n = 3. There are $2^{2^3} = 256$ ternary connectives. These are too many to list, but an interesting example is the "if x then y, else z" connective, whose value is y if $x = 1$ and z if $x = 0$. Another one is the majority connective:

$$\text{maj}(x, y, z) = \begin{cases} 1 & \text{if the majority of } x, y, z \text{ is } 1 \\ 0 & \text{otherwise} \end{cases}$$

For each wff A we indicate the fact that A contains only propositional variables among $p_1, \ldots, p_n$ by writing $A(p_1, \ldots, p_n)$. This simply means that $\text{supp}(A) \subseteq \{p_1, \ldots p_n\}$. It does not mean that A contains *all* of the propositional variables $p_1, \ldots, p_n$.

Definition 1.6.2 For each wff $A(p_1, \ldots, p_n)$, we define an n-ary truth function $f_A^n : \{0, 1\}^n \to \{0, 1\}$ by

$$f_A^n(x_1, \ldots, x_n) = \text{ (the truth value of } A \text{ given by the valuation } \nu(p_i) = x_i).$$

In other words, f_A is the truth function corresponding to the truth table of the wff A.

Examples 1.6.3

(i) For any binary connective $*$, if $A = (p_1 * p_2)$, then f_A is the truth function f_* corresponding to $*$. Similarly, if $A = \neg p_1$, then $f_A = f_\neg$, and so on.

(ii) If $A = ((p_1 \wedge p_2) \vee (\neg p_1 \wedge p_3))$, then f_A is the truth function of the "if. . . then. . . else. . . " connective.

Remark. Strictly speaking, each wff A gives rise to infinitely many truth functions f_A^n: one for each $n \geq n_A$, where n_A is the least number m for which $\text{supp}(A) \subseteq \{p_1, \ldots, p_m\}$. This is the same situation that we face when we have a polynomial, like $x + y$, which we can view as defining a function of two variables x, y, but also as a function of three variables x, y, z, which only depends on x, y, etc. However when the n is understood or irrelevant we just write f_A.

Note that by definition

$$A \equiv B \text{ iff } f_A = f_B.$$

The main fact about truth functions is the following:

Theorem 1.6.4 *Every truth function is realized by a wff containing only* $\neg, \wedge, \vee$. *That is, if* $f : \{0,1\}^n \to \{0,1\}$ *with* $n \geq 1$, *there is a wff*

$$A(p_1, \dots, p_n)$$

containing only $\neg, \wedge, \vee$ *such that*

$$f = f_A \ (= f_A^n).$$

Proof. If f is identically 0, take $A = (p_1 \wedge \neg p_1)$.

Otherwise, for each entry $s = (x_1, \dots, x_n) \in \{0,1\}^n$ in the truth table of f, introduce the wff

$$A_s = \epsilon_1 p_1 \wedge \epsilon_2 p_2 \wedge \cdots \wedge \epsilon_n p_n$$

where

$$\epsilon_i = \begin{cases} \text{nothing} & \text{if } x_i = 1 \\ \neg & \text{if } x_i = 0 \end{cases}.$$

Example. $s = (1,0,1) \to A_s = p_1 \wedge \neg p_2 \wedge p_3$.

Notice that $f_{A_s}(x_1, \dots, x_n) = 1$, but $f_{A_s}(x_1', \dots, x_n') = 0$ if $(x_1', \dots, x_n') \neq (x_1, \dots, x_n)$. Enumerate in a sequence $s_1, \dots, s_m$ $(m \leq 2^n)$ all $s \in \{0,1\}^n$ such that $f(s) = 1$ (and there is at least one such) and put

$$A = A_{s_1} \vee A_{s_2} \vee \cdots \vee A_{s_m}.$$

Then $f_A(s) = 1$ iff at least one of $f_{A_{s_i}}(s) = 1$ iff $s \in \{s_1, \dots, s_m\}$ iff $f(s) = 1$, so $f_A = f$. $\dashv$

Example 1.6.5 Suppose that f is given by the following truth table.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Then f is realized by the wff

$$A = (\neg p_1 \wedge p_2 \wedge \neg p_3) \vee (p_1 \wedge \neg p_2 \wedge \neg p_3) \vee (p_1 \wedge p_2 \wedge p_3).$$

