To make some extra cash during the semester you take up a part-time job at CUHK. On your first morning of work, your boss – let's call him Bob – gives you an inventory of the watermelon reserves across campus:

location	stock
330 Cafe	3 watermelons
S. H. Ho Canteen	1 watermelon
Morningside Canteen	7 watermelons
University Guesthouse	2 watermelons
Canteen at Medical School	1 watermelon

In anticipation of the lunch hour rush, Bob would like to redistribute the watermelons like this:

location	request
330 Cafe	2 watermelons
S. H. Ho Canteen	4 watermelons
Morningside Canteen	1 watermelon
University Guesthouse	1 watermelon
Canteen at Medical School	6 watermelons

The distance between any pair of consecutive locations, give or take, is about 100 meters.

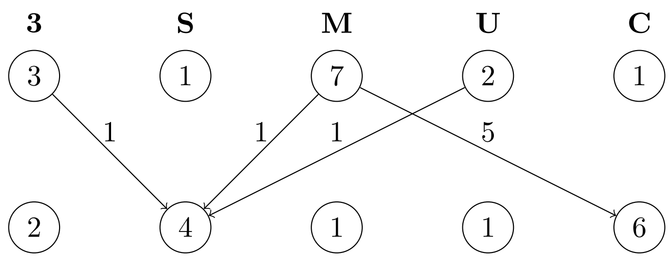
$$\mathbf{3} \xleftrightarrow{\text{100m}} \mathbf{S} \xleftrightarrow{\text{100m}} \mathbf{M} \xleftrightarrow{\text{100m}} \mathbf{U} \xleftrightarrow{\text{100m}} \mathbf{C}$$

Shuttling watermelons is demanding work. A porter asks for 10 HKD for every watermelon carried over a distance of 100 meters. If he were to carry, for example, two watermelons from the 330 Cafe to Morningside – at a distance of 200 meters – that would cost Bob 40 HKD. Being on a tight budget, Bob asks his other employee Jason, a student in the Business School, to come up with an economical way of moving the watermelons around.

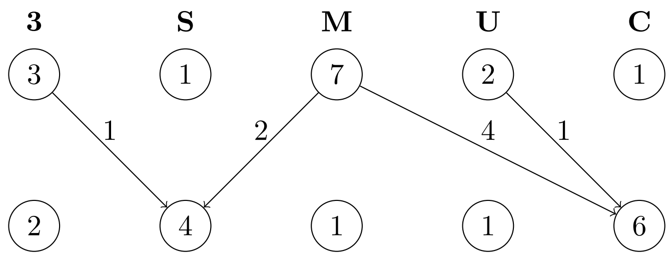
You, a promising young engineer in pursuit of an exciting assignment, overhear Jason shouting the following instructions to the porter:

> Move 5 watermelons from Morningside to the Canteen at the Med School. Then take one watermelon from each of the 330 Cafe, Morningside, and the University Guesthouse and move those to S. H. Ho.

Jason's plan involves moving two watermelons at a distance of 100 meters and another six at a distance of 200 meters, for a total cost $2 \times 10 + 6 \times 20 = 140$ dollars. You take out a piece of paper and make a quick sketch of it:



You notice something fishy: The trips out of Morningside and the University Guesthouse cross paths. This is clearly wasteful, so why not move the watermelons around like this instead:



Indeed, now you are moving 4 watermelons at 100 meters and another 4 at 200 meters for a total cost of 120 calories! Bursting with excitement, you go and show your improvement to Bob. He is happy that you will save him 20 HKD. Then he asks: “Since you are so bright, can you save me another 20 HKD and do the whole operation for 100?”

You scratch your head for a few minutes, but you are at a loss; nothing seems to work. You are a bit embarrassed to admit your failure to Bob. What if your rival Jason impresses Bob with an even better solution?

Proofs

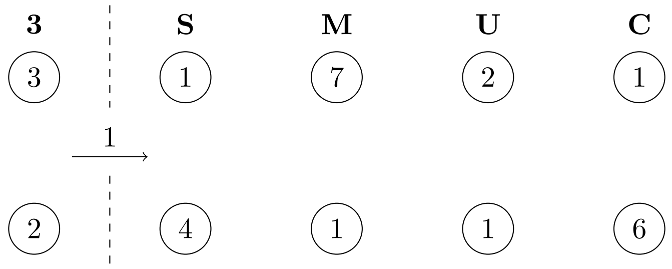
Much of the mathematics you study in school is about *calculating* things. In first grade you learn how to add single digit numbers. Later you move on to larger numbers and more complicated calculations, like multiplications, divisions, and square roots. In high school, you may be calculating roots of quadratic equations, sines and cosines, and doing some complex number algebra. At university, you take derivatives and compute integrals, solve systems of linear equations and differential equations.

