The Whole Framework

Satisfaction relation

$M \models \phi$ "$\models$" is read as "satisfies"

> M is some sort of situation or model of a system, and ϕ is a specifi- cation, a formula of that logic, expressing what should be true in situation M.
> -LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.13

The problem here is how to

syntax, semantics
a) formally express the specification "ϕ"

inference
b) do the computation "$\models$"

The point here is "logic" contains 3 parts:

1. syntax
2. semantics
3. inference

> The aim of logic in computer science is to develop languages to model the situations we encounter as computer science professionals, in such a way that we can reason about them formally.
> -LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.17

> we developed propositional logic by examining it from three different angles: its proof theory (the natural deduction calculus), its syntax (the tree-like nature of formulas) and its semantics (what these formulas actually mean).
> -LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.109

Propositional Logic

"proposition" is the synonym of "declarative sentences" which are in principle capable of being declared true or false.

Declarative sentence (命題)

- atomic proposition
- composition proposition
  - a) $\neg$ negation
  - b) $\vee$ disjunction
  - c) $\wedge$ conjuction
  - d) $\longrightarrow$ implication

This is about how to encode specifications

These are operators to produces propositions which are terms of the target language

Natural Deduction (Inference/Proof Rules)

Syntax :

premises conclusion

sequent $\phi_1, \phi_2, \cdots, \phi_n \vdash \psi$

A sequent is VALID if $\exists$ a proof (under some model/logic)

> The important thing to realise, though, is that any putative proof can be checked for correctness by checking each individual line, starting at the top, for the valid application of its proof rule.
> -LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.23

In HC ϕ_i and ψ are types/objects, "$\vdash$" are arrows

1 Conjunction Rules

- "And-introduction" rule ($\wedge i$)

$$\frac{\phi \quad \psi}{\phi \wedge \psi}\ \wedge i$$

this is just the name of the rule

the computation of the rule is just pattern matching.

- "And-elimination" rules ($\wedge e_1$) ($\wedge e_2$)

$$\frac{\phi \wedge \psi}{\phi}\ \wedge e_1 \qquad \frac{\phi \wedge \psi}{\psi}\ \wedge e_2$$

2 Negation Rule

- "double negation" rule ($\neg\neg e$.) ($\neg\neg i$)

$$\frac{\neg\neg\phi}{\phi}\ \neg\neg e \qquad \frac{\psi}{\neg\neg\psi}\ \neg\neg i$$

3 Implication Rule

("$\rightarrow$" constructs a function type)

- "eliminating implication"

$$\frac{\phi \qquad \phi \rightarrow \psi}{\psi}\ \rightarrow e$$

modus tollens

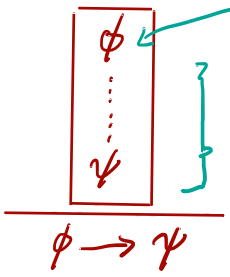
$$\frac{\phi \to \psi \quad , \neg\psi}{\neg\phi}\ MT$$

* This is not true in Constructive Logic

MT, $\neg\neg e \vdash$ 反证法 : "$\neg p \to \neg q, q \vdash p$"

* • Implication Introduction

This one is non-trivial, since "$\to$" is about Causality which have a link with "$\vdash$"



temporary assumes ϕ

$\phi \vdash \psi$: a proof of ψ given ϕ.

The box is the namespace where ϕ resides

> It says: in order to prove $\phi \to \psi$, make a temporary assumption of ϕ and then prove ψ.
> -LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.28

> There are rules about which formulas can be used at which points in the proof. Generally, we can only use a formula ϕ in a proof at a given point if that formula occurs prior to that point and if no box which encloses that occurrence of ϕ has been closed already.
> -LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.28

> **Remark 1.12** Indeed, this example indicates that we may transform any proof of $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ in such a way into a proof of the theorem
>
> $$\vdash \phi_1 \to (\phi_2 \to (\phi_3 \to (\cdots \to (\phi_n \to \psi) \ldots)))$$
>
> by 'augmenting' the previous proof with n lines of the rule $\to$i applied to $\phi_n, \phi_{n-1}, \ldots, \phi_1$ in that order.

nontrivial

Curying This is exactly $(a, b) \to c \cong a \to b \to c$

4. Disjunction Rules

- introduction rule

$$\frac{\phi}{\phi \vee \psi}\ \vee i_1 \qquad \frac{\psi}{\phi \vee \psi}\ \vee i_2$$