Corollary 1.6.6 *Any wff is equivalent to a wff containing only $\neg, \wedge$ and to a wff containing only $\neg, \vee$. So any truth function can be realized by a wff containing only $\neg, \wedge$ or only $\neg, \vee$.*

Proof. Given a wff $A(p_1, \ldots, p_n)$, consider the truth function $f_A : \{0,1\}^n \rightarrow \{0,1\}$ and let $A'(p_1, \ldots, p_n)$ be a wff containing only $\neg, \wedge, \vee$ such that

$$f_A = f_{A'},$$

i.e.

$$A \equiv A'.$$

We can now systematically eliminate $\vee$ by using the equivalence

$$C \vee D \equiv \neg(\neg C \wedge \neg D) \tag{1.1}$$

to obtain an equivalent formula $A'' \equiv A'$ containing only $\neg, \wedge$. Similarly using

$$C \wedge D \equiv \neg(\neg C \vee \neg D) \tag{1.2}$$

we can find an equivalent formula $A''' \equiv A'$ containing only $\neg, \vee$. $\dashv$

Remark. Another way to prove this corollary is by using the equivalences (1.1) and (1.2) above, as well as

$$(C \Rightarrow D) \equiv (\neg C \vee D)$$

$$(C \Leftrightarrow D) \equiv (\neg C \vee D) \wedge (\neg D \vee C)$$

to systematically eliminate all connectives except $\neg, \vee$ or $\neg, \wedge$.

Example 1.6.7 Suppose A is

$$((p \Rightarrow q) \wedge \neg(r \Rightarrow s)) \vee (s \vee p),$$

and we want to find an equivalent wff involving only $\neg, \wedge$. We have the following equivalent wffs, successively:

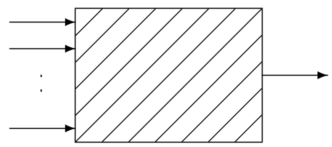
$$((p \Rightarrow q) \wedge \neg(r \Rightarrow s)) \vee (s \vee p) = A,$$

$$((\neg p \vee q) \wedge \neg(\neg r \vee s)) \vee (s \vee p),$$

$$(\neg(p \wedge \neg q) \wedge (r \wedge \neg s)) \vee \neg(\neg s \wedge \neg p),$$

$$\neg(\neg(\neg(p \wedge \neg q) \wedge (r \wedge \neg s))) \wedge (\neg s \wedge \neg p)).$$

Remark. If by a *(logic) circuit* we understand any device which accepts n binary inputs and produces a binary output, then the previous results show that any circuit can be built out of (*not*, *and*) or (*not*, *or*) gates only.



1.6.B Completeness of Binary Connectives

Definition 1.6.8 A set of connectives $C \subseteq \{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$ is called *complete* if any wff is equivalent to a wff whose only connectives are in C.

Examples 1.6.9

(i) $\{\neg, \wedge\}$, $\{\neg, \vee\}$ are complete sets, as seen in corollary 1.6.6.

(ii) $\{\neg, \Rightarrow\}$ is complete, because $(A \vee B) \equiv (\neg A \Rightarrow B)$.

(iii) $\{\neg, \Leftrightarrow\}$ is not complete: To see this notice that if $A(p_1, p_2)$ is any wff which only contains $\neg, \Leftrightarrow$ and a, b, c, d are the values of A under the four possible truth assignments to the variables p_1, p_2, then viewing a, b, c, d as members of $\mathbb{Z}_2$ we claim that

$$a + b + c + d = 0.$$

We can prove this by induction on the construction of A. If $A = p_1$ or $A = p_2$, then this is true since $a + b + c + d = 1 + 1 + 0 + 0 = 0$. If $A = \neg B$ and a, b, c, d are the values of B, then the values of A are $1 - a, 1 - b, 1 - c, 1 - d$ and their sum is

$$4 - (a + b + c + d) = 0 - 0 = 0.$$

Finally if $A = (B \Leftrightarrow C)$ and a_1, b_1, c_1, d_1 and a_2, b_2, c_2, d_2 are the corresponding values of B, C (i.e., corresponding to the same truth assignments to the variables p_1, p_2), then the truth values of A are $a_1 \Leftrightarrow a_2 = 1 - (a_1 + a_2)$, $1 - (b_1 + b_2)$, $1 - (c_1 + c_2), 1 - (d_1 + d_2)$, whose sum is

