William W. Cohen · Charles K. Cohen

A Computer Scientist's Guide to Cell Biology

Second Edition

Springer

A Computer Scientist's Guide to Cell Biology

William W. Cohen • Charles K. Cohen

A Computer Scientist's Guide to Cell Biology

Second Edition

Springer

William W. Cohen
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA, USA

Charles K. Cohen
Johns Hopkins University
Baltimore, MD, USA

ISBN 978-3-031-55906-8 ISBN 978-3-031-55907-5 (eBook)
https://doi.org/10.1007/978-3-031-55907-5

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Heiti Paves / Alamy Stock Photo

If disposing of this product, please recycle the paper.

Acknowledgments

Charlie:

This book was a family affair.

The first edition of this volume was almost entirely the work of my father and co-author, William. None of this would exist if not for him—while I was struggling to survive high school, he's the one who made the time to do the research, organize the information, drew the original figures, and write the original words. This was, and remains, his book, and I am forever grateful that he gave me the opportunity to work with him on this project and run roughshod over the text he spent so long writing.

My mother, Susan Cohen, proofread and indexed both editions (my late grandfather, William Daniel Kundin, also proofread the first edition). I'm sorry we put you through such a painful ordeal. Hopefully it was worth it.

My sister, Cassandra Cohen, updated a number of the figures, despite not knowing what she was working on half the time. If something's wrong, don't blame her.

Dr. Helen Rich gave us very helpful comments on an early draft of the second edition, including discussing important advances in biology in recent years.

As for me, I provided the words in this edition and not much else. The book already existed; all I did was go back and repeat it in a funny voice. Feel free to blame me if something's wrong.

On a personal note, I would never have finished the task if not for the tireless support of my partner, Ru—you're my home and my harbor, and without you I would have washed away long before finishing.

Finally, a big shout-out to the librarians at the Hazelwood Public Library for letting me camp out in front of the public computers for hours on end. Sorry!

William:

Writing the first edition of this book was, like many writing tasks, slow going for most of the way, although worthwhile in the end. I worked hard in organizing and curating the content of the first edition, but while that edition had many good points, I don't think any of the readers found it actually fun to read.

Writing the second edition with Charlie was rewarding on a totally different level. If there is anything more enjoyable than watching someone else take your precise but somewhat stilted prose, and make it engaging, it must be having one of your offspring do that. Charlie's modest comments in the acknowledgments above definitely undersell his contributions: he brought a new perspective to the book, as (unlike myself) he has actually spent years not just studying biology but doing real biology in real labs. Working with him was very rewarding technically, not just personally, and the end product of the collaboration is not only more up-to-date and readable, but a work that is a better bridge between how biologists and computer scientists might think about the field. It is also, in more than one place, more correct.

I'd like to thank my wife Susan for all her support over the years, for her standout job indexing and proofreading this book—and, of course, for her patience with Charlie and me for getting her drafts later than promised. I'm also grateful to my daughter Cassie for her work adding figures for the new material and updating the existing figures.

Finally, I'd like to thank the many readers of the first edition who approached me with questions, corrections, and occasionally thanks and encouragement for that work.

Contents

Introduction

As the amount of biological data grows, the task of understanding existing data becomes increasingly important, and this is largely a task best undertaken by computer science. This book is for the many curious souls who are coming into biology from backgrounds in computer science, especially the fields of information retrieval, natural language processing, and/or machine learning.

One major difference between biology and computer science is that in computer science, the world we explore is in large part our own creation, and a large part of what we do is make our creation understandable by finding useful abstractions, and then building more complex things by combining these abstractions together. For example, a deterministic finite state machine is a useful abstraction for computations that process discrete inputs sequentially with limited memory—we study this, and study stack data structures, and then study the result of combining them to make a push-down automaton. These abstractions might be compromised when we optimize our systems for performance, but they are rarely abandoned completely, because comprehensibility, elegance, and simplicity are practically important for systems that must be maintained and improved by humans.

In contrast, biology doesn't lend itself to clean and comprehensible abstract models: evolution relentlessly marches toward improved performance without worrying much about simplicity. Even the "simplest" forms of life are seemingly endless in their unique complexities, and almost every general statement about how organisms function comes with an asterisk. And unlike in computer science, the details that underlie the complexity of the real

W. W. Cohen, C. K. Cohen, *A Computer Scientist's Guide to Cell Biology*,
https://doi.org/10.1007/978-3-031-55907-5_1

systems are not something we can or should ignore, hoping they will be cleaned up in the next version—instead, the awkward details are, collectively, the real subject of the science of biology.

For the purposes of this book, we have broken the field down into three parts:

- **Biological mechanics** are the actual nitty-gritty details of how things work at the cellular level—protein pathways, chloroplasts, and so on. This is the typical focus of introductory biology classes and textbooks, and you would correctly suppose this is the essence of what biologists actually study. However, it's a surprisingly small part of what biologists write and talk about.
- **Experimental methods**, on the other hand, *are* what biologists spend most of their time talking about. If you pick up a typical biology paper, the actual *conclusions*, the newly discovered details about how these systems function, are compact enough to be laid out in the abstract.

As you read through this book, you'll find that it's mostly about methods. Biologists spend most of their word count talking about how they conducted their experiments, how cells were cultured, and what assays were run and a host of other details. The results, in isolation, tell you very little—the only way to tell the difference between good research and bad research is to examine how data was initially gathered. But to an outsider, that can be far from a simple task. The language of biology is rich, detailed, and almost impenetrable to the average layperson; learning its intricacies is as important as learning about biological mechanisms or experimental techniques.

- **Language and nomenclature** can be considered a "part" of biology in its own right. Without spending at least a little bit of time learning how to speak it, this book would be pretty useless.

If you like, you can think of biology as a journey to some strange, exotic land. The inhabitants speak a strange and often incomprehensible language, the customs and practices may be like nothing seen before, and even the most basic of tasks appear completely alien. With that in mind, our goal is to provide a short introduction to the three core aspects of cell biology—a travel guide, to continue the previous metaphor, focusing on high-level principles, and relating as much as possible to familiar concepts from computer science.

Consequently, in this book, we will gloss over some concepts and oversimply others, setting aside many otherwise-fascinating theories and details. Biology is fractal; no matter how deep you look, there is always another layer of complexity. For a more comprehensive background on biology, there are many excellent textbooks, written by people far more qualified—the last chapter of this book will introduce several of our favorites.

How Cells Work: The Basics

What Life Is Made Of

DNA

Deoxyribonucleic acid—**DNA**—is perhaps the most famous molecule in the world. If you close your eyes, you can probably picture its iconic double-helix shape, like a rope ladder twisted around its central axis. The "ropes" are long strings of simple sugars known as **the sugar-phosphate backbone**, the "rungs" much smaller organic molecules called **nucleotides**.

Of course, it's probably not a good idea to use a strand of DNA as an actual ladder. Issues of scale aside, the "rungs" aren't quite as solid as they look at first glance. Each one is actually a *pair* of nucleotides, either **adenine** and **thymine** or **guanine** and **cytosine** (usually abbreviated as ATGC). And while nucleotides are firmly attached to their parent backbone by powerful phosphodiester bonds, the **hydrogen bonds** that connect nucleotide to nucleotide are comparatively weak. Apply too much pressure, and the hydrogen bonds start to break, allowing the two strands of DNA to separate into individual ribbons, known as the **3′ and 5′ strands** (three prime and five prime).

The two strands are perfect mirror images of one another. The sugars that make up their backbones "point" in different directions, and they have complementary nucleotides. When the 3′ strand has an A, the 5′ strand will have a T, where the 5′ has a G, and the 3′ will have a C. Nucleotides can only pair with their specific partner, meaning that there's only one way to connect two long strands of DNA. This specificity also means that you only need one

W. W. Cohen, C. K. Cohen, *A Computer Scientist's Guide to Cell Biology*,
https://doi.org/10.1007/978-3-031-55907-5_2

strand to reconstruct the entire double-helix, a property that—as we'll see later—is crucial to DNA's ability to replicate itself.