- elimination rule

$$\frac{\phi \vee \psi \qquad \begin{array}{c}[\phi]\\ \vdots \\ \chi\end{array} \qquad \begin{array}{c}[\psi]\\ \vdots \\ \chi\end{array}}{\chi}\ \vee e.$$

$$\frac{\phi \vee \psi,\ \phi \to \chi,\ \psi \to \chi}{\chi}\ \vee e$$

5. Principle of Explosion

Ex falso quodlibet

Coq has the "exfalso" tactic

$\phi \wedge \neg\phi \vdash \psi$

This is denoted as the bottom "$\bot$" or "False", which is also the init object in Category Theory.

- bottom elimination rule

$$\frac{\bot}{\phi}\ \bot\text{e.}$$

This is constructive since $\bot$ is the initial object in Hask Category

- not-elimination

$$\frac{\phi \quad \neg\phi}{\bot}\ \neg\text{e.}$$

- proof rule $\neg$i

$$\frac{\begin{array}{|c|}\hline \phi \\ \vdots \\ \bot \\ \hline \end{array}}{\neg\phi}\ \neg\text{i.}$$

6 Summary

The basic rules of natural deduction:

	introduction	*elimination*
$\wedge$	$\dfrac{\phi \quad \psi}{\phi \wedge \psi}\,\wedge\text{i}$	$\dfrac{\phi \wedge \psi}{\phi}\,\wedge\text{e}_1$ $\quad$ $\dfrac{\phi \wedge \psi}{\psi}\,\wedge\text{e}_2$
$\vee$	$\dfrac{\phi}{\phi \vee \psi}\,\vee\text{i}_1$ $\quad$ $\dfrac{\psi}{\phi \vee \psi}\,\vee\text{i}_2$	$\dfrac{\phi \vee \psi \quad \begin{array}{c}[\phi]\\ \vdots\\ \chi\end{array} \quad \begin{array}{c}[\psi]\\ \vdots\\ \chi\end{array}}{\chi}\,\vee\text{e}$
$\rightarrow$	$\dfrac{\begin{array}{c}[\phi]\\ \vdots\\ \psi\end{array}}{\phi \rightarrow \psi}\,\rightarrow\text{i}$	$\dfrac{\phi \quad \phi \rightarrow \psi}{\psi}\,\rightarrow\text{e}$
$\neg$	$\dfrac{\begin{array}{c}[\phi]\\ \vdots\\ \bot\end{array}}{\neg\phi}\,\neg\text{i}$	$\dfrac{\phi \quad \neg\phi}{\bot}\,\neg\text{e}$
$\bot$	(no introduction rule for $\bot$)	$\dfrac{\bot}{\phi}\,\bot\text{e}$
$\neg\neg$		$\dfrac{\neg\neg\phi}{\phi}\,\neg\neg\text{e}$

Some useful derived rules:

$$\frac{\phi \rightarrow \psi \quad \neg\psi}{\neg\phi}\,\text{MT} \qquad \frac{\phi}{\neg\neg\phi}\,\neg\neg\text{i}$$

$$\frac{\begin{array}{c}[\neg\phi]\\ \vdots\\ \bot\end{array}}{\phi}\,\text{PBC} \qquad \frac{}{\phi \vee \neg\phi}\,\text{LEM}$$

Intuitionistic logicians argue that, to show $\phi \vee \neg\phi$, you have to show ϕ, or $\neg\phi$. If neither of these can be shown, then the putative truth of the disjunction has no justification. Intuitionists reject $\neg\neg$e since we have already used this rule to prove LEM and PBC from rules which the intuitionists do accept.
-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.46

the intuitionists favour a calculus containing the introduction and elimination rules shown in Figure 1.2 and excluding the rule $\neg\neg$e and the derived rules.
-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.47

Intuitionistic logic turns out to have some specialised applica- tions in computer science, such as modelling type-inference systems used in compilers or the staged execution of program code; but in this text we stick to the full so-called classical logic which includes all the rules.
-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.47

Def Propositional logic Formula

```
data Form a = Atom a
            | ¬ (Form a)
            | ∧ (Form a) (Form a)
            | ∨ (Form a) (Form a)
            | → (Form a) (Form a)
```