$$4 - (a_1 + b_1 + c_1 + d_1 + a_2 + b_2 + c_2 + d_2) = 0 - 0 = 0.$$

Since the values of $p_1 \wedge p_2$ are 1, 0, 0, 0 whose sum is 1, it follows that $p_1 \wedge p_2$ cannot be equivalent to any wff built using only $\neg, \Leftrightarrow$, so $\{\neg, \Leftrightarrow\}$ is not complete.

Recall that we actually have 16 binary connectives. We can introduce a symbol for each one of them (as we have already done for the most common ones) and build up formulas using them. So we can generalize the preceding definition to say that any set C of binary connectives is *complete* if any wff is equivalent to one in which the only connectives are contained in C. (In terms of truth functions this simply means that every truth function can be expressed as a composition of the truth functions contained in C. To see this notice that if $*$ is a binary connective, then $f_{A*B}(\bar{x}) = *(f_A(\bar{x}),\ f_B(\bar{x}))$.)

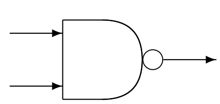
A single binary connective $*$ is complete if $C = \{*\}$ is complete, i.e., every wff is equivalent to one using only $*$. It turns out that the nand ($|$), and nor ($\downarrow$) connectives are complete. This is easily seen, because

$$\begin{aligned} \neg p &\equiv p|p \equiv p \downarrow p \\ (p \vee q) &\equiv (p|p)|(q|q) \\ (p \wedge q) &\equiv (p \downarrow p) \downarrow (q \downarrow q). \end{aligned}$$

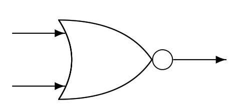
We will see in Assignment #3 that these are the *only* complete binary connectives.

Remark. In terms of circuits this implies that they can all be built using only *nor* or *nand* gates:

nand gate



nor gate



1.6.C Normal Forms

Definition 1.6.10 A wff A is in *disjunctive normal form* (*dnf*) if

$$A = A_1 \vee \cdots \vee A_n,$$

with

$$A_i = \ell_1^{(i)} \wedge \cdots \wedge \ell_{k_i}^{(i)},$$

and each $\ell_j^{(i)}$ a *literal*, i.e., p or $\neg p$ for some propositional variable p. We call A_i the *disjuncts* of A.

Examples 1.6.11

(i) $p, p \wedge q, (p \wedge q) \vee (\neg r \wedge s)$, $p \vee \neg q \vee (s \wedge \neg t \wedge u)$ are in dnf.

(ii) $p \Rightarrow q$ and $(p \vee (q \wedge r)) \wedge s$ are *not* in dnf.

Theorem 1.6.12 *For every wff A we can find a wff B in dnf such that $A \equiv B$.*

Proof. This follows from the proof of Theorem 1.6.4. If $A = A(p_1, \ldots, p_n)$ and A is contradictory, take $B = p_1 \wedge \neg p_1$. Otherwise, let

$$X = \{(\epsilon_1, \ldots, \epsilon_n) : \text{ the truth assigment } p_i \mapsto \epsilon_i \text{ satisfies } A\} \subseteq \{0,1\}^n,$$

and let

$$B = \bigvee_{(\epsilon_1,\ldots,\epsilon_n)\in X} \bigwedge_{i=1}^{n} \epsilon_i p_i,$$

where $\epsilon_i p_i = p_i$ if $\epsilon_1 = 1$ and $\neg p_i$ if $\epsilon_i = 0$. (The notation $\bigwedge_{i=1}^n$, like $\sum_{i=1}^n$, means to connect the specified values with $\wedge$, as i runs from 1 to n.) Then B is the desired formula. $\dashv$

Definition 1.6.13 A wff A is in *conjunctive normal form* (cnf) iff

$$A = A_1 \wedge \cdots \wedge A_n,$$

with

$$A_i = \ell_1^{(i)} \vee \cdots \vee \ell_{k_i}^{(i)}$$

and each $\ell_j^{(i)}$ a literal. We call A_i the *conjuncts* of A.

