# COSC 410 Lecture 1
# Classical Propositional Logic

Willem Labuschagne
University of Otago

2016

## Introduction

COSC 410 is about the logic you need to know if you want to understand research in artificial intelligence.

The best textbook introduction is still Genesereth and Nilsson (1987): *Logical Foundations of Artificial Intelligence*, but there are no newer editions and so the book doesn't cover the exciting developments after 1987 that form the main focus of COSC410. As a result we mention this textbook as an interesting source of additional reading only.

In the place of a suitable textbook, self–contained lecture notes will be provided on the COSC410 coursework webpage a few days before the scheduled lecture. Treat these notes as your textbook — read them, summarise them, scribble on them.

You should bring some paper and a pen to class in order to take additional notes and do exercises. Exercises are of key importance and you should set aside a minimum of 12 hours per week for unsupervised study, i.e. doing exercises and reading the lecture notes. (In general, budget two 7-hour working days per week for each paper you're doing and another two days per week for your project.)

At the start of lectures 2 – 9 a short quiz will be given, each worth 1%. There are also two written assignments due on 21 March and 16 May, worth 11% each. Submit your assignments either electronically as pdf files emailed to Willem or as hardcopy handed in to Willem. As always, we expect scrupulous honesty – the assignments are meant to be your own work.

For any questions related to lectures 1 through 9, come and see Willem, and for questions about lectures 10 through 13, consult Richard O'Keefe.

## Beady-eye agents

Why does one need to know some logic if one is interested in artificial intelligence?

AI is a very broad field and there are different ways to look at it. One perspective regards AI as the study of artificial agents. Agents differ from rocks: instead of simply enduring passively, agents respond to changes in their environment and take action to produce further changes.

A very low-level example of such an agent is a thermostat, which senses changes in the temperature of its environment and acts by increasing or decreasing the amount of heat provided by a furnace, stove, or heat pump.

If one were content to work with low-level agents, one would not need to know anything about logic. But what if one is interested in designing high-level agents that more closely approximate humans?

The simplest type of high-level agent is the BDI agent, where B stands for "Belief", D for "Desire", and I for "Intention".

Every BDI agent has an internal representation of its environment. This representation uses declarative sentences in some language. Such sentences are the agent's *beliefs*.

Declarative sentences are the sort of sentence you would use to state a fact, for example: "Sicily is an island". A natural language such as English also has other kinds of sentences, e.g. questions and commands. We will not be including non-declarative sentences like questions in our study. Think of the communication between BDI agents as being limited to stating what the agent believes. One can extend this to include speech acts like requesting or commanding, but that makes everything much more complicated, so we'll stick to sentences used for informing other agents of what is believed.

The *desires* of an agent are its long-term goals or motivations (for example, to become an All Black, or to visit Paris, or to learn how to grow prize-winning cucumbers).

The *intentions* of an agent are the shorter-term subgoals that act as stepping stones in reaching the long-term goals.

When planning, an agent typically selects intentions that will carry it closer to realising its desires. Of course, the plan needs to be realistic. This is where the beliefs come in. If the agent has the desire to visit Paris, then a sensible plan might consist of the intention to earn enough money for airfare to Paris. It would not be a good plan to form the intention of walking there, since among the beliefs we would expect the agent to possess would be the belief that there's an ocean in the way. Beliefs record the agent's awareness of the constraints imposed by its environment.

Anyone who wants to work with BDI agents will need to know some logic in order to be able to deal with the way the agent acquires and changes beliefs, and the ways in which an agent may draw inferences from its beliefs.

Sometimes we may use the word "knowledge" instead of talking about "beliefs". If we talk about knowledge then we're assuming that the agent's information about its environment is perfectly accurate. Knowledge is always true. As your own experience will reveal, an agent doesn't often have the guarantee that all the information it acquires is accurate. So knowledge is a much more limited concept than belief. That means it's easier to work with, and, until halfway through the 20th century, logic used to focus almost exclusively on knowledge. But in AI we can't get very far if we insist that our agents must have knowledge. For example, any agent having a sensor for vision will face the possibility of being mistaken about an optical illusion.

Many of the recent developments in logic that we'll be exploring were driven by this transition from working with infallible knowledge (certainty) to working with *defeasible* beliefs (uncertainty). The fact that beliefs may be defeated, i.e. may turn out to be false, causes all sorts of interesting problems for agents (especially us humans). Should a convicted murderer be hanged when we lack certainty because an element of doubt remains? How much certainty do we require before we act in a particular way? How sure must we be that Sally will say yes before we ask her to the dance? As you can see, we are dealing with life's central mysteries here :-)

Let's return to the question of what a BDI agent does with its beliefs. We saw that beliefs are used for planning, in other words the agent *thinks* with the help of the beliefs. So one of the things we study in AI is the thinking of artificial agents. But isn't thinking the sort of thing best left to psychologists?

