predicate* or *sentence* or *statements&, or even in related contexts names like /assertion* or *constraint*). For the time being, we just generically speak about "formulas" here and leave differentiations for later. Anyway, when it comes to the semantics, the meaning of a given formula, it's the question of whether it's true or not (at least in classical settings...) . Alternative and equivalent formulations is whether it *holds* or not and whether its *satisfied* or not.

That's only a rough approximation, insofar that, given a formula, one seldomly can stipulate unconditionally that it holds or not. That, generally, has to do with the fact that formulas typically has fixed parts and "movable" parts, i.e., parts for which an "interpretation" has to be chosen before one can judge the truth-ness of the formula. What exactly is fixed and what is to be chosen depends on the logic, but also on the setting or approach.

To make it more concrete in two logics one may be familiar with (but the lecture will cover them to some extent). For the rather basic *boolean logic* (or propositional logic), one deals with formulas of the form $P_1 \wedge P_2$, where $\wedge$ is a logical connective and the P's here are atomic propositions (or propositional variables or propositional symbols, depending on your preferred terminology). No matter how it's called, the $\wedge$ part is fixed (it's always "and"), the two P's is the "movable part" (it's for good reason why they are sometimes called propositioal *variables*...). Anyway, it should be clear that asking whether $P_1 \wedge P_2$ is true or holds cannot be asked *per se*, if one does not know about P_1 and P_2, the truth or falsehood is relative to the choice of truth or falsehood of the propositional variables: choosing both P_1 and P_2 as "true" makes $P_1 \wedge P_2$ true.

There are then different ways of notationally write that. Let's abbreviate the mapping $[P_1 \mapsto \top, P_2 \mapsto \top]$ as σ, then all of the formulations (and notations) are equivalent

- $\sigma \models \varphi$ (or $\models_\sigma \varphi$):
  - σ satisfies φ
  - σ models φ (σ is a model of φ)
- $[\![\varphi]\!]^\sigma = \top$:
  - with σ as propositional variable assignment, φ is true or φ holds
  - the semantics of φ under σ is $\top$ ("true")

Of course, there are formulas which truth-ness does *not* depend on particular choices, being unconditionally true (or other unconditionally false). They deserve a particular name like (propositional) "tautology" (or "contradiction" in the negative case).

Another name for a generally true formula or a formula which is true under all circumstances is to say it's *valid*. For propositional logic, the two notions (valid formula, tautology) coincide.

If we got to more complex logics like first-order logics, things get more subtle (and same for modal logics). In that case, there are more ingregients in the logic that are potentially "non-fixed", but movable. Basically, in that setting, one can distinguish of "movable parts". First-order logic is define relative to a so-called signature (to distinguish them form other forms of signatures, it's sometimes called first-order signature. It's just the "alphabet" one agrees upon to work with. It contains functional and relational symbols (with fixed arity or sorts). Those operators define the "domain(s) of interest" one intends to talk about and their syntactic operators. For example, one could fix a signature containing operators `zero`, `succ`, and `plus` on a single domain (a single-sorted setting where the names indicate that one plans to interpret the single domain as natural numbers. We use in the discussion here `typewriter` font to remind that the signature and their operators are intended as *syntax*, not as the semantical interpretation, presumable the known mathematical entities 0, the successor function, and +, i.e., addition). There are also syntactic operators which constitute the logic itself (like the binary operator $\wedge$, or maybe we should write `and`...), which are treated as really and absolutely fixed (once one has agree on that one does classical first order logic or similar).

The symbols of a chosen signature, however, are generally *not* fixed. They are not typically though of as *variables*, but choosing a semantics for them is one of the non-fixed, variable parts when talking about the semantics of a first-order formula. That part, fixing the functional and relational symbols of a given signature is called often an *interpretation*. There is, however, a second level of "non-fixed" syntax in a first-order formula, on which the truthness of a formula depends: those are (free) *variables*. For instance, assuming that we have fixed the interpretation of `succ`, `zero`, `leq` (for less-or-equal) and so an by the standard meaning implied by their name, the truth of the formula `leq(succ x, y)` depends on the choices for the free variables `x` and `y`.

To jugdge, whether a formula with free variables holds or not, one this needs to fix two parts, the interpretation of the symbols of the alphabet (sometimes called the interpretation), as well as the choice of values for the free variables. Now that the situation is more involved, with two levels of choices, the terminolgy becomes also a bit non-uniform (depending on the text-book, one might encouter slightly contradicting use of words).

One common interpretation is to call the choice of symbols the *interpretation* or also *model*. To make a distinction one sometimes say, the *model* (let's call it M is the mathematic structure to part

1.1.0.2.1 Two separate worlds: model theory and proof theory?

proof theory

model theory

calculus

1.2 Propositional logic

The kind of simplest form of logic is known as *propositional* or also *boolean* (in honor of George Boole).[2] It's also underlying binary hardware, binary meaning "two-valued". The two-valuedness is the core of *classical* logics in general, the assumption that there is some *truth* which is either the case or else not (true or else false, nothing in between). This is embodied the classical propositional logic. Formulas

1.2.0.1 Non-classical logics

1.2.0.2 Syntax

$$\begin{array}{rcll} \varphi & ::= & P \mid \top \mid \bot & \text{atomic formula} \\ & \mid & \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \to \varphi \mid \ldots & \text{formulas} \end{array}$$

1.2.0.3 Semantics

- truth values
- σ
- different "notations"
  - $\sigma \models \varphi$
  - evaluate φ, given σ $[\![\varphi]\!]^\sigma$

1.2.0.4 Proof theory

- decidable, so a "trivial problem" in that sense
- truth tables (brute force)
- one can try to do better, different derivation strategies (resolution, refutation, ...)
- SAT is NP-complete

SAT is not a model checking problem. If we see σ as model, then model checking $\sigma \models \varphi$ is complexity-wise linear (compositional, divide-and-conquer).

[2] Like later for first-order logic and other logics, there are variations of that, not only syntactical, some also essential. We are dealing with classical propositional logics. One can also study intuitistic versions. One of such logic is known as *minimal* logic (which may or may not make then the "simplest") which has implication as the only constructor.

1.3 Algebraic and first-order signatures

1.3.0.1 Signature

- fixes the "syntactic playground"
- selection of
  - *functional* and
  - *relational*

  symbols, together with "arity" or sort-information

1.3.0.2 Sorts

- **Sort**
  - name of a domain (like `Nat`)
  - restricted form of type
- single-sorted vs. multi-sorted case
- single-sorted
  - one sort only
  - "degenerated"
  - *arity* = number of arguments (also for relations)

1.3.0.3 Terms

- given: signature Σ
- set of variables X (with typical elements $x, y', \ldots$)

$$\begin{array}{rcll} t & ::= & x & \text{variable} \\ & | & f(t_1, \ldots, t_n) & f \text{ of arity } n \end{array} \tag{1.1}$$

- $T_\Sigma(X)$
- terms without variables (from $T_\Sigma(\emptyset)$ or short T_Σ): *ground* terms

The definition here makes use of the simple single-sorted case. The terms must be "well-typed" or "well-sorted" in that a function symbol that expects a certain number of arguments (as fixed by the signature in the form of the symbol's arity) must be a applied on exactly that number of arguments. The number n can be 0, in wich case the function symbol is also called a *constant* symbol.

As a simple example: with the standard interpretation in mind, a symbol `zero` would be of arity 0, i.e., represents a constant, `succ` would be of arity 1 and `plus` of arity 2.