In order to be a good engineer, you certainly need to be a master at various calculations that come up routinely in your discipline. But calculating is not enough. You will need to learn to set up and solve problems in a confident manner.

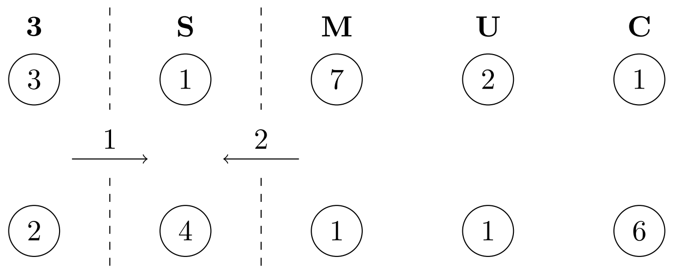
In Bob’s watermelon transportation task, a mediocre engineer (Jason) might be satisfied with a solution that “feels” good to him. A great engineer, like you, wants more: You want to be sure that your solution is the best possible one. For this, calculations are not particularly helpful; you need to reason things out.

After thinking for a while, you are quite sure that it is impossible to spend less than 120 dollars on the watermelons. But how do you explain this to Bob?

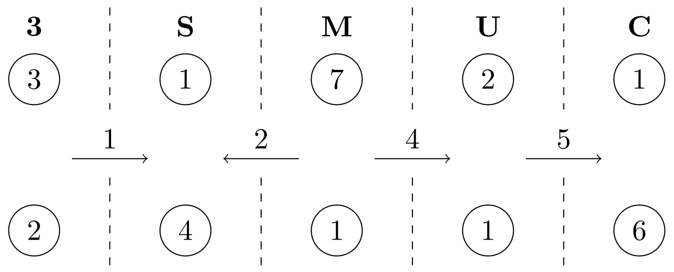
Here is how. You notice that at the 330 Cafe, there are three watermelons in the morning and we need to end up with two in the afternoon; so *no matter how things are moved*, at least one watermelon will have to be carried out of the 330 Cafe:



Next, if you add the number of watermelons at the 330 Cafe and S. H. Ho, there are 4 available but 6 are needed; so no matter how the watermelons are carried, at least two will have to be brought in from Morningside or beyond:



Continuing your reasoning in this way, you come up with the following picture:



It is now clear that Bob must spend at least 10 HKD moving crates between the 330 Cafe and S. H. Ho, at least 20 HKD between S. H. Ho and Morningside, and so on. Adding all the expenses together amounts to exactly 120 HKD. Your solution was indeed the best possible.

What we just saw is an example of a *proof*: A deduction of an interesting *proposition* ("No matter how the watermelons are carried, Bob must pay at least 120 HKD") by a sequence of clear, rigorous *logical deductions*. The ability to come up with proofs and present them clearly is important for computer science and other engineering disciplines. In the next few weeks, we will talk about various types of proofs in some detail.

1 Propositions

A *proposition* is a statement that is either true or false. Here are two examples of propositions.

> $1 + 1 = 2$.
> Thursday is the day after Tuesday.

The first proposition is about numbers; the second one is about days of the week. The first proposition is true; the second one is false. This can be figured out by most people with a first grade education (where we all learned the meaning of "1", "+", "Tuesday", and so on).

Telling whether a proposition is true or false is not always easy, but the *meaning* of a proposition must always be clear. For example, consider the following statements:

> Smoking kills.
> I will always love you.
> You may have cake, or you may have ice cream.

What do they mean exactly? Who does smoking kill? Will you love me even if I moved to Australia for good? May I have ice cream on top of my cake? In daily life, this kind of ambiguity is tolerable and even desirable, but it is not acceptable in mathematics and much of computer science (in programming, for example). We will not call such statements propositions.