> $\phi ::= p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi)$
> *-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.49*

> Such reasoning is greatly facilitated by the fact that the grammar in (1.3) satisfies the inversion principle, which means that we can invert the process of building formulas: although the grammar rules allow for five different ways of constructing more complex formulas – the five clauses in (1.3) – there is always a unique clause which was used last.
> *-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.49*

The definition of the formula and inference rules together gives the syntax part of propositional logic

The Semantics of Propositional Logic

Def Valuation or Model of a Formula

Given a formula ϕ, a Valuation or Model of ϕ is an assignment of each propositional atom to a truth value (Boolean)

Semantics of Logic Connectivities e.g. $\vee$, $\wedge$, $\neg$, $\rightarrow$, are defined through truth tables.

Theorem The semantics of

$$\phi \rightarrow \psi \overset{s}{\Longleftrightarrow} \neg\phi \vee \psi$$

"$\rightarrow$" preserves truth.

Def Soundness

Soundness is a property of a logic's sytax is consistant with its semantics.

> In that case, we said that the sequent ϕ1, ϕ2,...,ϕn ψ is valid. Do we have any evidence that these rules are all correct in the sense that valid sequents all 'preserve truth' computed by our truth-table semantics?
> -*LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.61*

Def Semantic Entailment

If $\forall$ valuation/model in which all $\phi_1, \ldots, \phi_n$ evaluates to True

i.e.

$M \models \phi_1, \ldots, \phi_n$

we have ψ also evaluates to True

i.e.

$M \models \psi$

we say that

$\phi_1, \phi_2, \ldots, \phi_n \models \psi$

holds and call $\models$ the semantic entailment.

Def Soundness

A logic is sound if

$\forall\ \phi_1, \phi_2 \ldots \phi_n \vdash \psi$ is valid

we have

$\phi_1, \phi_2, \ldots, \phi_n \models \psi$.

Theorem 1.35 (Soundness) *Let $\phi_1, \phi_2, \ldots, \phi_n$ and ψ be propositional logic formulas. If $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ is valid, then $\phi_1, \phi_2, \ldots, \phi_n \models \psi$ holds.*

Soundness speaks all proofs are sound.

Def Completeness

Soundness tells us:

$\vdash \Rightarrow \models$

while completeness is the backward direction:

$\models \Rightarrow \vdash$

Completeness speaks all truth has a proof.

Def (tautology)

A formula of propositional logic ϕ is called a tautology if ϕ evaluates to True i.e.

$\models \phi$.

Definition 1.36 A formula of propositional logic ϕ is called a *tautology* iff it evaluates to T under all its valuations, i.e. iff $\vDash \phi$.

Theorem 1.37 *If $\vDash \eta$ holds, then $\vdash \eta$ is valid. In other words, if η is a tautology, then η is a theorem.*

Definition 1.44 Given a formula ϕ in propositional logic, we say that ϕ is *satisfiable* if it has a valuation in which is evaluates to T.

Proposition 1.45 *Let ϕ be a formula of propositional logic. Then ϕ is satisfiable iff $\neg\phi$ is not valid.*

We reduce the problem of determining whether any ϕ is valid to the problem of computing an equivalent ψ ≡ ϕ such that ψ is in CNF and checking, via Lemma 1.43, whether ψ is valid.
-LOGIC IN COMPUTER SCIE...soning about Systems, p.74

Definition 1.42 A *literal* L is either an atom p or the negation of an atom $\neg p$. A formula C is in *conjunctive normal form* (CNF) if it is a conjunction of clauses, where each clause D is a disjunction of literals:

$$\begin{aligned} L &::= p \mid \neg p \\ D &::= L \mid L \vee D \\ C &::= D \mid D \wedge C. \end{aligned} \tag{1.6}$$

CNF

To decide the satisfiability, by Prop 1.45, need to check the validity of $\neg\phi$ where ϕ is in CNF, here the Lemma goes in

Lemma 1.43 *A disjunction of literals $L_1 \vee L_2 \vee \cdots \vee L_m$ is valid iff there are $1 \leq i, j \leq m$ such that L_i is $\neg L_j$.*

Horn clause

Fortunately, there are practically important subclasses of formulas which have much more efficient ways of deciding their satisfiability.
-LOGIC IN COMPUTER SCIENCE:...d Reasoning about Systems, p.81

Horn formulas are conjunctions of Horn clauses. A Horn clause is an impli- cation whose assumption A is a conjunction of propositions of type P and whose conclusion is also of type P.
-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.82

Definition 1.46 A *Horn formula* is a formula ϕ of propositional logic if it can be generated as an instance of H in this grammar:

$$\begin{aligned} P &::= \bot \mid \top \mid p \\ A &::= P \mid P \wedge A \\ C &::= A \to P \\ H &::= C \mid C \wedge H. \end{aligned} \tag{1.7}$$

We call each instance of C a *Horn clause*.