Well, it's true that cognitive psychologists study thinking. But we can draw a subtle distinction between the interest psychologists have in thinking and the interest a logician has in thinking. As the philosopher John Pollock said:

> *Psychology is about thinking both when it goes wrong and when it goes right.*
> *Logic is about what it is for thinking to go right.*

So logic is the science of how to think. Some people object to saying that an artificial agent thinks. All right, then — instead of "think" we can say that an agent "transforms information" and that logic is the study of sensible ways to transform information as opposed to careless or crazy ways to do so.

One important way of transforming information is *inference*, the movement from a given piece of information (your *premiss*) to the reasonable expectations you may form on the basis of your premiss (your *conclusions*). The different kinds of logic we shall encounter are different partly because they impose different constraints on this movement from premiss to conclusion. Today's lecture will introduce the very strict constraint giving us *classical logic*.

## Syntax versus semantics: two kinds of representation

Imagine that an agent wants to achieve a goal. Some part of the universe is relevant to the agent's goal and the rest is not. For example, an agent sitting in his car and waiting to cross an intersection is concerned just with the small system consisting of the traffic lights and the oncoming traffic (and perhaps the nearby pedestrians, passing dogs, and any falling aeroplanes that may impact the intersection). For the purposes of deciding whether to cross an intersection in Dunedin, political events in the Middle East are an irrelevant distraction, as are the cost of education in Denmark and whether to buy a puppy. Let's call the small relevant part of the universe the *system of interest.*

In most cases, the agent's planning can't really get started unless he has some idea what the *state* of the system currently is. So the key question is: "What are the facts?"

What information would the driver waiting at an intersection need? He wants the answers to questions such as

- Is the traffic light red?
- Is the oncoming car stopping?

Without answers to these questions, driving across the intersection would be an act of recklessness, not rationality.

The state of a system is made up of answers to all the questions that are relevant to the agent's goal. The answers will typically change as time passes and the system changes its state. More generally, the list of questions that are relevant may change as the agent's intention changes.

It is quite commonly the case that the agent may not be able to find answers to all the relevant questions, so we have the interesting situation of agents having to plan with limited information. We'll talk more about this aspect in future lectures.

An agent can represent information about a state of the system in two ways.

A *symbolic* (or *syntactic*) representation uses text, or more specifically uses declarative sentences of some language, i.e. beliefs. We might say or write that the light is red, that the oncoming car is not stopping, or that a passing poodle has gone mad and is biting a policeman in the middle of the road. We call this a symbolic representation because it uses words built up from symbols. Or we could call it a syntactic description since a language is defined by a grammar and *syntax* is another word for grammar. The main point is that it uses language.

A *semantic* representation doesn't use language. Instead the information is provided by means of a mental image. In the case of vision, we form images that are pictures or stylised pictures (e.g. diagrams or maps). A chess-player pondering his next move will visualise a chessboard, or at least the relevant part of the chessboard. In the case of taste or smell, the mental image is harder to describe, which incidentally shows that it

is non-verbal. Can you recognise what a peach smells like? Then that is because you have an olfactory (smell-based) image against which to compare new smells.

The semantic representations (mental images) are what the symbolic representations (sentences) are **about**.

In our human heads, we do both kinds of representation as patterns of neural excitation. Some patterns produce bits of language (symbols). Others give us images, and these form the semantics of the language. It's hard to talk about such neural patterns, so we won't try to describe the neuroscience.

In logic the symbolic representations of information are done using a formal language rather than a natural language like English, because it is simpler. The semantic representations will be done not by drawing pictures or maps but by using mathematics, because we want to be able to prove things. Our mathematical descriptions of states will start off being quite simple and later will become more interesting and complex.

To summarise: there are two ways to encode information about a state. The agent can describe the state by making assertions about the features of the system in a language, and for the first 9 lectures we will use *propositional* languages for this purpose. Alternatively we may use a mathematical way to represent the state, giving us a subsymbolic idealisation of the analog images that provide the semantics in our human brains.

Let's look more closely at the two kinds of representation by means of an example.