And it's the nucleotides that carry the actual genetic information. If a computer's code is ultimately binary, DNA is quaternary. Instead of a sequence of 1s and 0s, the code of life is written as a sequence of adenine, thymine, guanine, and cytosine groups. A sequence of three nucleotides is called a **codon**; it's these codons that the cell reads when the time comes to build proteins.

Cells also make use of DNA's cousin, ribonucleic acid (**RNA**). Chemically, RNA is very similar to DNA—the sugars that make up their backbone are slightly different, and it uses a fifth nucleotide called **uracil** in place of thymine—but it doesn't form a double-helix, making it less well-suited to long-term data storage. Instead, individual strands of RNA float freely around the cell, waiting to be used. It's usually much more short-lived than DNA, although some simple organisms use it as their sole type of genetic material.

Proteins

Proteins are massive biomolecules with a staggering variety of roles in the cell. Some are purely structural; others catalyze chemical reactions. They receive signals, transport smaller molecules across membranes, break down sugars, replicate DNA, and perform a thousand other tasks. If you're talking about a molecule *doing* something in a cell, it's almost certainly going to be a protein.

Just as DNA is a long molecule built using only four different blocks—adenine, guanine, cytosine, and thymine—proteins are long molecules made from simpler building blocks called **amino acids**. There are a total of 20 different amino acids, each with different chemical properties. Once assembled, these properties quickly drive the protein to start **folding** into complicated three-dimensional forms through a complex system of interactions that we still don't fully understand. Even small proteins are typically made up of hundreds or even thousands of amino acids; the largest are made of multiple amino acid chains twisting together into a single structure.

Twenty different amino acids appear in proteins, which are analogous to the four different nucleotides in DNA: every protein can be represented as a string over those 20 letters, just as every DNA molecule can be represented as a string over four letters. Recall that a DNA codon contains three base pairs: generally, each possible codon maps to one of the 20 amino acids.[1]

[1] There are also **start** and **stop codons**, which tell the cell's machinery when to start and stop transcribing DNA.

Each amino acid is connected to its neighbors in the linear chain by strong **covalent bonds** that stem from two atoms "sharing" some of the same electrons. The folding process, however, depends on weaker (and longer-range) connections such as **ionic bonds** (where oppositely charged atoms attract one another magnetically), **hydrogen bonds** (where an entire hydrogen atom is shared between two molecules), and **van der Waals forces**, which are sort of like atomic static electricity. Some amino acids are **hydrophobic** and try to minimize their exposure to water; others are **hydrophilic** and happily co-exist with water molecules.

When you take all this into account, you wind up with unique, complicated molecules. The details of a protein's shape determine what kind of molecules it can interact with, and what kind of chemical reactions it can take part in—and given how complicated those shapes can be, those interactions can be highly specific. Many proteins can only "stick" to a small number of other proteins—this type of interaction ultimately drives most cellular activity. Predicting the shape of a protein based solely on its **primary sequence** (the sequence of amino acids that defines it) as a complex computational problem, as is predicting if two proteins will interact.

Proteins don't just interact with one another, either. Some have evolved to detect small signaling molecules, or interact with certain metabolic products. Others attach themselves to highly specific DNA sequences (e.g., "TTAGCTCTA").

And just to make everything that much more confusing, protein shapes—and, thus, their functions—aren't static. Most of the time, the very process of binding to its preferred target causes one (or both!) proteins to change shape and behave in new and different ways.

Polymers

A molecule that is composed of two identical subunits is a dimer; three identical subunits compose a trimer; and N identical subunits compose a polymer. An enzyme in which binding sites do not behave independently is an allosteric enzyme; in the example here, the enzyme exhibits cooperative binding.

Lipids and Membranes

Hydrophobic and hydrophilic reactions are also crucial for the third type of important biomolecule: **phospholipids**. "Lipid" is a broad category of large organic molecules that don't mix well with water, such as fats and waxes. Phospholipids, however, have a hydro*philic* "head," meaning that one side

wants to avoid water and the other wants to embrace it. Dump a bunch of phospholipids into water, and they will naturally assemble into a back-to-back formation with their "heads" pointed out and their "tails" pointed in, forming a very stable two-layer sheet called a **phospholipid bilayer**. These bilayers are the main component of all cellular membranes—both the **plasma membrane** that encloses the entire cell, and the individual membranes that encircle smaller subcellular compartments like the nucleus.

Plasma membranes also contain many proteins—pores, receptors, and several other varieties. Because the interior of a membrane is hydrophobic (i.e., doesn't like water), similarly hydrophobic protein elements can "hide" inside and force the rest of the protein to remain adjacent to the membrane.

Types of Life

Prokaryotes

At the most basic level, all living organisms can be divided into two categories: **eukaryotes** and **prokaryotes**, depending on whether or not they keep their genetic material sectioned off from the rest of the cell.

Prokaryotes, at first glance, seem to be a pretty monotonous lot—bacteria, bacteria, and more bacteria. (To be fair there are also **archaea**, but they look so much like bacteria that it wasn't until the 1970s that people noticed that they were genetically different.) Structurally speaking, prokaryotes are more or less just bags of proteins and genetic material, where the "bag" is a plasma membrane—which, in many prokaryotic cells, is reinforced by a layer of stiff starchy armor called a **cell wall**. Numerous proteins are anchored in the plasma membrane, and inside is a convoluted soup of proteins, sugars, nucleic acids, and other biomolecules known as the **cytoplasm**. **Prokaryotic DNA** takes the form of a simple loop, which floats somewhere inside the plasma membrane.

If you look a little closer, though, you'll find an absolutely unbelievable amount of diversity among prokaryotes. These seemingly simple organisms can be found in every nook and cranny of the Earth, from hot springs to ice fields to deep-sea vents, and feed on everything from sunlight to elemental sulfur. In 2006, it was even discovered that some prokaryotes can sense magnetic fields.

Magnetotactic bacteria contain chains of tiny magnetic crystals enclosed in lipid bilayer membrane. The chain acts like a compass needle—it can be used to orient the bacteria relative to the earth's magnetic field.

Perhaps the most famous and most-studied prokaryote is *Escherichia coli* (*E. coli* to its friends), a bacterium normally found in the human intestine.

Eukaryotes

Eukaryotes are probably more familiar, at least in the sense that most living things you've ever seen are eukaryotes. As a rule of thumb, anything you can see with your naked eye is a eukaryote. It's certainly true that all multicelled organisms are eukaryotes, but the category also includes a number of single-celled organisms like yeast and amoebas. Despite seeming like they have more in common with their prokaryotic cousins than, say, an elephant, the structure and biochemistry of single-celled eukaryotes like yeast is far closer to that of the pachyderm than to a bacteria.

For a start, eukaryotic cells are *big*, much more so than prokaryotes. The famous *E. coli* bacterium, for instance, is about 2 µm long, but a typical mammalian cell is 10–30 µm long (Fig. 1). You might as well be comparing a hamster to a human, or a human to a 60-foot sperm whale.

Eukaryotes also have a much richer inner life. Every cell, regardless of whether or not it's part of a more complex organism, has its own set of internal organs, conveniently enough known as **organelles**, each one enclosed in its own little plasma membrane. The **nucleus** keeps the DNA safe and segregated, **mitochondria** house all the machinery needed to turn sugars into cellular energy, the **endoplasmic reticulum** is a vast factory for protein synthesis, to name a few of the most prominent examples, but that's just scratching the surface—check out Fig. 2 for a (marginally) more comprehensive overview.

Not only is eukaryotic DNA tucked away in its own compartment, there's also much, much more of it, stored in a much more complicated way. For a start, there's more than one loop—instead, we have many separate sections of DNA, each one wound around thousands of proteins called **histones**. Histones are then packed together to form **nucleosomes**; nucleosomes are compressed into **supercoils**, and ultimately thousands of supercoils collect into a single large complex known as a **chromosome**.

