Introduction to Logic

Alwen Tiu

The Australian National University

Summer Schools in Logic and Learning
26 January – 6 February 2009, Canberra

Logic and its applications in Computer Science

- Logic is about formalizing human reasoning.
- Not every form of reasoning can be given a precise model.
- Logic is used extensively in CS:
  - At the processor level: logic gates.
  - Hardware and software verification: floating point arithmetic verification, microkernel verification, etc.
  - High level programming: logic and constraint programming.
  - Artificial intelligence: planning, scheduling, diagnosis, agents, etc.

Outline of the lectures

Lecture 1 Propositional logic: syntax, semantics, basic notions such as models, boolean satisfiability, normal forms.

Lecture 2 First-order logic: syntax, semantics, some metatheory.

Lecture 3 Modal logic: syntax, semantics, several standard modal logics.

Part I

Propositional Logic

Propositional logic

- Propositional logic is concerned with *propositions*, i.e., statements which can be either true or false, and *compositions* of their truth values.
- *Atomic propositions* can be any sentences, e.g.,
  - It is raining.
  - Joe takes his umbrella.
  - $x < 0$.
  - $x = 0$.
- These sentences are considered atomic, i.e., the particular subjects/objects they mention are irrelevant.
- Atomic sentences are denoted by letters, such as, a, b, c, etc. These are called *propositional variables*.

Composing propositions

- Atomic propositions can be composed to form complex sentences, e.g.,

  It is raining and Joe takes his umbrella.

  Or

  $$x \leq 0 \;\text{ or }\; x = 0.$$

- We are interested in studying the truth value of the combined propositions, and how their truth can be systematically computed.
- The *satisfiability problem*: given a (complex) formula, how do we assign truth values to the atomic propositions in the formula so that the formula becomes true?

The language of propositional logic

The language of formulae is defined as the *least set of expressions* satisfying the following:

- Every propositional variable is a formula (also called an *atomic formula*).
- $\top$ ('true') and $\bot$ ('false') are formulae.
- If A is a formula then $\neg A$ ('not A') is a formula.
- If A and B are formulae then so are:
  - $A \wedge B$ (A 'and' B),
  - $A \vee B$ (A 'or' B),
  - and $A \rightarrow B$ (A 'implies' B).

Or, equivalently, in BNF notation:

$$F ::= p \mid \bot \mid \top \mid \neg F \mid F \wedge F \mid F \vee F \mid F \rightarrow F$$

where p is a propositional variable.

Syntax vs Semantics

- *Syntax* describes the *form* of a logical statement. *Semantics* describes its intended *meaning* (i.e., some mathematical structures, e.g., boolean algebra).
- *Soundness*: does a given syntactic proof procedure "respect" the semantics?
- *Completeness*: can all semantically valid logical statements be proved using a purely syntactic procedure?

Assigning truth values to formulae

Assuming we know the truth values of propositional variables, the truth value of a complex formula can be calculated as follows:

1. $\top$ is always true; $\bot$ is always false.
2. $A \wedge B$ is true if and only if A is true and B is true.
3. $\neg A$ is true if and only if A is false.
4. $A \vee B$ is true if and only if A is true or B is true.
5. $A \rightarrow B$ is true if and only if A is false, or A is true and B is true.

Boolean valuations and models

- A *boolean valuation* is a mapping from propositional variables to the set $\{0, 1\}$ (representing, respectively, 'false' and 'true').
- Example: the boolean valuation

$$\{x \mapsto 1, y \mapsto 0, z \mapsto 1, \ldots\}$$

  assigns x to true, y to false and z to true.
- A *model* of a formula F is a boolean valuation M such that F evaluates to 1 ('true') under the valuation M.
- We write $M \models F$ if M is a model for F.
- Note: the boolean valuation M is by definition an infinite set, but in practice we only show the relevant mappings for variables in F.

Boolean functions

- A formula can be seen as a *boolean function*, i.e., a function that maps boolean variables (propositionla variables) to the set $\{0, 1\}$.
- A boolean function can be defined easily by a *truth table*:
  - the columns correspond to the variables and the output of the function,
  - the rows of the tables correspond to all possible combination of input and their output.

Truth tables for standard connectives

x	y	$(x \wedge y)$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$(x \vee y)$
0	0	0
0	1	1
1	0	1
1	1	1

x	y	$(x \rightarrow y)$
0	0	1
0	1	1
1	0	0
1	1	1

x	$\neg x$
0	1
1	0

Satisfiability and validity

- An important question in propositional logic, and logic in general, is under what valuation a formula is true (or false).
- A formula F is *satisfiable* if it has a model, i.e., there exists a boolean valuation M such that $M \models F$. It is *unsatisfiable* if it has no model.
- A formula F is *valid* if it is true under *all* boolean valuation. Valid formulae are also called *tautologies*.
- Duality in logic: a formula F is valid if and only if $\neg F$ is unsatisfiable.

Closure under substitutions

- A useful property of valid formulae is that they remain valid if we replace its variables with arbitrary formulae.
- Example: $x \vee \neg x$ is valid. If we replace x with $(y \vee z)$ then

$$(y \vee z) \vee \neg(y \vee z)$$

  is also valid.

Logical equivalence

- A particular class of useful tautologies involves *logical equivalence*.
- Logical equivalence between two formulae A and B, written with $A \equiv B$, is defined as

$$(A \rightarrow B) \wedge (B \rightarrow A).$$

  That is, $A \equiv B$ is true if and only if the above formula is a tautology.
- Logical equivalence $\equiv$ satisfies the properties of an equivalent relation, i.e.,
  - reflexivity: $A \equiv A$
  - transitivity: $A \equiv B$ and $B \equiv C$ implies $A \equiv C$
  - symmetry: $A \equiv B$ implies $B \equiv A$.
- Logical equivalence is also a *congruence*, that is, if $A \equiv B$, then any occurence of A in a formula F can be replaced by B without changing the meaning of F.

Some useful tautologies

Units:	$A \wedge \top \equiv A$	$A \vee \top \equiv \top$
	$A \wedge \bot \equiv \bot$	$A \vee \bot \equiv A$
	$A \vee \neg A \equiv \top$	$A \wedge \neg A \equiv \bot$
Idempotency:	$A \wedge A \equiv A$	$A \vee A \equiv A$
Commutativity:	$A \wedge B \equiv B \wedge A$	$A \vee B \equiv B \vee A$

Associativity:
$$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$$
$$A \vee (B \vee C) \equiv (A \vee B) \vee C$$

Distributivity:
$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$
$$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$$

Implication:
$$A \rightarrow B \equiv \neg A \vee B$$

de Morgan: $\neg\neg A \equiv A \qquad \neg(A \wedge B) \equiv \neg A \vee \neg B \qquad \neg(A \vee B) \equiv \neg A \wedge \neg B$

Excluded middle:
$$A \vee \neg A$$

Contrapositive:
$$A \rightarrow B \equiv \neg B \rightarrow \neg A.$$

Normal forms

In checking validity or satisfiability, certain forms of formulae are easier to work with than others. Three important normal forms that are commonly used:

- Negation normal form (NNF).
- Disjunctive Normal Form (DNF).
- Conjunctive Normal Form (CNF).

Simplifying assumptions:

- We shall work only with formulae which are implication-free. That is, every implication $A \rightarrow B$ is converted into its equivalent $\neg A \vee B$.
- We assume the formulae do not contain the nullary operators $\perp$ and $\top$.