"⊥" bottom unsatifiable
"⊤" top tautology
"P" propositional atom

SAT (satisfiability) Solver

Horn Marking Algorithm

1. It marks $\top$ if it occurs in that list.
2. If there is a conjunct $P_1 \wedge P_2 \wedge \cdots \wedge P_{k_i} \to P'$ of ϕ such that all P_j with $1 \leq j \leq k_i$ are marked, mark P' as well and go to 2. Otherwise (= there is no conjunct $P_1 \wedge P_2 \wedge \cdots \wedge P_{k_i} \to P'$ such that all P_j are marked) go to 3.
3. If $\bot$ is marked, print out 'The Horn formula ϕ is unsatisfiable.' and stop. Otherwise, go to 4.
4. Print out 'The Horn formula ϕ is satisfiable.' and stop.

The marking algorithm for Horn formulas computes marks as constraints on all valuations that can make a formule true. By (1.8), all marked atoms have to be true for any such valuation.
-LOGIC IN COMPUTER SCIENCE: Mo...and Reasoning about Systems, p.84

Predicate Logic (Fist Order Logic)
谓词罗辑

> Propositional logic dealt quite satisfactorily with sentence components like not, and, or and if ... then, but the logical aspects of natural and artificial languages are much richer than that. What can we do with modifiers like there exists ..., all ..., among . . . and only ... ?
> *-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.109*

A predicate difines a Relation on 2^{Term}

> Variables are written u, v, w, x, y, z,... or x1, y3, u5,... and can be thought of as place holders for concrete values (like a student, or a program state).
> *-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.110*

> This is where we need to introduce quantifiers ∀ (read: 'for all') and ∃ (read: 'there exists' or 'for some') which always come attached to a variable, as in ∀x ('for all x') or in ∃z ('there exists z', or 'there is some z').
> *-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.110*

Predicate logic 支持三段论的表达

Two sorts of things involved in predicate logic

1) Term

   denotes the objects we are talking about, include

   - Individules : a, b, x, y
   - Function symbols : m(a), m(x)

2) Formula

   denote truth values (facts) including

   - predicate with arguments: Y(x, m(x)).

> A predicate vocabulary consists of three sets: a set of predicate symbols P, a set of function symbols F and a set of constant symbols C. Each predicate symbol and each function symbol comes with an arity, the number of arguments it expects.
> *-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.115*

Def Term

Definition 2.1 Terms are defined as follows.

- Any variable is a term.
- If $c \in \mathcal{F}$ is a nullary function, then c is a term.
- If $t_1, t_2, \ldots, t_n$ are terms and $f \in \mathcal{F}$ has arity $n > 0$, then $f(t_1, t_2, \ldots, t_n)$ is a term.
- Nothing else is a term.

In Backus Naur form we may write

$$t ::= x \mid c \mid f(t, \ldots, t)$$

where x ranges over a set of variables var, c over nullary function symbols in $\mathcal{F}$, and f over those elements of $\mathcal{F}$ with arity $n > 0$.

- the first building blocks of terms are *constants* (nullary functions) and *variables*;
- more complex terms are built from function symbols using as many previously built terms as required by such function symbols; and
- the notion of terms is dependent on the set $\mathcal{F}$. If you change it, you change the set of terms.

Def Formula

Definition 2.3 We define the set of formulas over $(\mathcal{F}, \mathcal{P})$ inductively, using the already defined set of terms over $\mathcal{F}$:

- If $P \in \mathcal{P}$ is a predicate symbol of arity $n \geq 1$, and if $t_1, t_2, \ldots, t_n$ are terms over $\mathcal{F}$, then $P(t_1, t_2, \ldots, t_n)$ is a formula.
- If ϕ is a formula, then so is $(\neg\phi)$.
- If ϕ and ψ are formulas, then so are $(\phi \wedge \psi)$, $(\phi \vee \psi)$ and $(\phi \to \psi)$.
- If ϕ is a formula and x is a variable, then $(\forall x\, \phi)$ and $(\exists x\, \phi)$ are formulas.
- Nothing else is a formula.