For clarity we used here (at least for a while) `typewriter` font to refer to the symbols of the signature, i.e., the syntax, to distinguish them from their semantic meaning. Often, as in textbooks, one might relax that, and just also write in conventional situations like here + and 0 for the *symbols* as well.

The situation for multi-sorted situations is not really different, it does not pose fundametally more complex challenges (neither syntactically nor also what proof theory or models or other questions are concerned).

In practical situations (i.e., tools), one could allow *overloading*, or or other "type-related" complications (sub-sorts for examples). Also, in concrete syntax supported by tools, there might be questions of *associativity* or *precedence* or whether one uses *infix* or *prefix* notations. For us, we are more interested in other questions, and allow ourselves notations like x `plus` y or $x + y$ instead and similar things, even if the grammar seems to indicate that it should be `plus x y`. Basically, we understand the grammars as *abstract syntax* (i.e., as describing trees) an assume that educuated readers know what is meant if we use more conventional concrete notations.

1.3.0.4 Substutition

- **Substitution** = *replacement*, namely of variables by terms
- notation $t[s/x]$

Other notations for substitution exist in the literature. We will later use (mutatis mutandis) substution also on first-order formulas (actually, one can use it everywhere if one has "syntactic expression" with "variables"): formulas will contain, besides logical constructs and relational symbols also variables and terms. The substitution will work the same as here, with one technical thing to consider (which is not covered right now): Later, variables can occur *bound* by quantifiers. That will have two consequences: the substitution will apply only to not-bound occurrences of variables (also called *free* occurrences). Secondly, one has to be careful: a naive replacement could suffer from so-called *variable-capture*, which is to be avoided (but it's easy enough anyway).

1.3.0.5 First-order signature (with relations)

So far we have covered standard signatures for terms (also known as algebraic signatures). In first-order logic, one also adds a second kind of syntactic material to the signatures, besides function symbols, those are *relational* symbols. Those are intended to be interpreted "logically". For instance, in a one-sorted case, if one plans to deal with natural numbers, one needs relational symbols on natural numbers, like the binary relation `leq` (less-or-equal, representing $\leq$) or the unary relation `even`. One can call those relations also **predicates** and the form later then the *atomic* formulas of the first-order logic (also call (first-order) predicate logic).

- add **relational** symbols to Σ
- typical elements P, Q
- relation symbols with fixed arity n-ary predicates or relations)
- standard binary symbol: $\doteq$ (equality)

1.3.0.5.1 Multi-sorted case and a sort for booleans The above presentation is for the single-sorted case again. The multi-sorted one, as mentioned, does not make fundamental touble.

In the hope of not being confusing, I would like to point out the following in that context. If we assumed a many-sorted case (maybe again for illustration dealing with natural numbers and a sort `nat`), one can of course add a second sort intended to represent the booleans, maybe call it `bool`. Easy enough. Also one cound then could think of relations as boolean valued function. I.e., instead of thinking of `leq` as relation-symbol, one could attempt to think of it as a *function symbol* namely of sort $\texttt{nat} \times \texttt{nat} \to \texttt{bool}$. Nothing wrong with that, but one has to be careful not confuse oneself. In that case, `leq` is a function symbol, and `leq(5,7)` (or `5 leq 8`) is a term of type `bool`, presumably interpreted same as term `true`, but it's not a predicate as far as the logic is concerned. One has chosen to use the surrounding logic (FOL) to speak about a domain intended to represent booleans. One can also add operator like `and` and `or` on the so-defined booleans, but those are *internal* inside the formalization, they are *not* the logical operators $\wedge$ and $\vee$ that part part of the logic itself.

1.3.0.5.2 0-arity relation symbols On principle, in the same way that one can have 0-arity function symbols (which are understood as constants), one can have 0-arity relation symbols or predicates. When later, we attach meaning to the symbols, like attaching the meaning $\leq$ to `leq`, then there are basically only two possible interpretations for 0-arity relation symbols: either "to be the case" i.e., true or else not, i.e., false. And actually there's no need for 0-arity relations, one has fixed syntax for those to cases, namely "true" and "false" or similar which are reserved words for the two only such trivial "relations" and their interpretation is fixed as well (so there is no need to add more alternative such symbols in the signature).

Anyway, that discussion shows how one can think of propositional logic as a special case of first-order logic. However, in boolean logic we assume many propositional symbols, which then are treated as *proposional variables* (with values true an false). In first order logics, the relational symbols are not thought of as variables, by fixed by choosing an interpretation, and the variable part are the variables inside the term as members of the underlying domain (or domains in the multi-sorted case).

1.3.0.5.3 Equality symbol The equality symbol (we use $\doteq$) plays a special role (in general in math, in logics, and also here). One could say (and some do) that the equality symbol is one particular binary *symbol*. Being *intended* as equality, it may be captured by certain laws or axioms, for instance, along the following lines: similar like requiring `x leq x` and with the intention that `leq` represents $\leq$, this relation is *reflexive*, one could do the same thing for equality, stating among other things `x eq x` with `eq` intended to represent equality. Fair enough, but equality is *so central* that, no matter what one tries to capture by a theory, equality is at least *also part of the theory*: if one cannot even state that two things are equal (or not equal), one cannot express anything at all. Since one likes to have equality anyway (and since it's not even so easy/possible to axiomatise it in that it's really the identity and not just some equivalence), one simply says, a special

binary symbol is "reserved" for equality and not only that: it's agreed upon that it's *interpreted* semantically as equality. In the same way that one always interprets the logical $\wedge$ on predicates as conjuction, one always interprets the $\doteq$ as equality.

As a side remark: the status of equality, identity, equivalence etc is challenging from the standpoint of *foundational* logic or maths. For us, those questions are not really important. We typically are not even interested in alternative interpretations of other stuff like `plus`. When "working with" logics using them for specifications, as opposed to investigate meta-properties of a logic like its general expressivity, we just work in a framework where the symbol `plus` is interpreted as $+$, end of story. Logicians may ponder the question, whether first order logic is expressive enough that one can write axioms in such a way that the only possible d interpretation of the symbols correspond to the "real" natural numbers and plus thereby is really $+$. Can one get an axiomatization that characterizes the natural numbers as the *only* model (the answer is: *no*) but we don't care much about questions like that.

1.4 First-order logic

1.4.1 Syntax

1.4.1.1 Syntax

- given: first order signature Σ

$$\begin{array}{rcll} \varphi & ::= & P(t,\ldots,t) \mid \top \mid \bot & \text{atomic formula} \\ & \mid & \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \to \varphi \mid \ldots & \text{formulas} \\ & \mid & \forall x.\varphi \mid \exists x.\varphi & \end{array}$$

The grammar shows the syntax for first-order logic. We are not overly obsessed with concrete syntax (here), i.e., we treat the syntax as *abstract syntax*. We silently assume proper priorities and associativities (for instance, $\neg$ binds by convention stronger than $\wedge$, which in turn binds stronger than $\vee$ etc. In case of need or convenience, we use parentheses for disambiguation.