Negation normal form

- A formula is in *negation normal form* if the negation operator is applied only to variables.
- Every formula can be transformed into an equivalent NNF using the following tautologies:

$$\begin{array}{ll} \text{DM1:} & \neg(A \wedge B) \equiv \neg A \vee \neg B \\ \text{DM2:} & \neg(A \vee B) \equiv \neg A \wedge \neg B \\ \text{DM3:} & \neg\neg A \equiv A \end{array}$$

NNF Algorithm

function $\mathrm{NNF}(A)$

case A **of**

A is a literal: **return** A

A is $\neg\neg B$: **return** $\mathrm{NNF}(B)$

A is $B \wedge C$: **return** $\mathrm{NNF}(B) \wedge \mathrm{NNF}(C)$

A is $B \vee C$: **return** $\mathrm{NNF}(B) \vee \mathrm{NNF}(C)$

A is $\neg(B \wedge C)$: **return** $\mathrm{NNF}(\neg B) \vee \mathrm{NNF}(\neg C)$

A is $\neg(B \vee C)$: **return** $\mathrm{NNF}(\neg B) \wedge \mathrm{NNF}(\neg C)$

end case

Note: a *literal* is a variable or a negated variable, e.g., $\neg x$.

Conjunctive Normal Form

- A *literal* is a propositional variable or a negation of a propositional variable, e.g., x, $\neg y$, etc.
- A formula is in *conjunctive normal form* (CNF) if it is of the form

$$(A_{11} \vee \cdots \vee A_{1k_1}) \wedge \cdots \wedge (A_{m1} \vee \cdots \vee A_{mk_m})$$

  where each A_{ij} is a literal.
- In other words, a CNF formula is a conjunction of disjunctions of literals.
- Every NNF formula can be transformed into an equivalent CNF formula using the distributivity laws. Hence, every formula can be transformed into an equivalent CNF formula.

CNF Algorithm (1)

Input: a formula in NNF.
Output: a CNF formula equivalent to the input.
function $\mathrm{CNF}(A)$

  case A **of**

    A is a literal: **return** A
    A is $B \wedge C$: **return** $\mathrm{CNF}(B) \wedge \mathrm{CNF}(C)$
    A is $B \vee C$: **return** $\mathrm{DistCNF}(\mathrm{CNF}(B), \mathrm{CNF}(C))$

  end case

$\mathrm{DistCNF}$ distributes a CNF over another CNF, using the distributive laws.

CNF Algorithm (2)

Input: two CNF formulae A and B.
Output: a CNF of $A \vee B$.

function $\mathrm{DistCNF}(A, B)$
  if A is $C \wedge D$ **then**
    return $\mathrm{DistCNF}(C, B) \wedge \mathrm{DistCNF}(D, B)$
  else if B is $E \wedge F$ **then**
    return $\mathrm{DistCNF}(A, E) \wedge \mathrm{DistCNF}(A, F)$
  else
    return $A \vee B$.
  end if

Disjunctive Normal Form

- A formula is in *disjunctive normal form* (DNF) if it is of the form

$$(A_{11} \wedge \cdots \wedge A_{1k_1}) \vee \cdots \vee (A_{m1} \wedge \cdots \wedge A_{mk_m})$$

  for some m and n. As in CNF, each A_{ij} is a literal.
- Every NNF formula can be transformed into an equivalent DNF formula using the distributivity laws.
- Exercise: give an algorithm for DNF transformation.

The resolution proof method

- The resolution proof method is really a *refutation procedure*.
- Given a formula in CNF, it checks whether the formula is unsatisfiable.
- Since validity is dual to unsatisfiability, we can use resolution to check for validity: F is valid if $\neg F$ is unsatisfiable.

A representation of CNF using sets

- To simplify presentation, we use another notation to represent CNF as sets of sets.
- A clause is a set of literals, e.g.,

  $$\{x, y, \neg z\}.$$

  It represents the disjunction of the literals in the set, e.g., the above clause corresponds to $x \vee y \vee \neg z$.
- A formula in CNF is represented as a set of clauses. Each clause corresponds to a conjunct in the CNF.
- For example, $(a \vee b \vee c) \wedge (\neg a \vee \neg b) \wedge \neg c$ is represented as

  $$\{\{a, b, c\}, \{\neg a, \neg b\}, \{\neg c\}\}.$$

  We call this representation the clausal form of the original formula.

The inference rule for resolution

- The basic mechanism of the resolution proof method is an *inference rule* for forming a new clause given two existing ones:

$$\frac{\{l_1, \ldots, l_m, x\} \qquad \{\neg x, k_1, \ldots, k_n\}}{\{l_1, \ldots, l_m, k_1, \ldots, k_n\}} \; res$$

  Here l_i and k_j are literals, and x is a variable.
- The rule says that, given two clauses such that one clause contains a complement of a literal in the other, form the union of the two clauses, minus that literal and its complement.

Saturated sets of clauses

- Given clauses C_1 and C_2, we write $res(x, C_1, C_2)$ to denote the resulting clause obtained by resolving C_1 and C_2 on the variable x.
- A set of clauses Δ is saturated if for all $C_1 \in \Delta$ and $C_2 \in \Delta$, if $res(x, C_1, C_2)$ is defined, then

$$res(x, C_1, C_2) \in \Delta.$$

- In other words, Δ is closed under the resolution rule; applying resolution to any two clauses in Δ results in another clause already in Δ.

Unsatisfiability testing with resolutions

Input: a set of clauses Δ
Output: return true if Δ is unsatisfiable, otherwise return false.
function unsat(Δ):

- **if** $\{\} \in \Delta$ **then**
  - return true
- **else if** Δ is saturated **then**
  - return false
- **else**
  - select C_1, C_2 and x such that $res(x, C_1, C_2)$ is defined and $res(x, C_1, C_2) \notin \Delta$
  - return unsat($\Delta \cup \{res(x, C_1, C_2)\}$)
- **end if**

Examples

- Construct a resolution proof for

$$\{\{\neg x, y\}, \{\neg y, \neg z\}, \{x, \neg z\}, \{z\}\}.$$

- Clause reuse: some clauses may be used more than once. Example:

$$\{\{a, b\}, \{c, d\}, \{\neg a, \neg c\}, \{\neg a, \neg d\}, \{\neg b, \neg c\}, \{\neg b, \neg d\}\}.$$

Unit resolution

- *Unit resolution* is a special case of the resolution rule where one of the clauses to be resolved is a *unit clause*, i.e., a set containing only one literal.
- Unit resolution is obviously less expensive than the general resolution. But it is incomplete.
- Example: the set of clauses

  $$\{\{a, b\}, \{\neg a, b\}, \{\neg b, c\}, \{\neg b, \neg c\}\}$$

  is unsatisfiable but cannot be refuted using unit resolution alone.
- Unit resolution is often used as a simplifying step in satisfiability testing.

Soundness and completeness of resolution

Theorem

The resolution method is sound. That is, given a clausal form Δ, if unsat(Δ) returns true, then Δ is unsatisfiable.

Theorem

The resolution method is complete. That is, if a clausal form Δ is unsatisfiable, then unsat(Δ) returns true.

Satisfiability testing

- Satisfiability testing (SAT) is the problem of determining whether a given propositional formula is satisfiable or not.
- Assume input in clausal form (or equivalently, conjuctive normal form).
- Satisfiability testing is the first problem shown to be NP-complete. See:
  S. A. Cook. *The complexity of theorem-proving procedures*. Proceedings of the third annual ACM symposium on Theory of Computing, 1971.
- There is a large amount of research done in finding heuristics for efficient SAT solving.
- SAT competition: `http://www.satcompetition.org`.