All this wrapping is extraordinarily effective. A single human cell is microscopic, but if you took all of its DNA and stretched it out end-to-end, it would make a strand almost six feet long. A multicellular organism like you has *billions* of miles worth of DNA in their body.

As another way of emphasizing the relative complexity of eukaryotes, it's very possible that a eukaryotic cell's organelles were once completely independent prokaryote-like creatures. According to the theory of **endosymbiosis**, smaller prokaryotes once took shelter inside the membranes of their larger

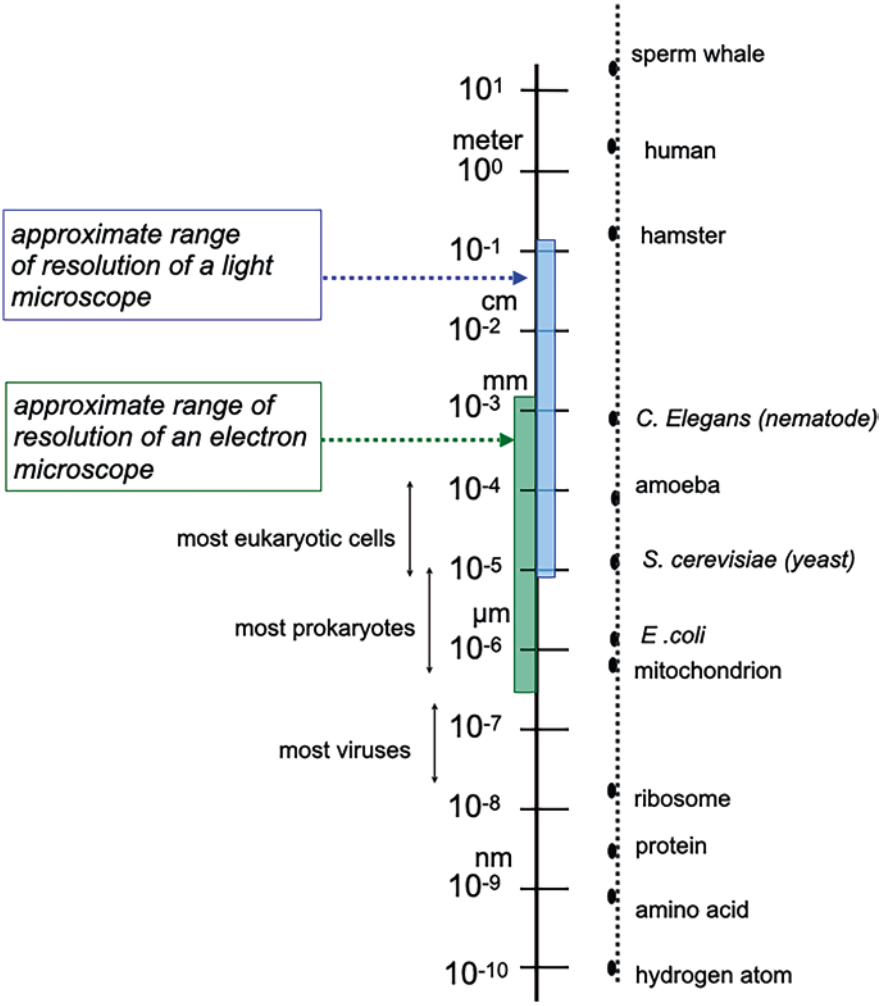


Fig. 1 Relative sizes of various biological objects

cousins, providing some sort of service—photosynthesis, say—in exchange for safety and a share of the cell's resources. Over time, the symbiotic cells became smaller and more specialized, delegating more and more functions and shifting more and more DNA to the host cell until they were little more than internal organs.

The two biggest candidates for this role are mitochondria and **chloroplasts**, the organelle plants use to turn sunlight into energy. Even today, these organelles retain scraps of their original genomes, with DNA sequences that use completely different codes than those found in the cells' nucleus.

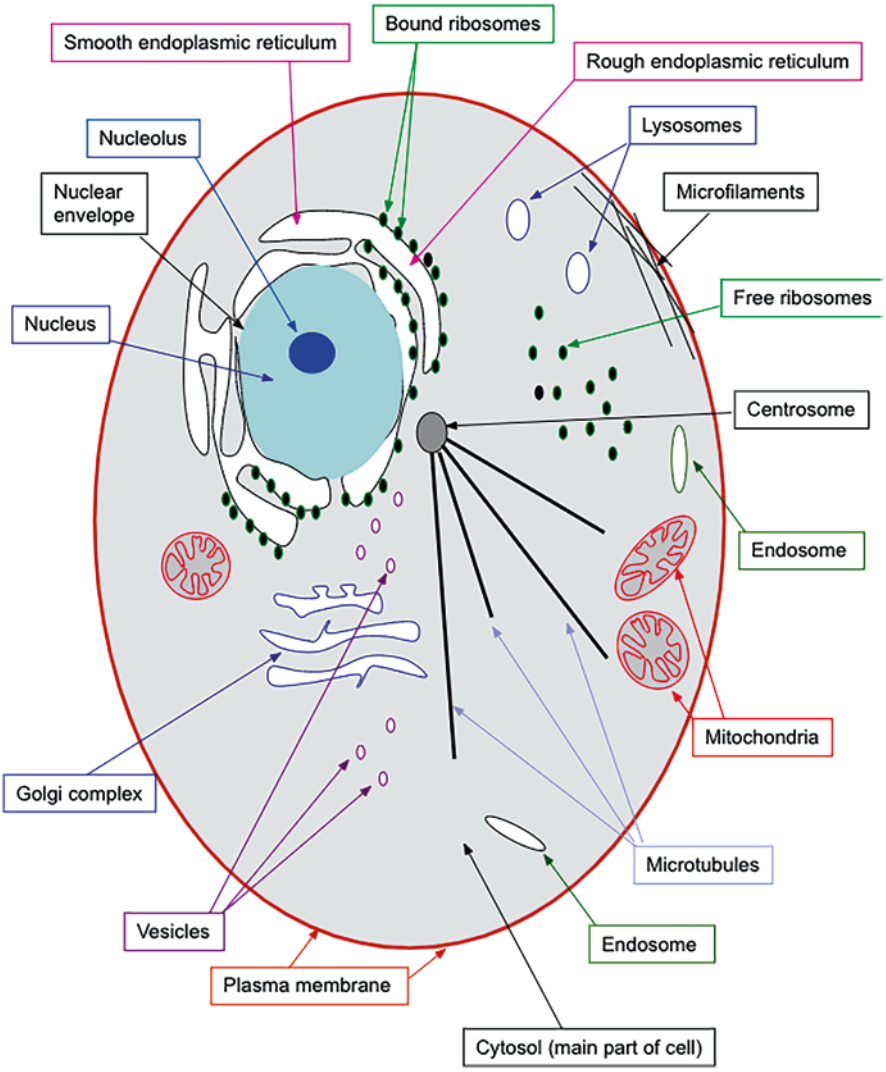


Fig. 2 Internal organization of a eukaryotic animal cell

Multicellular Life, Tissues, and Signaling

The human body—indeed, most any kind of multicellular body—is made up of a staggering variety of different cell types. Red blood cells carry oxygen, muscle cells contract and expand our muscles, nerve cells carry electrical signals across (comparatively) vast distances, and a thousand more. And yet, somehow, they all contain the same exact sequences of DNA.

How does one set of instructions give rise to such a staggering variety of end products? The answer is surprisingly simple—different cells use different parts of the genome. Some genes, such as those related to DNA transcription or metabolic activity, are active in all cells. Other, more specific sequences are only found in certain types of cells.

In addition to **differentiating** into different types, cells (even bacterial cells) communicate among each other, a process called **signaling**.

Viruses

In studying biology, prokaryotes are a good start because they are the simplest living things. Eukaryotes, in particular multicellular ones, are relevant because, well, that's what we are—and as far as we know, they are the most complex living things. But we're not done surveying types of life yet, because there are also the entities that are only *sort of* alive, such as **viruses**.