Example 1.6.14

$p, p \vee q, (p \vee q) \wedge (\neg r \vee s)$, $p \wedge \neg q \wedge (s \vee \neg t \vee u)$ are in cnf.

$p \Rightarrow q$, $(p \vee (q \wedge r)) \wedge s$, and $(p \wedge q) \vee r$ are *not* in cnf.

Corollary 1.6.15 *For every wff A we can find a wff B in cnf such that $A \equiv B$.*

Proof. Apply the preceding theorem to $\neg A$ to find a wff B' in dnf such that

$$\neg A \equiv B',$$

so

$$A \equiv \neg B'.$$

Now

$$B' = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{k_i} \ell_j^{(i)},$$

so, by de Morgan,

$$\neg B' \equiv \bigwedge_{i=1}^{n} \bigvee_{j=1}^{k_i} \neg \ell_j^{(i)}.$$

If some $\ell_j^{(i)}$ is of the form $\neg p$ replace $\neg \ell_j^{(i)}$ by p to obtain $\tilde{\ell}_j^{(i)}$. Otherwise let $\tilde{\ell}_j^{(i)} = \ell_j^{(i)}$. Thus

$$\neg B' \equiv \bigwedge_{i=1}^{n} \bigvee_{j=1}^{k_i} \tilde{\ell}_j^{(i)} = B,$$

where each $\tilde{\ell}_j^{(i)}$ is a literal, so B is in cnf and $A \equiv B$. $\dashv$

Remark. Given a wff A in dnf it is easy to check if it is satisfiable or not: If

$$A = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{k_i} \ell_j^{(i)},$$

then A is satisfiable iff one of the disjuncts is satisfiable, which is true iff there is $1 \leq i \leq n$ so that $\{\ell_j^{(i)} : 1 \leq j \leq k_i)$ does not contain both a propositional variable and its negation. Similarly given a wff A in cnf, it is easy to check if it is a tautology or not: it is a tautology iff all the conjuncts are tautologies, which happens iff for all $1 \leq i \leq n$, $\{\ell_j^{(i)} : 1 \leq i \leq k_i\}$ contains both a propositional variable and its negation.

Examples 1.6.16

(i) $(p \wedge q) \vee (p \wedge \neg r \wedge s) \vee (p \wedge s \wedge \neg p)$ is satisfiable.

(ii) $(p \vee q) \wedge (r \vee s \vee \neg r) \wedge (t \vee p \vee \neg s \vee \neg t)$ is not a tautology.

This is one reason that cnf and dnf are useful forms to be able to put a wff in. However it is not known how to transform efficiently any given wff A to an equivalent one in dnf (or cnf).

1.7 König's Lemma and Applications

We will now temporarily take a break from propositional logic and discuss a basic result in combinatorics, which we will use in the next section to prove an important result about propositional logic known as the *Compactness Theorem*.

1.7.A Graphs and Trees

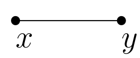
Definition 1.7.1 A (nondirected, simple) *graph* consists of a set of *vertices* V and a set $E \subseteq V^2$ of *edges* with the property that

$$(x, y) \in E \text{ iff } (y, x) \in E,$$

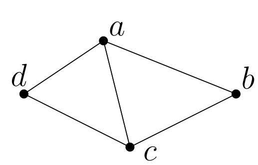
$$(x, x) \notin E.$$

If $e = (x, y) \in E$ is an edge, we say that x, y are *adjacent*.

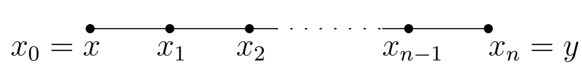
We represent a graph geometrically by drawing points for vertices and connecting adjacent vertices with lines, as shown.



Example 1.7.2 This is a graph with 4 vertices a, b, c, d and 5 edges (a, b), $(b, c), (c, d), (a, d), (a, c)$. (Formally, E has size 10.)