Heuristics for SAT solving

- Most of current SAT solvers are based on DPLL algorithm (after Davis, Putnam, Logemann and Loveland). See:
  - M. Davis and H. Putnam. *A computing procedure for quantification theory*. Journal of the ACM, Vol. 7 (3), 1960.
  - M. Davis, G. Logemann, and D. Loveland. *A machine program for theorem proving*. Communications of the ACM, Vol. 5 (7), 1962.
- Much improvement has been done on DPLL, e.g., using various sorts of "conflict analysis". See, e.g.,:
  J.P. Marques-Silva and K.A. Sakallah. *GRASP: A search algorithm for propositional satisfiability*. IEEE Transactions on Computers, Vol. 48 (5), 1999.
- Modern SAT solvers have achieved a high level of efficiency that they are often used as the solver for various NP-complete problems.

Representing boolean functions

- In some application domains, such as formal verification and diagnoses (of digital circuits), propositional logic is used as a representation language for systems and their properties.
- Two main problems related to this use of propositional logic:
  - Optimality of representation: how much space needed to represent boolean functions.
  - Efficiency of (logical) operations: what is the complexity of logical operations on this representation.

Binary Decision Tree (BDT)

A *binary decision tree* is a labelled binary tree satisfying:

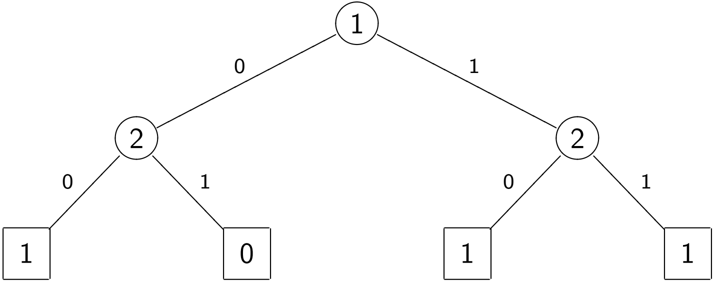
1. The leaves are labelled with either 0 (false) or 1 (true).
2. The non-leaf nodes are labelled with positive integers.
3. For every non-leaf node labelled with i has two child nodes, both labelled with $i + 1$.
4. The branches of the trees are labelled with either 0 (the *low branch*) or 1 (*the high branch*). Every non-leaf node has a low branch and a high branch.

Representing formulae as BDTs

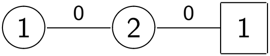
- Assume variables are totally ordered, e.g., by assigning indices to them.
- Let F be a formula with variables $x_1, \ldots, x_n$. A BDT T is a representation of F if
  - the internal nodes of T are labelled with $\{1, \ldots, n\}$,
  - every path from the root to a 1-leaf (a 0-leaf) represents a valuation which makes F true (false).
- In other words, a BDT for F is just another way of writing the truth table of F.
- Its size is exponential in the number of variables in F.

Example: a binary decision tree

BDT for $x \vee \neg y$, with variable ordering: $x < y$.



Notice that the path



corresponds to the valuation

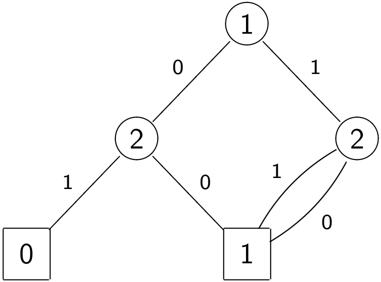
$$\{x \mapsto 0, y \mapsto 0\}$$

Binary Decision Diagram (BDD)

- A more economical representation of BDT is to allow sharing of subtrees.
- A *binary decision diagram* of n variables is a rooted directed acyclic graph G satisfying the following condition:
  1. Every *terminal* vertex of G is labelled with a value $\mathit{value}(v) \in \{0, 1\}$.
  2. Every *nonterminal* vertex is labelled with an index $\mathit{index}(v) \in \{1, \dots, n\}$ and has two children $\mathit{low}(v)$ and $\mathit{high}(v)$.
  3. For every non terminal vertex v, if $\mathit{low}(v)$ is nonterminal then
  $$\mathit{index}(v) < \mathit{index}(\mathit{low}(v)).$$
  Similarly, if $\mathit{high}(v)$ is nonterminal, then
  $$\mathit{index}(v) < \mathit{index}(\mathit{high}(v)).$$
- NOTE: if v is a terminal vertex then $\mathit{index}(v) = n + 1$.

Example: a BDD

A BDD representing $x \vee \neg y$, assuming the ordering $x < y$.



Isomorphic BDDs

Definition

Two BDDs G and G' are isomorphic if there exists a one-to-one mapping σ from vertices of G onto the vertices of G' such that for any vertex v, if $\sigma(v) = v'$ then either

- both v and v' are terminal vertices with $value(v) = value(v')$,
- or both v and v' are nonterminal vertices with $index(v) = index(v')$, $\sigma(low(v)) = low(v')$ and $\sigma(high(v)) = high(v')$.

Definition

For any vertex v in a BDD G, the subgraph rooted by v is the subgraph of G consisting of v and all of its decendants.

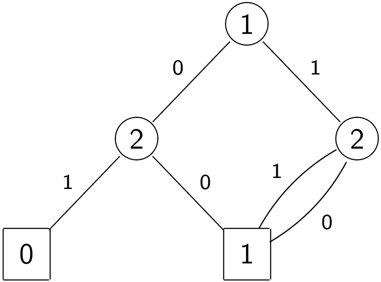
Reduced BDD

Definition

A BDD G is *reduced* if it contains no vertex v with $low(v) = high(v)$, nor does it contain distinct vertices v and v' such that the subgraphs rooted by v and v' are isomorphic.

Example: a non-reduced BDD

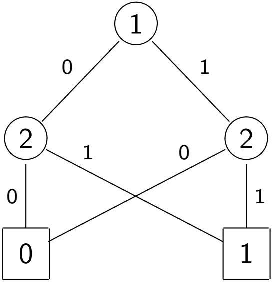
A BDD for $x \vee \neg y$, with $x < y$.

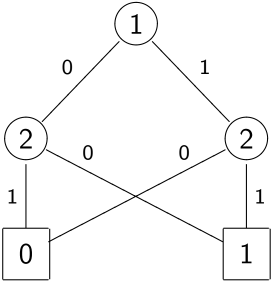


If x is true, then the value of y does not matter, since the function will always evaluate to true.

Example: a reduced BDD

Which BDD is reduced?





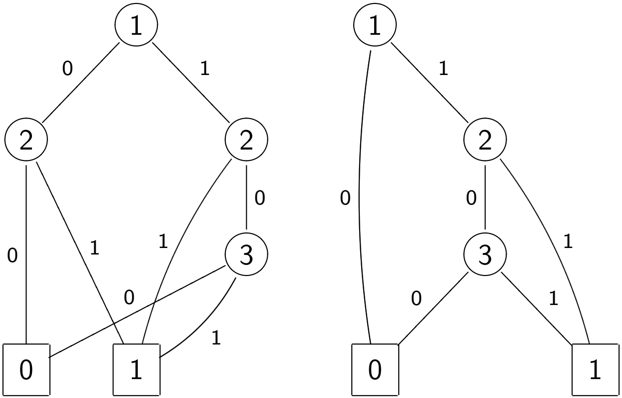
Reducing BDDs

- Given a BDD G, there is a polynomial algorithm that computes its reduced form. See:
- The algorithm works by identifying isomorphic subgraphs, starting with the terminal nodes.
- See the following paper for details:
  Randal E. Bryant. *Graph-Based Algorithms for Boolean Function Manipulation.* IEEE Transactions on Computers, Vol. 35 (8), pages 677 – 691, 1986.