Propositional logic and truth tables

We can modify and combine propositions using *operators* such as AND, OR, and NOT. For example, the proposition

$$\text{NOT } (1+1=3)$$

is true, while the proposition

$$(1+1=2) \text{ AND } (1+1=3)$$

is false.

In general, given an arbitrary proposition P, we can build the proposition NOT P. The proposition NOT P is false when P is true, and true when P is false. We can describe the effect of NOT compactly in a *truth table*:

P	NOT P
T	**F**
F	**T**

Given two propositions P and Q, we can form the *compound propositions* P AND Q, P OR Q. Here are their truth tables:

P	Q	P AND Q
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

P	Q	P OR Q
T	**T**	**T**
T	**F**	**T**
F	**T**	**T**
F	**F**	**F**

This is a different from the way the word "or" is used in common English. When you see a dinner set in a restaurant that comes with "beer or wine", it is usually understood that you cannot have both. In contrast, in mathematics and computer science, P OR Q is also true when P is true and Q is true.

The English meaning of "You can have beer or wine with your dinner" is captured by the operator XOR, which stands for "exclusive or":

P	Q	P XOR Q
T	**T**	**F**
T	**F**	**T**
F	**T**	**T**
F	**F**	**F**

We say P and Q are *logically equivalent* if they take the same truth value. The operator IFF (short for "if and only if") describes logical equivalence:

P	Q	P IFF Q
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**T**

We can go on and on, but in fact every compound proposition is logically equivalent to a propositional formula that uses only the operators AND, OR, and NOT. For example

$$P \text{ XOR } Q \quad \text{is logically equivalent to} \quad ((\text{NOT } P) \text{ AND } Q) \text{ OR } (P \text{ AND } (\text{NOT } Q)).$$

One way to verify this is to compute the truth table of the second formula and compare it to the truth table for XOR:

P	Q	NOT P	(NOT P) AND Q	NOT Q	P AND (NOT Q)	((NOT P) AND Q) OR (P AND (NOT Q))
T	**T**	**F**	**F**	**F**	**F**	**F**
T	**F**	**F**	**F**	**T**	**T**	**T**
F	**T**	**T**	**T**	**F**	**F**	**T**
F	**F**	**T**	**F**	**T**	**F**	**F**

2 Quantifiers

A *predicate* is a proposition whose truth may depend on one or more *free variables*. For example, "n is even" is a predicate (about integer numbers) with free variable n. It is true when $n = 2$ and false when $n = 3$. The predicate "$n = 2 \times m$" is true when $n = 4, m = 2$ and false when $n = 2, m = 4$.

A predicate can be turned into a proposition by *quantifying* over the free variables. For example, the statement

For all n, n is even

is a proposition. This proposition is false because when $n = 3$, "n is even" becomes false. On the other hand, the proposition

There exists an n such that n is even

is true because when $n = 2$, "n is even" becomes true.

A proposition like "10 is even" can itself be written using quantifiers: A number is even if it equals twice *some* other number, namely

There exists an m such that $10 = 2 \times m$.

This proposition is true because $10 = 2 \times 5$. The proposition "For all numbers n, n is even" can be written as

For all n there exists an m such that $n = 2 \times m$

As we saw, this one is false.

Both the *names* of the quantified variables and the *order* in which they appear matters in such statements. Do not be careless with them! For example, if we change the role of m and n, we obtain the proposition

For all m there exists an n such that $n = 2 \times m$

which is true. Now if we change the order of the quantifiers in the last statement, we obtain

There exists an n such that for all m, $n = 2 \times m$

which is false again.

The implies operator

The IMPLIES operator, which we also write as $\longrightarrow$, captures the meaning of the English conditional "If P then Q". It has the following truth table:

P	Q	$P \longrightarrow Q$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

So the proposition

$1 + 1 = 2 \longrightarrow 1 + 1 = 3$

is false. On the other hand, the propositions

$1 + 1 = 3 \longrightarrow 1 + 1 = 2$
$1 + 1 = 3 \longrightarrow 1 + 1 = 4$

are both true. This may sound a bit strange at first, but it makes perfect sense: If $1 + 1 = 3$, which is clearly false, then anything goes. *A false proposition implies any other proposition.*

The *implies* operator comes in handy when reasoning about predicates. For example, let's take the following proposition about integer numbers:

(1) Every even number is the sum of two odd numbers.

