MATH 448 FALL 2011, COMPUTER ALGEBRA

SREEKAR M. SHASTRY

CONTENTS

1. STRUCTURAL INDUCTION

The principle of mathematical induction is no doubt known to the reader, not only as a proof technique, but also as a means of constructing objects of a combinatorial nature. The idea of the proof technique goes as follows. We want to know that a statement $P(n)$ is true for all $n \geqslant 0$. Suppose we have proved $P(1)$ and $P(i) \Rightarrow P(i+1)$. Then $P(n)$ is true because it appears in the chain of implications $P(1) \Rightarrow P(2) \Rightarrow \cdots \Rightarrow P(n) \Rightarrow \cdots$. The idea of the construction technique is exemplified by the factorial function; to define $n!$ for all $n \geqslant 1$ it suffices to define $1! = 1$ and $n! = n \cdot (n-1)!$. The underlying mechanism which makes induction work is that there is a family of statements (resp. objects) which are indexed by the positive integers $\mathbb{Z}_{>0}$ and we have an ordering on $\mathbb{Z}_{>0}$ which allows us to deduce the sought statement (resp. to construct the sought object) in terms of integers which are smaller with respect to the ordering.

We will often find ourselves in a more general situation. The set which will index our statements and objects will not be the set of integers, but some well-founded set Σ (definition below). Part of what makes this sort of proof difficult to understand and to devise is that, unlike for $\mathbb{Z}_{>0}$, the partial order on Σ which comes with the well-foundedness property is often not explicitly stated nor even particularly obvious, and since we only have a partial order, we may have many induction hypotheses to assume and many induction steps to prove.

Definition 1.1. Let Σ be a set together with a partial order $\prec$. Σ is called *well-founded* if every nonempty subset $S \subset \Sigma$ has a minimal element with respect to $\prec$ i.e.

$$\exists m \in S,\ \forall x \in S,\ x \not\prec m.$$

Principle of Structural Induction. Let Σ be a well-founded set. In order to show that a statement $P(x)$ holds for all $x \in \Sigma$ it suffices to show that if $P(y)$ is true for all $y \prec x$ then $P(x)$ is true.

Date: December 1, 2011.

1.2. To be convinced that this is a legitimate principle of proof, it suffices to consider a minimal counterexample to the family of statements $P(x)$ as x varies through Σ.

1.3. The statement $P(y)$ is true for all $y \prec x$ corresponds to the induction hypothesis and the proof that this hypothesis implies $P(x)$ corresponds to the the induction step in classical mathematical induction. But unlike the classical case there is no longer a single induction hypothesis step nor a single induction step. Likewise there may be many base cases; in terms of the partial ordering we can say that for each minimum with respect to $\prec$ there exists a base case to verify.

Remark 1.4. Structural induction is also known as Noetherian induction.

Example 1.5. If $n \geqslant 8$ then n can be written as $n = 3a + 5b$ with $a, b \geqslant 0$.
Proof. Write $P(n)$ for the statement that $n = 3a + 5b$ with $a, b \geqslant 0$.

The base cases are $P(8), P(9), P(10)$ which are proved, respectively, by $8 = 3 + 5, 9 = 3 * 3, 10 = 5 * 2$. Note very well that none of $P(i)$ is true for $i = 0, 1, 2, 4, 7$.

Suppose that $n \geqslant 10$ and that $P(11), \ldots, P(n)$ are all true. We shall prove $P(n+1)$. Since $n-2 \geqslant 8$ we have by assumption that $P(n-2)$ is true, i.e. $n-2 = 3a + 5b$ for some $a, b \geqslant 0$. Thus $n + 1 = 3(a + 1) + 5b$ with $a + 1, b \geqslant 0$, as required. ///

This is not a classical mathematical induction because we required multiple induction hypotheses; the preceeding proof is a structural induction with respect to the well-founded set $\{n \in \mathbb{Z}_{>0} : n \geqslant 8\}$.

Example 1.6. We recursively define a tree. A tree T is

$$T := \begin{cases} \text{a single vertex called the } \textit{root}, \text{ or} \\ \text{a root } r, \text{ trees } T_1, \ldots, T_m, \text{ and an edge from } r \text{ to the roots of each } T_i. \end{cases}$$

We write $T = T(r)$ in the first case and $T = T(r, T_1, \ldots, T_m)$ in the second case.[1]

This definition provides us with a well-founded set over which we can perform structural induction. Namely, consider the set of all trees. We define a partial order by $T_\alpha \prec T_\beta$ if there exists a finite sequence $T_1, T_2, \ldots T_m$ such that $T_1 = T(r_1, T_\alpha, \ldots), T_2 = T(r_2, T_1, \ldots), \ldots, T_m = T(r_m, T_{m-1}, \ldots)$ and $T_m = T_\beta$.

The well-foundedness is clear; given any set S of trees, take for a minimal element the tree in S with the smallest number of nodes (there are many choices for the minimal element; we only need one of them). In fact, what we have shown with this definition is that any chain with respect to $\prec$ has a unique minimal element.

Now, to prove by structural induction a statement $P(T)$ is true for all trees T, we must first prove it for the base case $T(r)$ and assume the induction hypotheses $P(T')$ for all trees $T' \prec T$, i.e. for all trees T' smaller than T. In fact, since $T = T(r, T_1, \ldots, T_m)$ it suffices to assume $P(T_1), \ldots, P(T_m)$. Then we must show that $P(T)$ follows from these assumptions. Here is an example.

Every tree has one more node than it has edges.
Proof. Write $n(T)$ for the number of nodes in T and $e(T)$ for the number of edges. The statement $P(T)$ is $n(T) = e(T) + 1$.

[1] To be completely rigorous we should only define A-labelled trees where A is a given set; an element r which labels the root is then required to be an element of A. Without this requirement, one runs into the usual bevy of set theoretic paradoxes.

The base case is $P(T(r))$ in which we have $n(T(r)) = 1$ and $e(T(r)) = 0$. Now write $T = T(r, T_1, \ldots, T_m)$. We have

$$\begin{aligned} n(T) &= 1 + \sum_{i=1}^{m} n(T_i) \\ &= 1 + \sum_{i=1}^{m} (e(T_i) + 1) \\ &= 1 + m + \sum_{i=1}^{m} e(T_i) \\ &= 1 + e(T) \end{aligned}$$

where the second equality is comes from the induction hypotheses and the last equality follows from the construction of $T(r, T_1, \ldots, T_m)$. ///

Example 1.7. We would like to define numerical expressions of the form $2, 3, 2 + 3, 2 * 3, 2 \circledast 3, (2 + 3) \circledast 5, x + 2, x \circledast n, \ldots$ with the idea of writing a computer program which is capable of evaluating such expressions. Here, $n \circledast a = n^a$ signifies exponentiation.

An *expression* is defined recursively as follows. The basic expressions are numbers and letters. Intuitively, the letters stand for variables into which we may later substitute numbers. The recursive definition goes as follows:

If E and F are expressions then so are $E+F, E*F, E \circledast F,$ and (E).

Now that we have the definition of an expression, let us use structural induction to prove the following fact.

Every expression has an equal number of left and right parentheses.
Proof. Given an expression G, write $l(G)$ for the number of left parentheses of G and $r(G)$ for the number of right parentheses. Write $P(G)$ for the statement $l(G) = r(G)$.

For a basic expression in which G is just a number or a letter, $l(G) = r(G) = 0$.

Now we assume that $P(E), P(F)$ is true for all expressions E, F and must prove each of the following

$$P(E + F), P(E * F), P(E \circledast F), P((E))$$

since these are all of the ways that G could be built up from smaller expressions. In the first three cases we have $l(G) = l(E) + l(F) = r(E) + r(F) = r(G)$. In the last case we have $l(G) = l(E) + 1$ and $r(G) = r(E) + 1$. Thus in all cases we have proved that $l(G) = r(G)$. ///

Observe that unlike the previous example we did not explicitly state the well-founded set nor the partial order on it. Often in such proofs, one omits mention of those two pieces of information. Nevertheless, it is best to know how to construct the underlying well-founded set with its partial order since that can be crucial to discovering a structural induction proof.

2. Finite State Automata

In this section of the lecture notes, we closely follow chapter 1 of "Word Processing in Groups" by Epstein.

Some other references:

- ⋆ Hopcroft, Motwani, Ullman, Introduction to Automata Theory, Languages, and Computation (2e),
- ⋆ Kozen, Automata and Computability,

⋆ Sipser, Introduction to the Theory of Computation (2e).

Definitions 2.1. An *alphabet* A is simply a finite set. A *letter* is an element of A. A *string* over the alphabet is a finite sequence of letters, or equivalently, a map $\{1,\dots,n\} \to A$. If $n = 0$ the domain is the empty set $\varnothing$ and there is a unique such string, denoted ε and called the empty string. Typically, A will be clear from the context, but we will occasionally write ε_A instead of ε in case of ambiguity.

A nonempty string will be written by simply writing the succesive values left to right. Thus 00101110 is the representation of a map $\{1,\dots,8\} \to \{0,1\}$. If ω is a string $\{1,\dots,n\} \to A$ then we call n the *length* of ω, and denote it by $|\omega|$.

We write A^* for the set of all strings over A.

Definitions 2.2. A *semigroup* is a set S together with a map $\mu : S \times S \to S$, such that

$$\mu(\mu(a,b),c) = \mu(a,\mu(b,c)) \text{ for all } a,b,c \in S.$$

Usually we will write $a\cdot b$ or simply ab for $\mu(a,b)$. A *monoid* M is a semigroup with a distinguished element $1 \in M$ such that

$$1 \cdot a = a \cdot 1 = a \text{ for all } a \in M.$$

A *group* G is a monoid together with an additional map $i : G \to G$, also written $i(a) = a^{-1}$, such that

$$aa^{-1} = a^{-1}a = 1 \text{ for all } a \in G.$$

Examples 2.3. (a) The set $2\mathbb{Z} = \{\dots,-4,-2,0,2,4,\dots\}$ is a semigroup under multiplication and is a group under addition. It is not a monoid (and therefore not a group) under multiplication.

(b) The set $\mathbb{Z}_{>0} = \{1,2,3,\dots\}$ is a monoid but not a group under multiplication and a semigroup but not a monoid under addition.

(c) The set $\mathbb{Q}$ of rational numbers is a group under addition but only a monoid under multiplication. The set $\mathbb{Q}-\{0\}$ is a group under multiplication.

Definitions 2.4. Given two strings $\omega : \{1,\dots,n\} \to A$ and $\tau : \{1,\dots,m\} \to A$, the *concatenation* $\omega\tau$ of ω and τ is defined to be the string $\{1,\dots,m+n\} \to A$ given by

$$(\omega\tau)(i) := \begin{cases} \omega(i) & \text{if } 1 \leqslant i \leqslant n, \\ \tau(i-n) & \text{if } n+1 \leqslant i \leqslant n+m. \end{cases}$$

In other words, $\omega\tau$ is simply the result of writing out ω and then writing out τ, left to right.

If we have $w = puq$ for some (possibly empty) strings w, p, q, u over A. We say that p is a *prefix* of w, that q is a *suffix* of w, and that u is a *substring* of w.

Given an integer $t \geqslant 0$ write $w(t)$ for that prefix of w of length t, or w itself if $t > |w|$.

Example 2.5. Under concatenation of strings, A^* is a monoid. We will see later that A^* is in fact the free monoid on the set A.

Definition 2.6. A *language* over A is a subset of A^*. A will usually be clear from the context and will not be mentioned. However, to be precise, we must take note of A. For instance, we must distinguish the the empty language over $\{x\}$ from the empty language over $\{x,y\}$.

2.7. For a language to be useful, there should be some way of understanding it. One way to interpret this requirement is to demand that there is some sort of machine capable of examining strings in A^* and responding with a "Yes" if and only if the string is in the language under consideration. In this case we say that the machine recognizes or accepts the language. In general in theoretical computer science, the machine may or may not ever give an answer if the string is not in the language. More on this point later.

Definitions 2.8. Let U and V be languages over A. Their *concatenation* UV is

$$UV := \{uv \in A^* : u \in U, v \in V\}.$$

The *star closure* (also known as the *Kleene closure*) of U is

$$U^* := \bigcup_{n=0}^{\infty} U^n,$$

where $U^0 := \{\varepsilon\}$ and $U^n = U^{n-1}U$ for $n > 0$. If $U = A$ then this agrees with our earlier definition of A^*.

Definition 2.9. Given a language A we form the alphabet A_{reg} as follows. Consider the set of five elements given by

$$\Psi = \{(,), *, \vee, \varepsilon_\varnothing\}.$$

We pronounce $\vee$ as "or," $*$ as "star," and $\varnothing$ as "null" in this context. Let

$$A_{\text{reg}} := A \sqcup \Psi$$

be the disjoint union of A and Ψ. Note well the distinct meanings of the symbols $\varepsilon_\varnothing \in \Psi$, $\varepsilon_A \in A^*$, and $\varepsilon_{A_{\text{reg}}} \in A^*_{\text{reg}}$.

The motivation for introducing these symbols goes as follows. The parentheses are to denote grouping, $*$ is to indicate repetition, $\vee$ is how we combine alternative patterns, and $\varepsilon_\varnothing$ matches the empty string ε_A of A.

We now inductively define a *regular expression* over A, which is a certain type of string over A_{reg}, and *regular languages* which are certain languages over A associated to regular expressions, as follows.

(1) $\varepsilon_\varnothing$ is a regular expression which gives rise to the language $L(\varepsilon_\varnothing) = \{\varepsilon_A\}$,

(2) a is a regular expression for $a \in A$ and $L(a) = \{a\}$,

(3) $()$ is a regular expression and $L(()) = \varnothing$.

(4) If r is a regular expression then (r) is a regular expression, and $L((r)) := L(r)$.

(5) If r is a regular expression then (r^*) is a regular expression which gives rise to the language $L((r^*)) = (L(r))^*$.

(6) If r and s are regular expressions then so is $(r \vee s)$ and the latter gives the language $L((r \vee s)) = L(r) \cup L(s)$.

(7) If r and s are regular expressions then so is (rs) and the latter gives the language $L((rs)) = L(r)L(s)$.

All regular expressions arise by being built up from the primitive expressions (1),(2),(3) by means of the rules (4),(5),(6),(7). If r is a regular expression and $w \in L(r)$ then we say that w *matches* r.

Note well: this recursive definition enables us to use structural induction over the set of regular expressions. See example 1.7.

Remarks 2.10. (1) The languages $L(a)$ and $L(\varepsilon_\varnothing)$ have one element each and the language $L(())$ has no elements.

(2) The assignment $r \mapsto L(r)$ from regular expressions to regular languages is far from injective.

(3) If $r = a \in A$ then $L((a^*)) = \{\varepsilon_A, a, a^2, a^3, \dots\}$.

(4) Often, we will omit the parentheses when writing out regular expressions, in analogy with familiar rules from arithmetic: $*$ is a sort of exponentiation which has a higher precedence than $\vee$, the latter being the analogue of multiplication.

Proposition 2.11. *The set of languages over A forms a monoid under concatenation. The identity element is the language whose only element is the empty string. The set of regular languages over A forms a submonoid. The set of finite languages (i.e. the languages consisting of finite subsets of A) form a submonoid of the monoid of regular languages.*

Proof. Unwind the definitions. ∎

Definition 2.12. A *finite state automaton* (or simply *automaton*) over A is a quintuple

$$(S, A, \mu, Y, s_0)$$

consisting of

(1) a finite set S, called the set of *states*,
(2) a finite set A, called the *alphabet*,
(3) $\mu : S \times A \to S$ is a mapping, called the *transition function*,
(4) $Y \subset S$ is a possibly empty set of *accept states*, and
(5) s_0 is the *start* or *initial* state.

We usually suppress μ from the notation and write $sx = \mu(s, x)$.