The grammar, choice of symbols, and presentation (even terminology) exists in variations, depending on the textbook. For instance, one could su

1.4.1.1.1 Minimal representation and syntactic variations The above presentation, as in the proposition or boolean case, is a bit generous wrt.\ offered syntax. One can be more economic in that one restrics oneselves to a *minimal* selection of constrict (there are different possible choices for that). For instance, in the presence of (classical) negation, one does not need both $\wedge$ and $\vee$ (and also $\to$ can be defined as syntactic sugar). Likewise, one would need only one of the two quantification operators, not both. Of course, in the presence of negation, *true* can be defined using *false*, and vice versa. In the case of the boolean constants *true* and *false*, one could even go a step further and define them as $P \vee \neg P$ and $P \wedge \neg P$ (but actually it seems less forced to have at least one as native

contruct). One could also explain *true* and *false* as propositions or relations with arity 0 and a fixed interpretation. All of that representation can be found here and there, but they are inessential for the nature of first-order logic and as a master-level course we are not over-obsessed with representational questions like that. Of course, if one had to interact with a tool that "supports" for instance first-order logics (like a theorem prover or constraint solver) or if one wanted to implement such a tool oneself, syntactial questions would of course matter and one would have to adhere to stricter standards of that particular tool.

1.4.2 Semantics

1.4.2.1 First-order structures and models

- given Σ
- assume single-sorted case

1.4.2.1.1 first-order model model M

$$M = (A, I)$$

- A some domain/set
- **interpretation** I, respecting arity
  - $[\![f]\!]^I : A^n \to A$
  - $[\![P]\!]^I : A^n$
- cf. first-order structure

1.4.2.1.2 First-order structure (left out from the slide)

- single-sorted case here
- domain A together with functions and relations
- NB: without relations: *algebraic structure*
- many-sorted case: analgously (interpretation respecting sorts)

A model here is a pair, namely a domain of interpretation together with I, that associates functions and relations appropriately to the corresponding syntactic vocabulary from the signature. A set, equipped with functions and relations is also called a first-order *structure*. Often, the structure itself is also called the model (leaving the association between syntax and its interpretation implicit, as it's obvious in many cases anyway). For instance,

$$(\mathbb{N}; 0, \lambda x.x + 1, +, *, \leq, \geq)$$

is a structure, whose domain are the natural numbers and which is equipped with 4 functions (one of which is 0, which is of zero arity and this called usually a constant rather than function) and two binary relations $\leq$ and $\geq$. That can be a "model" of a signature Σ with one sort say `Ñat` and function symbols `zero`, `succ`, `plus` and `times` and relational symbols `leq` and `geq`. Technically, though the model is the mapping I.

Strictly speaking, nothing would forbid us to interpret the syntax differently in the same structure, for instance setting $[\![\texttt{times}]\!]^I = +$ or $[\![\texttt{leq}]\!]^I = \geq$.

In this (and similar) cases the association is obvious, thereby sometimes left implicit, and some people also call the structure a *model* of a signature (but for preciseness sake, it should be the structure *and* making clear which elements of the structure belong to what syntax).

That may sounds nitpicking, but probably it's due to the fact that when dealing with "foundational" questions like model theory, etc. one should be clear what a *model* actually is (at least at the beginning). But also practically, one should not forget that the illustration here, the natural numbers, may be deceivingly simple. If one deals with more mundane stuff, like capturing real word things as for instance is done in ontologies, there may be hundreds or thougsangs of symbols, predicates, functions etc. and one should be clear about what means what (Ontologies is related to "semantics techniques" that try to capture and describe things and then query about it (which basically means, asking questions and draw conclusions from the "data base" of collected knowledge) and the underlying language is often (some fragment of) first-order logic.

1.4.2.2 Giving meaning to variables

1.4.2.2.1 Variable assignment

- given Σ and model

$$\sigma : X \to A$$

- other names: *valuation*, *state*

1.4.2.3 (E)valuation of terms

- σ "straightforwardly extended/lifted to terms"
- how would one define that (or write it down, or implement)?

Given a signature Σ and a corresponding model $M = (A, I)$, the value of term t from $T_\Sigma(X)$, with variable assignment $\sigma : X \to A$ (written $[\![\varphi]\!]^I_\sigma$) is given inductively as follows

$$[\![\varphi]\!]^I_\sigma$$

1.4.2.4 Free and bound occurrences of variables

- quantifiers *bind* variables
- *scope*
- other binding, scoping mechanisms
- variables can *occur* free or not (= *bound*) in a formula
- careful with substitution
- how could one define it?

1.4.2.5 Substitution

- basically:
  - generalize substitution from terms to formulas
  - careful about binders especially don't let substitution lead to variables being "captured" by binders

1.4.2.5.1 Example

$$\varphi = \exists x.x + 1 \doteq y \qquad \theta = [y/x]$$

1.4.2.6 Satisfaction

1.4.2.6.1 $\models$ $M, \sigma \models \varphi$

- Σ fixed
- in model M and with variable assignment σ formula φ is true (holds
- M and σ satisfy φ
- minority terminology: M, σ model of φ

In seldom cases, some books call the pair of (M, φ) a model (and the part M then called interpretation or something else. It is a terminology question (thus not so important), but it may lead to different views, for instance what "semantical implication" means. The standard default answer what that means is the following (also indepdendent form the logic). A formula φ_1 implies semantically a formula φ_2, if all models of φ_1 are also models of formula φ_2 (the satisfaction of φ_1 implies satisfaction of φ_2).

Now it depends in if one applies the word "model" to M or to the pair M, σ). That leads to different notions of semantical implications, at least if one had formulas with free variables. For closed formulas, it does not matter, so some books avoid those finer points but just defining semantical implication on closed formulas.

1.4.2.7 Exercises

- substitutions and variable assignments: similar/different?
- there are infinitely many primes
- there is a person with at least 2 neighbors (or exactly)
- every even number can be written as the sum of 2 primes

1.4.3 Proof theory

1.4.3.1 Proof theory

- how to infer, derive, deduce formulas (from others)
- mechanical process
- soundness and completeness
- *proof* = deduction (sequence or tree of steps)

- theorem
  - syntactic: derivable formula
  - semantical a formula which holds (in a given model)
- (fo)-theory: set of formulas which are
  - derivable
  - true (in a given model)

- soundness and completeness

1.4.3.2 Deductions and proof systems

A *proof system* for a given logic consists of

- *axioms* (or *axiom schemata*), which are formulae assumed to be true, and

- *inference rules*, of approx. the form

$$\frac{\varphi_1 \quad \cdots \quad \varphi_n}{\psi}$$

- $\varphi_1, \ldots, \varphi_n$ are **premises** and ψ **conclusion**.

1.4.3.3 A simple form of derivation

1.4.3.3.1 Derivation of φ

Sequence of formulae, where each formula is

- an axiom or
- can be obtained by applying an inference rule to formulae earlier in the sequence.

- $\vdash \varphi$
- more general: set of formulas Γ

$$\Gamma \vdash \varphi$$

- proof = derivation
- theorem: derivable formula (= last formula in a proof)

A proof system is

- *sound* if every theorem is valid.
- *complete* if evey valid formula is a theorem.

We do not study soundness and completeness for validity of FOL in this course.