Note how the arguments given to predicates are always terms. This can also be seen in the Backus Naur form (BNF) for predicate logic:

$$\phi ::= P(t_1, t_2, \ldots, t_n) \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \to \phi) \mid (\forall x\, \phi) \mid (\exists x\, \phi) \tag{2.2}$$

where $P \in \mathcal{P}$ is a predicate symbol of arity $n \geq 1$, t_i are terms over $\mathcal{F}$ and x is a variable. Recall that each occurrence of ϕ on the right-hand side of the ::= stands for any formula already constructed by these rules. (What role could predicate symbols of arity 0 play?)

Def Free and bound variables

Definition 2.6 Let ϕ be a formula in predicate logic. An occurrence of x in ϕ is free in ϕ if it is a leaf node in the parse tree of ϕ such that there is no path upwards from that node x to a node $\forall x$ or $\exists x$. Otherwise, that occurrence of x is called bound. For $\forall x\,\phi$, or $\exists x\,\phi$, we say that ϕ – minus any of ϕ's subformulas $\exists x\,\psi$, or $\forall x\,\psi$ – is the scope of $\forall x$, respectively $\exists x$.

Def Substitution

Definition 2.7 Given a variable x, a term t and a formula ϕ we define $\phi[t/x]$ to be the formula obtained by replacing each free occurrence of variable x in ϕ with t.

You can see that the process of substitution is straightforward, but requires that it be applied only to the free occurrences of the variable to be substituted.
-LOGIC IN COMPUTER SCIE...soning about Systems, p.121

Proof Rules

Proof rules for Equality

Here equality does not mean syntactic, or intensional, equality, but equality in terms of computation results.
-LOGIC IN COMPUTER SC...ning about Systems, p.123

$$\frac{}{t = t}\ {=}\mathrm{i}$$

$$\frac{t_1 = t_2 \quad \phi[t_1/x]}{\phi[t_2/x]}\ {=}\mathrm{e}.$$

- Reflexive

Substitution principle

- Symetric

$t_1 = t_2 \vdash t_2 = t_1$

- Transitive

$t_1 = t_2, t_2 = t_3 \vdash t_1 = t_3$

Proof Rules for "∀"

$$\frac{\forall x\, \phi}{\phi[t/x]}\ \forall x\,\mathrm{e}.$$

$$\frac{\begin{array}{|l} x_0 \\ \vdots \\ \phi[x_0/x] \end{array}}{\forall x\, \phi}\ \forall x\,\mathrm{i}.$$

the box is to stipulate the scope of the 'dummy variable' x0 rather than the scope of an assumption.
-LOGIC IN CO...ystems, p.126

It says: If, starting with a 'fresh' variable x0, you are able to prove some formula ϕ[x0/x] with x0 in it, then (because x0 is fresh) you can derive ∀x ϕ.
-LOGIC IN CO...ystems, p.126

Proof rules for "∃"

$$\frac{\phi[t/x]}{\exists x \phi}\ \exists x\,\mathrm{i}.$$

$$\frac{\exists x\,\phi \qquad \boxed{\begin{array}{c} x_0\ \phi[x_0/x] \\ \vdots \\ \chi \end{array}}}{\chi}\ \exists \mathrm{e}.$$

The reasoning goes: We know ∃x ϕ is true, so ϕ is true for at least one 'value' of x. So we do a case analysis over all those possible values, writing x0 as a generic value representing them all. If assuming ϕ[x0/x] allows us to prove some χ which doesn't mention x0, then this χ must be true whichever x0 makes ϕ[x0/x] true. And that's precisely what the rule ∃e allows us to deduce.
-LOGIC IN COMPUTER S...ning about Systems, p.129

Quantifier Equivalencies

Predicate Logic Semantics

By 'separate,' we mean that the meaning of the connectives is defined in a different way; in proof theory, they were defined by proof rules providing an oper- ative explanation. In semantics, we expect something like truth tables. By 'equivalent,' we mean that we should be able to prove soundness and completeness, as we did for propositional logic – although a fully fledged proof of soundness and completeness for predicate logic is beyond the scope of this book.
-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.138