Remark 2.13. The letter Y stands for "Yes." The motivation for this definition is as follows. We are given a string $\omega = a_1 \dots a_n$ over A which we imagine to be printed left to right on a tape. The first letter tells us where to go from s_0: $s_1 = s_0 a_1$ The next letter tells us where to go from s_1: $s_2 = s_1 a_2$. And so on. If, after consuming all of w, the final state s_n is in Y, we interpret the machine as having said "Yes" in answer to a question about the given string w. We now proceed to make these ideas precise.

Definitions 2.14. Let $M = (S, A, \mu, Y, s_0)$ be a finite state automaton and let $\mathrm{End}(S)$ be the monoid of all set theoretic maps $S \to S$. The map $A \to \mathrm{End}(S)$ may be extended in a unique way to a monoid homomorphism

$$A^* \to \mathrm{End}(S).$$

We say that a string w is *recognized* or *accepted* by M if the corresponding element of $\mathrm{End}(S)$ takes s_0 to Y. The set

$$L(M) := \{w \in A^* : w \text{ is recognized by } M\}$$

is called the language recognized by M, or simply the *language of* M.

Remark 2.15. We may express the definition as follows. Let $a \in A$. Then we define a function $f_a : S \to S$ by $f_a(x) := \mu(x, a)$.

Some notation: for any mappings $f : X \to Y$ and $g : Y \to Z$ we write $f; g = g \circ f$ so that for all $x \in X$ we have $(f; g)(x) = g(f(x))$.

Now, let $w = a_1 \cdots a_n \in A^*$. Then we define a map $f_w : S \to S$ by

$$f_w = f_{a_1}; f_{a_2}; \cdots; f_{a_n}.$$

If we write $\mu(x, a) = xa$ then the above is simply expressing

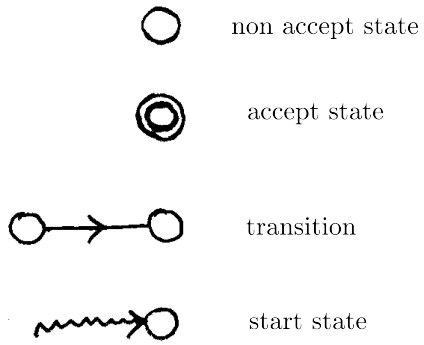
$$x(a_1 \cdots a_n) = ((((xa_1)a_2) \cdots a_n) \cdots)$$

and the associativity of A^* and of $\mathrm{End}(S)$ guarantees that it makes no difference how we arrange the parentheses. (By using the language of monoids

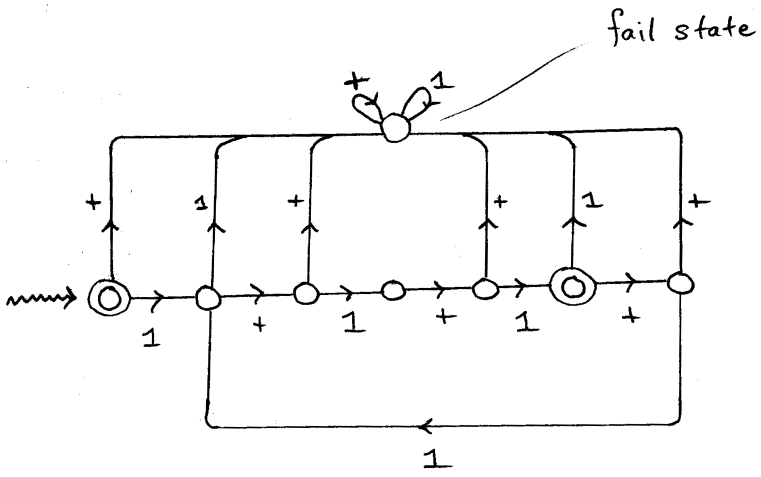
and their homomorphisms, we don't have to explicitly mention associativity.)

Very shortly we will study homomorphisms and the extension of maps from A to the free monoid generated by A.

2.16. We draw a finite state automaton by means of a finite directed graph. The set of vertices is the set S of states, and the set of directed edges is the set of pairs $\{(s, \mu(s,a)) : a \in A\}$. The subset of accept states, $Y \subset S$, is indicated by a double circle around that vertex and the initial state is indicated by a wavy arrow. See the figure.



Example 2.17. The alphabet in this case is $\{1, +\}$.



2.18. We can often transform a finite state automaton to a simpler one without changing the language it recognizes, as follows. (1) Remove all of the automaton's *inaccessible states*, those that cannot be reached from the start state by traversing a sequence of edges. (2) A state is called *live* or *dead* according as there is or isn't an accept state that can be reached from it. (Observe: it is possible for a state to be both live and inaccessible.) We collapse all of the automaton's dead states into a single *failure state* which is by definition a vertex which transitions to no states other than itself and is not an accept state.

An automaton is said to be *normalized* if the above procedure has already been applied to it. Henceforth we will only work with normalized automata.

2.19. A finite state automaton is also known as a *deterministic finite state automaton*. An *arrow* is a triple (s, α, t) where $s, t \in S$ and α is called the *label* of the arrow. α may come from any set, s is called the source of the arrow, and t the target of the arrow. An arrow labelled by α is also called an *α-transition.*

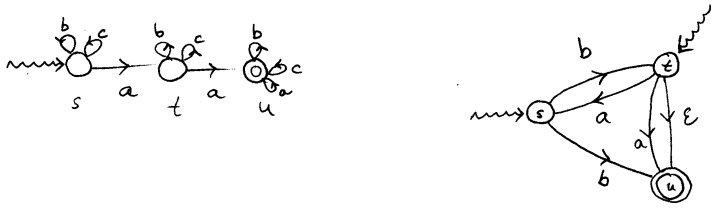
Definition 2.20. A *nondeterministic finite state automaton* is a quintuple

$$(S, A, \mu, Y, S_0)$$

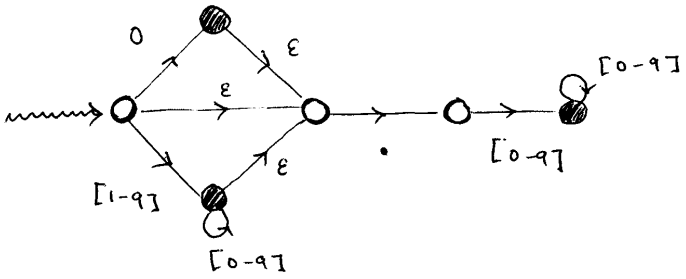
where
(1) the *alphabet* A is a finite set,
(2) the *state set* S is another finite set,
(3) $S_0 \subset S$ is the set of *start states*,
(4) $Y \subset S$ is the set of *accept states*
(5) μ is a set of arrows with labels in the set $A \sqcup \{\varepsilon\}$ (disjoint union; ε is to be thought of as an empty string).

2.21. Thus the differences between a nondeterministic automaton and an automaton (= deterministic automaton) are as follows.
(1) There are multiple points of entry: the single start state s_0 has been replaced by a set of start states S_0.
(2) It is possible to leap between states without consuming an element of A; this is what it means to have an arrow labelled by ε.
(3) The function $\mu : S \times A \to S$ has been replaced by a relation on the set of states. Recall that a relation is a subset of the set of pairs: $R \subset S \times S$. A function is a special type of relation which satisfies "the vertical line test" familiar from calculus. Thus, saying that μ is a set of arrows with labels in $A \sqcup \{\varepsilon\}$ means that we have a relation $R \subset S \times S$ and a map $R \to A \sqcup \{\varepsilon\}$ which gives the labels.

For concreteness, suppose that we have states $S = \{s, t, u\}$ and alphabet $A = \{a, b, c\}$. The figure on the left is what a deterministic automata looks like, and the figure on the right is what a nondeterministic automata could look like.



Moreover, we are to interpret the labels as telling us which of several possibilities we may take. For instance, if we are at state t, the label a on the figure on the right only tells us that there are two choices for the next step; thus the next step is not determined. In the figure on the left, the label a does determine our next step, from any given state.



Example 2.22. The alphabet for the nondeterministic automaton of this figure is

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, .\}$$

consisting of eleven symbols. The only nondeterminism in this example is the use of ε-transitions. The other source of nondeterminism, which is not present in this example, is the possible entry to the automaton from one of several start states — as with the ε-transition, that start state is not determined by the automaton, hence the adjective nondeterministic.

Definitions 2.23. A *generalized finite state automaton* is defined in the same way as a nondeterministic finite state automaton except that each edge is now labelled by a regular expression over A instead of simply by an element of A. Let M be a generalized finite state automaton. We seek to define the language $L(M)$ *recognized*, or *accepted* M.

A *path of arrows* in M is defined to be a sequence

$$P = (s_1, u_1, s_2, u_2, \ldots, u_n, s_{n+1})$$

where $n \geqslant 0$ is the called the *length* $\ell(P)$ of the path and each u_i is the label of an arrow (s_i, u_i, s_{i+1}) with source s_i and target s_{i+1} as defined earlier. s_1 is called the *source* and s_{n+1} the *target* of the given path of arrows. Given a path P we will also write $s(P)$ and $t(P)$ for the source and target of P, respectively. Now, each u_i is given by a regular expression. Let w be the concatenation of these regular expressions, $w = u_1 \cdots u_n$. Thus w is a regular expression, which we call the *label* of the given path, and it has an associated regular language $L(w)$. Finally we define $L(M) \subset A^*$ by

$$L(M) := \bigcup_{P \in \mathscr{P}(S_0, Y)} L(w(P))$$

where $\mathscr{P}(S_0, Y)$ is the set of paths with source in S_0 and target in Y, S_0 and Y came with the given automaton M, and $w(P)$ is the label associated to the path P.

The foregoing equally well defines the language accepted by a nondeterministic finite automaton, where we regard each element of the A as a regular expressions over A. Put differently, a nondeterministic finite automaton is also a generalized finite automaton in a natural way.

Definition 2.24. Given a nondeterministic automaton M with state set S we define the *ε-topology* on S as follows. Let $x \in S$ and put

$$Z_x := \left\{ t(P) : P \text{ is a path of arrows s.t. } s(P) = x \text{ and } w(P) = \varepsilon^{\ell(P)} \right\}$$

and for $E \subset S$ put

$$Z_E := \bigcup_{x \in E} Z_x.$$

Considering paths of length zero, it is clear that $E \subset Z_E$. Since S is a finite set, this definition realizes all closed sets in a topology on S.

Theorem 2.25 (Kleene, Scott, Rabin)**.** *Let A be an alphabet. The following conditions on a language over A are equivalent.*

(a) The language is recognized by a finite state automaton.
(b) The language is recognized by a nondeterministic finite state automaton.
(c) The language is recognized by a generalized finite state automaton.
(d) The language is defined by a regular expression.

Proof. We have (a) $\Rightarrow$ (b) $\Rightarrow$ (c) by definition.

We now show that (c) $\Rightarrow$ (d). We are given a generalized finite state automaton with states S, start states S_0, and accept states Y. Given a path of arrows $(s_1, u_1, \ldots, u_n, s_{n+1})$, say that it visits the states $s_2, \ldots, s_n$ (observe that the path needn't visit its own source and target). Let $s_1, s_2 \in S$ and $X \subset S$ and define $L(s_1, s_2, X) := \bigcup L(w(P))$ where the P range over those paths with source s_1, target s_2, and visiting only states in X. We shall show by induction on $\#X$ that $L(s_1, s_2, X)$ is defined by a regular expression.

First suppose that $X = \varnothing$ so that $L(s_1, s_2, X)$ is simply the finite set of arrows with source s_1 and target s_2. Let $r_i (i = 1, \ldots, k)$ be the label of the ith arrow; r_i is a regular expression. Then, recalling that $\vee$ of regular expressions corresponds to $\bigcup$ of languages, we have

$$L(s_1, s_2, X) = \begin{cases} L(r_1 \vee r_2 \vee \cdots \vee r_k) & \text{if } k > 0 \\ \varnothing & \text{if } k = 0 \text{ and } s_1 \neq s_2 \\ \{\varepsilon\} & \text{if } k = 0 \text{ and } s_1 = s_2 \end{cases}$$

so that $L(s_1, s_2, \varnothing)$ is indeed defined by a regular expression.

Now suppose that $X \neq \varnothing$ and let $s \in X$. Put

$$\begin{aligned} L_1 &= L(s_1, s_2, X - \{s\}) \\ L_2 &= L(s_1, s, X - \{s\}) \\ L_3 &= L(s, s, X - \{s\}) \\ L_4 &= L(s, s_2, X - \{s\}). \end{aligned}$$

Then we have that

$$L(s_1, s_2, X) = L_1 \cup (L_2 L_3^* L_4).$$

The induction hypothesis guarantees us that each of the terms on the rhs is defined by a regular expression, and thus so too is $L(s_1, s_2, X)$. To complete the proof that (c) implies (d), observe that we have

$$L(M) = \bigcup_{\substack{s_1 \in S_0, \\ s_2 \in Y}} L(s_1, s_2, S).$$

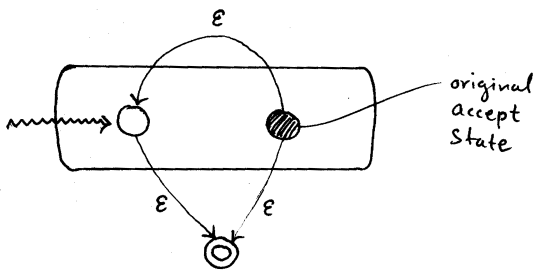
Thus $L(M)$ is a finite union of languages each defined by a regular expression and we are done.

We now seek to prove (d) $\Rightarrow$ (b). We shall use structural induction to construct a nondeterministic finite state automaton M which accepts the same language as the given regular expression R. In all cases, let the automaton M have as start states S_0 the singleton set $\{s_0\}$ and as accept states Y the singleton set $\{y\}$ with $s_0 \neq y$.

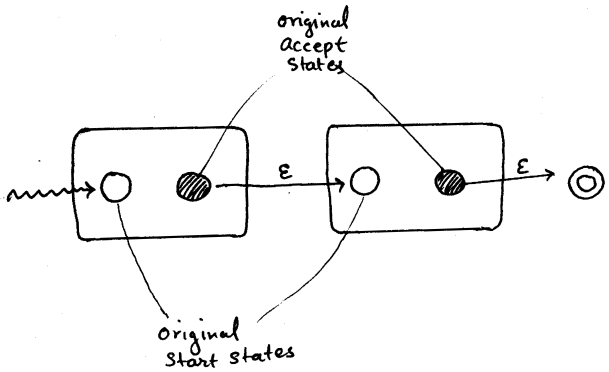
If the regular expression if $()$ then the associated language is the empty language and the automaton has only the start state and accept state as vertices, with no arrows at all.

If the regular expression is a single symbol $x \in A \cup \{\varepsilon\}$ then the automaton consists of a single arrow labeled by x going from the start state to the accept state. The start and accept are the only states in this case as well.

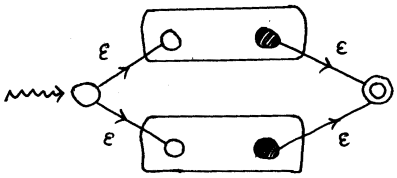
If r is a regular expression corresponding to the machine M (this is a structural induction hypothesis) then we may construct a machine corresponding to r^* as follows. Create a single new ε-arrow from the accept state of M to the start state.



If r_1 and r_2 are two regular expressions accepted by nondeterministic automata M_1 and M_2 (again this is a structural induction induction hypothesis) then the following figure gives the automata which accepts the concatenation $r_1 r_2$.



If r_1 and r_2 are two regular expressions corresponding to M_1 and M_2 (structural induction hypothesis) then we may construct an automaton which accepts $r_1 \vee r_2$ as follows. Draw the two automata M_1 and M_2 side by side, create a new start state and accept state, and draw ε-arrows from the new start states to the old ones, and from the old accept states to the new ones. Tracing through the figure shows that a string is accepted by the new automaton if and only if it was accepted by M_1 or M_2; this is what it means to be in the union $L(r_1) \cup L(r_2) = L(r_1 \vee r_2)$.