## The traffic system

Imagine that you are waiting in your car at an intersection for an opportunity to safely cross. To keep things very simple, let's suppose that only two features of the system are relevant:

- Is the traffic light currently red for traffic crossing from your left and right?
- Is the oncoming cross traffic going to stop?

Focusing on these two features allows us to build a simple language for symbolically representing information about the traffic system.

Remember the agent asks "What are the facts?"

The two assertions "The traffic light for cross traffic is red" and "The oncoming car is stopping" are called *atomic sentences* because they express elementary facts that might hold for our system of interest. To save us the tedium of writing out these atomic sentences repeatedly, let us abbreviate them. For the rest of this lecture, we use $p$ to abbreviate "The traffic light is red" and $q$ for "The oncoming car is stopping".

As a further simplification we assume that the agent will build up other sentences from the atomic sentences in a few restricted ways, using the words:
**not**, **and**, **or**, **if ... then**, and **if and only if**.

We abbreviate these *connectives* by symbols as follows:

| **Word** | **Symbol** | **Connective Name** |
|---|---|---|
| not | $\neg$ | negation |
| and | $\wedge$ | conjunction |
| or | $\vee$ | disjunction |
| if ... then | $\rightarrow$ | the conditional |
| if and only if | $\leftrightarrow$ | the biconditional |

A sentence such as "The traffic light is not red or the oncoming car is stopping" can now be represented by combining symbols to give $\neg p \vee q$.

We have to be a bit careful not to have ambiguous strings, so will use parentheses where necessary. Thus $\neg p \vee q$ could conceivably mean either $\neg(p \vee q)$ or $(\neg p) \vee q$, where the parentheses indicate the scope of the negation. For simplicity, let's agree that negation binds the shortest sentence following it, so that $\neg p \vee q$ would unambiguously indicate $(\neg p) \vee q$. We are now ready to say what a propositional language is.

**Definition 1** *(Propositional language) Let $A$ be a nonempty set, called the set of atomic sentences. The propositional language $L_A$ generated by $A$ is the set of sentences defined recursively as follows:*

- *If $\alpha \in A$ then $\alpha$ is a sentence of $L_A$*
- *If $\beta$ and $\gamma$ are sentences of $L_A$, then so are $(\neg\beta)$, $(\beta \wedge \gamma)$, $(\beta \vee \gamma)$, $(\beta \rightarrow \gamma)$, and $(\beta \leftrightarrow \gamma)$.*

Suppose the traffic light is red and the oncoming car is not stopping. This state of the system can be described in $L_{\{p,q\}}$ by means of the sentence $p \wedge \neg q$ which tells us, in an abbreviated way, that "The traffic light is red and it is not the case that the oncoming car is stopping". So here we have a symbolic description that identifies the state by using sentences of the agent's knowledge representation language $L_{\{p,q\}}$.

How could we represent the same state semantically?

A very simple semantic way to represent states is to use functions called *truth assignments* instead of sentences in $L_{\{p,q\}}$. Here is how it works.

Our set of atomic sentences is

$$A = \{p, q\}.$$

A sentence may have either of two possible truth values, i.e. each sentence is either true or false, and we indicate truth and falsity respectively by the numbers 1 and 0. So a truth assignment is any function

$$f : A \longrightarrow \{0, 1\}.$$

For example, $f$ might be the function given by $f(p) = 1$ and $f(q) = 0$. Since $p$ abbreviates "The traffic light is red" and $q$ abbreviates "The oncoming car is stopping,"the truth assignment $f$ is a semantic representation of the state in which the traffic light is red ($p$ is true) and the oncoming car is not stopping ($q$ is false).

**Definition 2** *(States) The set of all truth assignments $f : A \longrightarrow \{0, 1\}$ will be denoted by $W_A$ and the set of states of a system of interest will be denoted by $S$.*

For now, we simply assume $S = W_A$. In Lecture 2 we will look at cases in which we might want $S \neq W_A$.

**Notation 1** *(Strings) To avoid mindless tedium we often abbreviate truth assignments as binary strings. For example, the function $f$ that has $f(p) = 1$ and $f(q) = 0$ is abbreviated by writing just 10, since this shorthand tells us that $p$ gets the truth value 1 and $q$ gets the truth value 0. As long as we remember the order in which to consider the atomic sentences, the little string of binary digits tells us exactly which truth assignment we're dealing with.*

It now becomes obvious that the traffic system has exactly four possible states:

$$S = \{11, 10, 01, 00\}.$$

## Satisfaction and models

In the *basic epistemic scenario*, an agent wants to figure out "What is the current state of the system?".