Viruses don't have plasma membranes, or cytoplasm, or metabolic processes, or any of the other machinery normally involved in keeping a cell alive. Instead, they infect more complex organisms and hijack their processes—just as an email virus uses existing programs on an infected machine to propagate.

A typical virus, such as the **lambda phage**, is made up of only two components—a **protein coat**, and a strand of DNA or RNA with instructions for how to make said coat. In spite of this simplicity, the lambda phage has a rather interesting life cycle.

When it encounters the right type of host cell, the coat binds to the cell's plasma membrane and injects its payload of genetic material directly into the cytoplasm. Once it's there, the host cell has no way of telling the difference between its own nucleic acids and those of the invader. The organelles, whose job it is to make proteins, simply "read" the new code and build the new proteins, using the cell's own resources to do it.

Just to make matters more confusing for the poor cell, the first thing many of these illicitly produced proteins do is splice the virus's DNA into one of the host's chromosomes—in the case of the lambda phage, the protein that does this is called the **lambda integrase**. Once the integrase has done its dirty work, the cell continues to grow and reproduce, and as it does so, it passes the viral sequences on to its daughters. Eventually, some environmental signal tells the cells to start copying viral DNA and producing viral proteins as fast as they can. Even as its own processes wither away, the cell continues making new viruses, until it finally bursts and releases thousands upon thousands of viruses to seek new targets.

If we think of DNA as a cell's source code, then a virus like the lambda phage is a sort of self-modifying program. Not only does it hijack the cell's machinery to make copies of itself, but it permanently changes the original code. Often a phage's alterations will eventually be corrected by the cell or its descendants, but sometimes non-executable fragments of phage "code" stay in the genome. After millions of years of evolution, our genome is littered with the remains of viral insertions, sequences known as **transposons**.

Plasmids

And yet, viruses still aren't the simplest possible things that can replicate themselves, however. If a virus is nothing but a bit of DNA or RNA in a protein capsule, a **plasmid** ditches its shell in favor of existence as a free-floating loop of nucleic acid. Once scooped up by a larger cell, the plasmid takes advantage of the same vulnerabilities as the virus and uses the cell's own machinery to make new copies of itself.

Plasmids are particularly common in prokaryotes, where they serve as a way for bacteria to swap bits of genetic information back and forth. A common—and troublesome—example of this is the transmission of antibiotic resistances from one species of bacteria to another. Experimental biologists can also exploit the process to quickly introduce genes they are interested in.

Prions

Plasmids may be the simplest things that qualify as "sort of alive," but there is another type of ultra-simple self-replicating biomolecule out there—**prions**, misfolded proteins most famous as the cause of "mad cow disease,[2]" along with a number of similar neurodegenerative diseases such as kuru and Creutzfeldt-Jakob disease.

A normal, "living" copy of something called the **major prion protein (PrP)** is an important player in the nervous and immune systems. Its exact roles are still unknown, but it's been connected to circadian rhythm, long-term memory, neural plasticity (the brain's ability to change and adapt), and the activation of various immune cells. Whatever it does is clearly vital, though—prion diseases inevitably lead to death.

[2] More properly known as **bovine spongiform encephalopathy** in cows, and **variant Creutzfeldt-Jakob disease** in humans unfortunate enough to eat said cows.

It is possible, however, for healthy PrP (referred to as PrP^{C}, for cellular) to mutate, dramatically changing its three-dimensional structure. The new, misfolded variants (or PrP^{Sc}, after one of the earliest known prion diseases, scrapie) proceed to attach themselves to healthy PrP^{C} molecules and somehow twist them into the same misfolded shape. PrP^{C} becomes PrP^{Sc}, and both copies are released to infect more proteins.

So basically, prions are like zombies—they used to be "living," healthy proteins, but after mutation, they begin to "bite" other healthy proteins and turn them into more zombies…until the brain is nothing but a zombie-infested wasteland.

Cellular Activity

The Central Dogma

Regardless of whether an organism is prokaryotic or eukaryotic, at the cellular level everything boils down to the same process, known as the **central dogma of biology** (Fig. 3).

Long, helical molecules of DNA form a sort of cellular blueprint, containing all the plans and instructions necessary for a cell to function. Individual sections, or genes, are then **transcribed** to smaller molecules of messenger RNA. These, in turn, travel to giant molecular factories known as **ribosomes**, where they will be read and **translated** into the proteins that make up most of a cell's machinery. At that point, the gene is considered to have been **expressed**.

Or, in computer terms, DNA is a stored program, which is "executed" by transcription to RNA and expression as a protein.

Types of RNA

Messenger RNA, ribosomal RNA, and transfer RNA are abbreviated as mRNA, rRNA, and tRNA, respectively. As time goes by, more and more internal uses are discovered for RNA molecules. (Some of these are discussed below, such as CRISPR and gene silencing.) A gene product is a generic term for a molecule (RNA or protein) that is coded for by a gene.

This basic process, the progression from DNA to RNA to protein, is carried out by all living organisms. In fact, many of the genes involved in transcription and translation are identical across wildly different species, or at least highly similar.

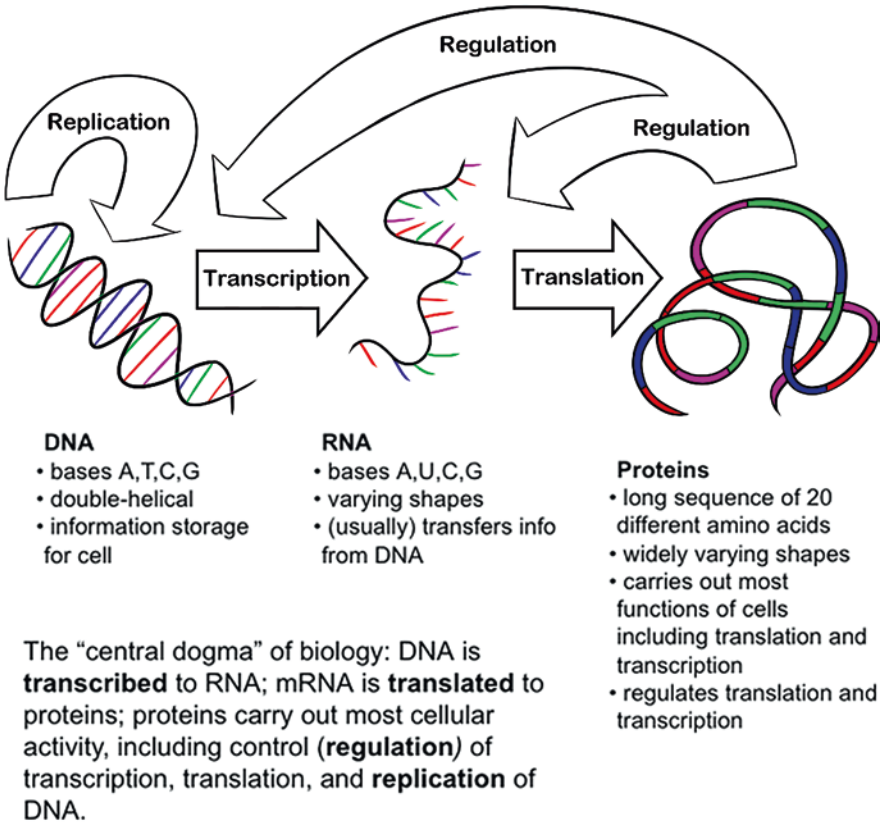


Fig. 3 The "central dogma" of biology

This is, of course, a dramatic oversimplification. The more complicated eukaryotic cells add in an entirely new step of messenger RNA editing—**splicing**—in which sequences known as **introns** are removed and discarded. (Not always the same introns, though—there can be multiple ways to splice the mRNA for a gene, so a single gene can produce many different proteins. Just in case eukaryotes weren't complicated enough.)

Additionally, some RNA molecules perform useful functions without ever being translated into proteins—for example, key parts of ribosomes are made of pure "ribosomal RNA", rather than proteins or lipids.