This completes the proof that (d) $\Rightarrow$ (b).

Finally, we must prove that (b) $\Rightarrow$ (a). Let S be the state set of the given nondeterministic automaton M. If $X \subset S$, we consider the closure $\overline{X}$ of X with respect to the ε-topology. We define the required deterministic automaton M' as follows. The state set S' is the set of all ε-closed subsets of S. Thus the new automaton has as states *collections* of states from the old automaton. If X is a state in M' and $x \in A$ then we put $\mu(X, x) :=$ the ε-closure of the set of all targets of arrows in M with source in X and labelled with x. The new start state is the ε-closure of the original set of start states and the new accept states are those which contain an old accept state.

To see that the new deterministic automaton M' accepts the same language as the given nondeterministic automaton M, consider a string accepted by M. This string corresponds to a particular path P in M. We must show that P uniquely corresponds to a path P' in M'. P gives rise to such a P' for the following reasons:

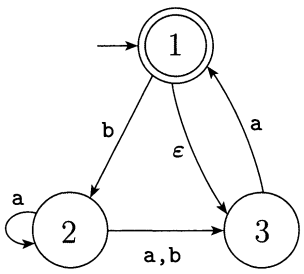
(1) the ambiguity in the start state goes away by definition of the start state of M',

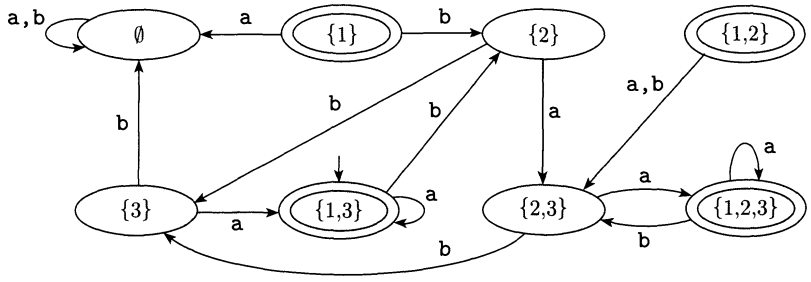
(2) the transition *relation* in M becomes a transition *function* in M' by the definition of $\mu(X, x)$, possibly with ε-transitions, but then

(3) the ε-transitions go away because μ is defined by means of the closure with respect to the ε-topology.

Upon passing from P' back to P it is clear that $P \leftrightarrow P'$ is a bijection. ■

Example 2.26. This example is taken from Sipser, page 57ff. The first figure is a nondeterministic automaton and the second figure is the corresponding deterministic automaton.





Theorem 2.27 (reversal)**.** *If* L *is a regular language over an alphabet* A *then the language consisting of the strings in* L *written in the reverse order is also regular.*

Proof. Let M be a nondeterministic automaton accepting L and form the automaton M' from M by reversing all arrows and interchanging the set of start states with the set of accept states. We have crucially used the definition of nondeterminism; there must be allowed more than a single start state for this construction to work. ■

Theorem 2.28 (Myhill-Nerode)**.** *Given a language* L *over an alphabet* A *consider the following equivalence relation on* A^**. We define* $w \sim w'$ *iff*

$$\forall u \in A^* : wu \in L \Leftrightarrow w'u \in L.$$

Then L *is regular iff* A^* *is partitioned into only finitely many equivalence classes under this relation.*

Proof. First suppose that the set of equivalence classes is finite. Define an deterministic automaton M with state set the set of equivalence classes. Write $[w]$ for the equivalence class, and also state of M, corresponding to $w \in A^*$. Put $[w]x := [wx]$. We define the start state to be $[\varepsilon]$ and the accept states are the $[\ell]$ such that $\ell \in L$. Then M accepts L.

Conversely, suppose that L is regular. Then it is given by some deterministic automaton M, and we assume without loss that M has no inaccessible states. Given a state s, choose a string $w_s \in A^*$ which takes the start state to s, and then consider the equivalence class of w_s. This class is independent of the choice of w by the definition of the equivalence relation, so that the map $s \mapsto [w_s]$ is well defined. Now, given $v \in A^*$, put $s := \mu(s_0, v)$. Then we have that $v \sim w_s$ i.e. $[v] = [w_s]$. Thus we have produced a surjective map from the finite set of states to the set of equivalence classes. ■

Remark 2.29. The first part of the preceeding proof shows that for a given regular language L there is a unique minimal deterministic automaton accepting it.

Definition 2.30. If L is a language over A, the *prefix closure* of L is the set

$$L_{\text{pre}} := \{w \in A^* : w \text{ is a prefix of some } \ell \in L\}.$$

If $L = L_{\text{pre}}$ then we say that L is *prefix closed*. Note that since the empty string ε is a prefix of any string, it follows that a prefix closed language must contain the empty string.

Theorem 2.31 (prefixes)**.** *Let* L *be a regular language. Then the prefix closure of* L *and the largest prefix closed sublanguage of* L *are both regular languages.*

Proof. Let M be a deterministic automaton accepting L. As usual, we assume that M is normalized. Now construct M' by turning every non-failure state of M into an accept state. Then M' accepts precisely the prefix closure of L.

Next, construct M'' from M by removing all non-accept states and all arrows either into or out of such a non-accept state. (In case the start state was not an accept state, then M'' has no start state; but in this case, the nullstring is not in L so that L has no prefix closed subsets.) Then M'' accepts the largest prefix closed sublanguage of L. ■

Definitions 2.32. Let A and B be alphabets and let $f : A \to B$ be a map. Then there exists a unique extension of f to a monoid homomorphism $f^* : A^* \to B^*$. (We will discuss this in much greater detail in the sequel.) Given a language L_A over A, the *image* of L_A under f is defined to be $f^*(L_A)$. If L_B is a language over B then the *inverse image* or *preimage* of L_B is defined to be $(f^*)^{-1}(L_B)$.

Now suppose that f is a map from A to the set of regular expressions over B. In this case, f is called a *substitution.* To define the image of L_A under f we proceed as follows. Extend f to a monoid homomorphism from A^* to the set of regular languages over B and put

$$f(L_A) := \bigcup_{u \in L_A} f(u).$$

Lemma 2.33. *Let L_A be a regular language over A and L_B a regular language over B. Given $f : A \to B$ or $f : A \to B^*$ we have that $f(L_A)$ resp. $f^{-1}(L_B)$ are regular languages over B resp. A. Given a substitution f, the image $f(L_B)$ is a regular language over B.*

Proof. Let M_A be a deterministic automaton accepting L_A. Replace each arrow in M_A with an arrow labelled by $f(a)$. This gives a new automaton over B which accepts $f(L_A)$; note well that this new automaton is generalized. But that is sufficient to prove regularity of the image.

Let M_B be a deterministic automaton over B accepting L_B. Define a deterministic automaton over A with the same states as M_B. Given a state s and $a \in A$, put $sa := sf(a)$. This new automaton accepts $f^{-1}(L_B)$. ■

Theorem 2.34 (pumping lemma)**.** *Let L be a regular language. Then there is a number $n > 0$ such that any $x \in L$ of length $\geqslant n$ is of the form $x = uvw$ with $|v| > 0$ and $uv^i w \in L$ for all $i \geqslant 0$.*

Proof. Let n be the number of states in a deterministic automaton accepting L. If an accepted string x has length $\geqslant n$ then it must visit some state twice (pigeonhole principle) and therefore is has a substring v whose corresponding path of arrows forms a loop. Traversing this loop i times gives the result. ■

2.35 (Regular Expressions in Practice)**.** Go to `http://regexpal.com/` and click the checkbox which says `^$ match at line breaks`. Copy any paste any text (for instance something from Wikipedia or some news website) into that site[2]. For example, I have input the following words from my `/usr/share/dict/words` to that site

```
craniums
crank
```

[2]You don't have to go to any website to try out regular expressions: the Unix command line tools `grep` or `ack` can be used to search text matching a given regular expression.

```
crank's
crankcase
crankcase's
crankcases
cranked
cranker
crankest
crankier
crankiest
crankiness
crankiness's
cranking
cranks
crankshaft
crankshaft's
crankshafts
cranky
crannied
crannies
cranny
```

Next, I have tried out the searches

```
c.*n
co
ank
c.*iu
c..nn
^c
n$
n*
.
..
....
```

In the practice of computer programming the notion of regular expression has a somewhat distinct, but closely related, meaning to the one we have defined in these notes. In computer programming, a regular expression is a pattern made from letters, numbers, and the special characters

```
^  $  .  *  (  )
```

This is a substantial simplification of the actual notion of regular expression, but it conveys the idea. The meanings of the special symbols are as follows. Any string of letters and numbers is a regular expression. Let `x` stand for a regular expression. Then `^x` accepts those strings which start with a string matching `x`. `x$` accepts those strings ending in `x`. Let `y` be another regular expression. Then `x.y` accepts all those strings with prefix matching `x`, followed by any single character, and with suffix matching `y`. The expression `.` alone matches any single character. `x*` matches any number of repetitions of strings matching `x`. Parentheses denote grouping as in an algebraic expression.

Now, given a regular expression in the sense of Definition 2.9, we see how to construct a regular expression in the computer programming sense. To construct a pattern which searches for those strings in $\mathrm{L}(\mathtt{r} \vee \mathtt{s})$ we simply input as a search pattern `rs`. For $\mathrm{L}(\mathtt{r}^*)$ we search for `r*`.

Thus we see that a regular expression in the sense of computer programming searches for those substrings of a given piece of text which are in the language accepted by a regular expression in the sense of Definition 2.9.

3. Presentations of Groups and Monoids

In this section we closely follow "Computation with Finitely Presented Groups" by Sims.

Proposition 3.1. *Let M be a monoid. The intersection of any nonempty family of submonoids of M is a submonoid of M. The intersection of any nonempty family of subgroups of M is a subgroup of M.*

Proof. Unwind the definitions. ∎

Definition 3.2. Let Y be a subset of a monoid M. Let S be the set of submonoids of M which contain Y. $S \neq \varnothing$ since $M \in S$. By the above

$$\mathrm{Mon}\langle Y\rangle := \bigcap_{N \in S} N$$

is a submonoid of M, called the *monoid generated by* Y.

3.3. It is clear that $\mathrm{Mon}\langle Y\rangle$ consists of all elements of M which may be expressed as a product $y_1 \cdots y_t$ with $y_i \in Y$ for all i. In this notation, if $t = 0$, we mean the element $1 \in M$.

Definitions 3.4. The *group of units* of a monoid M is the set

$$\{a \in M : \text{there exists } a^{-1} \in M\}.$$

It is a subgroup of M. Let Y be a subset of the group of units of M. We define

$$\mathrm{Grp}\langle Y\rangle := \bigcap_{\text{subgroups } H \subset M} H.$$

It is a subgroup of M. Writing $Y^{-1} := \{y^{-1} : y \in Y\}$ we have that

$$\mathrm{Grp}\langle Y\rangle = \mathrm{Mon}\langle Y \cup Y^{-1}\rangle.$$

Proposition 3.5. *If a group G is generated as a group by n elements then G is generated as a monoid by $n+1$ elements.*

Proof. Let $x_1, \ldots x_n$ generate G as a group so that $x_1, x_1^{-1}, x_2, x_2^{-1}, \ldots, x_n, x_n^{-1}$ generate G as a monoid. Put $y := x_1^{-1} \cdots x_n^{-1}$. Then we have

$$x_i^{-1} = x_{i-1}x_{i-2} \cdots x_1 y x_n x_{n-1} \cdots x_{i+1}$$

for each $i = 1, \ldots, n$. Thus $G = \mathrm{Mon}\langle x_1, \ldots, x_n, y\rangle$. ∎

Definitions 3.6. A monoid is *cyclic* if it is generated as a monoid by one element. A group is *cyclic* if it is generated as a group by one element. If $x \in G$ is an element of a group then the *order* of x is the cardinality of the subgroup $\mathrm{Grp}\langle x\rangle \subset G$ generated by x provided that that cardinality is finite. If it is not finite then x is said to be of infinite order.

Proposition 3.7. *Let M be a finitely generated monoid. Every generating set for M contains a finite generating set. Likewise for groups.*

Proof. Let X be a finite generating set for M and let Y be any generating set. Write each element $x \in X$ as a finite product of elements of Y; for each x we fix one such product decomposition. Let X' be the set of all those elements of Y which appear in the product of some $x \in X$ as chosen in the previous sentence. X' is the sought finite subset of Y which generates M. ∎

3.8. As we have seen in the section on finite state automata, the set A^* of all strings over an alphabet A is a monoid. We shall call it the *free monoid* generated by A. The following proposition justifies the terminology "free."

Proposition 3.9. *Let A be a set and let M be a monoid. For each set theoretic function $f : A \to M$ there is a unique extension of f to a homomorphism of monoids again denoted f, $f : A^* \to M$.*

Proof. Let $w = a_1 \cdots a_n \in A^*$ with each $a_i \in A$. Defining

$$f(w) := f(a_1) \cdots f(a_n)$$

does the job. ■

Proposition 3.10. *Let $f : M \to N$ be a homomorphism of monoids. If H is a submonoid of M and K is a submonoid of N then $f(H)$ is a submonoid of N and $f^{-1}(K)$ is a submonoid of M. If M and K are groups then $f^{-1}(K)$ is a subgroup of M.*

Proof. Unwind the definitions. ■

Definition 3.11. Let M be a monoid. A subset $I \subset M$ is an *ideal* if for all $x \in I, y \in M$ we have $xy \in I$ and $yx \in I$ i.e.

$$IM \subset I \text{ and } MI \subset I.$$

Examples 3.12. (1) Let A be a set. For any integer $k \geqslant 1$, the set

$$I_k := \{f \in \mathrm{End}(A) : \#f(A) \leqslant k\}$$

is an ideal of the monoid $\mathrm{End}(A)$.

(2) Let A be a set. For any $k \geqslant 0$ the set

$$I_k := \{w \in A^* : |w| \geqslant k\}$$

is an ideal of A^*.

(3) Let $U \subset A^*$. Then

$$I_U := \{w \in A^* : \text{ some element of } U \text{ is a subword of } w\}$$

is an ideal of A^*.

Definition 3.13. I is a *right ideal* if $IM \subset M$. I is a *left ideal* if $MI \subset I$. Thus an ideal is both a left ideal and a right ideal.

Examples 3.14. (1) Let $B \subset A$. Then

$$\{f \in \mathrm{End}(A) : f(A) \subset B\}$$

is a left ideal of $\mathrm{End}(A)$ and

$$\{f \in \mathrm{End}(A) : f \text{ is not injective}\}$$

is a right ideal of $\mathrm{End}(A)$.

(2) Let $U \subset A^*$. Then

$$\{w \in A^* : \text{ some element of } U \text{ is a prefix of } w\}$$

is a right ideal of A^*.

Definitions 3.15. Let I be an ideal of a monoid M. A *generating set* for I is a subset Y of M such that

$$I = MYM = \{xyz : x \in M, y \in Y, z \in M\}.$$

In particular, $Y \subset I$. A generating set for a right ideal J is a subset $T \subset J$ such that $J = TM$. Similarly for left ideals.

A *minimal generating set* for an ideal I is a set Y which does not properly contain any other generating set, i.e. is minimal with respect to the partial order given by inclusion. In general, an ideal may have no minimal generating sets, and it may have many. More can be said in the case of A^* as evinced by the following proposition.