In proof theory, the basic object which is constructed is a proof. Let us write Γ as a shorthand for lists of formulas ϕ1, ϕ2,...,ϕn. Thus, to show that Γ ψ is valid, we need to provide a proof of ψ from Γ. Yet, how can we show that ψ is not a consequence of Γ? Intuitively, this is harder; how can you possibly show that there is no proof of something? You would have to consider every 'candidate' proof and show it is not one. Thus, proof theory gives a 'positive' characterisation of the logic; it pro- vides convincing evidence for assertions like 'Γ ψ is valid,' but it is not very useful for establishing evidence for assertions of the form 'Γ ϕ is not valid.'
-LOGIC IN COMPUTER SCIENCE:...nd Reasoning about Systems, p.138

Semantics, on the other hand, works in the opposite way. To show that ψ is not a consequence of Γ is the 'easy' bit: find a model in which all ϕi are true, but ψ isn't.
-LOGIC IN COMPUTER SCIENCE: M...nd Reasoning about Systems, p.139

Evaluating Formula

- Connectivities
  Evaluation of "$\wedge$", "$\vee$", "$\neg$" and "$\rightarrow$" is the same as Propositional Logic.
- Quantifiers : $\forall$, $\exists$

evaluations of formulas require a fixed universe of concrete values, the things we are, so to speak, talking about. Thus, the truth value of a formula in predicate logic depends on, and varies with, the actual choice of values and the meaning of the predicate and function symbols involved.
-LOGIC IN COMPUTER SCIENCE:...nd Reasoning about Systems, p.140

- Predicates : A "Model" is needed for predicate's semantics

we cannot just assign truth values to P directly without knowing the meaning of terms. We require a model of all function and predicate symbols involved. For example, terms could denote real num- bers and P could denote the relation 'less than or equal to' on the set of real numbers.
-LOGIC IN COMPUTER SCIENCE: M...d Reasoning about Systems, p.140

Definition 2.14 Let $\mathcal{F}$ be a set of function symbols and $\mathcal{P}$ a set of predicate symbols, each symbol with a fixed number of required arguments. A model $\mathcal{M}$ of the pair $(\mathcal{F}, \mathcal{P})$ consists of the following set of data:

1. A non-empty set A, the universe of concrete values;
2. for each nullary function symbol $f \in \mathcal{F}$, a concrete element $f^{\mathcal{M}}$ of A
3. for each $f \in \mathcal{F}$ with arity $n > 0$, a concrete function $f^{\mathcal{M}} \colon A^n \to A$ from A^n, the set of n-tuples over A, to A; and
4. for each $P \in \mathcal{P}$ with arity $n > 0$, a subset $P^{\mathcal{M}} \subseteq A^n$ of n-tuples over A.

The distinction between f and $f^{\mathcal{M}}$ and between P and $P^{\mathcal{M}}$ is most important. The symbols f and P are just that: symbols, whereas $f^{\mathcal{M}}$ and $P^{\mathcal{M}}$ denote a concrete function (or element) and relation in a model $\mathcal{M}$, respectively.

we are forced to interpret formulas relative to an environment.
-LOGIC IN COMPUTER SCIEN...asoning about Systems, p.143

Definition 2.17 A look-up table or environment for a universe A of concrete values is a function $l \colon \mathsf{var} \to A$ from the set of variables var to A. For such an l, we denote by $l[x \mapsto a]$ the look-up table which maps x to a and any other variable y to $l(y)$.

Definition 2.18 Given a model $\mathcal{M}$ for a pair $(\mathcal{F}, \mathcal{P})$ and given an environment l, we define the satisfaction relation $\mathcal{M} \vDash_l \phi$ for each logical formula ϕ over the pair $(\mathcal{F}, \mathcal{P})$ and look-up table l by structural induction on ϕ. If $\mathcal{M} \vDash_l \phi$ holds, we say that ϕ computes to $\mathtt{T}$ in the model $\mathcal{M}$ with respect to the environment l.