Let's give the number a name; let's call it n. Proposition (1) says that if n happens to be even, then n must be the sum of two odd numbers:

(2) For every n, (n is even) $\longrightarrow$ (n is the sum of two odd numbers)

Let's give the predicate "(n is even) $\longrightarrow$ (n is the sum of two odd numbers)" a name: We'll call it $P(n)$. Then $P(0)$ is true because 0 is even and it is the sum of two odd numbers (1 and -1). $P(1)$ is true because 1 is not even; $P(8)$ is true because 8 is even and it is the sum of 3 and 5. It looks like this proposition may be true.

Using the definitions of "even number" and "odd number", we can further expand proposition (2) like this:

> (3) For every n, (There exists an m such that $n = 2 \times m$) $\longrightarrow$ (There exist a and b such that ($n = a + b$) AND (there exists a c such that $a = 2 \times c + 1$) AND (there exists a d such that $b = 2 \times d + 1$)).

Propositions (1), (2), and (3) are different formulations of the same statement. Which is the best one to use? It all depends on context. For example, if you are not sure whether the proposition is true and want to ask your teacher about it, you should be as concise as possible and use formulation (1): "Is it true that every even number is the sum of two odd numbers?" If you want to reason about the truth of the proposition by trying out different cases as we just did, then formulation (2) is more suitable. Formulation (3) is too detailed for most purposes and would be rarely used in practice. Such formulations might come up, for instance, in automated theorem proving.

You will need to become comfortable at translating between formulations of the same proposition at different levels of detail inside your head.

From an English sentence to a proposition

Let's practice translating some English statements into propositions with quantifiers. The propositions will be about people in a group (Alice, Bob, Charlie, ...) and friendships among them. We'll write $F(x, y)$ for the predicate "x and y are friends". When writing propositions formally, it is customary to use the symbol $\exists$ for "there exists" and $\forall$ for "for all".

Example 1. Let's take the statement "Alice has friends". It means that there is *someone* out there who is Alice's friend, which calls for an existential quantifier:

$$\exists x\colon F(\text{Alice}, x).$$

Example 2. By default, Facebook allows friends of friends to view your profile. How do you write "Alice is a friend of a friend of Bob"? This statement says there is someone out there who is a friend of Bob and also a friend of Alice:

$$\exists x\colon F(\text{Alice}, x) \text{ AND } F(\text{Bob}, x).$$

Example 3. How about "Alice and Bob have the same friends"? This statement says that *anyone* who is friends with Alice is friends with Bob, and vice versa. The quantifier here is universal, and the operator IFF comes in handy:

$$\forall x\colon F(\text{Alice}, x) \text{ IFF } F(\text{Bob}, x).$$

Example 4. "Everyone has a friend" is expressed as $\forall x \exists y\colon F(x, y)$, while "Someone in the group is everyone's friend" is expressed as $\exists y \forall x\colon F(x, y)$.

Example 5. "Alice has no friends" says there does not exist a person who is Alice's friend, or NOT $\exists x\colon F(\text{Alice}, x)$. How about Alice has "exactly one friend"? This proposition is a bit tricky because it says two different things: (1) Alice has at least one friend ($\exists x\colon F(\text{Alice}, x)$) and (2) Alice has no more than one friend. One way to formulate (2) is to say "Any two friends of Alice must in fact be the same person", or "If x and y are both friends with Alice, then $x = y$". We can express (2) as:

$$\forall x, y\colon (F(\text{Alice}, x) \text{ AND } F(\text{Alice}, y)) \longrightarrow x = y$$

and so the statement "Alice has exactly one friend" becomes

$$(\exists x\colon F(\text{Alice}, x)) \text{ AND } ((\forall x, y\colon F(\text{Alice}, x) \text{ AND } F(\text{Alice}, y)) \longrightarrow x = y).$$

Mathematical books and manuscripts, for example our textbook, rarely use such formal notation because it is very difficult to read. In your own mathematical writing – including your homework solutions – I also encourage you to express your propositions in plain English as much as possible. However, it is important that the meaning of your proposition always be *clear* and *unambiguous*; if the need arose, you ought to be able to express your proposition using logical symbols.

3 Quantifier logic

Here are two rules that come in handy for quantified predicates.