Proposition 3.16. *Let* I *be an ideal of* X^* *and let*

$$U := \{w \in I : w \text{ does not contain an element of } I \text{ as a proper subword}\}.$$

Then U *generates* I *and* U *is a subset of every generating set for* I*. Thus* U *is the unique minimal generating set for* I*.*

Proof. Let $w \in I$. Choose a subword v of w of minimal length such that v is in I. Then $v \in U$ and w is in the ideal generated by U. Thus U is a generating set for I. Now let T be any generating set for I and let $u \in U$. There exist words $p, q \in A^*$ and $v \in T$ such that $u = pvq$. By the definition of U we have $p = q = \varepsilon$ and thus $u \in T$, as required. ∎

Proposition 3.17. *Let* $I_1 \subseteq I_2 \subseteq \cdots$ *be an infinite sequence of ideals in* A^**. Suppose that there is an integer* m *such that the minimal generating set of each of the* I_i *has no more than* m *elements. Then there is an integer* N *such that* $I_j = I_N$ *for all* $j \geqslant N$*.*

Proof. Consider the ideal

$$I := \bigcup_j I_j.$$

Let U be the minimal generating set for I from the previous proposition and suppose by way of contradiction that we could find distinct elements $u_1, \ldots, u_{m+1}$ in U. Then there is an index j_0 such that I_{j_0} contains all of the u_i. Let V be the minimal generating set for I_{j_0}. Then not all of the u_i are in V so that some u_i contains an element $v \in V$ as a proper subword. But $v \in I$ and thus $u_i \notin U$. Thus $\#U \leqslant m$ and U is finite. Thus there is an index N such that $I_N \supset U$. Finally, $I_N = I$ and $I_j = I_N$ for $j \geqslant n$. ∎

Definition 3.18. Let X be a subset of a group G. The *normal closure* of X is the smallest normal subgroup of G containing X and is denoted $\mathrm{Grp}\langle X^G \rangle$.

Definition 3.19. A group G satisfies the *ascending chain condition* on subgroups if there is no strictly increasing infinite chain of subgroups:

$$H_1 \subsetneq H_2 \subsetneq \cdots.$$

Proposition 3.20. *A group* G *satisfies the ascending chain condition iff all subgroups of* G *are finitely generated.*

Proof. Suppose that $H \subset G$ is a subgroup which is not finitely generated. Take $x_1 \in H$. Then $H_1 := \mathrm{Grp}\langle x_1 \rangle \neq H$ so we may take $x_2 \in H - H_1$. Then $H \neq H_2 := \mathrm{Grp}\langle x_1, x_2 \rangle \supsetneq H_1$. And so on. The $H_i := \mathrm{Grp}\langle x_1, \ldots, x_i \rangle$ form an infinite strictly increasing chain of subgroups.

Conversely, suppose that all subgroups of G are finitely generated. Let $H_1 \subset H_2 \subset$ be an infinite increasing chain of subgroups. Then $H = \bigcup H_i$ is a finitely generated subgroup and hence there exists an index n such that H_n contains this generating set. Thus $H = H_i$ for all $i \geqslant n$. ∎

Definition 3.21. Let M be a monoid. A *congruence* on M is an equivalence relation $\sim$ on M which is compatible with the multiplication law on M as follows: for all $x, y, m \in M$ we have

$$x \sim y \Rightarrow mx \sim my.$$

Proposition 3.22. *(1) Let* $f : M \to N$ *be a homomorphism of monoids. For* $x, y \in M$ *put* $x \sim y$ *iff* $f(x) = f(y)$*. Then* $\sim$ *is a congruence on* M*.*

(2) Conversely, given a congruence $\sim$ *on* M*, there exists a monoid* N *and a homomorphism* $f : M \to N$ *which recovers* $\sim$ *by the construction in (1).*

Proof. (1) Unwind the definitions.

(2) Let N be the set of equivalence classes of M under $\sim$. Write $[x] \in N$ for the class of $x \in M$. Then $[x][y] := [xy]$ with $[1]$ as identity endows N with a monoid structure because $\sim$ is a congruence (not merely an equivalence relation). N is called as the quotient monoid of M mod $\sim$, also written $N = M/\sim$, and the homomorphism f is $x \mapsto [x]$ is the quotient map. ■

Proposition 3.23. *Let* M *be a monoid and let* $S \subset M \times M$*. The intersection* $\sim$ *of all congruences containing* S *is a congruence.*

Proof. There exists one congruence containing S, namely $M \times M$ itself. Thus the interesection is over a nonempty indexing set. One need only observe that the property

$$x \sim y \Rightarrow mx \sim my$$

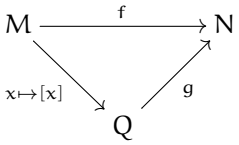
is preserved under the formation of intersection of subsets of $M \times M$. ■

Definitions 3.24. The congruence $\sim$ of the previous proposition is known as the *congruence generated by* S. A *right congruence* on M is an equivalence relation $\sim$ such that

$$x \sim y \Rightarrow xm \sim ym$$

for all $m \in M$. Likewise one defines a *left congruence.*

Proposition 3.25. *Let* M *be a monoid and* Q *be the quotient of* M *mod the congruence* $\sim$ *generated by* $S \subset M \times M$*. Let* $f : M \to N$ *be a monoid homomorphism such that* $f(s) = f(t)$ *for all* $(s, t) \in S$*. Then there is a unique* $g : Q \to N$ *such that*



commutes.

Proof. For $x, y \in M$, define $x \equiv y$ iff $f(x) = f(y)$. As we have seen, $\equiv$ is a congruence on M which contains S by hypothesis. Now, $xSy \Rightarrow f(x) = f(y) \Rightarrow x \equiv y$ and therefore $S \subset \equiv$. Since $\sim$ is generated by S it follows that $\sim \subset \equiv$. Thus we have $x \sim y \Rightarrow f(x) = f(y)$. Therefore we have a well defined map $g : Q \to N$ taking $[x]$ to $f(x)$ for all $x \in M$. To verify that g is a homomorphism, we check that $g([1]) = f(1) =$ the identity of N and that

$$g([x][y]) = g([xy]) = f(xy) = f(x)f(y) = g([x])g([y])$$

as required. ■

Remark 3.26. In category theoretic terminology we could say that Q is initial for all monoid homomorphisms out of M which respect S.

Definitions 3.27. Let X be a set and let $\mathcal{R}$ be a subset of $X^* \times X^*$. The monoid $\mathrm{Mon}\langle X|\mathcal{R}\rangle$ is defined to be the quotient monoid Q of X^* modulo the congruence generated by $\mathcal{R}$. The pair $(X, \mathcal{R})$ is said to be a *monoid presentation* for Q and for any monoid isomorphic to Q. The presentation is *finite* if both X

and $\mathcal{R}$ are finite. Q is then said to be *finitely presented.* If $X = \{x_1, \ldots, x_s\}$ and $\mathcal{R}$ consists of the pairs (U_i, V_i) with $1 \leqslant i \leqslant t$ then Q is sometimes written

$$\mathrm{Mon}\langle x_1, \ldots, x_s | U_1 = V_1, U_2 = V_2, \ldots, U_t = V_t \rangle$$

and the equations $U_i = V_i$ are called *defining relations* for Q. Under the natural map $[-] : X^* \to Q$, we have $[U_i] = [V_i]$ for each i. In general, given a monoid homomorphisms $f : X^* \to M$ and $U, V \in X^*$ such that $f(U) = f(V)$, we say that the relation $U = V$ holds in M relative to f. If $f(U) = 1 \in M$ then we say that $U = 1$ holds in M. The elements of $\mathrm{Mon}\langle X|\mathcal{R}\rangle$ are equivalence classes of words. The class containing U will be denoted $[U]$. In the sequel, we will not always use this notation to denote an equivalence class as the exposition is often clearer with less notation. The context should resolve any ambiguities.

Proposition 3.28. *Suppose that $\mathcal{R}, \mathcal{S}$ are two subsets of $X^* \times X^*$ with $\mathcal{R} \subset \mathcal{S}$. Then there is a surjective homomorphism from $\mathrm{Mon}\langle X|\mathcal{R}\rangle$ onto $\mathrm{Mon}\langle X|\mathcal{S}\rangle$.*

Proof. Let $\sim$ and $\equiv$ be the congruences on X^* generated by $\mathcal{R}$ and $\mathcal{S}$, respectively. Then $\mathcal{R} \subset \equiv$ so that $\sim \subset \equiv$. Therefore each $\sim$-class is contained in a unique $\equiv$-class. One verifies that the resulting map is a monoid homomorphism. ∎

Definition 3.29. Let X be a set and let $X^{\pm} := X \times \{1, -1\}$. Write x^{α} for $(x, \alpha) \in X^{\pm}$ and identify x with x^1. Write $X^{\pm *}$ for the monoid $(X^{\pm})^*$. Let $\mathcal{R} \subset X^{\pm *} \times X^{\pm *}$ be the set of pairs of the form $(x^{\alpha}x^{-\alpha}, \varepsilon)$ with $x \in X$ and $\alpha \in \{1, -1\}$. Put $F := \mathrm{Mon}\langle X^{\pm}|\mathcal{R}\rangle$. Write $[U]$ for the element of F containing the word U. F is called the *free group* generated by X.

Proposition 3.30. *The monoid F is a group. If $f : X \to G$ is any map of X into a group G then there is a unique homomorphism from F into G extending f.*

Proof. To see that F is a group it suffices to observe that F is generated as a monoid by $\{[x^{\alpha}] : x \in X, \alpha \in \{1, -1\}\}$ and then to observe that each such $[x^{\alpha}]$ is invertible:

$$[x^{\alpha}][x^{-\alpha}] = [x^{\alpha}x^{-\alpha}] = [\varepsilon] = 1 = [\varepsilon] = [x^{-\alpha}x^{\alpha}] = [x^{-\alpha}][x^{\alpha}].$$

To define the extension of f to F, we proceed as in the case of monoids but now we moreover demand that $f(x^{-1}) := f(x)^{-1}$. ∎

Definitions 3.31. We will write $\mathrm{FGRel}(X)$ for the relation $\mathcal{R}$ which defines a free group by $F := \mathrm{Mon}\langle X^{\pm}|\mathcal{R}\rangle$. The congruence on $X^{\pm *}$ generated by $\mathcal{R}$ is called *free equivalence.* A word U in $X^{\pm *}$ is called *freely reduced* if it contains no subword of the form $x^{\alpha}x^{-\alpha}$. If X is finite then $|X|$ is called the *rank* of F. Given a free group, it is not obvious that the rank depends only on the isomorphism class of F. We will see that this is the case in the next lemma. If $\mathcal{S} \subset X^{\pm *} \times X^{\pm *}$. By $\mathrm{Grp}\langle X|\mathcal{S}\rangle$ we shall mean $G := \mathrm{Mon}\langle X^{\pm}|\mathrm{FGRel}(X) \cup \mathcal{S}\rangle$. Then there is a surjective homomorphism from F onto G and we call $(X, \mathcal{S})$ a *group presentation* for G. If $\mathcal{S} = \{(U_i, V_i) : 1 \leqslant i \leqslant t\}$ then we also write $\mathrm{Grp}\langle x_1, \ldots, x_s | U_1 = V_1, \ldots, U_t = V_t \rangle$.

Lemma 3.32. *Let X be a finite set and let F be the free group generated by X. Then*

$$|\mathrm{Hom}(F, \mathbb{Z}/2)| = 2^{|X|}.$$

It follows that the rank of F depends only on the isomorphism class of F.

Proof. Exercise. ∎

Proposition 3.33. *Let* $\pi : X^{\pm *} \to F$ *be the natural map. Then* $\mathrm{Grp}\langle X|\mathcal{S}\rangle$ *is isomorphic to* F/N*, where* N *is the normal closure in* F *of the elements* $\pi(V)^{-1}\pi(U)$ *with* $(U, V) \in \mathcal{S}$*.*

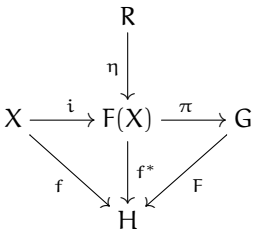
Proof. This follows from 3.25 once we note that the set theoretic quotient F/N is a group iff N is a normal subgroup. ■

Definition 3.34. Let $(X, \mathcal{R})$ be a monoid presentation for the monoid M and write $\sim$ for the congruence on X^* generated by $\mathcal{R}$. If $U, V \in X^*$ are such that $U \sim V$ then we say that (U, V) is a *consequence* of $\mathcal{R}$.

3.35. For the next few results we will follow Johnson, Presentations of Groups, second edition.

Proposition 3.36 (Substitution Test)**.** *Given* $G = \mathrm{Grp}\langle X|R\rangle$*, a group* H*, and a set map* $f : X \to H$*. Then* f *extends to a group homomorphism* $F : G \to H$ *if and only if* $f(r) = 1 \in H$ *for all* $r \in R$*. By this we mean take the group homomorphism* $f^* : F(X) \to H$ *and consider* $f^*(r) \in H$*; we demand that this be* $1 \in H$ *for all* r*.*

Proof. Consider the commutative diagram



with η, i inclusions and π the projection. The condition of the proposition is equivalent to $R \subseteq \ker(f^*)$. Since $\ker(f^*)$ is normal in $F(X)$ and $\mathrm{Grp}\langle R^G\rangle = \ker(\pi)$, the condition $R \subseteq \ker(f^*)$ is equivalent to the condition $\ker(\pi) \subseteq \ker(f^*)$.

Thus, given the above commutative diagram, we are reduced to showing that $\ker(\pi) \subseteq \ker(f^*)$ iff there exists $F : G \to H$. This follows from basic group theory (see for instance Dummit and Foote, Abstract Algebra, third edition, page 100). ■

Proposition 3.37. *If* $G = \mathrm{Grp}\langle X|R\rangle$ *and* $H = \mathrm{Grp}\langle Y|S\rangle$ *then their direct product* $G \times H$ *has the presentation*

$$\mathrm{Grp}\langle X, Y|R, S, [X, Y]\rangle$$

where $[X, Y] := \{x^{-1}y^{-1}xy : x \in X, y \in Y\}$*.*

Proof. Write $D := \mathrm{Grp}\langle X, Y|R, S, [X, Y]\rangle$. The substitution test implies that the set maps $X \to D$, $Y \to D$ induce homomorphisms $\varphi : G \to D, \psi : H \to D$. The relators $x^{-1}y^{-1}xy$ ensure that φ and ψ commute in D and thus we obtain a homomorphism $\alpha : G \times H \to D$ by $\alpha((g, h)) = \varphi(a)\psi(b)$.

On the other hand, the substitution test tells us that the map $X \cup Y \to G \times H$ which is defined by $x \mapsto (x, 1_H), y \mapsto (1_G, y)$ extends to a homomorphism $\beta : D \to G \times H$. Since $\alpha \circ \beta$ and $\beta \circ \alpha$ both fix generating sets in $G \times H$ respectively D, we have the sought isomorphism. ■

Proposition 3.38. *Let* $F = F(X)$ *and* $G = \mathrm{Grp}\langle X|R\rangle$ *and let* $w, r \in F$ *with* w *arbitrary and* $r \in \mathrm{Grp}\langle R^F\rangle - R$*. Let* y *be a symbol not in* X*. Then the natural*

maps

$$\alpha : X \to \mathrm{Grp}\langle X|R, r\rangle$$
$$\beta : X \to \mathrm{Grp}\langle X, y|R, y^{-1}w\rangle$$

both give rise by means of the canonical map $X \to G$ *to isomorphisms with source* G.

Proof. The substitution test guarantees that α and β extend to homomorphisms again denoted α, β with source G because the targets have the elements of R appearing as relations. To find the inverses, observe that the maps

$$\alpha' : X \to G \text{ and } \beta' : X \cup \{y\} \to G$$

which send y to $w \in G$ extend to homomorphisms out of G. Then α and β are isomorphisms because α, α' respectively β,β' are mutual inverses because generating sets are fixed in all cases. ■

3.39. The four isomorphisms of the previous proposition show how to go from a presentation $G = \mathrm{Grp}\langle X|R\rangle$ to a new presentation $G = \mathrm{Grp}\langle X'|R'\rangle$ of the same group. These are called the *Tietze transformations*. Write $F = F(X)$.