The traffic system has four possible states, namely $11, 10, 01, 00$. Which is the real one? The more information the agent is able to bring to bear, the more possibilities the agent will be able to rule out. Ideally, the agent would be able to get enough information to rule out all but one state, but this ideal is seldom realised in practice — even high-quality sensors like eyes and ears may be unable to narrow down the possibilities to a single candidate for being the actual state.

Suppose the agent waiting at the intersection can see that the traffic light is red but has no information about whether the oncoming car is going to stop. How can the agent represent this very limited information about the state of the Traffic System?

Symbolically, using the propositional language, the agent's best attempt at describing the actual state is just to assert that the traffic light is red, i.e. to assert $p$.

Semantically, using truth assignments, the agent can narrow down the set $S = \{11, 10, 01, 00\}$ of states by excluding the two states in which the light is not red, leaving $\{11, 10\}$ as the candidates that, as far as the agent can tell, might be the actual state.

What is the connection between the agent's symbolic description $p$ and the semantic representation $\{11, 10\}$? The set $\{11, 10\}$ consists of all the states that *satisfy* the sen-

tence $p$, in other words all the states that make $p$ true. We call a state that satisfies a sentence a *model* of that sentence. Every state will satisfy some sentences and fail to satisfy (i.e. make false) the remaining sentences. Similarly, a sentence will usually be true in some states and false in others.

It is easy to figure out whether a state satisfies an atomic sentence like $p$. After all, states are (or are closely associated with) functions that assign truth values to atomic sentences. So we immediately see that the states 11 and 10 satisfy the sentence $p$ whereas 01 and 00 don't satisfy $p$. Similarly 11 and 01 satisfy $q$ but 10 and 00 don't.

But what about a more complex sentence built up from atomic sentences with the help of connectives?

**Definition 3** *(Satisfaction) Let $\alpha \in L_A$ and let $f$ be any state of the system. Then $f$ satisfies $\alpha$ if and only if one of the following is the case:*

- $\alpha \in A$ *and* $f(\alpha) = 1$
- $\alpha = \neg\beta$ *and $f$ fails to satisfy* $\beta$
- $\alpha = \beta \wedge \gamma$ *and $f$ satisfies both* $\beta$ *and* $\gamma$
- $\alpha = \beta \vee \gamma$ *and $f$ satisfies at least one of* $\beta$ *and* $\gamma$
- $\alpha = \beta \rightarrow \gamma$ *and $f$ satisfies* $\gamma$ *or fails to satisfy* $\beta$
- $\alpha = \beta \leftrightarrow \gamma$ *and $f$ satisfies either both* $\beta$ *and* $\gamma$ *or else neither.*

The definition of satisfaction gives us an algorithm by which we can work out whether a state satisfies a sentence or not. Break the sentence down to its atoms, see what truth values the state gives to those, then build up the sentence step by step, keeping track of the truth value each component gets according to the definition of satisfaction.

For example, to check whether 11 satisfies $p \wedge \neg q$, we note that 11 satisfies both atomic sentences $p$ and $q$, and so (by the definition of satisfaction) 11 fails to satisfy $\neg q$, and so (again by the definition of satisfaction) 11 fails to satisfy $p \wedge \neg q$.

As a second example, does 11 satisfy $q \rightarrow p$? Yes, because 11 satisfies the atomic sentence $p$ and thus (by the definition of satisfaction) satisfies $q \rightarrow p$.

Notice that every clause in the definition of satisfaction has a shadow side. The clause for conjunctions tells us directly that a state $f$ satisfies a sentence $\beta \wedge \gamma$ if and only if $f$ satisfies both $\beta$ and $\gamma$. The shadow side of this clause in the definition tells us that if $f$ for some reason fails to satisfy any of $\beta$ and $\gamma$, then that ensures $f$ does not satisfy the conjunction $\beta \wedge \gamma$.

To use the definition of satisfaction, you need to be comfortable with seeing and using the shadow side as well. The exercises will give you practice.

Recall that if a sentence $\alpha$ is satisfied by a state $f$ then $f$ is a model of $\alpha$.

**Definition 4** *(Set of models) The set consisting of all the states that satisfy a sentence $\alpha$ is denoted by $Mod(\alpha)$, which we read: the set of models of $\alpha$.*

A sentence will usually be true in some states and false in the other states. But there are exceptions. Let $\varnothing$ stand for the empty set (i.e. the set with no members).