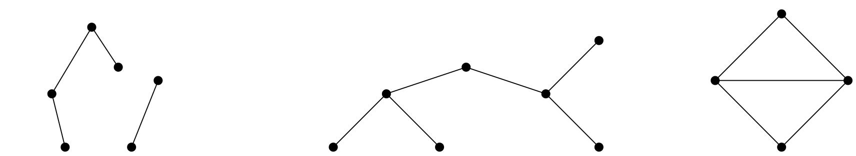
Definition 1.7.3 A *path* from $x \in V$ to $y \in V (x \neq y)$ is a finite sequence $x_0 = x, x_1, \ldots, x_n = x$ of distinct successively adjacent vertices (i.e., each (x_i, x_{i+1}) is an edge).



Definition 1.7.4 A graph is *connected* if for every $x, y \in V (x \neq y)$ there is a path from x to y.

Definition 1.7.5 A graph is a *tree* if for any $x \neq y$ there is a *unique* path from x to y.

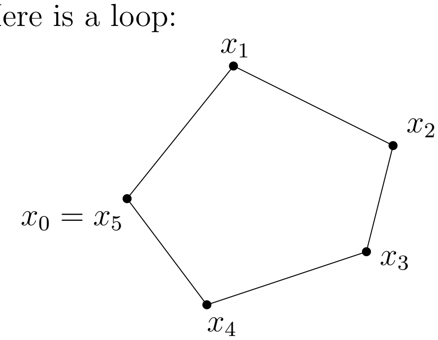
Examples 1.7.6
Here are some examples of graphs that are and are not trees.



not a tree (not connected) | tree | *not* a tree (has loops)

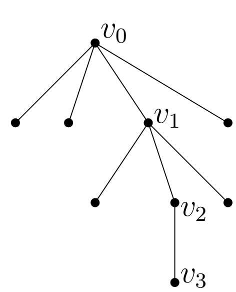
Equivalently, a connected graph is a tree exactly when it contains no *loops* (or *cycles*), where a *loop* is a sequence of successively adjacent vertices $x_0, x_1, \ldots, x_n$ with $x_0 = x_n$ and $x_0, \ldots, x_{n-1}$ distinct.

Example 1.7.7 Here is a loop:

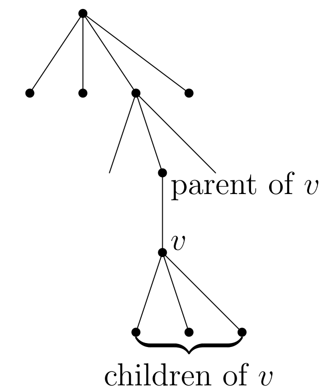


Definition 1.7.8 A *rooted* tree is a tree with a distinguished vertex called the *root*.

The root is usually denoted by v_0. For every vertex $v \neq v_0$ there is a unique path $v_0, v_1, \ldots, v_n = v$.

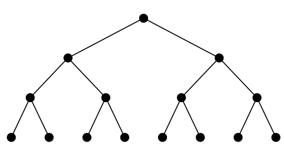


Definition 1.7.9 The *parent* of v is the vertex v_{n-1} and the *children* of v are all the vertices v' such that $v_0, v_1, \ldots, v_n, v_{n+1} = v'$ is the unique path from v_0 to v'.

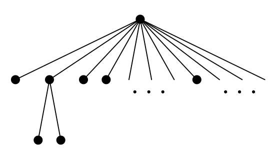


1.7.B König's Lemma

Definition 1.7.10 A tree is *finite splitting* if every vertex has only finitely many children (thus only finitely many adjacent vertices).

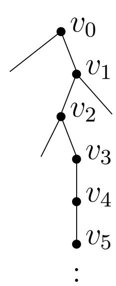


finite splitting

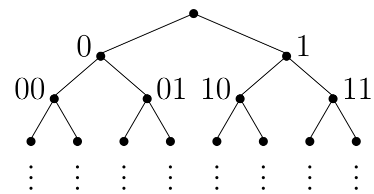


not finite splitting

Definition 1.7.11 An *infinite branch* of a rooted tree is an infinite sequence $v_0, v_1, v_2, \ldots,$ where v_{n+1} is a child of v_n.

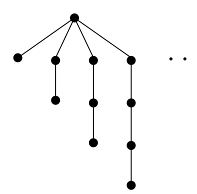


For example, if T is the infinite binary tree (in which each vertex has exactly two children), then the infinite branches correspond exactly to the infinite binary sequences $a_1, a_2, a_3, a_4, \ldots$ (each $a_i = 0$ or 1), where we interpret 0 as going left and 1 as going right.