Dependency on variable ordering

Ordering of variables affects the size of the reduced BDD of a formula F.
Consider the formula $(x \wedge z) \vee y$.

$x < y < z$ $\qquad$ $x < z < y$



Part II

First-Order Logic

First-order logic

- First-order logic extends propositional logic by allowing certain forms of reasoning about individual objects in logical statements.
- In particular, it allows
  - representation of *relations* between individuals (also called *predicates*),
  - representation of individuals and functions on individuals, and
  - quantification over individuals: "for all" and "exists".

Predicates, functions and constants

We assume a countably infinite set $\mathbf{V}$ of variables.
A first-order language is determined by specifying:

1. A countable set $\mathbf{R}$ of *relation symbols*, or *predicate symbols*. Each predicate symbol $P \in \mathbf{R}$ has an *arity*, which is a non-negative integer, denoting the number of arguments P takes.
2. A countable set $\mathbf{F}$ of *function symbols*, each of which is associated with an arity.
3. A countable set $\mathbf{C}$ of *constant symbols*.

The triple $\Sigma = \langle \mathbf{R}, \mathbf{F}, \mathbf{C} \rangle$ is called a (first-order) signature.

First-order terms

Let $\Sigma = \langle \mathbf{R}, \mathbf{F}, \mathbf{C} \rangle$ be a first-order signature. The set of Σ-terms is the smallest set satisfying

- Any variable in $\mathbf{V}$ is a term.
- Any constant symbol in $\mathbf{C}$ is a term.
- If f is a function symbol of arity n, and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

Formulae of first-order logic

Let $\Sigma = \langle \mathbf{R}, \mathbf{F}, \mathbf{C} \rangle$ be a first-order signature. The set of Σ-formulae is defined as follows:

- The expression $R(t_1, \ldots, t_n)$, where $R \in \mathbf{R}$ is a predicate symbol of arity n and each t_i is a Σ-term, is a formula. It is called an *atomic formula*.
- $\top$ and $\bot$ are formulae.
- If F is a formula then $\neg F$ is a formula.
- If F and G are formulae, then $F * G$ is a formula, for any binary propositional connective $*$.
- If F is a formula then so are $\forall x.F$ and $\exists x.F$.

The symbol $\forall$ is the universal quantifier and $\exists$ is the existential quantifier.

Notational convention

To simplify presentation, certain alphabet symbols are associated with certain syntactic categories:

- Variables: x, y, z.
- Constants: a, b, c and d.
- Function symbols: f, g, h.
- Predicate symbols: P, Q, R.
- Terms: s, t and u.
- Formulae: A, B, C, D, F and G.

We also use descriptive words such as "plus", "minus", "equal", etc. to represent function/predicate symbols.

Some example formulae

- "All men are mortal": Define a signature $\Sigma = \langle \mathbf{R}, \mathbf{F}, \mathbf{C} \rangle$ as follows:

  $$\mathbf{R} = \{man/1, mortal/1\} \qquad \mathbf{F} = \{\} \qquad \mathbf{C} = \{\}$$

  Then the sentence can be represented as the first-order formula:

  $$\forall x.(man(x) \to mortal(x)).$$

- Consider the signature Σ defined as follows:

  $$\mathbf{R} = \{equal/2\} \qquad \mathbf{F} = \{plus/2\} \qquad \mathbf{C} = \{\}$$

  Suppose that *plus* denotes the addition operator, and *equal* denotes equality on natural numbers. Then commutativity of addition can be stated as the formula:

  $$\forall x \forall y. equal(plus(x, y), plus(y, x)).$$

  Alternatively, we can use more familiar symbols, written in infix notation, e.g.,

  $$\forall x \forall y.(x + y = y + x).$$

Free and bound variables

- The variable x in the formula $\forall x.P(x)$ is a *bound variable*, whose *scope* is $P(x)$.
- The two occurrences of x in

$$(\forall x.P(x)) \wedge Q(x)$$

  must be distinguished. The right-most occurrence of x is called a *free variable*, since it is not under the scope of any quantifier.
- The free variables of a formula is defined inductively as follows:
  - The free variables of an atomic formula are all the variables occuring in that formula.
  - The free variables of $\neg F$ are the free variables of F.
  - The free variables of $F * G$, where $*$ is a binary connective, are the free variables of F together with the free variables of G.
  - The free variables of $\forall x.F$ and $\exists x.F$ are the free variables of A, except for x.

Variable-naming convention

- Bound variables in formulae can be renamed without changing its meaning: for example,

$$\forall x.P(x) \qquad \text{and} \qquad \forall y.P(y)$$

  have the same meaning, i.e., they state that the predicate P holds for all individual.
- To simplify discussions, we adopt the following naming conventions in writing formulae:
  Bound variables are always chosen so that they are distinct from free variables.

Semantics: models

A model for the first-order language determined by $\Sigma = \rangle \mathbf{R}, \mathbf{F}, \mathbf{C} \rangle$ is a pair $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ where:

- $\mathbf{D}$ is a non-empty set, called the domain of $\mathbf{M}$, and
- $\mathbf{I}$ is a mapping, called an interpretation that associates:
    - every constant symbol $c \in \mathbf{C}$ with some element $c^{\mathbf{I}} \in \mathbf{D}$;
    - every n-ary function symbol $f \in \mathbf{F}$ with some n-ary function $f^{\mathbf{I}} : \mathbf{D}^n \to \mathbf{D}$; and
    - every n-ary relation symbol $P \in \mathbf{R}$ with some n-ary relation $P^{\mathbf{I}} \subseteq \mathbf{D}^n$.

Semantics: assignments and valuation of terms

Let $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ be a model for the first-order language determined by $\Sigma = \langle \mathbf{R}, \mathbf{F}, \mathbf{C} \rangle$.

- An assignment in the model $\mathbf{M}$ is a mapping $\mathbf{A}$ from the set of variables to the domain $\mathbf{D}$.
- Given an assignment $\mathbf{A}$, to each Σ-term t, we associate a value $t^{\mathbf{I},\mathbf{A}}$ in $\mathbf{D}$ as follows:
  1. for a constant symbol c, $c^{\mathbf{I},\mathbf{A}} = c^{\mathbf{I}}$;
  2. for a variable v, $v^{\mathbf{I},\mathbf{A}} = \mathbf{A}(v)$;
  3. for a function symbol f,

$$[f(t_1, \dots, t_n)]^{\mathbf{I},\mathbf{A}} = f^{\mathbf{I}}(t_1^{\mathbf{I},\mathbf{A}}, \dots, t_n^{\mathbf{I},\mathbf{A}}).$$

Example

Suppose Σ is the signature defined by:

$$\mathbf{R} = \{\} \qquad \mathbf{F} = \{s/1, +/2\} \qquad \mathbf{C} = \{0\}$$

Consider the term $s(s(0) + s(x))$ (where $+$ is written in infix notation). Several choices of model $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ and assignment $\mathbf{A}$:

- $\mathbf{D} = \{0, 1, 2, \ldots\}$, $0^{\mathbf{I}} = 0$, $s^{\mathbf{I}}$ is the successor function and $+^{\mathbf{I}}$ is the addition operation. If $\mathbf{A}$ is an assignment such that $\mathbf{A}(x) = 3$ then
$$[s(s(0) + s(x))]^{\mathbf{I},\mathbf{A}} = 6.$$
- $\mathbf{D}$ is the collection of words over alphabet $\{a, b\}$, $0^{\mathbf{I}} = a$, and $s^{\mathbf{I}}$ is the operation of appending a to the end of a word, and $+^{\mathbf{I}}$ is the concatenation. If $\mathbf{A}(x) = aba$ then
$$[s(s(0) + s(x))]^{\mathbf{I},\mathbf{A}} = aaabaaa.$$

Semantics: assigning truth values to formulae

- Variants: Let x be a variable. The assignments $\mathbf{A}$ and $\mathbf{B}$ are *x-variants* if they assign the same value to every variable, except possibly x.
- Let $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ be a model for $\Sigma = \langle \mathbf{R}, \mathbf{F}, \mathbf{C} \rangle$ and let $\mathbf{A}$ be an assignment in $\mathbf{M}$. To each Σ-formula G, we associate a truth value $G^{\mathbf{I},\mathbf{A}}$ ($\mathbf{t}$ or $\mathbf{f}$) as follows:
  - $[P(t_1, \ldots, t_n)]^{\mathbf{I},\mathbf{A}} = \mathbf{t}$ if and only if $\langle t_1^{\mathbf{I},\mathbf{A}}, \ldots, t_n^{\mathbf{I},\mathbf{A}} \rangle \in P^{\mathbf{I}}$.
  - $\top^{\mathbf{I},\mathbf{A}} = \mathbf{t}$, $\bot^{\mathbf{I},\mathbf{A}} = \mathbf{f}$.
  - $[\neg G]^{\mathbf{I},\mathbf{A}} = \neg[G^{\mathbf{I},\mathbf{A}}]$, $[G * H]^{\mathbf{I},\mathbf{A}} = G^{\mathbf{I},\mathbf{A}} * H^{\mathbf{I},\mathbf{A}}$, for any binary connective $*$.
  - $[\forall x.G]^{\mathbf{I},\mathbf{A}} = \mathbf{t}$ if and only if $G^{\mathbf{I},\mathbf{B}} = \mathbf{t}$ for every assignment $\mathbf{B}$ that is an x-variant of $\mathbf{A}$.
  - $[\exists x.G]^{\mathbf{I},\mathbf{A}} = \mathbf{t}$ if and only if $G^{\mathbf{I},\mathbf{B}} = \mathbf{t}$ for some assignment $\mathbf{B}$ that is an x-variant of $\mathbf{A}$.

Satisfiability and validity

Let $\Sigma = \langle \mathbf{R}, \mathbf{F}, \mathbf{C} \rangle$.

- A Σ-formula G is true in the model $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ if $G^{\mathbf{I},\mathbf{A}} = \mathbf{t}$ for all assignments $\mathbf{A}$.
- A Σ formula G is valid if G is true in all models of the language.
- A set S of Σ-formulae is satisfiable in $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ if there is some assignment $\mathbf{A}$ (called a satisfying assignment) such that $G^{\mathbf{I},\mathbf{A}} = \mathbf{t}$ for all $G \in S$. S is satisfiable if it is satisfiable in some model.

Example

Let $\Sigma = \langle \mathbf{R}, \mathbf{F}, \mathbf{C} \rangle$ where $\mathbf{R} = \{R/2, \oplus/2\}$. Suppose $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ be a model for Σ.
Suppose $\mathbf{D} = \{1, 2, 3, \ldots\}$ and $\oplus^{\mathbf{I}}$ is the addition operator. Consider the following interpretations of R:

- $R^{\mathbf{I}}$ is the equality relation and let $G = \exists y.R(x, y \oplus y)$. Then $G^{\mathbf{I},\mathbf{A}}$ is true if and only if $\mathbf{A}(x)$ is an even number.
- $R^{\mathbf{I}}$ is the greater-than relation and $G = \forall x \forall y \exists z.R(x \oplus y, z)$. Then G is true in $\mathbf{M}$.
- $R^{\mathbf{I}}$ is the greater-than-by-4-or-more relation and $G = \forall x \forall y \exists z.R(x \oplus y, z)$. Then the formula G is not true in this model.

Example

Let $\Sigma = \langle \mathbf{R}, \mathbf{F}, \mathbf{C} \rangle$ where $\mathbf{R} = \{R/2, \oplus/2\}$. Suppose $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ be a model for Σ.

- Suppose $\mathbf{D}$ is the set of real numbers and $R^{\mathbf{I}}$ is the greater-than relation. Then the formula

$$\forall x \forall y [R(x, y) \rightarrow \exists z (R(x, z) \wedge R(z, y))]$$

  is true in $\mathbf{M}$ (it expresses the denseness of the reals).
- Suppose $\mathbf{D} = \{7, 8\}$ and $R^{\mathbf{I}} = \{(7, 8)\}$. Then the formula $\forall x \forall y [R(x, y) \rightarrow R(y, x)]$ (symmetry) is not true in $\mathbf{M}$.

Example

Drinkers paradox: There is someone in the pub such that if he/she is drinking then everybody in the pub is drinking.
More formally:

$$\exists x.(drinks(x) \rightarrow \forall y.drinks(y)).$$

This formula is valid.
Proof: ...

Example: Peano axioms

G. Peano introduced an axiomatization of natural number (1889). Some of the axioms are given below. We assume the following signature:

$$\mathbf{R} = \{= /2\} \qquad \mathbf{F} = \{s/1\} \qquad \mathbf{C} = \{0\}$$

1. $\forall x. x = x.$
2. $\forall x \forall y. x = y \rightarrow y = x.$
3. $\forall x \forall y \forall z.(x = y \wedge y = z) \rightarrow x = z.$
4. $\forall x. \neg(s(x) = 0).$
5. $\forall x \forall y. s(x) = s(y) \rightarrow x = y.$

Example: the standard model for Peano axioms

The "standard model": $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ where

- $\mathbf{D}$ is the set of natural numbers $\{0, 1, 2, \ldots\}$,
- $0^{\mathbf{I}} = 0$, $s^{\mathbf{I}}$ is the successor function (i.e., $s^{\mathbf{I}}(n) = n + 1$)
- $=^{\mathbf{I}}$ is the equality relation on natural numbers.

Each of the axioms is true in $\mathbf{M}$.

Example: no finite model for Peano axioms

There cannot be a model $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$, where $\mathbf{D}$ is finite, satisfying all the Peano axioms.

Proof.

Suppose otherwise, i.e., $\mathbf{D} = \{a_0, \dots, a_n\}$ for some $n \geq 0$. Suppose $0^{\mathbf{I}} = a_0$. Consider the following "contrapositive" form of Axiom 5:

$$\forall x \forall y. \neg(x = y) \rightarrow \neg(s(x) = s(y)).$$

It says that the successor function must be injective. But Axiom 4: $\forall x. \neg(s(x) = 0)$ implies that s is a mapping from

$$\{a_0, \dots, a_n\} \qquad \text{to} \qquad \{a_1, \dots, a_n\}.$$

Therefore, there must be a_i and a_j in $\mathbf{D}$ such that $a_i \neq a_j$ and

$$s^{\mathbf{I}}(a_i) = s^{\mathbf{I}}(a_j),$$

contradicting the injectivity of $s^{\mathbf{I}}$. $\square$

Example: a non-standard model for Peano axioms

A "non-standard model": $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$

- $\mathbf{D}$ is the set of natural numbers plus a new element ω (representing an "infinite number").
- $0^{\mathbf{I}} = 0$, $s^{\mathbf{I}}$ is the successor function on natural numbers, but

$$s^{\mathbf{I}}(\omega) = \omega.$$

- $=^{\mathbf{I}}$ is an equality relation on $\mathbf{D}$, with the additional requirements that:
  - $s^{\mathbf{I}}(\omega) = \omega$
  - $\omega \neq n$, for any natural number n.