(1) $R+$, adjoining a relator:

$$X' = X, R' = R \cup \{r\}$$

where $r \in \mathrm{Grp}\langle R^F\rangle - R$.

(2) $R-$, removing a relator:

$$X' = X, R' = R - \{r\}$$

where $r \in R \cap$ (the normal closure of $R - \{r\}$).

(3) $X+$, adjoining a generator:

$$X' = X \cup \{y\}, R' = R \cup \{y^{-1}w\}$$

where $y \notin X$ and $w \in F$.

(4) $X-$, removing a generator:

$$X' = X - \{y\}, R' = R - \{y^{-1}w\}$$

where $y \in X, w \in (X - \{w\})^*$, and $y^{-1}w$ is the only member of R involving y.

Example 3.40. Consider the group

$$T = \mathrm{Grp}\langle x, y, x|x = yzy^{-1}, y = zxz^{-1}, z = xyx^{-1}\rangle.$$

Using the last relation to eliminate z we have the equivalent presentation

$$T = \mathrm{Grp}\langle x, y|xyx = yxy\rangle.$$

Putting $xy = a$ we obtain

$$y = x^{-1}a, ax = x^{-1}a^2.$$

Write $x = a^{-1}b$ we obtain

$$T = \mathrm{Grp}\langle a, b|a^3 = b^2\rangle.$$

Theorem 3.41. *Given two finite presentations of the same group, one can be obtained from the other by a finite sequence of Tietze transformations.*

Proof. Given

$$G = \mathrm{Grp}\langle X|R(X) = e\rangle = \mathrm{Grp}\langle Y|S(Y) = e\rangle$$

we suppose that

$$X = X(Y), Y = Y(X)$$

are systems of equations expressing the generators X in terms of the generators Y and vice versa. We apply the Tietze transformations to the first presentation as follows.

$$\begin{array}{lllllll}
X+ : X, Y & R(X) = e & Y = Y(X) & & & & \\
R+ : X, Y & R(X) = e & Y = Y(X) & X = X(Y) & & & \\
R+ : X, Y & R(X) = e & Y = Y(X) & X = X(Y) & R(X(Y)) = e & & \\
R- : X, Y & & Y = Y(X) & X = X(Y) & R(X(Y)) = e & & \\
R+ : X, Y & & Y = Y(X) & X = X(Y) & R(X(Y)) = e & Y = Y(X(Y)) & \\
R- : X, Y & & & X = X(Y) & R(X(Y)) = e & Y = Y(X(Y)) & \\
X- : Y & & & & R(X(Y)) = e & Y = Y(X(Y)) & \\
R+ : Y & & & & R(X(Y)) = e & Y = Y(X(Y)) & S(Y) = e \\
R- : Y & & & & & & S(Y) = e
\end{array}$$

■

Remark 3.42. The steps of this proof involved adding $|Y|$ generators and removing $|X|$ generators. The operations $R+$ has been used $|X| + |R| + |Y| + |S|$ times and $R-$ has been used $2(|R| + |Y|)$ times.

3.43. What the above theorem does is take a given isomorphism between two finitely presented groups and produce an algorithm transforming one set of generators and relations to the other. This is quite distinct from the isomorphism problem which we now describe. The determination of an algorithm for deciding whether two finitely presented groups in a class of groups are isomorphic is known as the isomorphism problem for that class of groups. For instance, the isomorphism problem for abelian groups is known to have a solution. The isomorphism problem for the class of all finitely presented groups is known not to have a solution.

Two other problems along the lines of the isomorphism problem are the word problem and the conjugacy problem; these are also undecidable in general.

3.44. Let us warn the reader against a common error. Given $G = \mathrm{Grp}\langle X|R(X)\rangle$ and new generators Y with $X = X(Y)$, it is not the case that $G = \mathrm{Grp}\langle Y|R(X(Y))\rangle$. As an example, take $G = \mathrm{Grp}\langle x|x^3\rangle, x = y^2$. Then $\mathrm{Grp}\langle Y|R(X(Y))\rangle$ is cyclic of order 6 and thus is not equal to G. The correct presentation is on the fifth line of the display in the proof of the theorem, namely

$$G = \mathrm{Grp}\langle Y|R(X(Y)), Y = Y(X(Y))\rangle.$$

Example 3.45. Let $m, n \in \mathbb{Z}_{>0}$ relatively prime so that we have $um + vn = 1$ for some $u, v \in \mathbb{Z}$. Write C_n for the cyclic group of cardinality n written multiplicatively. Starting from

$$C_{mn} = \mathrm{Grp}\langle a|a^{mn} = e\rangle$$

we seek a new presentation in terms of the generators $x = a^m, y = a^n$. We have that $x^u y^v = a$ and expressing each set of generators in terms of the

other (as in the notation $X = X(Y), Y = Y(X)$ in the proof of the theorem) we obtain

$$a = x^u y^v \text{ and } x = a^m, y = a^n$$

and the relations with respect to the generators x, y are

$$(x^u y^v)^{mn} = e, x = (x^u y^v)^m, y = (x^u y^v)^n.$$

Thus

$$C_{mn} = \text{Grp}\langle x, y | (x^u y^v)^{mn} = e, x = (x^u y^v)^m, y = (x^u y^v)^n \rangle$$

is the presentation in the new generating set.

We can deduce a new set of relations as follows. Recalling that $a = x^u y^v$ we have that x and y commute with each other since they are both powers of a, thus we must have the relation $[x, y] = e$ in the group presentation of C_n with respect to the generators x, y. Next we raise $(x^u y^v)^m = x$ to the nth power to obtain $e = (x^u y^v)^{nm} = x^n$ and likewise $y^m = e$. Using the new relations and three R+'s followed by three R−'s we go from

$$C_{mn} = \text{Grp}\langle x, y | (x^u y^v)^{mn} = e, x = (x^u y^v)^m, y = (x^u y^v)^n \rangle$$

to

$$C_{mn} = \text{Grp}\langle x, y | x^n = y^m = [x, y] = e \rangle.$$

Finally, we obtain $C_{mn} \simeq C_m \times C_n$ by invoking 3.37.

Definition 3.46. A *van Kampen diagram* for the presentation $G = \text{Grp}\langle X | R \rangle$ is a finite planar connected graph $\Gamma \subset \mathbb{R}^2$ the edges of which are directed and labelled by elements of X such that every face of Γ is homeomorphic to a disc the boundary of which (with respect to some starting point and orientation) belongs to R. One can then show that the boundary label of the graph Γ itself is a relation in G. We use van Kampen diagrams to visualize the deduction of new relations from old. Thus, for instance, if a relation appears visually as a consequence of other relations, we may omit it from the presentation.

There is a sort of converse. It can be shown that every relator in G is the boundary label of some van Kampen diagram Γ for G and it may moreover be assumed that Γ is reduced in the sense that no nontrivial circuit in Γ carries a label that reduces in $F(X)$ to the empty word (by means of cancellations of elements against their inverses).

The fact that the graph Γ embeds in the plane imposes strong conditions on Γ (for instance on the Euler characteristic $\chi(\Gamma)$). These conditions may be used to go from local properties of Γ to properties of the boundary $\partial\Gamma$ corresponding to group-theoretical properties of G.

This was a not completely rigorous definition of van Kampen diagram, but is sufficient to work with them in this course. The actual definition is much more involved and we refer the reader to `http://en.wikipedia.org/wiki/Van_Kampen_diagram`.

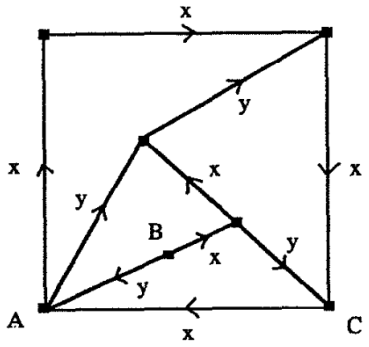
Example 3.47. The quaternion group of order 8 is given by

$$Q_4 = \text{Grp}\langle x, y | x^4 = e, x^2 = y^2, y^{-1}xy = x^{-1} \rangle.$$

We convert this presentation into relators (i.e. expressions of the form $r = 1$) a easy to use form by setting $r = x^4, s = x^2 y^{-2}, t = y^{-1}xyx$ we obtain the presentation

$$Q_4 = \text{Grp}\langle x, y | r = s = t = 1 \rangle$$

and the following van Kampen diagram.



Since the boundary label $r = x^4$ can be deduced from the internal faces, it follows that

$$Q_4 = \mathrm{Grp}\langle x, y | s = t = 1 \rangle.$$

Example 3.48. The group

$$F(2,4) = \langle a, b, c, d | ab = c, bc = d, cd = a, da = b \rangle$$

is generated by a and b since $c = ab$ and $d = bc = bab$. We claim that it is actually a cyclic group whose order divides 5 (therefore is of order 5 or 1). This follows from the following diagrams. The faces are labelled by

$$r_1 = abc^{-1}, r_2 = bcd^{-1}, r_3 = cda^{-1}, r_4 = dab^{-1}.$$

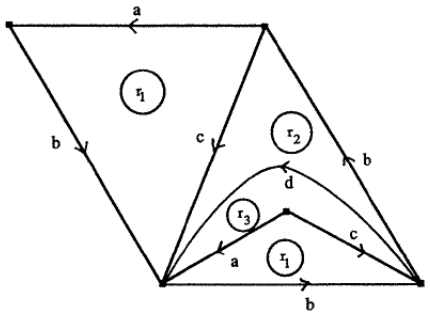


Fig. 11

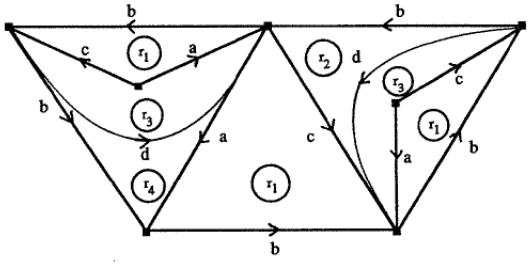


Fig. 12

Namely, the first figure shows that $a = b^{-3}$ which implies that the group is cyclic and the second shows that $b^5 = 1$ which implies that the generator of this cyclic group has order 1 or 5.

3.49. We will now investigate in detail the generators and relations of some important examples of groups. Typically the group G will be given by an

abstract definition which does not immediately lend itself to the determination of a presentation. The method by which we find a presentation goes as follows.

(1) Find a set of generators X of G.

(2) In terms of X find some relations $R = e$ that hold in G and conjecture that these are enough to define G.

(3) Show that the natural surjection (which exists because we know that X generates G)

$$\pi : \langle X|R\rangle \twoheadrightarrow G$$

is an isomorphism.

It is usually the last step which is the most difficult. For instance, it may be possible to bound $|G|$ from below by using properties of an action of G on some set and to bound $|\langle X|R\rangle|$ from above by looking at the presentation. If these bounds are equal and finite then π is an isomorphism. More generally, one may find a "normal form" for the elements of $\langle X|R\rangle$ i.e. write down a list L of elements of $(X^{\pm})^*$ and then to give a bijection between L and $\langle X|R\rangle$. Let us now look at some examples in detail.

Remark 3.50. Let us make explicit a notational convention which we have already used several times without comment. When the context clearly indicates whether we are working with groups or with monoids, we will simply write $\langle X|R\rangle$ to mean one of $\mathrm{Mon}\langle X|R\rangle$ or $\mathrm{Grp}\langle X|R\rangle$ according to the context.

Example 3.51 (The Quaternions)**.** Let us classify groups of order 8, as follows. First, if G is abelian then by the classification of finite abelian groups $G \simeq (\mathbb{Z}/2)^{\oplus 3}$ or $\mathbb{Z}/4 \oplus \mathbb{Z}/2$ or $G \simeq \mathbb{Z}/8$.

Suppose that G is nonabelian and let $x \in G$ be an element of maximal order m. Then $m = 2, 4$, or 8.

If $m = 2$ then G is abelian ($m = 2$ and $a, b \in G \Rightarrow a, b, ab$ are all of order 2 $\Rightarrow ba = a^2(ba)b^2 = a(abab)b = a(ab)^2b = ab$). If $m = 8$ then G is cyclic and therefore abelian. Thus, G nonabelian $\Rightarrow m = 4$.

Then $H := \langle x\rangle \lhd G$ since we know that a subgroup of index two is normal. Let $y \in G - H$. Then we have that $y^2, y^{-1}xy \in H$. It then follows by considering the order of group elements that

$$y^{-1}xy = x \text{ or } x^{-1} \text{ and } y^2 = e \text{ or } x^2.$$

(In detail, $y^{-1}xy$ has the same order in G as x thus it must be a generator of H since it is in H. Regarding y^2, if we know that it has order dividing 4 in H but if $y^2 = x$ or x^{-1} then G would be cyclic of order 8 with generator y. Therefore $y^2 = e$ or x^2.)

Now, if $y^{-1}xy = x$ then G is abelian and isomorphic to $\mathbb{Z}/4 \oplus \mathbb{Z}/2$. If $y^{-1}xy = x^{-1}$ and $y^2 = e$ then $G \simeq D_4$ is a dihedral group (we refer you to Dummit and Foote for more details). We are reduced the following. Is there a group of order 8 containing an element x of order 4 and an element $y \neq x$ with $y^2 = x^2$ and $y^{-1}xy = x^{-1}$ There is at most one such group since the information implies that all elements of G are of the form

$$\{x^i, x^iy : 0 \leqslant i \leqslant 3\}$$

and the multiplication table is given by

	x^j	x^jy
x^i	x^{i+j}	$x^{i+j}y$
x^iy	$x^{i-j}y$	x^{i-j+2}

where the powers of x taken with the understanding that x is of order 4. We must now show that this multiplication table actually defines a group. We could check this by hand but let us instead proceed as follows. We will construct a group of order 8 which has the above multiplication table; the point is that because of the way we will make the construction, we can see easily that the cardinality is 8, which is not at all clear from the above multiplication table.

Let $\mathbb{H}$ be the set of matrices in $\mathrm{GL}_2(\mathbb{C})$ of the form

$$\left\{\begin{pmatrix} z & w \\ -\overline{w} & \overline{z} \end{pmatrix}\right\}$$

One checks that if $A, B \in \mathbb{H}$ then so are $A+B, -A, AB,$ and A^{-1} (provided that $A \neq 0$ in the last case). Note that if $z = a + ib, w = c + id$ are both nonzero then likewise

$$\det A = z\overline{z} + w\overline{w} = a^2 + b^2 + c^2 + d^2 \neq 0.$$

Thus $\mathbb{H}$ is a division algebra over $\mathbb{R}$ (i.e. a noncommutative $\mathbb{R}$-algebra in which every nonzero element has a multiplicative inverse), known as the ring of quaternions. Writing

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, X = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix}, Y = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, Z = \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix}$$

then one can see using the coordinates that the elements

$$\{\pm I, \pm X, \pm Y, \pm Z\}$$

are 8 distinct elements of $\mathbb{H}$ and using matrix multiplication one can check that this set forms a group Q_4 of order 8. Finally the relations

$$X^4 = I, X^2 = Y^2, Y^{-1}XY = X^{-1}$$

all obtain in Q_4 so that Q_4 is isomorphic to the sought group of order 8:

$$Q_4 = \langle x, y | x^4 = e, x^2 = y^2, y^{-1}xy = x^{-1} \rangle.$$

Example 3.52 (The Symmetric Group)**.** Let

$$S_n := \mathrm{Sym}(\{1, 2, \ldots, n\})$$

be the symmetric group on n letters. To find a presentation for S_n we proceed in three steps.