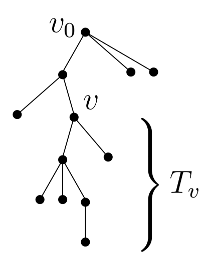


Theorem 1.7.12 (König's Lemma) *If a finite splitting tree has infinitely many vertices, then it has an infinite branch.*

Note that this fails if the tree is *not* finite splitting. Consider the following counterexample:



Proof. Let T be the given tree. For each vertex v of the tree, let T_v be the subtree of T consisting of v, the children of v, the grandchildren of v, etc., i.e., consisting of v and all its descendents, and all the edges connecting them. Denote by V the set of vertices of T, and by V_v the set of vertices of T_v.



We will use the following version of the *Pigeon Hole Principle*: If X is an infinite set and $X = X_1 \cup \cdots \cup X_n$, then some X_i, $1 \leq i \leq n$, is infinite.

Now, since T is finite splitting, v_0 has only finitely many children, say $c_1, \ldots, c_n$. Then $V = \{v_0\} \cup V_{c_1} \cup \cdots \cup V_{c_n}$, so some $V_{c_{i_1}}$ is infinite. Put $v_1 = c_{i_1}$. Let then $d_1, \ldots, d_m$ be the children of v_1. We have $V_{v_1} = \{v_1\} \cup V_{d_1} \cup \cdots \cup V_{d_m}$, so one of the V_{d_i}, say $V_{d_{i_2}}$ is infinite. Put $v_2 = d_{i_2}$, etc. Proceeding this way, we define an infinite path $v_0, v_1, v_2, \ldots$ (so that for each n, V_{v_n} is infinite). $\dashv$

1.7.C Domino Tilings

As an application of König's Lemma, we will consider the following tiling problem in the plane.

Definition 1.7.13 A *domino system* consists of a *finite* set $\mathcal{D}$ of domino types, where a *domino type* is a unit square with each side labeled.

Examples 1.7.14
Here are some example domino types:

$$\begin{array}{|ccc|}\hline & b & \\ a & & a \\ & c & \\ \hline\end{array} \qquad \begin{array}{|ccc|}\hline & b & \\ a & & c \\ & d & \\ \hline\end{array} \qquad \begin{array}{|ccc|}\hline & 0 & \\ 1 & & 2 \\ & 3 & \\ \hline\end{array}$$

Definition 1.7.15 A *tiling* of the plane by $\mathcal{D}$ consists of a filling-in of the plane by dominoes of type in $\mathcal{D}$, so that adjacent dominoes have matching labels at the sides where they touch. (Dominoes *cannot* be rotated.)

Example 1.7.16 Here is a tiling of the plane:

$$\begin{array}{c|ccc|ccc|ccc|c}
 & & \cdot\cdot & & & \cdot\cdot & & & \cdot\cdot & & \\ \hline
 & & b & & & b & & & d & & \\
\vdots & b & & a & a & & c & c & & b & \vdots \\
 & & c & & & d & & & a & & \\ \hline
 & & c & & & d & & & a & & \\
\vdots & c & & d & d & & b & b & & c & \vdots \\
 & & b & & & b & & & d & & \\ \hline
 & & \cdot\cdot & & & \cdot\cdot & & & \cdot\cdot & &
\end{array}$$

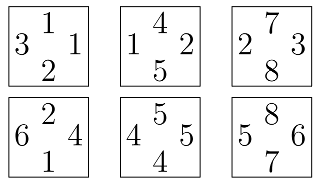
For this tiling, $\mathcal{D}$ contains at least

$$\begin{array}{|ccc|}\hline & b & \\ b & & a \\ & c & \\ \hline\end{array} \quad \begin{array}{|ccc|}\hline & b & \\ a & & c \\ & d & \\ \hline\end{array} \quad \begin{array}{|ccc|}\hline & d & \\ c & & b \\ & a & \\ \hline\end{array} \quad \begin{array}{|ccc|}\hline & c & \\ c & & d \\ & b & \\ \hline\end{array} \quad \begin{array}{|ccc|}\hline & d & \\ d & & b \\ & b & \\ \hline\end{array} \quad \begin{array}{|ccc|}\hline & a & \\ b & & c \\ & d & \\ \hline\end{array}$$

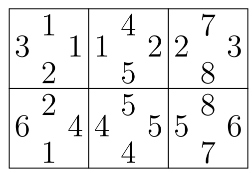
Problem. Given $\mathcal{D}$, can one tile the plane by $\mathcal{D}$?