**Definition 5** *(Special cases) If $Mod(\alpha) = \varnothing$ then $\alpha$ is called a contradiction; if $Mod(\alpha) = S$ then $\alpha$ is called a valid sentence or tautology.*

## Classical entailment

What inferences are sanctioned by classical propositional logic?

We abbreviate "The inference from premiss $\alpha$ to conclusion $\beta$ is allowable in classical logic" by saying that $\alpha$ *classically entails* $\beta$, or writing $\alpha \models \beta$. (The symbol $\models$ is called a double turnstile.) The following definition tells us which inferences are classical entailments.

**Definition 6** *(Classical entailment):*
*$\alpha \models \beta$ if and only if every state $x$ that satisfies $\alpha$ also satisfies $\beta$.*

For those who are mathematically inclined, the definition could be re-written to say that $\alpha \models \beta$ if and only if $Mod(\alpha) \subseteq Mod(\beta)$.

The idea underlying the definition is that a conclusion should follow from a premiss if and only if every state making the premiss true also makes the conclusion true. It must never be the case that the conclusion turns out to be false under circumstances that make the premiss true. Think of this as a very safe criterion — if we know that the premiss is true, we may have complete confidence in the conclusion.

For example, suppose the driver waiting at the intersection can see that the traffic light for cross traffic is red (and so he knows that $p$ is the case). Suppose the driver also knows that he lives in an ideal world in which there is a universally obeyed law to the effect that cars stop at red lights, which we can formalise as the sentence $p \rightarrow q$ and which, in this ideal world, the driver can consider to be true. (So in this imaginary universe, there are no reckless or careless drivers who drive across red lights.)

The two items give the driver two premisses, namely $p$ and $p \rightarrow q$, which we may combine into a single premiss of the form $p \wedge (p \rightarrow q)$.

The premiss $p \wedge (p \rightarrow q)$ entails the conclusion $q$ because every model of $p \wedge (p \rightarrow q)$ is also a model of $q$.

To see this, let us calculate the models of $p \wedge (p \rightarrow q)$. We can do this by considering each state in turn.

State 11 satisfies $p$ and (since 11 satisfies $q$) also satisfies $p \to q$, so that 11 satisfies $p \wedge (p \to q)$.

State 10 fails to satisfy $p \to q$ and thus fails to be a model of $p \wedge (p \to q)$.

State 01 fails to satisfy $p$ and thus fails to be a model of $p \wedge (p \to q)$.

Finally, state 00 fails to satisfy $p$ and thus fails to be a model of $p \wedge (p \to q)$.

Accordingly $Mod(p \wedge (p \to q)) = \{11\}$.

Now, does 11 satisfy the conclusion $q$? Yes, clearly it does. Thus

$$p \wedge (p \to q) \models q.$$

## Propositions

Let's talk about information. A sentence like $p$ expresses symbolically some information about what the current state of the system is, namely that it is a state in which the traffic light is red (since $p$ abbreviates "The light is red").

The set of models of $p$ is $M(p) = \{11, 10\}$ and this set of models expresses exactly the same information as the sentence $p$ itself, because the set of models $\{11, 10\}$ is formed from the set $S = \{11, 10, 01, 00\}$ of all states by throwing away the states 01 and 00 in which the light is not red.

So a sentence and its set of models may be regarded as expressing the same information. The sentence expresses the information symbolically, while the set of models expresses it semantically.

Now consider that any item of information can be expressed symbolically in more than one way. For example, saying that the light is red ( $p$) conveys exactly the same information as saying that the light is not not red ($\neg\neg p$). We can make this idea precise by using the notion of *equivalence*, where we shall abbreviate the phrase "is equivalent to" by the symbol $\equiv$.

**Definition 7** *(Equivalence)* $\alpha \equiv \beta$ *if and only if* $Mod(\alpha) = Mod(\beta)$.

To see why this matters, think of all the sentences of $L_A$. Notice that even when $A$ is a small set of atomic sentences like $\{p, q\}$, the set of sentences $L_A$ is very large, in fact there are an infinite number of sentences, because they can be made arbitrarily long:

$$p, \neg p, \neg\neg p, \neg\neg\neg p, \ldots$$

However, this doesn't necessarily mean we can say infinitely many things using our language. Lots of these sentences say the same thing, in the sense that they exclude the same states. For example, $p$ and $\neg\neg p$ both have 11 and 10 as their models and thus

convey the same information. We can group sentences into *equivalence classes* according to their models, and each equivalence class will have infinitely many sentences in it.