(1) Let us first find a generating set. We know that every permutation can be written as a product of disjoint cycles so that the cycles generate S_n. Since any cycle may be expressed as a product of transposition

$$(x_1 x_2 \cdots x_t) = (x_1 x_2)(x_1 x_3) \cdots (x_1 x_t),$$

in fact the transpositions generate S_n. If (ij) is a transposition with $1 \leqslant i < j \leqslant n$ (this accounts for all transpositions since $(ji) = (ij)$) then we have

$$(ij) = (i\ i+1)(i+1\ i+2) \cdots (j-2\ j-1)(j-1\ j)(j-2\ j-1) \cdots (i+1\ i+2)(i\ i+1).$$

Therefore S_n is generated by the adjacent transpositions

$$\{(i\ i+1) : 1 \leqslant i \leqslant n+1\}.$$

(2) The adjacent transpositions satisfy the following relations:

$$\begin{aligned} (i\ i+1)^2 &= e, & 1 \leqslant i \leqslant n-1, \\ ((i\ i+1)(i+1\ i+2))^3 &= e, & 1 \leqslant i \leqslant n-2, \\ (i\ i+1)(j\ j+1) &= (j\ j+1)(i\ i+1), & 1 \leqslant i, j \leqslant n-1, |i-j| \geqslant 2. \end{aligned}$$

Therefore we conjecture that the following is a presentation of S_n:

$$G_n := \langle x_1, \dots, x_{n-1} | R, S, R \rangle$$

where

$$\begin{aligned} R &= \{x_i^2 : 1 \leqslant i \leqslant n-1\} \\ S &= \{(x_i\ x_{i+1})^3 : 1 \leqslant i \leqslant n-2\}, \\ T &= \{[x_i, x_j] : 1 \leqslant i < j-1 < n-1\}. \end{aligned}$$

Here, T expresses the fact that disjoint transpositions commute.

Now, by the above we know that there is a surjection $G_n \twoheadrightarrow S_n$ and we must show that it is an isomorphism.

(3) To show that it is an isomorphism it suffices to show that $|G_n| \leqslant n!$. We prove this by induction on n. If $n = 1$ then G_1 is the trivial group so there is nothing to prove. Assume now that $n \geqslant 2$ and that $G_{n-1} \leqslant (n-1)!$. Let H be the subgroup of G_n generated by $x_1, \dots, x_{n-2}$ and put

$$y_0 := e \text{ and } y_i := x_{n-1} \dots x_{n-i} \text{ for } 1 \leqslant i \leqslant n-1.$$

Let

$$A := \{hy_i : h \in H, 0 \leqslant i \leqslant n-1\} \subset G_n.$$

We shall show that $A = G_n$. Let us first show that

$$Ax_j \subset A \text{ for } 1 \leqslant j \leqslant n-1.$$

We have to consider the terms hy_ix_j in six cases.

(i) $i = 0, j < n-1 \Rightarrow hy_ix_j = hx_j \in Hy_0 \subset A$

(ii) $i = 0, j = n-1 \Rightarrow hy_ix_j = hex_{n-1} = hy_1 \in Hy_1 \subset A$

(iii)

$$\begin{aligned} i > 0, j > n-i \Rightarrow hy_ix_j &= h(x_{n-1} \cdots x_j x_{j-1} \cdots x_{n-i})x_j \\ &= hx_{n-1} \cdots x_j x_{j-1} x_j \cdots x_{n-i} && \text{by } T \\ &= hx_{n-1} \cdots x_{j-1} x_j x_{j-1} \cdots x_{n-i} && \text{by } R \text{ and } S \\ &= (hx_{j-1})x_{n-1} \cdots x_{n-i} && \text{by } T \\ &\in Hy_i \subset A \end{aligned}$$

(iv)

$$\begin{aligned} i > 0, j = n \Rightarrow hy_ix_j &= hx_{n-1} \cdots x_{n-i}x_{n-i} \\ &= hx_{n-1} \cdots x_{n-i+1} && \text{by } R \\ &= hy_{i-1} \in Hy_{i-1} \subset A \end{aligned}$$

(v)

$$\begin{aligned} i > 0, j < n-i-1 \Rightarrow hy_ix_j &= (hx_j)y_i && \text{by } T \\ &\in Hy_i \subset A \end{aligned}$$

(vi)

$$\begin{aligned} i > 0, j < n-i-1 \Rightarrow hy_ix_j &= (hx_j)y_i && \text{by } T \\ &\in Hy_i \subset A \end{aligned}$$

This shows that $Ax_j \subset A$ for all j. Now, by R we have

$$Ax_j^{-1} = Ax_j \subset A \text{ for } 1 \leqslant j \leqslant n-1.$$

Thus there is an injection from G_n to A, i.e. A consists of the elements of G_n written in "normal form" with respect to the subgroup G_{n-1}. Since the relations in G_n which involve $x_1, \dots, x_{n-2}$ are exactly those from G_{n-1} it

follows that the map $\varphi : G_{n-1} \to G_n$ given by $x_i \mapsto x_i$ is a group homomorphism. Since $\mathrm{Im}(\varphi) = H$ (by definition) and $|G_{n-1} \leqslant (n-1)!$ by the induction hypothesis, it follows that

$$|G_n| \leqslant |A| \leqslant n|H| \leqslant n!$$

as required.

4. Rewriting Systems

4.1. Many questions we would like to ask about finitely presented groups are known to not have general algorithmic solutions. In other words, it has been shown that it is not possible to write down an algorithm which solves the problem in question.

The most basic of these unsolvability results pertains to the word problem. Let $M = \mathrm{Mon}\langle X|\mathcal{R}\rangle$ be a finitely presented monoid. The word problem for M is to decide, given $U, V \in X^*$, whether U and V define the same element of M. Put differently, we must decide whether $U \sim V$ where $\sim$ is the congruence on X^* generated by $\mathcal{R}$. It has been shown that there is no general algorithm which will solve the word problem for all monoids, and moreover it has been shown there are specific monoids for which no solution to the word problem exists. Likewise for groups. See Chapter 12 of Rotman, "The Theory of Groups" (2e) for a proof of this fact.

Now, it may not be possible to *decide* whether two words U and V define the same element of M. However, if U and V do represent the same element, then it is possible to *verify* this fact because the definition of the congruence $\sim$ generated by $\mathcal{R}$ is explicit enough for us to simply write down all the words in the equivalence class containing U. This is the content of the next proposition.

Proposition 4.2. *Let A and B be words in X^*. Then $A \sim B$ iff there is a sequence of words*

$$A = A_0, A_1, \ldots, A_t = B$$

such that for $0 \leqslant i < t$ the words A_i and A_{i+1} have the form CPD and CQD respectively with (P, Q) or (Q, P) in $\mathcal{R}$.

Proof. Write $A \equiv B$ if there is such a sequence. Then one checks that $\equiv$ is an equivalence relation. If $(P, Q) \in \mathcal{R}$ then $P \sim Q$ so that $CPD \sim CQD$ for all $C, D \in X^*$. Thus $A \equiv B \Rightarrow A \sim B$, or in other words, we have $\equiv \subset \sim$.

Next, let us suppose that $A \equiv B$ and let $A_0, \ldots, A_t$ be the corresponding sequence. If U is a word then the sequence $A_0U, A_1U, \ldots A_tU$ shows that $AU \equiv BU$. Likewise $UA \equiv UB$. Thus $\equiv$ is a congruence. If $(P, Q) \in \mathcal{R}$ then by definition $P \equiv Q$. Therefore we have $\sim \subset \equiv$ and thus $\equiv = \sim$, as required. ∎

4.3. We obtain the sought verification algorithm as follows. To list all words W such that $U \sim W$ then write down all words that can be reached using $\mathcal{R}$ by a sequence of length 1, then those which can be reached by a sequence of length 2, and so on. Eventually, every word in $[U]$ will appear in the list. Therefore, if $U \sim V$ then V will be listed and this procedure terminates.

The unsolvability of the word problem means that although we can list the elements of $[U]$, we cannot list the elements of $X^* - [U]$. If it were possible to list the elements of the latter, then given V, we could run both procedures and we would know in a finite amount of time whether V was in $X^* - [U]$ or in $[U]$. In other words, we could decide whether or not $U \sim V$.

Definition 4.4. Let S be a set. A *linear ordering* of S is a transitive relation $<$ on S such that for any $s, t \in S$, exactly one of the following holds: $s < t, s = t, t < s$. Write $s \leqslant t$ if $s < t$ or $s = t$.

The ordering is a *well-ordering* provided that there does not exist an infinite decreasing sequence $s_1 > s_2 > \cdots$ of elements of S.

Proposition 4.5. *If $<$ is a well-ordering on S then every nonempty subset of S has a least element.*

Proof. Exercise. ∎

4.6. If $<$ is a linear ordering of a set S and n is a positive integer then we may impose the lexicographic ordering on S^n by defining $(s_1, \ldots, s_n) < (t_1, \ldots, t_n)$ iff there is an i with $1 \leqslant i \leqslant n$ such that $s_j = t_j$ for $1 \leqslant j < i$ and $s_i < t_i$. We warn the reader that we use the same symbol $<$ for the ordering on both S and on S^n. This ordering on S^n is also sometimes called the left-to-right lexicographic ordering.

Proposition 4.7. *Given a linear ordering $<$ on S, the corresponding lexicographic orderings on S^n are linear orderings, which are moreover well-orderings if $<$ is a well-ordering on S.*

Proof. Exercise. ∎

Definition 4.8. Given a linear ordering $<$ on a set X we define the *lexicographic ordering* on X^* as follows. Let $U = u_1 \cdots u_m$ and $V = v_1 \cdots v_n$ be in X^* with each of the $u_i, v_j \in X$. We say that $U < V$ provided that one of the following holds:

(i) $m < n$ and $u_i = v_i$ for all $1 \leqslant i \leqslant m$.

(ii) There is an i with $1 \leqslant i \leqslant \min(m, n)$ such that $u_t = v_t$ for all $1 \leqslant t \leqslant i$ and $u_i < v_i$ in X.

Equivalently, $U < V$ iff U is a proper prefix of V or some prefix of U is less than the prefix of V of the same length i in the lexicographic ordering on X^i.

Proposition 4.9. *If $<$ is a linear ordering on X then the lexicographic ordering is a linear ordering on X^*.*

Proof. Exercise. ∎

4.10. If $|X| > 1$ then the lexicographic ordering on X^* is not a well-ordering, even if X is well-ordered. For instance, if $a, b \in X$ and $a < b$ then $ab > a^2b > a^3b > \cdots$ is a strictly decreasing sequence in X^*.

4.11. For $w \in X^*$ write $w(i)$ for the prefix of i of length i and write $|w|$ for the length of w. We define the *length-lexicographic ordering* as follows. Let $U, V \in X^*$. Then $U < V$ provided that either $|U| < |V|$ or that $|U| = |V|$ and $U < V$ in the lexicographic ordering on X^m where $m = |U| = |V|$.

Proposition 4.12. *The length-lexicographic ordering is a linear ordering on X^* and if $<$ is a well-ordering of X then X^* is also well-ordered.*

Proof. Let us show that a $<$ well-ordering on X induces a well ordering on X^*. Let $U_1 > U_2 > \cdots$ be a strictly decreasing sequence of words in X^*. Since $|U_i| \geqslant |U_{i+1}|$ for all i, from some point on all of the U_i have the same length m. But then we are reduced to the lexicographic ordering on X^m which is a well-ordering since $<$ on X is a well-ordering. Thus the sequence terminates.

The rest of the assertions of the proposition are left to the reader. ∎

Definition 4.13. An ordering $<$ on X^* is *translation invariant* provided that

$$u < v \Rightarrow AuB < AvB \text{ for all } A, B \in X^*.$$

Lexicographic orderings are not translation invariant: $a < b \Rightarrow a < a^2$ but $ab > a^2b$.

We say that $<$ is consistent with length provided that

$$u < v \Rightarrow |u| \leqslant |v|.$$

The length-lexicographic orderings are clearly consistent with length.

Proposition 4.14. *The length-lexicographic ordering of X^* is translation invariant.*

Proof. Given words $u < v$ we must show that $ux < vx$ and $xu < xv$ for all $x \in X$. If $|u| < |v|$ then $|ux| = |xu| < |vx| = |xv|$ and thus we assume without loss that $m = n$. Suppose that u and v differ first in the ith term. Then ux and vx also differ first in their ith terms, which coincide with the ith terms of u and v. Therefore $ux < vx$. Likewise $xu < xv$. ∎

Definition 4.15. A *reduction ordering* is a translation invariant well-ordering.

Proposition 4.16. *In a reduction ordering on X^* the empty word comes first.*

Proof. If $u < \varepsilon$ for some word u then by translation invariance we have $u^2 < \varepsilon u = u$, $u^3 < \varepsilon u^2 < u$, and so on. Thus we obtain the infinite strictly descending sequence

$$\varepsilon > u > u^2 > u^3 > \cdots$$

contrary to the well-ordering assumption. ∎

4.17. Given a finitely presented monoid M we seek a solution to the word problem on M in particular, even though we know that in the word problem is unsolvable in general. Suppose that $M = X^*/\sim$ is given as a quotient of a free monoid. One way to attack the word problem is to define for each element $u \in M$ a word $u \in X^*$, which is in some sense the "simplest" way of defining u. The reduction orderings defined above give us a way to make precise the notion that one element is simpler than another.

Definition 4.18. Fix a reduction ordering $<$ on X^*. For $u \in M$ the set of words defining u is nonempty and therefore has a smallest element u. We define u to be the *canonical form* for u relative to $<$. For a word v, write $\overline{v}$ for the canonical form of the $\sim$-class containing v. We remark that the assignment $u \mapsto u$ taking a word to its canonical form is a section of the natural projection $X^* \to M$.

Remark 4.19. This definition of canonical class is not constructive and given a finitely presented monoid and a reduction ordering, we have at present no way of computing a canonical forms. Put differently, we have no algorithm which takes $v \in X^*$ as input and produces $\overline{v} \in X^*$ as output.

If we could compute canonical forms, then we would have a solution to the word problem because $u \sim v$ iff $\overline{u} = \overline{v}$. Thus, the unsolvability of the word problem implies that there are finitely presented monoids for which no algorithm $v \mapsto \overline{v}$ exists.

Proposition 4.20. *If u is the canonical form for an element of M then each subword of u is the canonical form for some element of M.*

Proof. Let V be a subword of U with $U = AVB$. If V is not a canonical form then there is a W such that $V > W$ and $V \sim W$. Then $AVB \sim AWB$ and since $<$ is translation invariant, $AVB > AWB$. This implies that U is not the least element of its $\sim$-class, a contradiction. ■

4.21. Let (P, Q) be an element of a generating set $\mathcal{R}$ for $\sim$. If we replace (P, Q) by (Q, P) then $\sim$ is unchanged, so we may assume without loss that $P > Q$. We call such a $(P, Q) \in \mathcal{R}$ with $P > Q$ a *rewriting rule* with respect to $<$. If every element of $\mathcal{R}$ is a rewriting rule, then $\mathcal{R}$ is called a *rewriting system* with respect to $<$.

Let $\mathcal{R}$ be a rewriting system. Let $\mathcal{P}$ be the set of left sides of elements of $\mathcal{R}$, let $\mathcal{N}$ be the ideal of X^* generated by $\mathcal{P}$, and put $\mathcal{C} := X^* - \mathcal{N}$. For each $U \in \mathcal{N}$ there are A, B, P, Q such that $U = APB$ and $(P, Q) \in \mathcal{R}$. Let $V = AQB$. Then $P \sim Q$ so that $U \sim V$. Moreover, we have $P > Q$ by definition, so that $U > V$. If $V \in \mathcal{N}$ we repeat this procedure to obtain a word W such that $U \sim V \sim W$ and $U > V > W$. Since $<$ is a well-ordering, this process terminates and we obtain $C \in \mathcal{C}$ with $U \sim C$ and $U \geqslant C$.