Examples 1.7.17

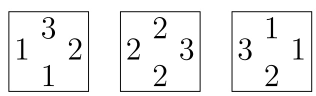
(i) Suppose $\mathcal{D}$ consists of



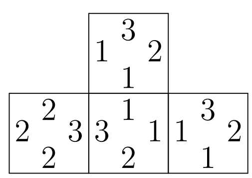
Then $\mathcal{D}$ can tile the plane, by repeating the following pattern:

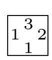
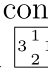
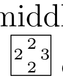


(ii) Suppose, on the other hand, that $\mathcal{D}$ consists of

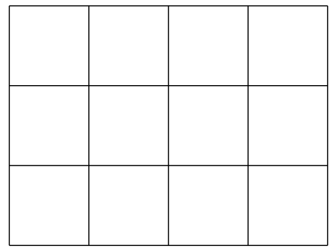


This $\mathcal{D}$ cannot tile the plane, as the following forced configuration shows:

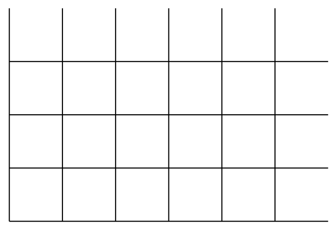


Starting from the  on the right, we are forced to form this configuration, which cannot be completed to a tiling. Starting from  in the middle, we are again forced to this configuration. Finally, starting from  on the left we are again forced to this configuration.

We can similarly define what it means to tile a rectangular finite region of the plane, like



or an infinite region, like



We just impose no restriction on the boundaries. Here is then a surprising fact:

Theorem 1.7.18 *For any given (finite) set $\mathcal{D}$ of domino types, $\mathcal{D}$ can tile the plane iff $\mathcal{D}$ can tile the upper right quadrant.*

Proof. We will actually show that the following are equivalent:

(i) $\mathcal{D}$ can tile the plane.

(ii) $\mathcal{D}$ can tile the upper right quadrant.

(iii) For each $n = 1, 2, \ldots,$ $\mathcal{D}$ can tile the $n \times n$ square.

It is clear that (i) implies (ii) implies (iii), so it is enough to show that (iii) implies (i).

So assume that for each $n = 1, 2, \ldots,$ $\mathcal{D}$ can tile the $n \times n$ square. We build a tree as follows:

The children of the root v_0 are all possible tilings of the 1×1 square, i.e., all domino types in $\mathcal{D}$. They are only finitely many. Let a be a typical one of them. Then its children are all the tilings of the 3×3 square consistent with a (viewed as being in the middle of the 3×3 square). Let b a typical one of them. Then its children are the tilings of the 5×5 square consistent with b, etc.

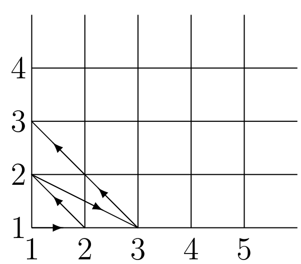
This tree is finite splitting, since for each fixed tiling of a $(2n+1)\times(2n+1)$ square there are only finitely many ways of extending it to a tiling of the $(2n+3)\times(2n+3)$ square, since $\mathcal{D}$ is finite. The tree is also infinite, since for each $n \geq 1$ there is some tiling of the $(2n-1)\times(2n-1)$ square, say u_n, and if we let $u_1, u_2, \ldots u_{n-1}$ be the tilings of the middle 1×1, 3×3, $\ldots, (2n-3)\times(2n-3)$ squares contained in u_n, then $u_0, u_1, \ldots, u_n$ are all vertices in this tree, so for each n the tree has at least n vertices, i.e., it is infinite. So by König's Lemma there is an infinite branch of the tree, say $u_0, u_1, u_2, \ldots u_n, \ldots$ This gives, in an obvious way, a tiling of the plane by $\mathcal{D}$. ⊣

Remark. It can be proved that there is *no* algorithm to check whether a given $\mathcal{D}$ can tile the plane or not.