Negating quantifiers The proposition "NOT (For every $x, P(x)$)" is equivalent to "There exists an x such that NOT $P(x)$".

For example, the proposition "Not every number is greater than 2" is equivalent to "There exists a number no greater than 2."

Universal instantiation If the proposition "There exists an x such that for all y, $P(x, y)$" is true, then the proposition "For all y there exists a x such that $P(x, y)$" is also true.

For example, take the (true) proposition "There is a day of the week when there are no classes" (about the CUHK class schedule). This is the same as saying "There is a d such that for every c, class c does not meet on day d of the week". By the universal instantiation rule, we can conclude that "For every c, there is a d such that class c does not meet on day d of the week", i.e., "For every class there is a day of the week when the class does not meet." Indeed, that day is Sunday.

The *converse* of this rule is invalid: For example, the predicate "Every class meets on some day of the week" is true, but "There exists a day of the week on which every class meets" is false.

In general, when we want to know how two quantified predicates relate to one another we cannot rely on truth-tables because there is an infinite number of possibilities to consider. For example, suppose we want to know if the two predicates

$(\forall x\colon P(x)) \text{ OR } (\forall x\colon Q(x))$
$\forall x\colon P(x) \text{ OR } Q(x)$

are equivalent to one another. How should we go about this?

One useful way to get started is to assign some meaning to the abstract symbols like x, P, and Q. For example, imagine a world of balls that may come in different shapes and weights. Suppose x represents a ball, $P(x)$ means "x is blue", and $Q(x)$ means "x is heavy". Then we can interpret the two propositions by the following English sentences:

All balls are blue, or all balls are heavy.
Any given ball is blue or heavy (or both).

If the first predicate is true, clearly so is the second one: Our world must consist of blue balls only or heavy balls only, and in either case any given ball is blue or heavy.

However, if the second predicate is true, the first one may not necessarily be so. For example, our world could consist of one ball that is blue and light, and another one that is red and heavy. In this world, the second predicate is true but the first one is false. So we can conclude that the two are not equivalent.

4 Reasoning about quantifiers

It is often tricky to figure out whether a proposition that involves quantifiers is true or false.

Let's start with the proposition

(1) Every even number is the sum of two odd numbers.

How do we go about figuring out if this is true? First, we identify that the leading quantifier refers to "every even number." We want to know if something is true for every even number, so we start by trying out some examples. Is the statement true for 2? Yes, because $2 = 1 + 1$. Is it true for 4? Yes, $4 = 3 + 1$. Is it true for 0? Yes, $0 = 1 + (-1)$. These "experiments" seem to indicate that the proposition is true. Should we leave it at that?

Unfortunately we can't. A predicate $P(n)$ may well turn out to be true for all the cases n we can think of checking, but the proposition "For all n, $P(n)$" could still be false. Here is a nice example:

(2) For every nonnegative number n, the number $n^2 + n + 41$ is prime.

To determine if this proposition is true, let's try out some examples. $0^2 + 0 + 41 = 41$, which is prime. $1^2 + 1 + 41 = 43$, which is prime. $2^2 + 2 + 41 = 47$, which is prime. Let's try something bigger. $10^2 + 10 + 41 = 151$, which also happens to be prime. Can we conclude that the proposition is true? If we did, we would be wrong: The number $41^2 + 41 + 41$ is not prime; it is the product of 41 and 43.

Does this mean that the effort we spent in checking cases was wasted? No! If you look carefully, in our analysis of proposition (1) all the examples we checked show a pattern: We write our even number as another number plus one. We are on to something: Every even number n is the sum of $n - 1$ and 1. But when n is even, $n - 1$ must be odd. So of course n is the sum of two odd numbers!

In contrast, for proposition (2), there is no discernible pattern in the examples we worked out. This may mean two things: Maybe the pattern is there but we cannot see it, or maybe the proposition is false. In this case the proposition turned out to be false.

Unfortunately, there is no foolproof method for finding the truth of propositions with quantifiers. Mathematicians can spend their whole lives trying to figure out just one or a handful of propositions. Here is one, Goldbach's Conjecture from the year 1742, that still eludes them:

Every even number greater than 2 is the sum of two primes.

Computers have checked all values up to 4000000000000000000, but we still do not know if Goldbach's conjecture is true.