1.4.3.4 Proof systems and proofs: remarks

- "definitions" from the previous slides: not very formal

in general: a proof system: a "mechanical" (= formal and constructive) way of conclusions from axioms (= "given" formulas), and other already proven formulas

- Many different "representations" of how to draw conclusions exists, the one sketched on the previous slide
  - works with "sequences"
  - corresponds to the historically oldest "style" of proof systems ("Hilbert-style"), some would say outdated ...
  - otherwise, in that naive form: impractical (but sound & complete).
  - nowadays, better ways and more suitable for computer support of representation exists (especially using trees). For instance natural deduction style system

for this course: those variations don't matter much

1.4.3.5 First order logic (commented out)

1.4.3.6 A proof system for prop. logic

We can axiomatize a subset of *propositional logic* as follows.

$$\varphi \to (\psi \to \varphi) \tag{Ax1}$$
$$(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi)) \tag{Ax2}$$
$$((\varphi \to \bot) \to \bot) \to \varphi \tag{DN}$$
$$\frac{\varphi \quad \varphi \to \psi}{\psi} \tag{MP}$$

1.4.3.7 A proof system

Example 1.4.1. $p \to p$ is a theorem of PPL:

$(p \to ((p \to p) \to p)) \to ((p \to (p \to p)) \to (p \to p))$	Ax_2	(1.1)
$p \to ((p \to p) \to p)$	Ax_1	(1.2)
$(p \to (p \to p)) \to (p \to p)$	MP on (1) and (2)	(1.3)
$p \to (p \to p)$	Ax_1	(1.4)
$p \to p$	MP on (3) and (4)	(1.5)

A proof can be represented as a tree of inferences where the leaves are axioms.

1.5 Modal logics

1.5.1 Introduction

The roots of logics date back very long, and those of modal logic not less so (Aristotle also hand his fingers in the origin of modal logic and discussed some kind of "paradox" that gave food for thought for future generations thinking about modalities). Very generally, a logic of that kind is concerned not with *absolute* statements (which are true or else false) but *qualified* statements, i.e., statements that "depend on" some. An example for a modal statement would be "tomorrow it rains". It's difficult to say in which way that sentence is true or false, only time will tell... It's an example of a statement depending on "time", i.e., *tomorrow* is a form of a *temporal* modality. But there are other modalities, as well (referring to *knowledge* or *believe* like "I know it rains", or "I believe it rains") or similar qualifications of absolute truth. Statements like "tomorrow it rains" or others where long debated often with philosphical and/or regligous connotions like: the the future deterministic (and pre-determined by God's providence), do persons have a free will, etc. Those questions will not enter the lecture, nonetheless: determinism vs. non-determinism is meaningful distinction when dealing with program behavior, and we will also encounter temporal logics that view time as *linear* which kind of means, there is only *one* possible future, or *branching*, which means there are many. It's however, not meant as a fundamental statement of the "nature of nature", it's just a distinction of how we want to treat the system. If we want to check individual runs, which are sequences of actions, then we are committing ourselves to a *linear* picture (even when dealing with non-deterministic programs). But there are branching alternatives to that view as well, which lead to branching temporal logics.

Different flavors of modal logic lead to different axioms. Let's write $\Box$ for the basic modal operator (whatever its interpretation), and consider

$$\Box\Box\varphi \to \Box\varphi \ , \tag{1.6}$$

with φ some ordinary statement (in propositional logic perhaps, for first-order logic). If we are dealing with a logic of belief, is that a valid formula: If I believe that I believe that something is true, do I believe the thing itself it true? What about "I believe that you believe that something is true"? Do I believe it myself? Not necessarily so.

As a side-remark: the latter can be seen as a formulation in *multi-modal* logic: it's not about one modality (being believed), but "person p believes", i.e., there's one belief-modality per person; thus, it's a *multi-modal* logic. We start in the presentation with a "non-multi" modal logic, where there is only *one* basic modality (say $\Box$). Technically, the syntax may feature two modalities $\Box$ and $\Diamond$, but to be clear: that does *not* yet earn the logic the qualification of "multi": the $\Diamond$-modality is in all considered cases expressible by $\Box$, and vice versa. It's analogous to the fact that (in most logics), $\forall$ and negation allows to express $\exists$ and vice versa.

Now, coming back to various interpretations of equation (1.6): if we take a "temporal" standpoint and interpret $\Box$ as *"tomorrow"*, then certainly the implication should not be valid. If we interpret $\Box$ differently, but still temporally, as *"in the future"* then again the interpretation seems valid.

If we take an "epistemologic" interpretation of $\Box$ as "knowledge", the left-hand of the implication would express (if we take a multi-modal view): "I know that you know that φ". Now we may ponder whether that means that then I *also* know that φ? A question like that may lead to philosophical reflections about what it means to "know" (maybe in contrast with "believe" or "believe to know" etc.).

The lecture will not dwell much on philospophical questions. The question whether equation (1.6) will be treated as *mathematical question*, more precisely a question of the assumed underlying *models* or *semantics*.

It's historically perhaps interesting: modal logic has attracted long interest, but the question of "what's the mathematics of those kind of logics" was long unclear. Long in the sense that classical logics, in the first half of the 20th century had already super-thoroughly been investigated and formalized from all angles with elaborate results concerning *model theory* and proposed as "foundations" for math etc. But no one had yet come up with a convicing answer for the question: "what the heck is a model for modal logics?". Until a teenager and undergrad came along and provided the now accepted answer, his name is *Saul Kripke*. Models of modal logics are now called *Kripke-structures* (it's basically transition systems).

1.5.1.1 Introduction

- **Modal** logic: logic of *"necessity"* and *"possibility"*, in that originally the intended meaning of the *modal* operators $\Box$ and $\Diamond$ was
  - $\Box\varphi$: φ is necessarily true.
  - $\Diamond\varphi$: φ is possibly true.
- Depending on what we intend to capture: we can interpret $\Box\varphi$ differently.

  temporal φ will always hold.

  doxastic I believe φ.

  epistemic I know φ.

  intuitionistic φ is provable.

  deontic It ought to be the case that φ.

We will restrict here the modal operators to $\Box$ and $\Diamond$ (and mostly work with a temporal "mind-set".

1.5.2 Semantics

1.5.2.1 Kripke structures

The definition below makes use of the "classical" choice of words to give semantics to modal logic. It's actually very simple, based on a relation (here denotate R) on some set. The "points" on that set are called *worlds* which are connected by an *accessibility* relation. Well, underlying the notion of modal model is thus just a relation, or we also could call it a *graph*, but traditionally it's called a *frame* (a Kripke frame). That kind of semantics is

also called *possible world semantics* (but not graph semantics or transition system system semantics, even if that would be an ok name as well).

The Kripke frame in itself not a model yet, in that it does not contain information to determine if a modal formula holds or not. The general picture is as follows: the elements of the underlying set W are called "worlds": one some world some formulas hold, and in a different world, different ones. "Embedded" in the modal logic is an "underlying"logic. We mostly assume *propositional* modal logic, but one might as well consider an extension of first-logic with modal operators. In this propositional setting, what then is needed for giving meaning to model formulas is an interpretation of the propositional variables, and that has to be done *per world*.

In the section of propopositional logic, we introduced "propositional variable assignments" $\sigma : P \to \mathbb{B}$, giving a boolean value to each propositional variable from σ. What we now call a *valuation* does the same *for each world* which we can model as a function of type

$$W \to P \to \mathbb{B} \ .$$

Alternatively one often finds also the "representation" to have valuations of type $W \to 2^P$: for each world, the valuation gives the set of atomic propositions which "hold" in that word. Both views are, of course equivalent in being *isomorphic*.