This has the following interesting implication. A *periodic tiling* by $\mathcal{D}$ consists of a tiling of an $(n\times m)$-rectangle, so that the top and bottom labels match and so do the right and left ones, so by repeating it we can tile the plane.

Theorem 1.7.19 *There is a domino system $\mathcal{D}$ which can tile the plane, but has no periodic tiling.*

Proof. If this fails, then for every $\mathcal{D}$ which can tile the plane, there is a periodic tiling. Then we can devise an algorithm for checking whether a given $\mathcal{D}$ can tile the plane or not, which is a contradiction. First enumerate, in some standard way, all pairs (n, m), $(n \geq 1, m \geq 1)$ (e.g. as shown in the figure), say (n_i, m_i), $i = 1, 2, \ldots$ Then generate all tilings of the $(n_1 \times m_1)$-rectangle, the $(n_2 \times m_2)$-rectangle (if any), etc.



For each fixed (n_i, m_i) this is a finite process. Stop the process when some (n_i, m_i) is found for which either there is a periodic tiling of the $(n_i \times m_i)$-rectangle, or else there is *no* tiling of the $(n_i\times m_i)$-rectangle. In the first case, $\mathcal{D}$ can tile the plane and in the second it cannot. The only thing left to prove is that this process *terminates*, i.e. for some i this must happen. But this is the case, since either $\mathcal{D}$ can tile the plane and so by our assumption there is a periodic tiling (of some $(n_i \times m_i)$-rectangle)) or else $\mathcal{D}$ cannot tile the plane, so, by the proof of Theorem 1.6.2, $\mathcal{D}$ cannot tile the $(n_i \times m_i)$-rectangle for some $n_i = m_i$. $\dashv$

1.7.D Compactness of $[0, 1]$

As another application, we will use König's Lemma to prove that the unit interval [0,1] is *compact*. This means the following: Let $(a_i, b_i),\ i = 1, 2, \ldots$ be a sequence of open intervals such that

$$[0, 1] \subseteq \bigcup_i (a_i, b_i).$$

Then there are $i_1, \ldots, i_k$ such that

$$[0, 1] \subseteq (a_{i_1}, b_{i_1}) \cup \cdots \cup (a_{i_k}, b_{i_k}).$$

To prove this, consider the so-called *dyadic* intervals which are obtained by successively splitting [0,1] in half. They can be pictured as a binary tree.

We will prove that there is an n such that every one of the dyadic intervals at the nth level of the tree (i.e., those with denominators 2^{-n}) is contained in some (a_i, b_i) (perhaps different (a_i, b_i) for different dyadic intervals). This proves what we want.

The proof is by contradiction: If this fails, then for each n there is some dyadic interval at the nth level which is not contained in any (a_i, b_i). Consider then the subtree T of the tree of dyadic intervals, consisting of all vertices (i.e., dyadic intervals) I which are not contained in any (a_i, b_i). Then for each n, there is an $I = I_n$ at the n-th level belonging to T, and if $I_0 = [0, 1],\ I_1, \ldots, I_n$ is the unique path from the root to I_n, it is clear that $I_0 \supseteq I_1 \supseteq \cdots \supseteq I_n$, so no one of $I_0, I_1, \ldots, I_n$ are contained in any (a_i, b_i), so the tree T has at least n vertices for each n, thus it is infinite. It is clearly finite splitting. So by König's Lemma, it has an infinite branch $I_0, I_1, I_2, \ldots$ Then $I_0 \supseteq I_1 \supseteq I_2 \supseteq \ldots$ are closed intervals and the length of I_n is 2^{-n}, so there is a unique point $x \in \bigcap_n I_n$. Since $[0, 1] \subseteq \bigcup_i (a_i, b_i)$, for some i we have $x \in (a_i, b_i)$, and if n is large enough so that $\min\{x - a_i, b_i - x\} > 2^{-n}$, then $I_n \subseteq (a_i, b_i)$, a contradiction.