The procedure just described wherein we replace subwords which are left sides of rewriting rules with the corresponding right sides is called *rewriting*. We call the elements of $\mathcal{C}$ *irreducible* or *reduced* with respect to $\mathcal{R}$ since no rewriting can be performed on them. The canonical form of every $\sim$-class is in $\mathcal{C}$.

Let $(P, Q) \in \mathcal{R}$. We write $P \to Q$ for (P, Q) and more generally write $U \to V$ or $U \xrightarrow{\mathcal{R}} V$ if V *is derived from* U *in one step using* $\mathcal{R}$ by which we mean: there are words A, B, P, Q such that $P \to Q \in \mathcal{R}$, $U = APB$, and $V = AQB$. Let us give the pseudocode for this procedure.

Algorithm 1 Rewriting of words wrt a rewriting system

```
1:  procedure REWRITE(X, R, U)
2:      Input:
3:      X = finite set;
4:      R = finite rewriting system on X* wrt a reduction ordering;
5:      U = word in X*.
6:      Output:
7:      V = word in X* irreducible wrt R
8:          and definining the same element of Mon⟨X|R⟩ as U.
9:
10:     Let P be the set of left sides in R;
11:     V := U;
12:     while V contains a subword in P; do
13:         Let V = APB with P → Q in R;
14:         V := AQB;
15:     end while
16: end procedure
```

It remains to be shown that the output of the above procedure depends only on the input U and not on the choices made in each execution of the body of the loop. This will be accomplished in 4.30, below.

Remarks 4.22. (1) Let us remark on the syntax used in the above pseudocode. The expression "Let $V = APB$" means given V we write it in the stated form. The expression $V := AQB$ means we define V to have a new value. It is this new value which is tested in the clause of the next iteration

of the loop and which is passed in to the body of the loop should the test succeed.

(2) We note for emphasis that it is only the fact that there is a reduction ordering in the background that ensures that this procedure always terminates in finitely many steps.

Definition 4.23. Let $\mathcal{R}$ be a rewriting system on X^* associated to the reduction ordering $<$. Let $\sim$ be the congruence generated by $\mathcal{R}$. Let $U, V \in X^*$. We say that V *is derivable from* U *by means of* $\mathcal{R}$ if there is a sequence of words

$$U = U_0, U_1, \ldots, U_t = V$$

with $t \geqslant 0$ such that $U_i \xrightarrow{\mathcal{R}} U_{i+1}$ for each $0 \leqslant i < t$, and we shall denote this phenomenon by

$$U \overset{\mathcal{R}}{\Longrightarrow} V.$$

The relation $\overset{\mathcal{R}}{\Longrightarrow}$ is reflexive and transitive on X^*. Moreover, $U \overset{\mathcal{R}}{\Longrightarrow} V$ implies that $U \sim V$ and $U \geqslant V$.

Proposition 4.24. *If* $U \overset{\mathcal{R}}{\Longrightarrow} V$ *then* $UW \overset{\mathcal{R}}{\Longrightarrow} VW$ *and* $WU \overset{\mathcal{R}}{\Longrightarrow} WV$ *for all* $W \in X^*$.

Proof. Unwind the definitions. ∎

Proposition 4.25. *Let* $U, V \in X^*$. *Then* $U \sim V$ *if and only if there is a sequence of words* $U = U_0, U_1, \ldots, U_t = V$ *such that for each* $0 \leqslant i < t$ *we have either* $U_i \overset{\mathcal{R}}{\Longrightarrow} U_{i+1}$ *or* $U_{i+1} \overset{\mathcal{R}}{\Longrightarrow} U_i$.

Proof. Write $U \equiv V$ if there is a sequence $U_0, \ldots, U_t$ as in the statement of the proposition. Since $U_i \sim U_{i+1}$ we have $U \equiv V \Rightarrow U \sim V$, or in other words $\equiv \subset \sim$. One checks that $\equiv$ is an equivalence relation. By the previous proposition, $\equiv$ is a congruence on X^*. For $(P, Q) \in \mathcal{R}$, we have $P \equiv Q$. Thus $\sim \subset \equiv$. ∎

Definitions 4.26. (1) We say that the relation $\overset{\mathcal{R}}{\Longrightarrow}$ has the *Church-Rosser property* if $U \sim V$ implies that there is a word Q such that $U \overset{\mathcal{R}}{\Longrightarrow} Q$ and $V \overset{\mathcal{R}}{\Longrightarrow} Q$.

(2) We say that $\overset{\mathcal{R}}{\Longrightarrow}$ is *confluent* if $W \overset{\mathcal{R}}{\Longrightarrow} U$ and $W \overset{\mathcal{R}}{\Longrightarrow} V$ implies that there is a word Q such that $U \overset{\mathcal{R}}{\Longrightarrow} Q$ and $V \overset{\mathcal{R}}{\Longrightarrow} Q$.

(3) We say that $\overset{\mathcal{R}}{\Longrightarrow}$ is *locally confluent* if $W \xrightarrow{\mathcal{R}} U$ and $W \xrightarrow{\mathcal{R}} V$ implies that there is a word Q such that $U \overset{\mathcal{R}}{\Longrightarrow} Q$ and $V \overset{\mathcal{R}}{\Longrightarrow} Q$.

Remark 4.27. The preceeding definitions are analgous to properties that we might ask of open sets in a topological space, as follows. Suppose that U and V are open subsets of a topological space X. Fix $x \in X$ and define an equivalence relation on the open subsets of X by $U \sim V$ iff $x \in U \cap V$. Next, suppose that we define $U_1 \overset{\mathcal{R}}{\Longrightarrow} U_2$ to mean $U_2 \subset U_1$ and $x \in U_2$. Thus $U_1 \overset{\mathcal{R}}{\Longrightarrow} U_2$ implies $U_1 \sim U_2$.

The Church-Rosser property then is analogous to the basic fact that $U \sim V \Rightarrow$ there exists an open set Q contained in both U and V which contains x. Of course, $Q = U \cap V$ will do.

Loosely speaking, all of the conditions, Church-Rosser, confluence, and local confluence are variations on the theme of "an intersection of open sets is open" in the context of a neighborhood basis of a point.

Proposition 4.28. *If the Church-Rosser property holds then every $\sim$-class contains a unique element of $\mathcal{C}$, namely the canonical form for said class.*

Proof. As we have seen earlier, the canonical form for each $\sim$-class is in $\mathcal{C}$. Let $U, V \in \mathcal{C}$ which lie in the same $\sim$-class. The Church-Rosser property tells us that there is a word Q such that $U \overset{\mathcal{R}}{\Longrightarrow} Q$ and $V \overset{\mathcal{R}}{\Longrightarrow} Q$. Since no nontrivial rewrites can be performed on the elements of $\mathcal{C}$, we have $U = V = Q$. ∎

Theorem 4.29. *Suppose given a rewriting system $\mathcal{R}$ relative to a reduction ordering $<$. The Church-Rosser property, confluence, and local confluence are equivalent notions for $\mathcal{R}$.*

Proof. Show that Church-Rosser implies confluence. Given that $W \overset{\mathcal{R}}{\Longrightarrow} U$ and $W \overset{\mathcal{R}}{\Longrightarrow} V$ we have that $W \sim U$ and $W \sim V$ and therefore $U \sim V$. The Church-Rosser property guarantees us the existence of a word Q such that $U \overset{\mathcal{R}}{\Longrightarrow} Q$ and $V \overset{\mathcal{R}}{\Longrightarrow} Q$.

Show that confluence implies local confluence. This is by definition.

Show that local confluence implies confluence. Confluence fails for a word W if there are words U, V with $W \overset{\mathcal{R}}{\Longrightarrow} U, W \overset{\mathcal{R}}{\Longrightarrow} V$ but no word Q with $U \overset{\mathcal{R}}{\Longrightarrow} Q$ and $V \overset{\mathcal{R}}{\Longrightarrow} Q$. Let $\mathcal{W}$ be the set of words at which confluence fails. Suppose for contradiction that $\mathcal{W}$ is nonempty. Since $<$ is a well-ordering, we can define $W_0 := \min \mathcal{W}$. If $W_0 \overset{\mathcal{R}}{\Longrightarrow} U$ and $W_0 \overset{\mathcal{R}}{\Longrightarrow} V$ then we seek a word Q with $U \overset{\mathcal{R}}{\Longrightarrow} Q$ and $V \overset{\mathcal{R}}{\Longrightarrow} Q$. If $U = W_0$ then we take $Q = V$. If $V = W_0$ then we take $Q = U$. Therefore, without loss, we may assume that $U \neq W_0$ and $V \neq W_0$.

There are words A, B such that $W_0 \overset{\mathcal{R}}{\to} A, W_0 \overset{\mathcal{R}}{\to} B$ such that $A \overset{\mathcal{R}}{\Longrightarrow} U, B \overset{\mathcal{R}}{\Longrightarrow} V$. Local confluence gives us a word C with $A \overset{\mathcal{R}}{\Longrightarrow} C$ and $B \overset{\mathcal{R}}{\Longrightarrow} C$. Since $A < W_0$, $A \notin \mathcal{W}$. Therefore there exists D with $U \overset{\mathcal{R}}{\Longrightarrow} D, C \overset{\mathcal{R}}{\Longrightarrow} D$. Therefore $B \overset{\mathcal{R}}{\Longrightarrow} D$. Now, $B < W_0$ implies that there is a Q such that $D \overset{\mathcal{R}}{\Longrightarrow} Q$ and $V \overset{\mathcal{R}}{\Longrightarrow} Q$. This implies that $U \overset{\mathcal{R}}{\Longrightarrow} Q$ and therefore confluence does not fail for W_0, a contradiction. Thus $\mathcal{W} = \varnothing$, as required.

Show that confluence implies Church-Rosser. Given $U \sim V$, we must show that there is a Q with $U \overset{\mathcal{R}}{\Longrightarrow} Q, V \overset{\mathcal{R}}{\Longrightarrow} Q$. By 4.25, we have a sequence $U = U_0, U_1, \ldots, U_t = V$ such that for all $0 \leqslant i < t$, $U_i \overset{\mathcal{R}}{\Longrightarrow} U_{i+1}$ or $U_{i+1} \overset{\mathcal{R}}{\Longrightarrow} U_i$. We now prove the claim by induction on t. If $t = 0$ then $Q = U = V$ will do. If $t = 1$ then put $Q := \min(U, V)$. Let $t \geqslant 2$. Then $U_1 \sim V$ and the induction hypothesis gives us a word A with $U_1 \overset{\mathcal{R}}{\Longrightarrow} A, V \overset{\mathcal{R}}{\Longrightarrow} A$. If $U_0 \overset{\mathcal{R}}{\Longrightarrow} U_1$, then we put $Q := A$. Suppose that $U_1 \overset{\mathcal{R}}{\Longrightarrow} U_0$. Confluence gives us Q such that $U_0 \overset{\mathcal{R}}{\Longrightarrow} Q$ and $A \overset{\mathcal{R}}{\Longrightarrow} Q$. Therefore $V \overset{\mathcal{R}}{\Longrightarrow} Q$, as required. ∎

Proposition 4.30. *If $\mathcal{R}$ is a confluent rewriting system on X^*, then the result V of* REWRITE$(X, \mathcal{R}, U)$ *depends only on $\mathcal{R}$ and U and not on the unspecified choices made in the execution of the algorithm.*

Proof. By 4.28 and 4.29, the value V is the canonical form for the $\sim$-class containing U. ∎

Remark 4.31. The algorithm REWRITE is only partially defined. This proposition says that it is "defined enough." Figuratively speaking, in topological terms we might say that there is a topology on the space of sample paths which is coarse enough to render irrelevant all of the arbitrary choices made

in REWRITE and which is such that REWRITE is well-defined and continuous with respect to that topology.

We have already had a hint of the precise formulation of these notions in the first part of the course on automata theory; the space of sample paths of the algorithm in question is akin to a regular language $L \subset A^*$ where A is an alphabet.

Definition 4.32. A rewriting system $\mathcal{R}$ is said to be *reduced* if for each $(P, Q) \in \mathcal{R}$, Q is irreducible with respect to $\mathcal{R}$, no word appears as P (i.e. appears as a left hand side) for two different rules, and no left hand side contains another left side as a proper subword. Put differently, $\mathcal{R}$ is reduced iff for each $(P, Q) \in \mathcal{R}$, both P and Q are irreducible with respect to $\mathcal{R} - \{(P, Q)\}$.

Proposition 4.33. *Let $<$ be a reduction ordering on X^*. Every congruence on X^* is generated by a unique reduced and confluent rewriting system with respect to $<$.*

Proof. Let $\sim$ be a congruence on X^* and let M be $X^*/\sim$. Let $\mathcal{C} := \{\overline{U} : U \in X^*\}$ be the set of canonical forms for the elements of M and let $\mathcal{N}$ be the ideal $X^* - \mathcal{C}$.

Let $\mathcal{P}$ be the unique minimal generating set for $\mathcal{N}$ (see 3.16). Put $\mathcal{S} := \{(P, \overline{P}) : P \in \mathcal{P}\}$. We now show that $\mathcal{S}$ is the unique reduced confluent rewriting system with respect to $<$.

We have $U \geqslant \overline{U}$ for any U. For $P \in \mathcal{N}, P \neq \overline{P}$ so that $P > \overline{P}$. Thus $\mathcal{S}$ is a rewriting system with respect to $<$. Let $\equiv$ be the congruence generated by $\mathcal{S}$. Since $P \sim \overline{P}$ for all $P \in \mathcal{P}$ we have $\equiv \subset \sim$. Now, any word which is irreducible with respect to $\mathcal{S}$ is in $\mathcal{C}$ and therefore REWRITE$(X, \mathcal{S}, U)$ returns $\overline{U}$. This means that $U \equiv \overline{U}$ for all words U and therefore that $\equiv$ and $\sim$ coincide. Note that $\mathcal{S}$ is reduced by definition. If $U \sim V$ then $U \overset{\mathcal{S}}{\Longrightarrow} \overline{U}$ and $V \overset{\mathcal{S}}{\Longrightarrow} \overline{V}$. Since $\overline{U} = \overline{V}$ the Church-Rosser property is satisfied and $\mathcal{S}$ is confluent.

Next, suppose that $\mathcal{T}$ is a reduced and confluent rewriting system with respect to $<$ which generates $\sim$. Fix $P \in \mathcal{P}$. Using $\mathcal{T}$ to rewrite P gives $\overline{P}$. It follows that P contains a left side of $\mathcal{T}$ as a subword. On the other hand, all proper subwords of P are in $\mathcal{C}$. Therefore $\mathcal{T}$ contains a rule of the form (P, Q) for some Q. The fact that $\mathcal{T}$ is reduced forces $Q \in \mathcal{C}$ and therefore $Q = \overline{P}$ and we obtain $\mathcal{S} \subseteq \mathcal{T}$. If $(U, V) \in \mathcal{T} - \mathcal{S}$ then $U \notin \mathcal{C}$ and it follows that U contains a subword $P \in \mathcal{P}$. By definition, P is a left side in $\mathcal{S}$, contradicting the fact that $\mathcal{T}$ is reduced. ∎

Definition 4.34. Given a reduction ordering $<$ on X^* and $\mathcal{R} \subset X^* \times X^*$, let $\sim$ be the congruence generated by $\mathcal{R}$. Write RC$(X, <, \mathcal{R})$ or RC$(X, <, \sim)$ for the reduced and confluent rewriting system which generates $\sim$.

If $\mathcal{R}$ is assumed to be confluent, the following proposition identifies RC$(X, <, \mathcal{R})$.