1.5.2.1.1 Labelling The valuation function V associates a propositional valuation to each word (or isomphically the set of all propositional atoms that are intended to hold, per world). As mentioned, a Kripke frame may also be called a graph or also transition system. In the latter case, the worlds may be called less pompously just *states* and the accessibility as *transitions*. That terminolgy is perhaps more familiar in computer science. The valuation function can also be seen to label the states with propositional information. A transition system with attached information is also called *labelled transition system*. But one has to be careful a bit with terminology. When it comes to *labelled* transition systems, additional information can be attached to transitions or states (or both). Often labelled transition systems, especially for some areas of model checking and modelling, are silently understood as *transition-labelled*. For such models, an edge between two states does not just expressed that one can go from one state to the other. It states that one can go from one state to the other *by doing such-and-such* (as expressed by the label of the transition. In an abstract setting, the transitions may be labelled just with letters from some alphabet.

As we will see later, going from a transition system with unlabelled *transitions* to one with transition labels correspond to a generalization from "simple" modal logic to multi-modal logic. But independent on whether one whether consider transitions as labelled or note, there is a "state-labelling" at least, namely the valuation that is needed to interpret the propsitions per world or state.

As a side remark: classical automata can be seen as labelled transitions as well, with the transitions being labelled. There are also variations of such automata which deal with input *and* output (thereby called I/O automata). There, two classical versions that are used in describing hardware (which is a form of propositional logic as well...) label the

transitions via the input. However, one version labels the states with the output (Moore-machines) whereas another one labels the transitions with the output (Mealy-machines), i.e., transitions contain both input as well as output information. Both correspond to different kinds of hardware circuitry (Moore roughly correspond to synchronous hardware, and Mealy to asynchronous one).

We will encounter automata as well, but in the form that fits to our modal-logic needs. In particular, we will look at Buechi-automata, which are like standard finite-state automata except that they can deal with infinite words (and not just finite ones). Those automata have connections, for instance, with LTL, a central temporal logic which we will cover.

Definition 1.5.1 (Kripke frame and Kripke model)**.**

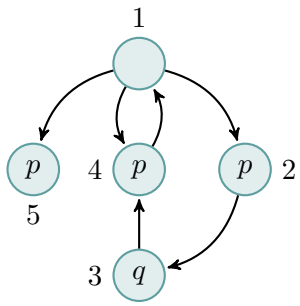
- A *Kripke frame* is a structure (W, R) where
  - W is a non-empty set of *worlds*, and
  - $R \subseteq W \times W$ is called the *accessibility relation* between worlds.
- A *Kripke model* M is a structure (W, R, V) where
  - (W, R) is a frame, and
  - V a function of type $V : W \to (P \to \mathbb{B})$ (called valuation).

isomorphically: $V : W \to 2^P$

Kripke models are sometimes called *Kripke structures*. The standard textbook about modelchecking Baier and Katoen [1] does not even mention the word "Kripke structure", they basically use the word *transition systems* instead of Kripke model with worlds called states (and Kripke frame is called *state graph*). I say, it's "basically" the same insofar that there, the (sometimes) also care to consider labelled transitions, and furthermore, their transition systems are equipped with a set of *initial states*. Whether one has initial states as part of the graph does not make much of a difference.

Also the terminology concerning the set P varies a bit (we mentioned it also in the context of propositional logics). What we call here propositional variables, is also known as propositional constants, propositional atoms, symbols, *atomic propositions*, whatever. The usage of "variable" vs. "constant" seem two irreconsilable choices of terminology. Perhaps the use of the word "variable" stresses that the value needs to be fixed by some interpretation, the word "constant" focuses in the fact that those are "0-ary" constructors (or atoms), and this "constant", not depending on subformulas (once their meaning is fixed). We prefer thinking of $\top$ and $\bot$ as the only two propositional constants and we don't call propositional atoms "constants".

1.5.2.2 Illustration



1.5.2.2.1 Kripke model Let $P = \{p, q\}$. Then let $M = (W, R, V)$ be the Kripke model such that

- $W = \{w_1, w_2, w_3, w_4, w_5\}$
- $R = \{(w_1, w_5), (w_1, w_4), (w_4, w_1), \dots\}$
- $V = [w_1 \mapsto \emptyset, w_2 \mapsto \{p\}, w_3 \mapsto \{q\}, \dots]$

Later, the graphic is slightly informal. There are 5 worlds, which are numbered for identification. In the Kripke model, they are referred to via $w_1, w_2, \dots$ (not as $1, 2, \dots$). Later, when we often call corresponding entities states, not worlds, we often use $s_1, s_2, \dots$ for typical states. For the valuation, we use a notation of the form $[... \mapsto ...]$ to denote a finite mappings.

In particular, we are dealing with finites mappings of type $W \to 2^P$, i.e., to subsets of the list of atomic propositions $P = \{p, q\}$. The sets are not explicitly noted in the graphical illustration, i.e., the set-braces $\{\dots\}$ are omitted. For instance, in world w_1, no propositional letter is mentioned, i.e., the valuation maps w_1 to the empty set $\emptyset$.

An isomporphic (i.e., equivalent) view on the valuation is, that it is a function of type $W \to (P \to \mathbb{B})$ which perhaps captures the intended interpretation better. Each propositional letter mentioned in a wold or state is intended to evaluate to "true" in that world or state. Propositional letter not mentioned are intented to be evaluate to "false" in that world.

As a side remark: we said, that we are dealing with finite mappings. For the examples, illustrations and many applications, that is correct. However, the definition of Kripke structure does *not* require that there is only a finite set of worlds, W in general is a *set*, finite or not.

1.5.2.3 Satisfaction

Now we come to the *semantics* of modal logic, i.e., how to interpret formulas of (propositional) modal formulas. That is done by defining the correspondent "satisfaction" relation, typically written as $\models$. After the introduction and discussion of Kripke models or transition systems, the satisfaction relation should be fairly obvious for the largest part, especially the part of the *underlying logic* (here propositional logic): the valuation V is made exactly so that it covers the base case of atomic propositions, namely give meaning to the elements of P depending on the current world of the Kripke frame. The treatment

of the propositional connectives $\wedge$, $\neg$, ... is identical to their treatment before. Remains the treatment of the real innovation of the logic, the modal operators $\Box$ and $\Diamond$.

Definition 1.5.2 (Satisfaction)**.** A modal formula φ is *true* in the world w of a model V, written $V, w \models \varphi$, if:

$$
\begin{array}{lll}
V, w \models p & \text{iff} & V(w)(p) = \top \\
\\
V, w \models \neg\varphi & \text{iff} & V, w \not\models \varphi \\
V, w \models \varphi_1 \vee \varphi_2 & \text{iff} & V, w \models \varphi_1 \text{ or } V, w \models \varphi_2 \\
\\
V, w \models \Box\varphi & \text{iff} & V, w' \models \varphi, \text{ for all } w' \text{ such that } wRw' \\
V, w \models \Diamond\varphi & \text{iff} & V, w' \models \varphi, \text{ for some } w' \text{ such that } wRw'
\end{array}
$$

As mentioned, we consider V as to be of type $W \to (P \to \mathbb{B})$. If we equivalently assumed a type $W \to 2^P$, the base case of the definition would read $p_1 \in V(w)$ instead.

For this lecture, we prefer the former presentation (but actually, it does not matter of course) for 2 reasons. One is, it seems to fit better with the presentation of propositional logic, generalizing directly the concept a boolean valuation. Secondly, the picture of "assigning boolean values to variables" fit better with seeing Kriple models more like transition systems, for instance capturing the behavior of computer programs. There, we are not so philosphically interested in speaking of "worlds" that are "accessible" via some accessibility relation R, it's more like states in a progam, and doing some action or step does a transition to another state, potentially changing the memory, i.e., the content of variable, which in the easiest case may be boolean variables. So the picture that one has a control-flow graph of a program and a couple of variables (propositional or Boolean variables here) whose values change while the control moves inside the graph seems rather straightforward and natural.