Even the basic structure of DNA can influence gene expression. Different cells pack the genome in different ways, exposing some segments and hiding others away in densely packed nucleosomes where they cannot be accessed by the cell's transcription machinery.

Cellular Signaling

It would be impossible for multicellular organisms to exist without some way of coordinating the activity of the cells they are made of. Cells are too small and too simple to use things like words and sounds, so instead they "talk" through a bewildering variety of chemical signals (typically some variety of specialized protein).

A **ligand** is a molecule that binds to a specific place on another molecule. The shape of a protein is called its **conformation**.

Once released into the environment, signals float around until they encounter matching **receptor sites** on proteins sticking out of other cells' plasma membranes. The signal binds to the receptor, triggering some sort of chemical change that the cell "understands"—typically, this means releasing more signals, which bind to more receptors, which release more signals, and on and on in long cascades known as **pathways**.

There are four well-studied categories of receptor:

- **Enzyme-linked receptors** stretch across the plasma membrane. When a signal—the receptor protein's ligand—binds to the receptor site outside the cell, it toggles the **enzyme** inside the cell on or off, directly changing what sorts of chemical reactions occur inside the cell.
- **G-protein coupled receptors** (**GPCRs**) are complexes of multiple proteins, both inside and outside the plasma membrane. When the exterior receptor is activated, the protein changes its **conformation** (or shape), releasing a smaller **G protein** *inside* the membrane to float off and activate other processes. (Although it's worth noting that the term "G protein" is something of a misnomer—like the receptor proteins, a single G protein is typically made of several smaller proteins bound together.)
- **Ion channels** are essentially doorways. They form pores in the plasma membrane; when activated, the pores open to allow the passage of molecules that wouldn't normally be able to cross the barrier. Some are activated by signal molecules of one sort or another, but other varieties exist, such as the **voltage-gated ion channels** found in nerve cells that respond to changes in the cell's electrical charge.
- **Nuclear receptors** respond to signals by binding directly to strands of DNA or RNA, changing how that particular gene is expressed.

Receptors—and signaling pathways in general—tend to be of particular interest to biologists working on drug development. It's hard to directly affect the inner workings of a cell: after all, they've had billions of years of evolution to teach them how to seal off their interiors to all but a few specific compounds. If you want to affect a cellular process with a new drug, it's much easier to target receptors—the cell's existing API, so to speak.

Cell Division

Perhaps the most important property separating living organisms from nonliving ones is their ability to reproduce. At the cellular level, the process is called **division**—the separation of a single cell into two identical **daughter cells**, each with their own fresh copy of their parent's genome.

In prokaryotes, the process is simple. First, the DNA "unzips," separating into its two component strands. Proteins called **DNA polymerases** travel along each strand and assemble a new matching strand, nucleotide by nucleotide. When they're done, each loop attaches to a different point on the plasma membrane, which then pinches inward until it splits off into two separate bubbles. This final division is known as **cytokinesis**.

In eukaryotes, however, the presence of multiple chromosomes complicates things. The cell not only has to copy its DNA, it has to make sure that each daughter cell receives one copy of every chromosome—no more, no less. The process is controlled by a set of proteins known as **cyclins** and **cyclin-dependent kinases** (**CDKs**), and can be divided into the four distinct phases of the **cell cycle**:

A **kinase** is a protein that modifies another protein by adding a phosphate group. This process is called **phosphorylation**.

- During the **G1 phase**—also known as the **growth phase**—the cell prepares for division by growing larger and making extra copies of organelles like mitochondria. At the end of the phase, the cell can either return to its normal state (**G0 phase**), or proceed to the next step of the cell cycle.
- During the **S phase**, the chromosomes are duplicated. The two copies, however, remain attached at a point called the **centromere**.
- During the **G2 phase**, the cell assembles a temporary protein scaffold, known as the **mitotic spindle**. At the same time, tumor-suppressant genes such as **p53** check for damaged DNA. If they find any, they'll either repair it or cause the cell to self-destruct; if not, the cell begins to physically divide.

- During the **M**, or **mitosis phase**, the cell separates the two copies of its DNA, pulling one set of chromosomes to each side of the cell before undergoing cytokinesis and finally splitting into two daughter cells.

Mitosis doesn't take long, but it's an impressively complicated process with its own set of distinct, named steps. They are, in sequence, as follows:

- **Prophase**—the DNA packs itself into its most condensed form, creating the familiar "x" shape, and the nucleus dissolves.
- **Prometaphase**—the nucleus breaks apart, allowing the chromosomes to move separately.
- **Metaphase**—the chromosomes form a straight line across the center of the cell and attach themselves to the mitotic spindle assembled during the G2 phase.
- **Anaphase**—the mitotic spindle contracts, pulling the chromosomes apart at the centromere and dragging one copy to each side of the cell.
- **Telophase**—a pair of new nuclei forms around each set of chromosomes.

As with most things in life, everything gets much more complicated when sex is involved. Pretty much every organism capable of sexual reproduction keeps multiple copies of its DNA around in its normal, or **somatic**, cells. Take us, for example—we have two complete sets of chromosomes, one inherited from each parent, making us **diploid** organisms. That's only the bare minimum, though, as organisms have been discovered with three, four, five, or more sets. (The black mulberry, for some reason, has a whopping 44 copies of each chromosome.)

Prokaryotic Sex

Prokaryotes might not enjoy the benefits—and complexities—of sexual reproduction and meiosis, but that doesn't mean they don't have ways of trading genes with each other. Remember when we mentioned plasmids earlier? One important subtype of those is fertility, f-plasmids, which contain all the genes necessary for a process called **conjugation**.

The prokaryote with the f-plasmid—the "male," if you will—constructs a sort of protein grappling hook called a **sex pilus**, which it uses to grab onto another, "female" cell. The two prokaryotes then form a conjugate bridge and freely exchange plasmids—including copies of the f-plasmid.

And if you think sex was complicated for humans, be glad you're not a bacteria. Conjugation typically involves groups of up to ten separate bacteria, with the "females" becoming "male" afterward, thanks to their new copies of the f-plasmid.

But wait. Fertilization involves two cells fusing into one. If both sperm and egg were diploid, the baby would end up with four complete copies of its DNA. Except that doesn't happen—a fertilized egg is diploid, with the usual two copies of DNA. What happens to the extra?

It turns out that our bodies get rid of the excess long before we actually get to the fun parts. Our egg and sperm cells, our **gametes**, have only a single set of DNA each. These **haploid** cells, as they're known, are produced by a variant of the normal cell division process called **meiosis** that results in *two* sets of daughter cells. The full process is broken down in Fig. 4.

But there is, as usual, another layer of complexity. Unlike the cell in Fig. 4, we have more than one chromosome pair. (Twenty-three, to be precise, at least in humans—chromosome numbers vary wildly between different species.)

Consider a diploid cell with *N* chromosome pairs. For convenience's sake, we'll label each chromosome as either A_N or B_N, with N representing the chromosome number and A and B representing the parent it was originally inherited from. When the cell undergoes meiosis, two of the four daughter cells will receive A1, and the other two will receive B1. But just because a particular cell happened to get A1 doesn't mean it'll get A2, A3, and all the rest of the set—chromosome pairs are distributed randomly. The daughter cells will get *either* A1 or B1, *either* A2 or B2, and so on, all the way down the line. That adds up to a whopping 2^N potential sets of DNA, meaning each gamete is almost guaranteed to be different.

And that matters, because there's nothing to say that A1 and B1 are identical sequences. Populations often have multiple varieties, or **alleles**, of many of their genes. (Among many other things, that's why we have different eye and hair colors.) Typically, one allele will be **dominant** over the other—the cell will make its version of the protein, while the other, **recessive allele** is effectively hidden.

And just to make things worse, meiosis *also* often involves some amount of **genetic recombination**, where individual genes are swapped between chromosome pairs. In humans, such **crossover events** typically happen two to three times per chromosome.

An organism with two copies of the same allele for a gene is **homozygous** for that gene. An organism with two different alleles for a gene is **heterozygous** for the gene.