Note: natural numbers cannot be characterized by finitely many first-order formulae. We need a "second-order" formula expressing the *induction principle* on natural numbers:

$$\forall P.[P(0) \land (\forall x.P(x) \rightarrow P(s(x)))] \rightarrow \forall x.P(x).$$

Some tautologies

- All propositional tautologies are also first-order tautologies.
- Some tautologies involving quantifiers:
  - $(\forall x.F) \land G \equiv \forall x(F \land G)$, $(\exists x.F) \land G \equiv \exists x(F \land G)$
  - $(\forall x.F) \lor G \equiv \forall x(F \lor G)$, $(\exists x.F) \lor G \equiv \exists x(F \lor G)$
  - $G \to \forall x.F \equiv \forall x(G \to F)$
  - $G \to \exists x.F \equiv \exists x(G \to F)$
  - $(\forall x.F) \to G \equiv \exists x(F \to G)$
  - $(\exists x.F) \to G \equiv \forall x(F \to G)$

  provided that x is not free in G.
- De Morgan law for quantifiers:
  - $\neg(\forall x.F) \equiv \exists x.\neg F$
  - $\neg(\exists x.F) \equiv \forall x.\neg F$

Prenex Normal Form

- A formula is in *prenex normal form* if it is of the form

$$Q_1 x_1 \cdots Q_n x_n . F$$

  where each Q_i is either $\forall$ or $\exists$ and F is a quantifier-free formula.
- Every formula is equivalent a formula in prenex normal form. This is done as follows:
  - Rename the bound variables so that they are pairwise distinct and also distinct from the free variables.
  - Apply the tautologies (in the previous slide) to bring each quantifier to the outer level of the formula.
- Prenex normal form transformation is often used as a pre-processing step for first-order automated theorem proving.

Example

Let F be the formula

$$\neg[P(x) \wedge \forall x.(Q(x) \rightarrow \exists y.R(x,y))].$$

To transform F into prenex normal form, first rename the bound variables:

$$\neg[P(x) \wedge \forall z.(Q(z) \rightarrow \exists y.R(z,y))].$$

Then apply first-order equivalences:

$$\begin{aligned}
&\neg[P(x) \wedge \forall z.(Q(z) \rightarrow \exists y.R(z,y))] \\
\equiv\ &\neg[P(x) \wedge \forall z \exists y(Q(z) \rightarrow R(z,y))] \\
\equiv\ &\neg[\forall z(P(x) \wedge \exists y(Q(z) \rightarrow R(z,y))] \\
\equiv\ &\neg[\forall z \exists y(P(x) \wedge (Q(z) \rightarrow R(z,y)))] \\
\equiv\ &\exists z.\neg[\exists y(P(x) \wedge (Q(z) \rightarrow R(z,y)))] \\
\equiv\ &\exists z \forall y.\neg[P(x) \wedge (Q(z) \rightarrow R(z,y))]
\end{aligned}$$

Metatheory: compactness

Theorem

Let S be a set of first-order formulae. If every finite subset of S is satisfiable, then so is S.

Corollary

Any set S of first-order formulae that is satisfiable in an arbitrary large finite model is satisfiable in some infinite model.

An important consequence of compactness theorem is that *the notion of being finite cannot be captured using the machinery of first-order logic.*

Metatheory: Löwenheim-Skolem Theorem

Theorem

Let S be a set of first-order formulae. If S is satisfiable, then S is satisfiable in a countable model.

Note: since the set of real numbers is uncountable, it has no first-order characterization.

Part III

Modal Logic

Modal logic

- Modal logic was originally developed to study reasoning about notions of "necessity" and "possibility".
- However, it is often used to describe a family of logics that capture different "modes" of truth:
  - Modal logic: "It is neccessary that ..", "It is possible that .."
  - Deontic logic: "It is obligatory that ...", "Its permitted that ..."
  - Temporal logic: "It will always be the case that ..", "It has always been the case that ..."
  - Epistemic logic: "Alice knows that ...", "Bob knows that Alice knows that .."
  - Dynamic logic: "Every execution of a program P leads to a state such that ..."

Possible worlds semantics

- A commonly used mathematical model of truth in modal logic is that of Kripke semantics (also called possible worlds semantics).
- Truth of a propositional statement is relative to the "world" it lives in.
- Different interpretation of worlds: A world can describe
  - a point in time;
  - the state of a running computer program;
  - knowledge of an (autonomous) agent;
  - etc.
- In general, worlds can be understood abstractly as a *relational structure*, i.e., a set with some defined relations on its elements.

Syntax

- Modal logic extends propositional logic with two modal operators: $\Box$ ('neccessity') and $\Diamond$ ('possibility').
- The language of basic modal logic:

$$F ::= p \mid \top \mid \bot \mid \neg F \mid F \wedge F \mid F \vee F \mid F \rightarrow F \mid \Box F \mid \Diamond F.$$

- Some examples:
  - $\Diamond A$: "A is possible"
  - $\neg\Diamond A$: "A is impossible"
  - $\neg\Diamond(\neg A)$: "not-A is impossible"
  - $\Box A$: "A is necessary"

Kripke Semantics: Frames and Models

- A frame is a pair $\mathcal{F} = \langle \mathcal{W}, \mathcal{R} \rangle$ such that
  - $\mathcal{W}$ is a non-empty set, also called the *worlds*, and
  - $\mathcal{R}$ is a binary relation on $\mathcal{W}$.

  A frame is basically just a (possibly infinite) directed graph, where $\mathcal{W}$ is the set of vertices and $\mathcal{R}$ the set of edges.
- A model for the basic modal language is a pair $\mathcal{M} = \langle \mathcal{F}, \mathcal{V} \rangle$, where $\mathcal{F}$ is a frame, and $\mathcal{V}$ is a function assigning each propositional variable to a subset of $\mathcal{W}$.
  Intuitively, $\mathcal{V}(p)$ is the set of worlds in which p is true.
- Notation: We shall sometimes write $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle$ if $\mathcal{F} = \langle \mathcal{W}, \mathcal{R} \rangle$.

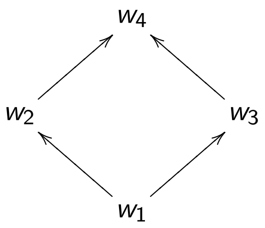
Kripke Semantics: Graphical Representation

Let $\mathcal{W} = \{w_1, w_2, w_3, w_4\}$, $\mathcal{R} = \{(w_1, w_2), (w_1, w_3), (w_2, w_4), (w_3, w_4)\}$ and $\mathcal{V}$ be a valuation such that

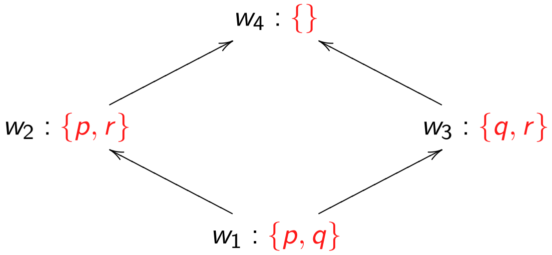
$$\mathcal{V}(p) = \{w_1, w_2\} \qquad \mathcal{V}(q) = \{w_1, w_3\} \qquad \mathcal{V}(r) = \{w_2, w_3\}$$

and $\mathcal{V}(u) = \emptyset$ for any u other than p, q, r.