Sometimes, other notations or terminology is used, for instance $w \models_M \varphi$. Sometimes, the model M is fixed (for the time being), indicated by the words like. "Let in the following M be defined as ... ", in which case one finds also just $w \models \varphi$ standing for "state w satisfies φ", or "φ holds in state w" etc. but of course the interpretation of a modal formula requires that there is alway a transition system relative to which it is interpreted.

Often one often finds also notations using the "semantic brackets" $[\![\_]\!]$. Here, the meaning (i.e., truth-ness of false-ness of a formula, depends on the Kripke model as well as the state, which means one could define a notation like $[\![\varphi]\!]^M_w$ as $\top$ or $\bot$ depending on wether $M, w \models \varphi$ or not. Remember that we had similar notation in first-order logic $[\![\varphi]\!]^I_\sigma$ We discussed (perhaps uneccessarily so) two isomorphic view of the valuation function V. Even if not relevant for the lecture, it could be noted that a third "viewpoint" and terminology exists in the literature in that context. Instead of associating with each world or state the set of propositions (that are intended to hold in that state), one can define a model also "the other way around": then one associate with each propositional variable the set of states in which the proposition is suppoed to hold, one would have a "valuation" $\tilde{V} : P \to 2^W$.

That's of course also an equivalent and legitimate way of proceeding. It seems that this representation is "popular" when doing Kripke models for the purpose of capturing systems and their transitions (as for model checking), but for Krikpe models of intuionistic logics. Kripke also propose "Kripke-models" for that kind of logics (for clarity, I am talking about intuitionnistic propositional logics or intuitionnistic first-order logice etc, not (necessarily) intuitionistic *modal* logics). In that kind of setting, the accessibility relation has also special properties (being a partial order), and there are other side conditions to be aware of. As for terminology, in that context, one sometimes does not speak of "w satisfies φ (in a model), for which we write "$w \models p$", but says "world w *forces* a formula φ", for which sometimes the notation $w \Vdash \varphi$ is favored. But those are mainly different traditions for the same thing.

For us, we sometimes use notations like $[\![\varphi]\!]^M$ to represent the set of all states in M that satisfy φ, i.e.,

$$[\![\varphi]\!]^M = \{w \mid M, s \models \varphi\} \ .$$

In general (and also other logics), $\models$- and $[\![\_]\!]$-style notations are interchangable and interdefinable.

1.5.2.4 "Box" and "diamond"

- modal operators $\Box$ and $\Diamond$
- often pronounced "nessecarily" and "possibly"
- mental picture: depends on "kind" of logic (temporal, epistemic, deontic ...) and (related to that) the form of accessibilty relation R:
- formal definition: see previous slide

The pronounciation of $\Box\varphi$ as "necessarily φ" and $\Diamond\varphi$ as "possibly φ" are *generic*, when dealing with specific interpretations, they might get more specific meanings and then be called likewise: "in all futures φ" or "I know that φ" etc. Related to the intended mindset, one imposes different restructions in the "accessibility" relation R. In a temporal setting, if we interpret $\Box\varphi$ as "tomorrow φ", then it is clear that $\Box\Box\varphi$ ("φ holds in the day after tomorrow") is not equivalent to $\Box\varphi$. If, in a different temporal mind-set, we intend to mean $\Box\varphi$ to represent "now and in the future φ", then $\Box\Box\varphi$ and $\Box\varphi$ are equivalent. That reflects common sense and reflects what one might think about the concept of "times" and "days". Technically, and more precisely, it's a property of the assumed class of frames (i.e., of the relation R). If we assume that all models are built upon frames where R is *transitive*, then $\Box\varphi \to \Box\Box\varphi$ is generally true.

We should be more explicit about what it means that a formula is "generally true". We have encountered the general terminology of a formula being "true" vs. being "valid" already. In the context of modal logic, the truth-ness requires a model (which is a frame with a valuation) and a state to judge the truth-ness: $M, w \models \varphi$. A model M is of the form (W, R, V), it's a frame (= "graph") together with a valuation V. A propositional formula is *valid* if it's true for all boolean valuation (and the notion coincided with being a propositional tautology). Now the situation get's more finegrained (as was the case in

first-order logics). A modal formula is *valid* if $M, w \models \varphi$ for all M and w. For that one can write

$$\models \varphi$$

So far so good. But then there is also a middle-ground, where one fixes the frame (or a class of frames), but the formula must be true *for all valuations* and all states. For that we can write

$$(W, R) \models \varphi$$

Let's abbreviate with F a frame, i.e., a tuple (W, R). We could call that notion *frame validity* and say for $F \models \varphi$ that "φ is valid in frame F". So, in other words, a formula is valid in a frame F if it holds in all models with F as underlying frame and for all states of the frame.

One uses that definition not just for a single frame; often the notion og frame-validity is applied to sets of frames, in that one says $F \models \varphi$ for all frames F such that ...". For instance, all frames where the relatiton R is *transitive* or *reflexive* or whatever. Those restrictions of the allowed class of frames reflect then the intentions of the modal logic (temporal, epistemic ...), and one could speak of a formula to be "transitivity-valid" for instance, i.e., for all frames with a transitive accessibility relation.

It would be an ok terminology, but it's not standard. There are (for historic reasons) more esoteric names for some standard classes, for instance, a formula could be S4-valid. That refers to one particular restriction of R which corresponds to a particular set of axioms traditionally known as S4. See below for some examples.

1.5.2.4.1 Further notational discussion and preview to LTL Coming back to the informal "temporal" interpretation of $\Box\varphi$ as either "tomorrow φ" vs. "now and in the future φ", where the accessibility relation refers to "tomorrow" or to "the future time, from now on". In the latter case, the accessibility relation would be reflexive and transitive. When thinking about such a temporal interpretation, there may also be another assumption on the frame, depending on how one sees the nature of time and future. A conventional way would be to see the time as *linear* (a line, fittingly called a *timeline*, connecting the past into the future, with "now" in the middle, perhaps measured in years or seconds etc.) With such a linear picture in mind, it's also clear that there is no difference between the modal operators $\Box$ and $\Diamond$.[3] In the informal interpretation of $\Box$ as "tomorrow", one should have been more explicit that, what was meant is "for all tomorrows" to distinguish it from $\Diamond$ that represent "there exist a possible tomorrow". In the linear timeline picture, there is only one tomorrow (we conventionally say "/the/ next day" not "for all possible next days" or some such complications). Consequently, if one has such a linear picture in mind (resp. works only with such linear frames), one does not actually *need* two modal operators $\Box$ and $\Diamond$, one can collapse them into one. Conventionally, for that collapsed one, one takes $\bigcirc$. A formula $\bigcirc\varphi$ is interpreted as "in the next state or world, φ holds" and

[3] Characterize as an exercise what *exactly* (not just roughly) the condition the accessibility relation must have to make $\Box$ and $\Diamond$ identical.

pronouced "next φ" for short. The $\bigcirc$ operator will be part of LTL (linear-time temporal logic), which is an important logic used for model checking and which will be covered later. When we (later) deal with LTL, the operator $\bigcirc$ corresponds to the modal operators $\Diamond$ and $\Box$ collapsed into one, as explained. Besides that, LTL will have *additional* operators written (perhaps confusingly) $\Box$ and $\Diamond$, with a *different interpretation* (capturing "always" and "eventually") Those are also temporal modalities, but their interpretation in LTL is different from the ones that we haved fixed for now, when discussing modal logics in general).