All of this variety and randomization is a big part of the reason why diploid species can be so genetically diverse, and why evolution can occur so much faster than in monoploids—there are far, far more potential genetic outcomes of any given mating than there are of a single cell division.

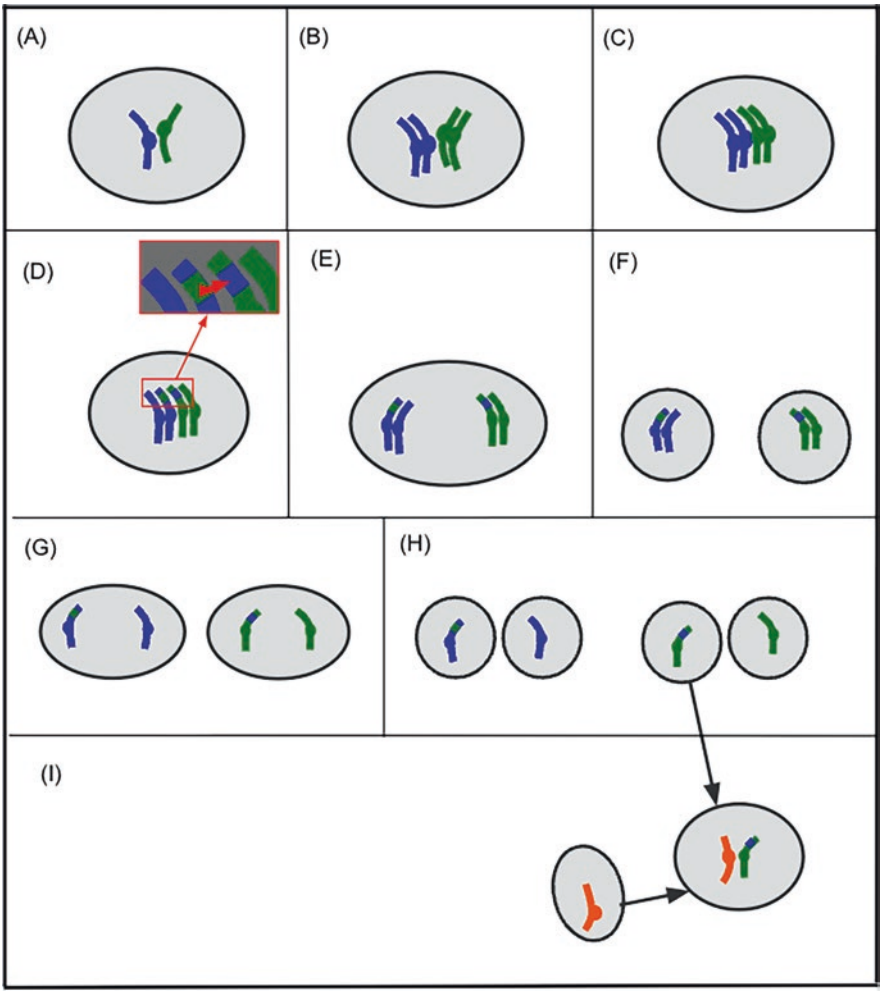


Fig. 4 How meiosis produces haploid cells. (**a**) A **diploid** cell, with one pair of homologous chromosomes. (**b**) After DNA replication the cell has a two pairs of **sister chromatids**. (**c**) The homologous chromatids pair to form a **bivalent** containing four chromatids. (**d**) DNA fragments **recombine**. (**e**) Bivalents are separated in preparation for **division I**. (**f**) The cell divides. Each daughter has two copies of a single parent's chromosome. (**g**) The sister chromatids in each daughter cell separate from each other in preparation for **division II**. (**h**) The daughter cells divide, producing four **haploid** cells, each of which contains a single representative of each chromosome pair from the original diploid cell. (**i**) In sexual reproduction, two haploids fuse to form a **diploid** cell with two homologous copies of each chromosome—one from each parent. Shown here is a cell formed from one of the daughter cells in (**h**), and a second haploid cell from another parent

Why Is Biology Hard?

Biology is a never-ending ocean of complexity. Although the basic mechanics of processes like DNA transcription are the same across most—if not all—kingdoms of life, billions of years of divergent evolution have left us with a bewildering variety of variations on the basic theme. And it can be *hard* to untangle one from the next, for many reasons.

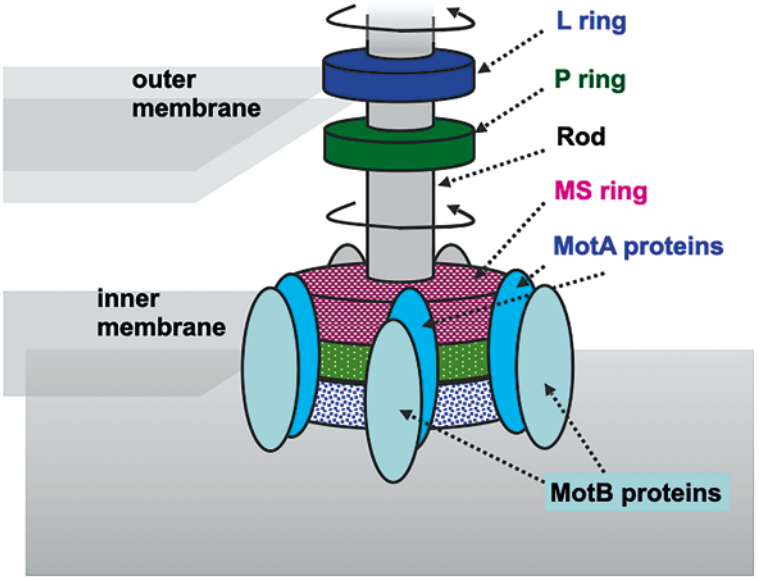
Proteins Interact in Complexes and Pathways

Each individual cell comes equipped with huge lengths of genetic material, and even the simplest single-celled organisms are filled with thousands upon thousands of different types of proteins. Each protein, in turn, is a three-dimensional shape where every twist and turn affect its function. These proteins, in turn, sometimes form **protein complexes**, made from multiple individual amino acid chains braided together.

How complex can a protein complex get? As one example, many bacteria have evolved to move by spinning long hair-like proteins called **flagellum**, like a boat spinning its propeller (see Fig. 1 for a *simplified* view). The "molecular motor" that they use to do so is made up of dozens of copies of 20 different sub-proteins—and it's far from the largest or most complicated protein complex out there.

And if protein structure is convoluted, their interactions—with each other, with the environment, or with their cell's nucleic acids—can be positively byzantine, full of wandering pathways and feedback loops. It's not the cells' fault, though. No one ever sat down and designed these processes (sorry

W. W. Cohen, C. K. Cohen, *A Computer Scientist's Guide to Cell Biology*,
https://doi.org/10.1007/978-3-031-55907-5_3



Structure of a bacterial flagellum (simplified). About 40 different proteins form this complex. The MS ring is made up of about 30 FliG subunits, and about 11 MotA/MotB protein pairs surround the MS ring. It is believed that these pairs, together with FliG, form an ion channel. As ions pass through the channel, conformational changes cause the MS ring to rotate, much like a waterwheel.

A similar "molecular motor" is used in ATP synthesis in a mitochondrion: rotation, driven by ions flowing through a channel, is the energy used to convert ADP to ATP. (See the section below, "Energy and Pathways").

Fig. 1 The bacterial flagellum (simplified)

"intelligent design" believers); they're the messy end result of 3.7 billion years of throwing everything at the wall and seeing what sticks.

Take, for instance, one of the most widely studied processes in *E. coli* bacteria, one that turns on the bacteria's ability to eat lactose when its preferred sugar, glucose, isn't available. Figure 2 shows a *simplified* summary of the process, and even that much is a nightmare involving multiple levels of gene expression, promoters, inhibitors, and feedback loops.

Almost every process taking place in any cell is at least this complicated.