Proposition 4.35. *Let $\mathcal{R}$ be confluent on X^* associated to $<$. Put*

$$\mathcal{P} := \{P : (P, Q) \in \mathcal{R} \text{ and no left side in } \mathcal{R} \text{ is a proper subword of } P\}.$$

Write $\overline{P}$ for the result of rewriting P by means of $\mathcal{R}$. Then we have

$$\text{RC}(X, <, \mathcal{R}) = \{(P, \overline{P}) : P \in \mathcal{P}\}.$$

Proof. Write $\sim$ for the congruence generated by $\mathcal{R}$. The confluence of $\mathcal{R}$ guarantees us that any word which is not the least element of its $\sim$-class must contain a subword which is a left side in $\mathcal{R}$. Therefore $\mathcal{P}$ is the minimal generating set for the ideal of noncanonical forms and the result follows. ∎

Proposition 4.36. *Suppose given a congruence $\sim$ on X^*, where as usual, X is assumed to be finite. Suppose that the set of $\sim$-classes is finite. Then* $\mathrm{RC}(X, <, \sim)$ *is finite for every reduction ordering $<$ of X^*.*

Proof. Put $\mathcal{S} := \mathrm{RC}(X, <, \sim)$ and let $(P, Q) \in \mathcal{S}$. Recall that we write $w(i)$ for the prefix of length i of a given word. Since each $P(i)$ is a canonical form, P_i and P_j are in different $\sim$-classes if $i < j < |P|$. It follows that $|P| \leqslant \#\{\sim\text{-classes}\}$. Since distinct elements of $\mathcal{S}$ have distinct left sides, it follows that $\mathcal{S}$ is finite. ■

Definition 4.37. If $\mathcal{R}$ is a confluent rewriting system on X^* then we say that $\mathrm{Mon}\langle X|\mathcal{R}\rangle$ is a *confluent presentation* for the monoid in question.

4.38. Given X, let $<$ be a reduction ordering on X^* and let $\mathcal{R}$ be a finite rewriting system on X^* with respect to $<$. If we suppose that $\mathcal{R}$ is confluent then we know how to solve the word problem in $M := \mathrm{Mon}\langle X|\mathcal{R}\rangle$ by using REWRITE. Thus if we could test a rewriting system for confluence, then we would have a way to say something about the solvability of the word problem for a given presentation.

We say that local confluence fails at a word W if there are $U, V \in X^*$ such that $W \xrightarrow{\mathcal{R}} U$ and $W \xrightarrow{\mathcal{R}} V$ but there is no Q such that $U \xRightarrow{\mathcal{R}} Q$ and $V \xRightarrow{\mathcal{R}} Q$.

Proposition 4.39. *Let W be a word such that local confluence fails at W but does not fail at any proper subword of W. Then one of the following holds:*

(1) W appears as the left side of two distinct elements of $\mathcal{R}$.

(2) W is a left side in $\mathcal{R}$ which contains another left side as a proper subword.

(3) $W = ABC$ where A, B, C are nonempty words such that AB and BC are left sides in $\mathcal{R}$.

Proof. This is Proposition 3.1 on page 58 of Sims. ■

Definition 4.40. If W is as in the proposition, then we call W an *overlap of left sides* in $\mathcal{R}$. If the third condition holds then we say that W is a *proper overlap*. Only proper overlaps can arise in a reduced rewriting system.

4.41. Since $\mathcal{R}$ is finite, the set $\mathcal{W}$ of words which are overlaps of left sides in $\mathcal{R}$ is also finite. For each $W \in \mathcal{W}$, write $\mathcal{U}_W$ for the finite set of words U such that $W \xrightarrow{\mathcal{R}} U$ is a derivation consisting of a single step. For each $U \in \mathcal{U}_W$ we put $V := \mathrm{REWRITE}(X, \mathcal{R}, U)$. As U varies, if more than one V is obtained, then $\mathcal{R}$ is not confluent. The reason is that in this case we have found two words which are irreducible with respect to $\mathcal{R}$ and define the same element of M.

On the other hand, if only one value of V is seen as U varies in $\mathcal{U}_W$, then local confluence does not fail at W.

Performing this test for all $W \in \mathcal{W}$, we have an algorithm for determining whether or not $\mathcal{R}$ is confluent. We now proceed to describe this algorithm formally.

Algorithm 2 Testing a rewriting system for confluence

```
1: procedure CONFLUENT(X, ℛ)
2:     Input:
3:     X = finite set;
4:     ℛ = finite rewriting system on X* wrt a reduction ordering;
5:     Output:
6:     True or False according as ℛ is or is not confluent.
7:
8:     for (P, Q) ∈ ℛ do
9:         for (R, S) ∈ ℛ do
10:            for B a nonempty suffix of P do
11:                Let U be the longest prefix common to both B and R;
12:                Let B = UD, R = UE;
13:                if D or E is the empty word then
14:                    P := AB;
15:                    V := REWRITE(X, ℛ, ASD);
16:                    W := REWRITE(X, ℛ, QE);
17:                    if V ≠ W then
18:                        return False;
19:                    end if
20:                end if
21:            end for
22:        end for
23:    end for
24:    return True;
25: end procedure
```

Example 4.42. Take $X := \{x, y, z\}$ and consider the finite rewriting system $\mathcal{S} := \{x^2 \to \varepsilon, yz \to \varepsilon, zy \to \varepsilon\}$. We claim that $\mathcal{S}$ is confluent. The claim is proved by tracing through the steps of the above algorithm.

4.43. Given a rewriting system $\mathcal{S}$ and a word $U \in X^{\pm *}$ there are typically many ways of rewriting U with the objective of obtaining an irreducible word. For the sake of writing a computer program which actually finds the sought irreducible word, we must fix a way of rewriting a given word by using a given rewriting system.

Let us informally describe a strategy as follows. If we are rewriting the word U, then we must write U as $U = APB$ and then replace it by $U = AQB$ where $(P, Q) \in \mathcal{S}$. How to choose which occurence of P to use in U? We choose the left side P in U which is maximal with respect to $<$ and subject to this condition, as far to the left in U as possible. Let us give the code of a refined version of the foregoing heuristic ideas.

We have the following invariants of the algorithm (i.e. the following statements are true at each step of the execution of the algorithm):

(1) the word VW is derivable from U.

(2) V is the longest prefix of VW known at the time of execution to be irreducible with respect to the set of rules.

Example 4.44. Let us trace through the execution of this procedure in an example. Let $X = \{a, b\}$ and consider the rewriting system $\mathcal{S} := \{a^2 \to \varepsilon, ab^2a \to bab, b^3 \to \varepsilon, baba \to ab^2, abab \to b^2a, b^2ab^2 \to aba\}$. Here is a "sample path" of the execution of REWRITELEFT$(X, \mathcal{S}, U)$.

Algorithm 3 Rewriting from left

```
1: procedure REWRITELEFT(X, ℛ, U)
2:     Input: X = generators, ℛ = rewriting system, U = a word;
3:     Output: the rewritten form of U
4:     V := ε, W := U;
5:     while W ≠ ε do
6:         Let W = xW_1 where x ∈ X; W := W_1, V := Vx;
7:         for i = 1, ..., n do
8:             if P_i is a suffix of V then
9:                 V := RP_i, W := Q_i W, V := R;
10:                break
11:            end if
12:        end for
13:    end while
14: end procedure
```

$$
\begin{aligned}
ab\underline{aa}babbaaabbabbbaba \\
\underline{abba}bbaaabbabbbaba \\
ba\underline{bbb}aaabbabbbaba \\
b\underline{aa}aabbabbbaba \\
b\underline{aa}bbabbbaba \\
\underline{bbb}abbbaba \\
a\underline{bbb}aba \\
\underline{aa}ba \\
ba
\end{aligned}
$$

5. THE KNUTH-BENDIX ALGORITHM

5.1. Let $(X, \mathcal{R})$ be a finite presentation of a monoid M. Given a reduction ordering $<$ on X^*, let $\mathcal{T} := RC(X, <, \mathcal{R})$. Assume that $\mathcal{T}$ is finite. The Knuth-Bendix algorithm takes as input $X, \mathcal{R}, <$ and produces $\mathcal{T}$. The algorithm makes an essential use of the hypothesis that $\mathcal{T}$ is finite; no procedure is given to decide whether or not $\mathcal{T}$ is finite. Moreover, the algorithm requires that $<$ be effective in the sense that $<$ is given together with an algorithm to determine whether or not the statement "$U < V$" is true.

The idea of the algorithm goes as follows. Apply CONFLUENT to the current set of rules. If the rules are confluent, we are done. Otherwise there exist irreducible words A and B such that $A \sim B$ where $\sim$ is the congruence generated by $\mathcal{R}$. Changing notation if need be, we have $A > B$ and (A, B) is appended to our set of rules.

We write $\mathcal{S} = \{(P_i, Q_i)\}_{i=1}^{n}$ for the set of rules that have been found so far. When the computation terminates, we will have $\mathcal{S} = \mathcal{T}$.

Let us now give the formal specifications of the algorithm. First we require a subroutine UPDATE(U, V) which will, if need be, add a new rule so as to ensure that there exists a word derivable from the given words U, V by means of the rules in $\mathcal{S}$.

Algorithm 4 Updating the list of rules

```
1: procedure UPDATE(S, U, V)
2:     Input: S = {(P_1, Q_1), (P_2, Q_2), ..., (P_n, Q_n)} a finite rewriting system;
       U, V = words;
3:     Output: none; the state of S is modified in place;
4:     A := REWRITELEFT(U);
5:     B := REWRITELEFT(V);
6:     if A ≠ B then
7:         if A < B then
8:             swap A and B;
9:         end if
10:        append (A, B) to S;
11:    end if
12: end procedure
```

We need one more subroutine, the procedure OVERLAP$(\mathcal{S}, i, j)$. A new rule is appended to $\mathcal{S}$ for each failure of local confluence.

Algorithm 5 Check the overlaps of P_i and P_j in which P_i is a prefix

procedure OVERLAP($\mathcal{S}, i, j$)
  Input: $\mathcal{S} = \{(P_1, Q_1), (P_2, Q_2), \ldots, (P_n, Q_n)\}$; i, j = positive integers $\leqslant |\mathcal{S}|$
  Output: none; the state of $\mathcal{S}$ is modified in place;
  for $k := 1, \ldots, |P_i|$ **do**
    Let $P_i = AB$ where $|B| = k$;
    Let U be the longest word which is a prefix of both B and P_j;
    Let $B = UD$ and $P_j = UE$;
    if $D = \varepsilon$ or $E = \varepsilon$ **then**
      UPDATE($\mathcal{S}, AQ_jD, Q_iE$);
    end if
  end for
end procedure

Algorithm 6 The Knuth-Bendix Algorithm

procedure KNUTHBENDIX($X, <, \mathcal{R}$)
  Input:
  X = a finite set, $<$ = reduction ordering on X^*, $\mathcal{R} \subset X^* \times X^*$ a finite subset;
  Output: $\mathcal{T} = RC(X, <, \mathcal{R})$ if it is finite

  $\mathcal{S} := \{\}$; $i := 1$;
  for $(U, V) \in \mathcal{R}$ **do**
    UPDATE($\mathcal{S}, U, V$);
  end for
  while $i \leqslant n$ **do**
    for $j := 1, \ldots, i$ **do**
      OVERLAP($\mathcal{S}, i, j$);
      if $j < i$ **then**
        OVERLAP($\mathcal{S}, j, i$);
      end if
    end for
    $i := i + 1$;
  end while
  Let $\mathcal{P} := \{P_i$: every proper subword of P_i is irreducible wrt $\mathcal{S}\}$;
  $\mathcal{T} := \{\}$;
  for $P \in \mathcal{P}$ **do**
    $Q :=$ REWRITELEFT($X, \mathcal{R}, P$);
    append (P, Q) to $\mathcal{T}$;
  end for
end procedure

Theorem 5.2. *If* $\mathrm{RC}(X, <, \mathcal{R})$ *is finite then* KNUTHBENDIX$(X, < \mathcal{R})$ *terminates and outputs* $\mathrm{RC}(X, <, \mathcal{R})$.

Proof. Tracing the steps of the execution of the algorithm, if the call UPDATE$(\mathcal{S}, U, V)$ has been made and the two calls to REWRITELEFT inside of update have completed, then $\mathcal{S} \cup \{(U, V)\}$ and $\mathcal{S} \cup \{(A, B)\}$ generate the same congruence on X^*. It follows that after the execution of the first for loop in KNUTHBENDIX (i.e. at line 11) the sets $\mathcal{R}$ and $\mathcal{S}$ generate the same congruence on X^*. In particular, $\mathcal{S}$ generates $\sim$ at any time during the execution of the while loop starting on line 12.

We observe that P_i is irreducible with respect to $\{(P_j, Q_j) : 1 \leqslant j < i\}$.

Suppose that KNUTHBENDIX does not terminate in finitely many steps. This implies that the while loop starting on line 12 produces infinitely many rewriting rules $(P_i, Q_i), i = 1, 2, \ldots$. Write $\mathcal{U}$ for the set of these rules.

Claim. $\mathcal{U}$ is confluent and generates $\sim$.

Proof of the Claim. $\mathcal{U}$ generates $\sim$ by the same proof that shows $\mathcal{S}$ does. If $\mathcal{U}$ is not confluent, put

$$W := \min\{V : \text{local confluence fails at } V\}.$$

Proposition 4.39 guarantees us that $W = ABC$ with $B \neq \varepsilon$ and one of the following is true:

(1) $W = P_j, B = P_i, j < i$ and no word can be derived from both AQ_iC and Q_j by means of $\mathcal{U}$.

(2) $AB = P_i, BC = P_j$ and no word can be derived from both AQ_j and Q_iC be means of $\mathcal{U}$.

If (1) holds then during the execution of KNUTHBENDIX, a call to OVERLAP(j, i) is made. Once this call has returned, there is a word W' which may be derived from both AQ_iC and Q_j using $\mathcal{S}$ in its current state. Since $\mathcal{S} \subset \mathcal{U}$, the word W' is derivable by means of $\mathcal{U}$, contrary to (1).

In case (2) holds, the call to OVERLAP(i, j) would give rise to a word which may be derived from AQ_j and Q_iC by means of $\mathcal{U}$.

In all cases we have a contradiction and therefore $\mathcal{U}$ is confluent. ///

To complete the proof of the theorem, let $\mathcal{C}$ be the set of canonical forms with respect to $\sim$ and put

$$\mathcal{P} := \{P \in X^* - \mathcal{C} : \text{every proper subword of } P \text{ is in } \mathcal{C}\}.$$

(This is precisely the same set defined on line 21.)

If we rewrite $P \in \mathcal{P}$ by means of $\mathcal{U}$, we obtain $\overline{P} \in \mathcal{C}$ such that $P \sim \overline{P}$. Therefore there is a rule in $\mathcal{U}$ with left side P and this rule is unique by the construction of $\mathcal{U}$. Put

$$\mathcal{V} := \{(P, Q) \in \mathcal{U} : P \in \mathcal{P}\}.$$

Since we are given that $\mathrm{RC}(X, <, \mathcal{R})$ is finite, it follows that $\mathcal{P}$ and $\mathcal{V}$ are finite. To prove this last statement, we consider the sequence of rewriting rules starting at P and terminating at an element of $\mathrm{RC}(X, <, \mathcal{R})$. Each such chain is finite in length, and there are only finitely many such chains. The set $\mathcal{P}$ is the set of first elements of such chains, and the set $\mathcal{V}$ is the set of first steps in such chains.

Let $n = \max\{n : (P_n, Q_n) \in \mathcal{V}\}$. We have $i > n \Rightarrow P_i \notin \mathcal{C}$ and is irreducible with respect to $\{(P_j, Q_j) : 1 \leqslant j < i\}$. This is a contradiction since $\mathcal{C}$ is the set of irreducible elements. Therefore $\mathcal{U}$ is finite and that the while loop starting on line 12 terminates.

The characterization of $\mathrm{RC}(X, <, \mathcal{R})$ in Proposition 4.35 gives us that the set $\mathcal{T}$ defined by the for loop starting on line 23 coincides with $\mathrm{RC}(X, <, \mathcal{R})$. ■