1.5.2.5 Different kinds of relations

R a *binary relation* on a set, say W, i.e., $R \subseteq W$

- reflexive
- transitive
- (right) Euclidian
- total
- order relation
- ....

Definition 1.5.3. A binary relation $R \subseteq W \times W$ is

- *reflexive* if every element in W is R-related to itself.

$$\forall a.\ aRa$$

- *transitive* if

$$\forall a\ b\ c.\quad aRb \wedge bRc \rightarrow aRc$$

- (right) *Euclidean* if

$$\forall a\ b\ c.\quad aRb \wedge aRc \rightarrow bRc$$

- *total* if

$$\forall a.\ \exists b.\quad aRb$$

The following remark may be obvious, but anyway: The quantifiers like $\forall$ and the other operators $\wedge$ and $\vee$ are not meant here to be vocabulary of some (first-order) logic, they are meant more as mathematical statements, which, when formulated in for instance English, would use sentences containing words like "for all" and "and" and "implies". One could see if one can formalize or characterize the defined concepts making use formally of a first-order logic, but that's not what is meant here. We use the logical connectives just as convenient shorthand for English words.

In that connection a word of caution: first-order logic seems like powerful, the swiss army knife of logics, perfect for formalizing everything if one us patient or obsessive enough. One should be careful, though, FOL has it's limitations (and not just because theorem-hood is undecidable). In some way, FOL is rather *weak* actually, for instance one cannot even characterize the natural numbers, at least not exactly (one way of getting a feeling for that is: Peano's axioms, that characterize the natural numbers, are *not* first-order. First-order logic is not strong enough to capture induction, and then one is left with a notation the

looks exactly like "natural numbers" but for which one cannot use *induction*. And that is thereby a disappointingly useless form of "natural numbers"...

In practice, some people use "applied forms" of first-order logics. For instance, one has a signature that captures the natural numbers, and then one *assumes* that the vocabulary is interpreted by the actual natural numbers as model. The assumption is, as just mention, not capturable by first-order logic itself, it's an external assumption. If one would like to capture that inside a formal logical system (and not just assuming it and explain that by English sentences), one would have to use stronger systems than available in first-order logics. As an example: Hoare-logic was mentioned in the second lecture, which is based traditionally on first-order logic. Those kind of logic is used to talk about programs and those program contain data stored in variables, like natural numbers and similar things. However, when talking about natural numbers or other data structures in Hoare logic, one is not concerned with "are those really expressible in pure first-order logic", one in interested in the program verification, so it's often simply assumed that those are the natural numbers as known from math etc. We may encounter not directly Hoare-logic, but perhaps dynamic logic, which is also a form of (multi)-modal logic. Actually Hoare-logic can be seen as a special case of dynamic logic.

1.5.2.6 Valid in frame/for a set of frames

If $(W, R, V), s \models \varphi$ for all s and V, we write

$$(W, R) \models \varphi$$

1.5.2.6.1 Samples

- $(W, R) \models \Box\varphi \to \varphi$ iff R is reflexive.
- $(W, R) \models \Box\varphi \to \Diamond\varphi$ iff R is total.
- $(W, R) \models \Box\varphi \to \Box\Box\varphi$ iff R is transitive.
- $(W, R) \models \neg\Box\varphi \to \Box\neg\Box\varphi$ iff R is Euclidean.

1.5.2.7 Some Exercises

Prove the double implications from the slide before!

1.5.2.7.1 Hints By "double implications", the iff's (if-and-only-if) are meant. In each case there are two directions to show.

- The forward implications are based on the fact that we quantify over *all* valuations and all states. More precisely; assume an arbitrary frame (W, R) which does *not* have the property (e.g., reflexive). Find a valuation and a state where the axiom does not hold. You have now the contradiction ...

- For the backward implication take an arbitrary frame (W, R) which *has* the property (e.g., Euclidian). Take an arbitrary valuation and an arbitrary state on this frame. Show that the axiom holds in this state under this valuation. Sometimes one may need to use an inductive argument or to work with properties derived from the main property on R (e.g., if R is euclidian then w_1Rw_2 implies w_2Rw_2).

1.5.3 Proof theory and axiomatic systems

We only sketch on the proof theory of modal logic, basically because we are more interested in model checking as opposed to verify that a formula is valid. There are connections between these two questions, though. As explained in the introductory part, proof theory is about formalizing the notion of proof. That's done by defining a formal system (a proof system) that allows to *derive* or *infer* formulas from others. Formulas a priori given are also called *axioms*, and rules allow new formulas to be derived from previously derived ones (or from axiom). One may also see axioms as special form of rule, namely one without *premises*.

The style of presenting the proof system here is the plain old Hilbert-style presentation. As mentioned, there are other styles of presentations, some better suited for interactive, manual proofs and some for automatic reasoning, and in general more elegant anyway. Actually, Hilbert-style may just be around for reasons of tradition, insofar it was historically the first or among the first. Other forms like natural deduction or sequent presentation came later, actually also to improve on the presentation of the proof system, for instance being more "natural". One difference between Hilbert-style and the natural deduction style presentations is that Hilbert's presentation put's more weight on the axioms, whereas the alternative downplay the role of axioms and have more deduction rules (generally speaking). That may in itself not capture the core of the differences, but it's an aspect. As discussed, different classes of frames (transitive, reflexive ...) correspond to axioms or selection of axioms, and we have seen some.

Since we intend (classical propositional) modal logics to encompass classical propositinal logic not just syntactically but also conceptually/semantically, we have all propositional tautologies as derivable. Furthermore, we have the standard rule of derivation, already present in the propositional setting, namely *modus ponens*.

That so far took care of the propositional aspects (but note that MP can be applied to all formulas, of course, not just propositional ones). But we have not really taken care of the modal operators $\Box$ and $\Diamond$. Now, having lot of operators is nice for the user, but puts more burden when formulating a proof system (or implementing one) as we have to cover more case. So, we treat $\Diamond$ as *syntactic sugar*, as it can be expressed by $\Box$ and $\neg$. Note: "syntactic sugar" is a well-established technical turm for such situations, mostly use in the context of programming languages and compilers. Anyway, we now need to cover only one modal operator, and conventionally, it's $\Box$, necessitation. The corresponding rule consequently is often called the rule of (modal) *necessitation*. The rule is below called Nec, sometimes also just N or also called G (maybe historically so).

Is that all? Remember that we touched upon the issue that one can consider special classes of frames, for instance those with *transitive* relation R or other special cases, that lead them to special *axioms* being added to the derivation system. Currently, we do *not*

impose such restrictions, we want general frame validity. So does that mean, we are done? At the current state of discussion, we have the propositional part covered including the possibility do to propositional-style inference (with modus ponens), we have the plausible rule of necessitation, introducing the $\Box$-modality, the the two aspects of the logic (the propositional part and modal part seem conceptually separated. Note: a formula $\Box p \to \Box p$ "counts" as (an instance of a) propositional tautology, even if $\Box$ is mentioned. A question therefore is: are the two parts of the logic somehow connected, even if we don't assume anything about the set of underlying frames?

The answer is, yes, and that connection is captured by the *axiom* stating that $\Box$ *distributes* over $\to$. The axiom is known as distribution axiom or transitionally also as axiom K. In a way, the given rules are somewhat the *base line* for all classical modal logics. Modal logics with propositional part covered plus necessitation and axiom K are also called *normal* modal logics.