$\mathcal{F} = \langle \mathcal{W}, \mathcal{R} \rangle$



$\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle$



Kripke Semantics: Satisfiability

Let $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle$. The relation $\mathcal{M}, w \models G$ means: G is *true* (or *satisfied*) in $\mathcal{M}$ at world w. We write $\mathcal{M}, w \not\models G$ to mean that $\mathcal{M}, w \models G$ does not hold.

The relation $\models$ is defined more precisely as follows:

- $\mathcal{M}, w \models p$ if and only if $w \in \mathcal{V}(p)$, where p is a propositional variable.
- $\mathcal{M}, w \models \bot$ never.
- $\mathcal{M}, w \models \top$ always.
- $\mathcal{M}, w \models \neg G$ if and only if $\mathcal{M}, w \not\models G$.
- $\mathcal{M}, w \models F \wedge G$ if and only if $\mathcal{M}, w \models F$ and $\mathcal{M}, w \models G$.
- $\mathcal{M}, w \models F \vee G$ if and only if $\mathcal{M}, w \models F$ or $\mathcal{M}, w \models G$.
- $\mathcal{M}, w \models F \rightarrow G$ if and only if $\mathcal{M}, w \not\models F$ or $\mathcal{M}, w \models G$.

Kripke Semantics: Satisfiability

- Satisfiability of modal operators:
  - $\mathcal{M}, w \models \Box G$ if and only if for all $v \in \mathcal{W}$, if $w\mathcal{R}v$ then $\mathcal{M}, v \models G$. (G is true in all successor worlds).
  - $\mathcal{M}, w \models \Diamond G$ if and only if there exists $v \in \mathcal{W}$ such that $w\mathcal{R}v$ and $\mathcal{M}, v \models G$.
- A formula G is universally true in a model $\mathcal{M}$ if it is satisfied in all worlds in $\mathcal{M}$:

$$\text{for all } w \in \mathcal{W}, \qquad \mathcal{M}, w \models G.$$

- A formula G is satisfiable in a model $\mathcal{M}$ if it is true in $\mathcal{M}$ in some world:

$$\text{there exists } w \in \mathcal{W}, \qquad \mathcal{M}, w \models G.$$

Example

Consider the model $\mathcal{F} = \langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle$ where

- $\mathcal{W} = \{w_1, w_2, w_3, w_4, w_5\}$
- $\mathcal{R} = \{(w_i, w_j) \mid j = i + 1\}$
- $V(p) = \{w_2, w_3\}$, $\mathcal{V}(q) = \{w_1, w_2, w_3, w_4, w_5\}$ and $V(r) = \{\}$.

Then

- $\mathcal{M}, w_1 \models \Diamond\Box p$.
- $\mathcal{M}, w_1 \not\models (\Diamond\Box p) \rightarrow p$
- $\mathcal{M}, w_2 \models \Diamond(p \wedge \neg r)$
- $\mathcal{M}, w_1 \models q \wedge \Diamond(q \wedge \Diamond(q \wedge \Diamond(q \wedge \Diamond q)))$.

Validity in a frame

- A formula G is valid in a frame $\mathcal{F} = \langle \mathcal{W}, \mathcal{R} \rangle$ if for every valuation $\mathcal{V}$ and for every world $w \in \mathcal{W}$

$$\langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle, w \models G.$$

  We write $\mathcal{F} \models G$ to mean "G is valid in $\mathcal{F}$".
- Let $\mathbf{F}$ be a *class of frames*. A formula G is valid in the class of frames $\mathbf{F}$, written $\models_{\mathbf{F}} G$, if G is valid in every frame in $\mathbf{F}$.
- A formula G is valid, written $\models G$, if it is valid in the class of all frames.
- Obviously, all propositional tautologies are valid.

Example

Let $\mathcal{F} = \langle \mathcal{W}, \mathcal{R} \rangle$ be the frame

$$
\begin{tikzcd}
& w_4 & \\
w_2 \arrow[ur] & & w_3 \arrow[ul] \\
& w_1 \arrow[ul] \arrow[ur] \arrow[uu] &
\end{tikzcd}
$$

Then the formula $\Box p \rightarrow \Box\Box p$ is valid in $\mathcal{F}$.

Example

Let $\mathcal{F} = \langle \mathcal{W}, \mathcal{R} \rangle$ be a frame where $\mathcal{W}$ is the set of natural numbers and $\mathcal{R}$ is the less-than relation on $\mathcal{W}$.
Then the formula $\Box\Diamond p \to \Diamond\Box p$ is not valid in $\mathcal{F}$.
Proof: Construct a countermodel by choosing a valuation $\mathcal{V}$ such that $\mathcal{V}(p)$ is the set of even numbers. Then show that there exists an $n \in \mathcal{W}$ such that

$$\langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle, n \models \Box\Diamond p \qquad \text{and} \qquad \langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle, n \not\models \Diamond\Box p.$$

This is the case for any natural number n.

Example

- Let **F** be the class of all reflexive frames, i.e., every frame $\mathcal{F} = \langle \mathcal{W}, \mathcal{R} \rangle$ satisifies
$$\forall u \in \mathcal{W}. u\mathcal{R}u.$$
Then the formula $p \to \Diamond p$ is valid in **F**.
- Let **F** be the class of all transitive frames, i.e., every frame $\mathcal{F} = \langle \mathcal{W}, \mathcal{R} \rangle$ satisfies: for every $u, v, w \in \mathcal{W}$, if $u\mathcal{R}v$ and $v\mathcal{R}w$ then $u\mathcal{R}w$. Then the formula $\Box p \to \Box\Box p$ is valid in **F**.
- The formula
$$\Box(p \to q) \to (\Box p \to \Box q)$$
is valid in all frames.

Axiomatic definition of modal logic

Another way of defining a logic is to define the set of formulae which are theorems of the logic.

Modal logic **K**

1. All instances of propositional tautologies are theorems of **K**.
2. All instances of the modal axioms

$$\begin{array}{rl} \text{(K)} & \Box(p \to q) \to (\Box p \to \Box q) \\ \text{(Dual)} & \Diamond p \equiv \neg\Box\neg p. \end{array}$$

   are theorems of **K**.
3. Modus ponens: If F is a **K**-theorem and $F \to G$ is a **K**-theorem, then G is a **K**-theorem.
4. Substitution: If G is a **K**-theorem and H is obtained from G by uniformly replacing propositional variables in G with arbitrary formulae, then H is a **K**-theorem.
5. Generalization: If G is a **K**-theorem then $\Box G$ is a **K**-theorem.

Soundness and completeness of modal logic **K**

The modal logic **K** is *sound* and *complete* with respect to the Kripke semantics.

Theorem

Soundness. Every theorem of **K** *is valid.*

Theorem

Completeness. Every valid formula is a theorem of **K**.

Normal modal logics

- Normal modal logics are a family of logics that extend the modal logic **K**.
- A normal modal logic is defined by extending **K** with a set of *modal axioms*.
- Let **L** be a set of modal axioms. The set of theorems of the normal logic defined by **L** is defined as in **K**, but with the additional condition:
  - All instances of the axioms in **L** are theorems of the modal logic **L**.

Some modal axioms

Several well-known modal axioms:

(T) $\Box p \rightarrow p$ (or equivalently, $p \rightarrow \Diamond p$).