P: If ϕ is of the form $P(t_1, t_2, \ldots, t_n)$, then we interpret the terms $t_1, t_2, \ldots, t_n$ in our set A by replacing all variables with their values according to l. In this way we compute concrete values $a_1, a_2, \ldots, a_n$ of A for each of these terms, where we interpret any function symbol $f \in \mathcal{F}$ by $f^{\mathcal{M}}$. Now $\mathcal{M} \vDash_l P(t_1, t_2, \ldots, t_n)$ holds iff $(a_1, a_2, \ldots, a_n)$ is in the set $P^{\mathcal{M}}$.
$\forall x$: The relation $\mathcal{M} \vDash_l \forall x\, \psi$ holds iff $\mathcal{M} \vDash_{l[x \mapsto a]} \psi$ holds for all $a \in A$.
$\exists x$: Dually, $\mathcal{M} \vDash_l \exists x\, \psi$ holds iff $\mathcal{M} \vDash_{l[x \mapsto a]} \psi$ holds for some $a \in A$.
$\neg$: The relation $\mathcal{M} \vDash_l \neg\psi$ holds iff it is not the case that $\mathcal{M} \vDash_l \psi$ holds.
$\vee$: The relation $\mathcal{M} \vDash_l \psi_1 \vee \psi_2$ holds iff $\mathcal{M} \vDash_l \psi_1$ or $\mathcal{M} \vDash_l \psi_2$ holds.
$\wedge$: The relation $\mathcal{M} \vDash_l \psi_1 \wedge \psi_2$ holds iff $\mathcal{M} \vDash_l \psi_1$ and $\mathcal{M} \vDash_l \psi_2$ hold.
$\rightarrow$: The relation $\mathcal{M} \vDash_l \psi_1 \rightarrow \psi_2$ holds iff $\mathcal{M} \vDash_l \psi_2$ holds whenever $\mathcal{M} \vDash_l \psi_1$ holds.

Definition 2.20 Let Γ be a (possibly infinite) set of formulas in predicate logic and ψ a formula of predicate logic.

1. Semantic entailment $\Gamma \vDash \psi$ holds iff for all models $\mathcal{M}$ and look-up tables l, whenever $\mathcal{M} \vDash_l \phi$ holds for all $\phi \in \Gamma$, then $\mathcal{M} \vDash_l \psi$ holds as well.
2. Formula ψ is satisfiable iff there is some model $\mathcal{M}$ and some environment l such that $\mathcal{M} \vDash_l \psi$ holds.
3. Formula ψ is valid iff $\mathcal{M} \vDash_l \psi$ holds for all models $\mathcal{M}$ and environments l in which we can check ψ.
4. The set Γ is consistent or satisfiable iff there is a model $\mathcal{M}$ and a look-up table l such that $\mathcal{M} \vDash_l \phi$ holds for all $\phi \in \Gamma$.

> *Validity in predicate logic.* *Given a logical formula ϕ in predicate logic, does $\models \phi$ hold, yes or no?*

> *The Post correspondence problem.* *Given a finite sequence of pairs $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$ such that all s_i and t_i are binary strings of positive length, is there a sequence of indices $i_1, i_2, \ldots, i_n$ with $n \geq 1$ such that the concatenation of strings $s_{i_1} s_{i_2} \ldots s_{i_n}$ equals $t_{i_1} t_{i_2} \ldots t_{i_n}$?*

same pair can be selected for multiple times.

The proof of undecidability takes the idea of showing if it is decidable then The Post Correspondence Problem is solvable.

> This is an immediate consequence of the definitional clause M I ¬φ for negation. Since we can't compute validity, it follows that we cannot compute satisfiability either.
> *-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.151*

The Problem of Reachability.

> The validation of many applications requires to show that a 'bad' state cannot be reached from a 'good' state.
> *-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.152*

> In its essence, deciding whether from a good state one can reach a bad state is the reachability problem in directed graphs.
> *-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.152*

> **Reachability:** Given nodes n and n' in a directed graph, is there a finite path of transitions from n to n'?

> can we find a predicate-logic formula φ with u and v as its only free variables and R as its only predicate symbol (of arity 2) such that φ holds in directed graphs iff there is a path in that graph from the node associated to u to the node associated to v?
> *-LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.153*

expressing Reachability using only R, i.e., neighbor relation.