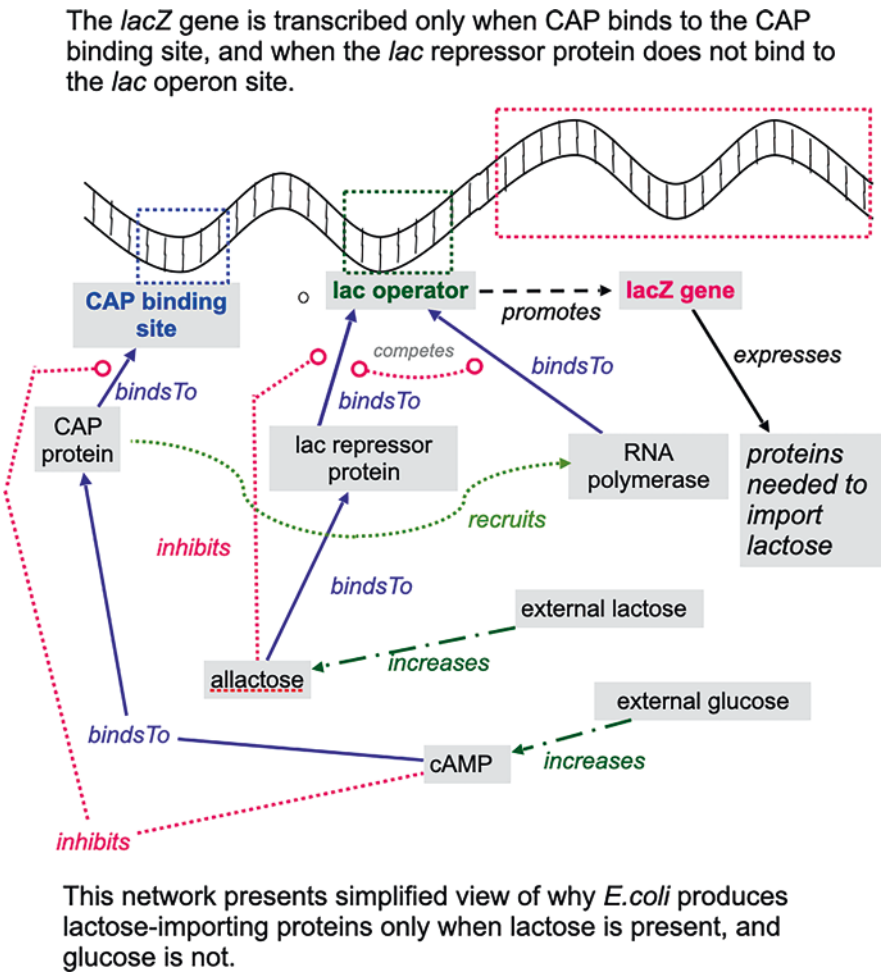


Fig. 2 How *E. coli* responds to nutrients

Individual Interactions Can Be Complicated

Enzymes Control Reaction Rates

To make matters worse, it's not just a question of qualitative interactions—this protein turns on that gene which inhibits this protein that digests this sugar, and on and on and on. There's also *quantitative* complexity, because these processes don't exist in purely linear relationships.

Most of the individual "steps" of a pathway—indeed, most biological reactions, period—are the result of a special type of protein called an **enzyme**. Its role is to bond to a specific target called a **substrate** and encourage, or **catalyze**, a given change, increasing the rate at which it happens by up to three orders of magnitude. But enzymes aren't actually *part* of the reaction they're catalyzing, as such; all they're doing is holding everything in place and making it easier for the reaction to occur. That means they're not used up in the process.

Imagine that you and a friend have just gone apple-picking, and now you're ready to make a pie. Your friend starts peeling apples; when they're done, they hand the apple to you to cut. If their job takes 2 minutes and yours only takes one, then your friend is the one holding things up. Add a second friend, and you're getting an apple a minute, just as fast as you can cut—your productivity just doubled! But now imagine a third friend wanders in and picks up a peeler, so now you're getting three apples every 2 minutes. That part of the process is moving faster, but you can't; not without cutting off a finger. The extra help has no effect, and you're still only finishing with one apple a minute (Fig. 3).

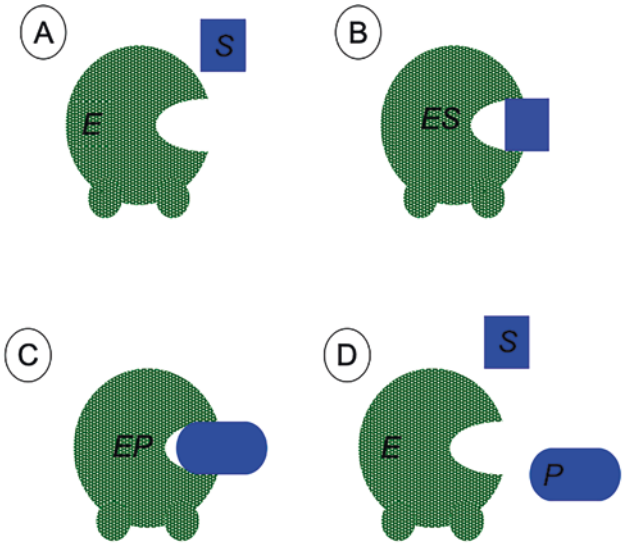
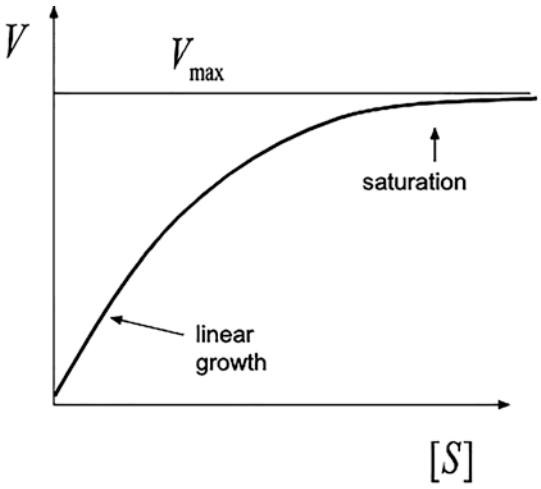


Fig. 3 How enzymes work. A cartoon showing how an enzyme catalyzes a change from *S* to *P*. (**a**) Initially, the enzyme *E* and "substrate" *S* are separate. (**b**) They then collide, and bind to form a "complex" *ES*. (**c**) While bound to *E*, forces on the substrate *S* cause it to change to form the "product" *P*. (**d**) The product is released, and the enzyme is ready to interact with another substrate molecule *S*. A chemist would summarize this as: E + S → ES → EP → E + P



Reaction velocity with a fixed quantity of an enzyme *E*, and varying amounts of substrate *S*. When little substrate is present, an enzyme *E* to catalyze the reaction is quickly found, so reaction velocity *V* grows linearly in substrate quantity [*S*]. For large amounts of substrate, availability of enzymes *E* becomes a bottleneck.

Fig. 4 Saturation kinetics for enzymes

The same dynamic takes place in cells. If we change your name to "enzyme E," the peeled uncut apples "substrate S," and the cut apples to "product P", then Fig. 3 illustrates the chemical process underlying "apple-cutting". We could also plot the speed at which the bowl fills up with sliced apples as "velocity V," and we can make a nice graph of the **saturation curve**, as we see in Fig. 4. Velocity will increase until all the enzymes are being used at maximum speed.

The math behind **saturation kinetics** is surprisingly accessible—it's based on basic probability and a few assumptions. If you're interested, Figs. 5 and 6 summarize the basic theory, which is called the **Michaelis-Menten model**.

Reaction Rates Can Be Highly Nonlinear

But then again, this is biology—there's always another layer of complexity. Enzymes can have different numbers of **active sites**, the point at which the substrate binds to the protein for processing. These active sites, in turn, can have multiple conformations, or shapes, where different shapes catalyze reactions at different rates.