The best way to learn how to reason about propositions is through practice. But first we have to agree on a standard by which we can agree that a proposition has been established as true: a mathematical proof.

5 Satisfiability*

The material in starred sections is covered in ESTR 2004. It is optional for ENGG 2440A students.

Truth-table calculations are handy for formulas that depend on 2 or 3 atomic propositions, but the work becomes significantly more difficult as the number grows. In general, the truth table of a compound formula built from propositions P_1, P_2, up to P_n has 2^n rows. When $n = 20$ the number of rows is over a million and when $n = 100$ it is a 30-digit number. Writing out the truth table is then out of the question even for the most powerful computers. In certain cases, there are much faster strategies.

Suppose we want to know if the proposition ϕ given by

$$(P \text{ AND } \overline{Q}) \text{ OR } (\overline{P} \text{ AND } \overline{R} \text{ AND } \overline{S}) \text{ OR } (Q \text{ AND } S) \text{ OR } (P \text{ AND } \overline{S})$$

is a *tautology*, namely always true. Here we use $\overline{P}$ as a shorthand for NOT P. One way is to write out the 16-row truth table and check if the last column is all true. Here is another method. First, we write out the proposition NOT ϕ which is given by the formula

$$(\overline{P} \text{ OR } Q) \text{ AND } (P \text{ OR } R \text{ OR } S) \text{ AND } (\overline{Q} \text{ OR } \overline{S}) \text{ AND } (\overline{P} \text{ OR } S)$$

(If you are not sure how NOT ϕ was derived from ϕ see *De Morgan's laws* in Section 3.4.2 of the textbook.) Instead of checking if proposition F is always true, we'll try to figure out if proposition NOT ϕ is always false. So let's investigate if is possible to find truth-values for P, Q, R, and S that make NOT ϕ evaluate to true. Such a setting of truth-values is called a *satisfying assignment* for the formula NOT ϕ.

Let's consider separately the cases when P is true and when P is false. If P was true, $\overline{P}$ must be false and the formula for NOT ϕ simplifies to

$$Q \text{ AND } (\overline{Q} \text{ OR } \overline{S}) \text{ AND } S.$$

To obtain a satisfying assignment, it better be that Q and S are both true. But then $\overline{Q}$ OR $\overline{S}$ is false. We conclude that there is no satisfying assignment for NOT ϕ in which P is true.

Let us now consider the case when P is false. The formula for NOT ϕ then becomes

$$(R \text{ OR } S) \text{ AND } (\overline{Q} \text{ OR } \overline{S}).$$

We again consider what happens depending on the truth-value of Q. When Q is true, the formula simplifies to R OR S, which is clearly satisfiable when R is true. We conclude that the assignment $P = \mathbf{F}$, $Q = \mathbf{T}$, $R = \mathbf{T}$ and, say, $S = \mathbf{T}$ is satisfying for the proposition NOT ϕ. Therefore NOT ϕ is not always false and so ϕ is not a tautology.

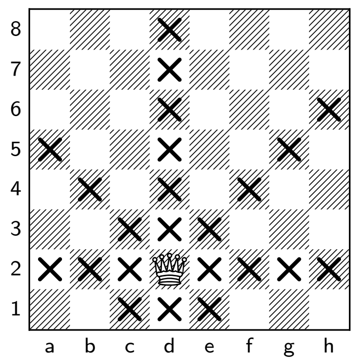
Beyond tautologies The *satisfiability problem* is the following algorithmic question: Given a propositional formula, find a satisfying assignment for it if one exist and output "unsatisfiable" otherwise. Despite its apparent simplicity, this problem is immensely important in virtually all areas of computer science (including for instance machine learning, optimization, and cryptography).

The propositional formulas that are relevant to such domains may contain thousands or even millions of atomic propositions, so solutions by *brute-force search* that look at the whole truth table are out of the question. There are other, more clever algorithms, such as the DPLL algorithm which we used to find our satisfying assignment to the formula NOT ϕ.[1]

Unforunately, there is no general-purpose satisfiability algorithm that is sound (it is never wrong), complete (it always produces an answer), and efficient. Sometimes, however, algorithms like DPLL are much faster than brute-force search. Here is one such example.

> **The eight queens puzzle:** Can you place eight queens on a chessboard so that no two of them attack one another?