Second Order Logic

- Existential Second-order Logic

> This can be realized by applying quantifiers not only to variables, but also to predicate symbols.
> -LOGIC IN COMPUTER SCIENCE: Modelling and Reasoning about Systems, p.155

$$\exists P \phi$$

Semantics

$$\mathcal{M} \models_l \exists P \phi \quad \text{iff} \quad \text{for some } T \subseteq A \times A,\ \mathcal{M}_T \models_l \phi.$$

Model $\mathcal{M}$ contains interpretation of all functions and predicates except P.

$\mathcal{M}_T$ is $\mathcal{M}$ with an interpretation of P added.

$P^{\mathcal{M}_T} = T \subseteq A \times A$

- Universal Second-order Logic

$\forall P \phi$

- Full-fledged Second-order Logic
  = Existential + Universal.

High-order logic

> If one wants to quantify over relations of relations, one gets third-order logic etc. Higher-order logics require great care in their design. Typical re- sults such as completeness and compactness may quickly fail to hold. Even worse, a naive higher-order logic may be inconsistent at the meta-level.
> *-LOGIC IN COMPUTER SCIENCE: M...and Reasoning about Systems, p.157*

罗素 paradox

Modal Logics and Agents

There might be different modes of truth, such as necessarily true, known to be true, currently true or true forever.

Basic Modal Logic

Definition 5.1 The formulas of basic modal logic ϕ are defined by the following Backus Naur form (BNF):

$$\phi ::= \bot \mid \top \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\phi \leftrightarrow \phi) \mid (\Box\phi) \mid (\Diamond\phi) \tag{5.1}$$

where p is any atomic formula.

Definition 5.3 A model $\mathcal{M}$ of basic modal logic is specified by three things:

1. A set W, whose elements are called worlds;
2. A relation R on W ($R \subseteq W \times W$), called the accessibility relation;
3. A function $L : W \rightarrow \mathcal{P}(\texttt{Atoms})$, called the labelling function.

We write $R(x, y)$ to denote that (x, y) is in R.

Intuitively, w ∈ W stands for a possible world and R(w, w) means that w is a world accessible from world w. The actual nature of that relationship depends on what we intend to model.
-LOGIC IN COMPUTER SCIENCE: Mode...g and Reasoning about Systems, p.325

These models are often called Kripke models, in honour of S. Kripke who invented them and worked extensively in modal logic in the 1950s and 1960s.
-LOGIC IN COMPUTER SCIENCE: M...d Reasoning about Systems, p.325

Definition 5.4 Let $\mathcal{M} = (W, R, L)$ be a model of basic modal logic. Suppose $x \in W$ and ϕ is a formula of (5.1). We will define when formula ϕ is true in the world x. This is done via a satisfaction relation $x \Vdash \phi$ by structural induction on ϕ:

$$
\begin{aligned}
x &\Vdash \top \\
x &\nVdash \bot \\
x &\Vdash p \quad \text{iff } p \in L(x) \\
x &\Vdash \neg\phi \quad \text{iff } x \nVdash \phi \\
x &\Vdash \phi \wedge \psi \quad \text{iff } x \Vdash \phi \text{ and } x \Vdash \psi \\
x &\Vdash \phi \vee \psi \quad \text{iff } x \Vdash \phi \text{ , or } x \Vdash \psi \\
x &\Vdash \phi \rightarrow \psi \quad \text{iff } x \Vdash \psi \text{ , whenever we have } x \Vdash \phi \\
x &\Vdash \phi \leftrightarrow \psi \quad \text{iff } (x \Vdash \phi \text{ iff } x \Vdash \psi) \\
x &\Vdash \Box\psi \quad \text{iff, for each } y \in W \text{ with } R(x, y), \text{ we have } y \Vdash \psi \\
x &\Vdash \Diamond\psi \quad \text{iff there is a } y \in W \text{ such that } R(x, y) \text{ and } y \Vdash \psi.
\end{aligned}
$$

When $x \Vdash \phi$ holds, we say 'x satisfies ϕ,' or 'ϕ is true in world x.' We write $\mathcal{M}, x \Vdash \phi$ if we want to stress that $x \Vdash \phi$ holds in the model $\mathcal{M}$.

Definition 5.5 A model $\mathcal{M} = (W, R, L)$ of basic modal logic is said to satisfy a formula if every state in the model satisfies it. Thus, we write $\mathcal{M} \vDash \phi$ iff, for each $x \in W$, $x \Vdash \phi$.