(B) $p \rightarrow \Box \Diamond p$.

(4) $\Box p \rightarrow \Box \Box p$.

(5) $\Diamond \Box p \rightarrow \Box p$.

(D) $\Box p \rightarrow \Diamond p$.

Naming convention: A normal modal logic that is obtained by extending **K** with a set of axioms S is named by listing the axioms next to **K**.
For example, **K**$\mathrm{T}4$ is the logic extending **K** with axioms T and 4.

Modal axioms and conditions on frames

- Modal axioms have a close correspondence with the "shape" of the frames that validate the axioms.
- For example, if the axiom T is valid in a frame $\mathcal{F} = \langle \mathcal{W}, \mathcal{R} \rangle$, then $\mathcal{F}$ must be reflexive, i.e., for all $u \in \mathcal{W}$, we have $u\mathcal{R}u$.
- Conversely, if a frame $\mathcal{F}$ is reflexive, then T must be valid in $\mathcal{F}$.

Shapes of frames

A frame $\mathcal{F} = \langle \mathcal{W}, \mathcal{R} \rangle$ is

- reflexive if for every $w \in \mathcal{W}$, $w\mathcal{R}w$.
- symmetric if for every $u, v \in \mathcal{W}$, $u\mathcal{R}v$ implies $v\mathcal{R}u$.
- transitive if for every $u, v, w \in \mathcal{W}$, if $u\mathcal{R}v$ and $v\mathcal{R}w$, then $u\mathcal{R}w$.
- euclidean if for every $u, v, w \in \mathcal{W}$, if $u\mathcal{R}v$ and $u\mathcal{R}w$ then $v\mathcal{R}w$.
- serial if for every $w \in \mathcal{W}$, there exists $v \in \mathcal{W}$ such that $w\mathcal{R}v$.

Modal axioms and shapes of frames

Name	Axiom	Frame shape
T	$\Box p \to p$	Reflexive
B	$p \to \Box\Diamond p$	Symmetric
4	$\Box p \to \Box\Box p$	Transitive
5	$\Diamond\Box p \to \Box p$	Euclidean
D	$\Box p \to \Diamond p$	Serial

Theorem

Let $\mathcal{F} = \langle \mathcal{W}, \mathcal{R} \rangle$. Then the axiom T (resp. B, 4, 5, D) is valid in $\mathcal{F}$ if and only $\mathcal{F}$ is reflexive (resp. symmetric, transitive, euclidean, serial).

Proof: ...

Some normal modal logics

- **K**T
- **K**D: also known as deontic logic; logic about "obligations"
- **K**$\mathrm{T4}$: also known as **S**4.
- **K**$\mathrm{T5}$: also known as **S**5.

Note 1: **S**5 can be equivalently defined as **S**4 plus the axiom B.
Note 2: This means that axiom 4 is also a theorem of **S**5.

Multi-modal logic

- Multi-modal logic generalizes modal logic by allowing a familiy of modal operators.
- Each modal operator in the family can be used to describe different modes of truth according to an agent.
- Let $\mathcal{A}$ be a set of *agents*. To each agent $a \in \mathcal{A}$ we introduce two modal operators $\Box_a$ and $\Diamond_a$.
- The language of multi-modal logic:

$$F ::= p \mid \top \mid \bot \mid \neg F \mid F \wedge F \mid F \vee F \mid F \rightarrow F \mid \Box_a F \mid \Diamond_a F.$$

where $a \in \mathcal{A}$.

Semantics of multimodal logic: frames and models

- The notions of frames and models are similar to modal logic.
- Let $\mathcal{A} = \{a_1, \ldots, a_n\}$ be a set of agents. Then an frame is the tuple

$$\mathcal{F} = \langle \mathcal{W}, \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$$

  where $\mathcal{W}$ is a set of worlds, and each $\mathcal{R}_i$ is a binary relation on $\mathcal{W}$.
- Valuations are defined as previously. A model is a pair $\mathcal{M} = \langle \mathcal{F}, \mathcal{V} \rangle$ of frame and valuation.

Semantics of multi-modal logic: satisfiability

Let $\mathcal{A} = \{a_1, \ldots, a_n\}$. Let $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}_1, \ldots, \mathcal{R}_n, \mathcal{V} \rangle$ be a model. The notion of a formula G being true in $\mathcal{M}$ at world w is defined analogously to the single agent case, with the following modification:

- $\mathcal{M}, w \models \Box_{a_i} G$ if and only if for all $v \in \mathcal{W}$, if $w\mathcal{R}_i v$ then $\mathcal{M}, v \models G$.
- $\mathcal{M}, w \models \Diamond_{a_i} G$ if and only if there exists $v \in \mathcal{W}$ such that $w\mathcal{R}_i v$ and $\mathcal{M}, v \models G$.

Semantics of multi-modal logic: validity

- Validity (in a frame) is defined similarly to the single agent case (left as exercise).
- The notion of a "shape" of a frame must now be defined with respect to all the relations $\mathcal{R}_i$.
- For example, a reflexive frame $\mathcal{F} = \langle \mathcal{W}, \mathcal{R}_1, \ldots, \mathcal{R}_n, \rangle$ satisfies: for every $i \in \{1, \ldots, n\}$, and for every $u \in \mathcal{W}$, we have $u\mathcal{R}_i u$.

Correspondence between modal axioms and shapes of frames

Let $\mathcal{A} = \{a_1, \ldots, a_i\}$. The axioms are stated for each modal operator for each agent. For every $a \in \mathcal{A}$:

(T) $\Box_a p \rightarrow p$.

(B) $p \rightarrow \Box \Diamond_a p$.

(4) $\Box_a p \rightarrow \Box_a \Box_a p$.

(5) $\Diamond_a \Box_a p \rightarrow \Box_a p$.

(D) $\Box_a p \rightarrow \Diamond_a p$.

The correspondence between the axioms and the shapes of frames also holds for multi-modal logic.

Epistemic logic

- Epistemic logic is a term used to describe a family of logics about 'knowledge' and 'belief'.
- The epistemic reading of multi-modal formulae:

$$\Box_a G \qquad \text{agent } a \text{ 'knows' } G.$$

- There are several axiomatizations of epistemic logic. Two of the better known ones are **S**4 and **S**5.

Epistemic logic **S**5

Recall that **S**5 is an extension of (multi-modal) **K** with

(T) $\Box_a p \rightarrow p$, for every agent a.

(5) $\Diamond_a \Box_a p \rightarrow \Box_a p$, for every agent a.

Recall also that Axiom 4 is a theorem in **S**5

(4) $\Box_a p \rightarrow \Box_a \Box_a p$.

Epistemic reading of the axioms 4 and 5:

- Axiom 4: If agent a knows p then it knows that it knows p.
- Axiom 5: consider the contrapositive form of 5:

$$\neg \Box_a p \rightarrow \Box_a \neg \Box_a p.$$

If agent a does not know p, then it knows that it does not know p.

The Unknown

> *As we know, there are known knowns. There are things we know we know.*
> *We also know there are known unknowns. That is to say we know there are some things we do not know.*
> *But there are also unknown unknowns, the ones we don't know we don't know.*
> (Donald Rumsfeld, Former US Secretary of Defense)

Exercise: Show why Rumsfeld is contradicting **S**5.

Some references

- M. Huth and M. Ryan. *Logic in Computer Science*, Cambridge University Press, 2nd Edition, 2004.
- M. Fitting. *First-order Logic and Automated Theorem Proving*, Springer-Verlag, 1990.
- P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.