If you are new to chess, it is enough for you to know that a queen attacks those pieces that share its row, column, or diagonal. For example a queen at position d2 attacks all the crossed squares on the following chessboard:



The eight queens puzzle can be modelled as satisfiability of a propositional formula ψ that represents the desired configuration of the chessboard. There are several possible ways to construct this formula; here is one that is particularly natural for the eight queens puzzle. We start with 64 atomic propositions $P_{\mathrm{a1}}, \ldots, P_{\mathrm{h8}}$, one for each square of the chessboard. The intended meaning of P_{a1} is "square a1 is occupied by a queen", and so on.

We would like the formula ψ to evaluate to true if and only if propositions P_{a1} up to P_{h8} describe a configuration in which no two queens attack one another. We can naturally break up the constraints imposed by the puzzle into the following three parts:

ψ_{row}: At least one square in each row is occupied (by a queen).

ψ_{col}: At least one square in each column is occupied.

ψ_{attack}: If two squares share a row, column, or diagonal, then they are not both occupied.

The propositional formulas for ψ_{row}, ψ_{col}, and ψ_{attack} are tedious to write out by hand but this can be easily done on the computer. Formula ψ_{row} looks like this:

$$(P_{a1} \text{ OR } P_{b1} \text{ OR } \cdots \text{ OR } P_{h1}) \text{ AND } \cdots \text{ AND } (P_{a8} \text{ OR } P_{b8} \text{ OR } \cdots \text{ OR } P_{h8})$$

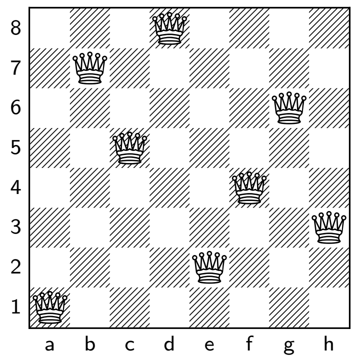
[1] DPLL and some other algorithms require that the formula is written in *conjunctive normal form* (CNF) – an AND of ORs of *literals*, each of which is an atomic proposition or its negation. There is a procedure for converting an arbitrary formula into a CNF (possibly with some extra literals) so that satisfying assignments are preserved.

Similarly, formula ψ_{col} looks like this:

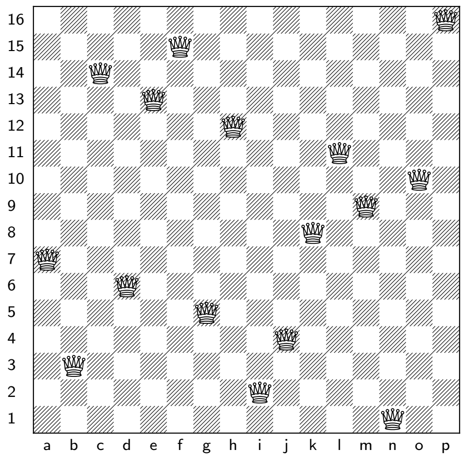
$$(P_{a1} \text{ OR } P_{a2} \text{ OR } \cdots \text{ OR } P_{a8}) \text{ AND } \cdots \text{ AND } (P_{h1} \text{ OR } P_{h2} \text{ OR } \cdots \text{ OR } P_{h8})$$

The formula ψ_{attack} is an AND of expressions of the form $(\overline{P}_{ij} \text{ OR } \overline{P}_{i'j'})$, where ij and $i'j'$ range over all pairs of positions that share a row ($j = j'$), a column ($i = i'$), or a diagonal ($|j - i| = |j' - i'|$). For example, ψ_{attack} contains the expression $(\overline{P}_{\text{b3}} \text{ OR } \overline{P}_{\text{f7}})$, but does not contain $(\overline{P}_{\text{c3}} \text{ OR } \overline{P}_{\text{e6}})$.

I wrote a computer program that generates the formula ψ and applies the minisat algorithm to search for a satisfying assignment. (The python satispy package came in handy for writing and running the code.) The program came up with a satsifying assignment that translates to the following eight queens puzzle solution:



The puzzle generalizes easily to other sizes. The same code produced the following solution for a 16×16 board:



References

This lecture is based on Chapter 3 of the text *Mathematics for Computer Science* by E. Lehman, T. Leighton, and A. Meyer. Material from slides by Prof. Lap Chi Lau were also used in the preparation.