For example, in the $p, q$-language, there are infinitely many sentences that all have the models 11 and 10, some of which are

$$p, p \wedge p, p \wedge (p \wedge p), p \wedge (p \wedge (p \wedge p)), \ldots$$

All these sentences are equivalent — they say the same thing in increasingly boring ways.

**Definition 8** *(Proposition) Let $X \subseteq S$ be any set of states. The set of all sentences $\alpha$ such that $Mod(\alpha) = X$ is called a proposition.*

Thus a proposition is an equivalence class of sentences, and each proposition is associated with a particular set of models (and thus with a particular item of information). It will be convenient, when speaking of a proposition $X$, to pick one of the sentences in the equivalence class and use that sentence as the generic representative of the proposition. For example, when we speak of the endless set of sentences each of which has the models 11 and 10, then we may simply say that the proposition is $p$, even though really $p$ is just one of the many sentences in the proposition.

So the propositions of a language convey the various pieces of information that can be expressed by that language. How many different pieces of information can be expressed by the language $L_{\{p,q\}}$, you may wonder? That's what exercises are for!

## Exercises

For these exercises, we assume that we are dealing with the traffic system and have a language $L_{\{p,q\}}$ generated by the two atomic sentences in $A = \{p, q\}$ with a set of states $S = W_A = \{11, 10, 01, 00\}$.

Quiz: The quiz question at the start of lecture 2 will come from exercise 3 below.

1. (Exercises on syntax) Which of the following symbols can appear in a sentence of $L_{\{p,q\}}$?
   - $\rightarrow$
   - $\models$
   - 00
   - $p$
   - $\leftrightarrow$
   - $\equiv$

2. (Exercises on satisfaction) For each of the following, determine whether the sentence is satisfied by the state 01 and show your reasoning:

   - $p \wedge \neg q$
   - $\neg(p \vee \neg q)$
   - $\neg(q \rightarrow p)$

   Here is a worked example showing how to set out your reasoning.

   > Suppose we want to show that 01 satisfies $p \vee (p \rightarrow q)$.
   >
   > The state 01 fails to satisfy $p$ but satisfies $q$.
   >
   > By the definition of satisfaction, 01 satisfies $p \rightarrow q$.
   >
   > Hence, by the definition of satisfaction, 01 satisfies $p \vee (p \rightarrow q)$.

3. (Exercises on models) For each of the following sentences $\alpha$, write down its set of models $Mod(\alpha)$:

   - $p \vee \neg p$
   - $p \vee q$
   - $p \vee \neg q$
   - $\neg p \vee q$
   - $\neg p \vee \neg q$
   - $p$
   - $q$
   - $p \leftrightarrow q$
   - $\neg(p \leftrightarrow q)$
   - $\neg q$
   - $\neg p$
   - $p \wedge q$
   - $p \wedge \neg q$
   - $\neg p \wedge q$
   - $\neg p \wedge \neg q$
   - $p \wedge \neg p$

   Remark: Do you see that we have given the 16 different propositions that can be expressed by the $p, q$-language?

4. (Exercises on equivalence) For each of the following, decide which of the 16 sentences listed in exercise 3 best expresses the same information and show your reasoning:

   - $q \vee p$
   - $q \wedge p$
   - $p \rightarrow q$
   - $p$ if $q$
   - Either $p$ or $q$ but not both.
   - $p \rightarrow p$
   - $p \rightarrow \neg p$
   - $\neg(p \wedge q)$
   - $\neg(p \vee q)$

5. (Exercises on entailment) Arrange the sentences of exercise 3 in rows such that all the sentences in the same row have the same number of models and, as you descend from the top row, the number of models becomes one smaller in each step. Leave a nice big gap between rows. Think of each sentence as a vertex in a graph. Now insert edges according to the following rule. Each edge goes upward from one row to the next row, and the edge connects a lower vertex $x$ to a higher vertex $y$ if and only if $x \models y$.

   Remark: The graph that you get is called the Lindenbaum-Tarski algebra. It is an example of a Boolean algebra and is the reason propositional logic is sometimes called Boolean logic.

6. (Exercises on designing languages) Using what you know, give a set $A$ of atomic sentences and a set $S$ of states appropriate for logically modelling each of the following systems of interest.

   - The light-fan system has two components, a light and a fan, each of which may be either on or off independently. The only features of interest are whether the components are on or not.
   - The light-fan-heater system has three components, a light, a fan, and a heater, each of which may be either on or off independently. The only features of interest are whether the components are on or not.
   - A stable has two horses, Andy and Bandy. In each case, we are interested in whether the horse has been fed and whether it has been watered.