As a side remark: there are also certain modal logics where K is dropped or replace, which consequently are *no longer* normal logics. Note that it means they don't have a Kripke-model interpretation any longer. Note that our interest in Kripke-models is that we use transition systems as representing steps of programs, so therefore, Kripke-style thinking is natural in the context of our course. Non-normal logics are more esoteric and "unconventional" and we don't go there.

1.5.3.1 Base line axiomatic system ("K")

$$\frac{\varphi \text{ is a propositional tautology}}{\varphi}\ \text{PL}$$

$$\frac{}{\Box(\varphi_1 \to \varphi_2) \to (\Box\varphi_1 \to \Box\varphi_2)}\ \text{K}$$

$$\frac{\varphi \to \psi \qquad \varphi}{\psi}\ \text{MP}$$

$$\frac{\varphi}{\Box\varphi}\ \text{G}$$

The distribution axiom K is written as "rule" without premises. The system also focuses on the "new" part, i.e., the modal part. It's not explicit about how the rules look like that allow to *derive* propositional tautologies (which would be easy enough to do, and includes MP anyway).

The sketched logic is is also known under the name K itself, so K is not just the name of the axiom. The presentation here is Hilber-style, but there are different ways to make a derivation system for the logic K. On the next slide, there are a few more axioms (with their traditional names, some of which are just numbers, like "axiom 4" or "axiom 5"), and in the literature, then one considers and studies "combinations" of those axioms (like K + 5), and they are traditionally also known under special, not very transparent names like "S4" or "S5". See one of the next slides.

1.5.3.2 Sample axioms for different accessibility relations

$$\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi) \quad \text{(K)}$$
$$\Box\varphi \to \Diamond\varphi \quad \text{(D)}$$
$$\Box\varphi \to \varphi \quad \text{(T)}$$
$$\Box\varphi \to \Box\Box\varphi \quad (4)$$
$$\neg\Box\varphi \to \Box\neg\Box\varphi \quad (5)$$
$$\Box(\Box\varphi \to \psi) \to \Box(\Box\psi \to \varphi) \quad (3)$$
$$\Box(\Box(\varphi \to \Box\varphi) \to \varphi) \to (\Diamond\Box\varphi \to \varphi)) \quad \text{(Dum)}$$

The first ones are pretty common and are connected to more or less straightforward frame conditions (except K which is, as said, generally the case for a frame-based Kripke-style interpretation). Observe that T implies D.

The are many more different axiom studied in the literature, how they related and what not. The axiom called Dum is more esoteric ("[among the] most bizzare formulae that occur in the literature" [3]) and actually, there are even different versions of that (Dum$_1$, Dum$_2$ ...).

1.5.3.3 Different "flavors" of modal logic

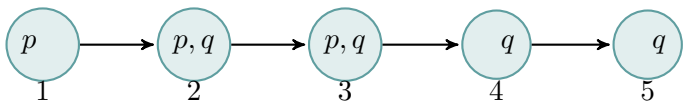
Logic	Axioms	Interpretation	Properties of R
D	K D	deontic	total
T	K T		reflexive
K45	K 4 5	doxastic	transitive/euclidean
S4	K T 4		reflexive/transitive
S5	K T 5	epistemic	reflexive/euclidean
			reflexive/symmetric/transitive
			equivalence relation

Concerning the terminology *doxastic* logic is about beliefs, *deontic* logic tries to capture obligations and similar concepts. Epistemic logic is about knowledge.

1.5.4 Exercises

1.5.4.1 Some exercises

Consider the frame (W, R) with $W = \{1, 2, 3, 4, 5\}$ and $(i, i+1) \in R$



Let the "valuation" $\tilde{V}(p) = \{2, 3\}$ and $\tilde{V}(q) = \{1, 2, 3, 4, 5\}$ and let the model M be $M = (W, R, V)$. Which of the following statements are correct in M and why?

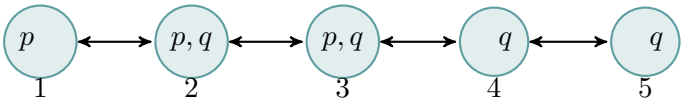
- $M, 1 \models \Diamond\Box p$
- $M, 1 \models \Diamond\Box p \to p$
- $M, 3 \models \Diamond(q \wedge \neg p) \wedge \Box(q \wedge \neg p)$
- $M, 1 \models q \wedge \Diamond(q \wedge \Diamond(q \wedge \Diamond(q \wedge \Diamond q)))$
- $M \models \Box q$

The answers to the above questions are

- yes
- no
- yes
- yes,
- yes. But why?

1.5.4.2 Exercises (2): bidirectional frames

1.5.4.2.1 Bidirectional frame A frame (W, R) is **bidirectional** iff $R = R_F + R_P$ s.t. $\forall w, w'(wR_Fw' \leftrightarrow w'R_Pw)$.



Consider $M = (W, R, V)$ from before. Which of the following statements are correct in M and why?

1. $M, 1 \models \Diamond\Box p$
2. $M, 1 \models \Diamond\Box p \to p$
3. $M, 3 \models \Diamond(q \wedge \neg p) \wedge \Box(q \wedge \neg p)$
4. $M, 1 \models q \wedge \Diamond(q \wedge \Diamond(q \wedge \Diamond(q \wedge \Diamond q)))$
5. $M \models \Box q$
6. $M \models \Box q \to \Diamond\Diamond p$

The notion of bidirectional The R can be separated into two disjoind relations R_F and R_P, which one is the inverse of the other.

1. $M, 1 \models \Diamond\Box p$: Incorrect
2. $M, 1 \models \Diamond\Box p \to p$: Correct
3. $M, 3 \models \Diamond(q \wedge \neg p) \wedge \Box(q \wedge \neg p)$: Incorrect
4. $M, 1 \models q \wedge \Diamond(q \wedge \Diamond(q \wedge \Diamond(q \wedge \Diamond q)))$: Correct
5. $M \models \Box q$: Correct ... but is it the same explanation as before?
6. $M \models \Box q \to \Diamond\Diamond p$

1.5.4.3 Exercises (3): validities

Which of the following are *valid* in modal logic. For those that are not, argue why and find a class of frames on which they become valid.

1. $\Box\bot$
2. $\Diamond p \to \Box p$
3. $p \to \Box\Diamond p$
4. $\Diamond\Box p \to \Box\Diamond p$

1. $\Box\bot$: Valid on frames where $R = \emptyset$.
2. $\Diamond p \to \Box p$: Valid on frames where R is a partial function.
3. $p \to \Box\Diamond p$: Valid on bidirectional frames.
4. $\Diamond\Box p \to \Box\Diamond p$: Valid on Euclidian frames.

As for further reading, [4] and [2] may be good reads.

Bibliography

[1] Baier, C. and Katoen, J.-P. (2008). *Principles of Model Checking*. MIT Press.

[2] Blackburn, P., de Rijke, M., and Venema, Y. (2001). *Modal Logic*. Cambridge University Press.

[3] Goré, R., Heinle, W., and Heuerding, A. (1997). Relations between propositional normal modal logics: an overview. *Journal of Logic and Computation*, 7(5):649–658.

[4] Harel, D., Kozen, D., and Tiuryn, J. (2000). *Dynamic Logic*. Foundations of Computing. MIT Press.

Index