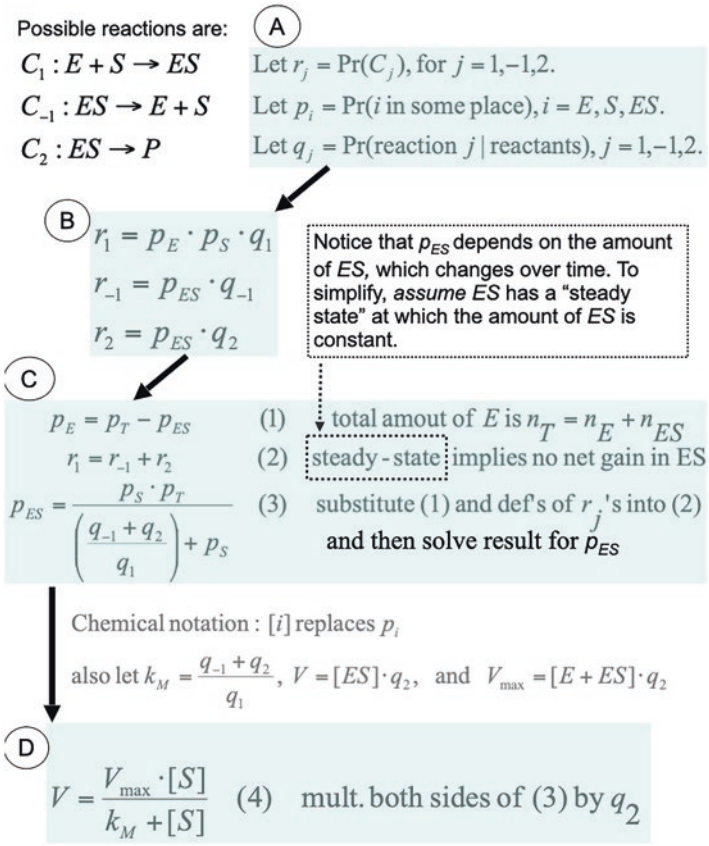


Fig. 5 Derivation of Michaelis-Menten saturation kinetics

To return to our previous metaphor, imagine if, partway through your task, you find one of those circular apple-cutters that core and chop an entire apple at once. That would let you cut apples twice as fast, effectively doubling your "velocity." If you looked at the graph of your progress, you'd see the line transition from your knife-velocity (VmaxSlow) to your cutter-velocity (VmaxFast). In your case, there wouldn't be any transition—you'd just switch from slow to fast—but that's because there's only one of you. If you had ten times as many peelers and ten times as many slicers, you'd see a more gradual change as one slicer after another switches implements.

Essentially the same thing happens when an enzyme switches from using slow-binding sites to fast-binding ones. The resulting shape is known as a **sigmoid curve**, shown at the top of Fig. 7, and serves as a smooth approximation of a step-function.

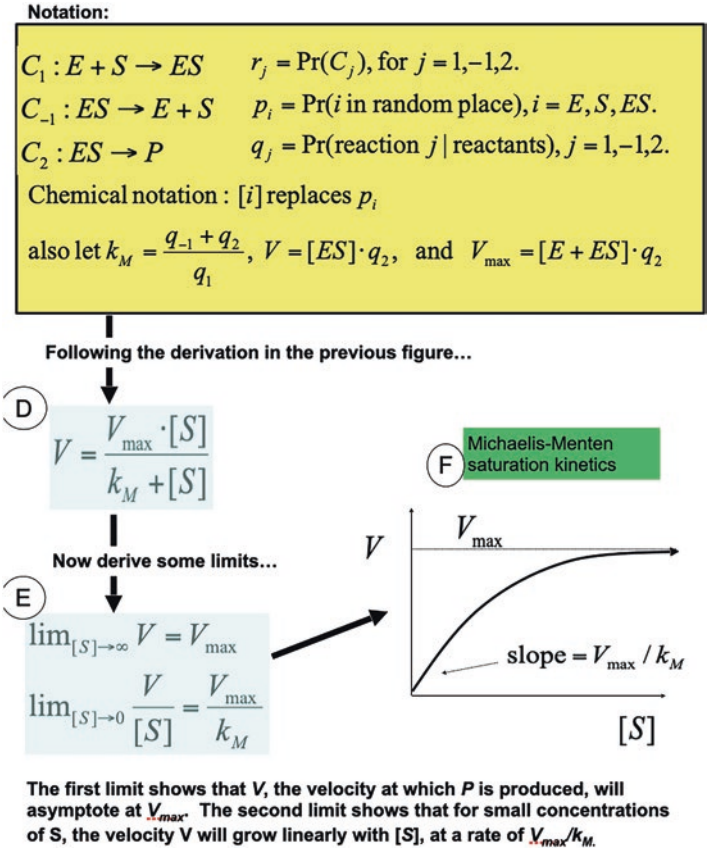
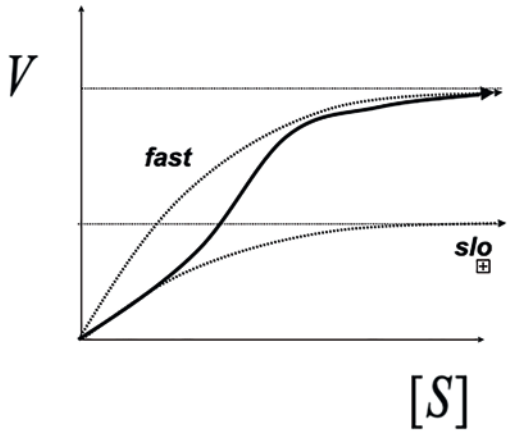
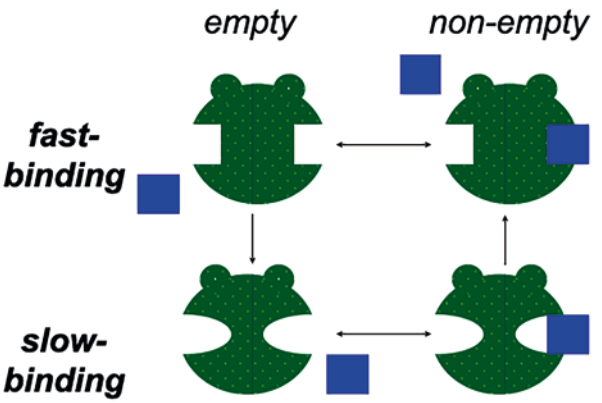


Fig. 6 Interpreting Michaelis-Menten saturation kinetics

Sigmoid curves crop up in computer science, too—especially machine learning, where they're commonly used to define **neural networks**. A neural network is simply a directed graph in which the "activation level" of each node is a sigmoid function of the sum of the activation levels of all its input (i.e., parent) nodes. It is well known that neural networks are very computationally expressive: for instance, finite-depth neural networks can compute any continuous function, or any Boolean function. Protein interaction networks are likely to be similarly expressive—possibly Turing-complete, in the case of feedback loops.



Allosteric enzymes switch from a slow-binding state to a fast-binding state, and tend to remain in the fast-binding state when the substrate *S* is common. Their kinetics follows a sigmoid curve.



A typical allosteric enzyme: when one half is being used, the whole molecule tends to shift to the fast-binding state.

Fig. 7 An enzyme with a sigmoidal concentration-velocity curve

Enzymatic Pathways Are Complicated

Cellular Energy

Running all the machinery of a cell takes energy. Enzymes can make reactions more efficient, but most are still **endothermic**—that is to say, they require more energy to begin than they release when they're done. Typically, this energy comes from sugars and other "food" molecules, but that's only the start of the process. Getting that energy where it needs to go can be just as important.

So how do cells do it? Simple—they use waterwheels to charge batteries, which they plug into proteins to power them.

Although that sounds absurd, it's a lot closer to reality than you'd think.

Once eaten, sugar molecules eventually make their way to an organelle called the **mitochondria**. There it is broken down by a series of **exothermic** reactions—those that release more energy than they need to start. (**Combustion**, for example, is a *very* exothermic reaction.) The actual processes are painfully complicated, and along the way a lot of positively charged hydrogen ions wind up being forced into a second, smaller membrane.

The result—a bubble full of hydrogen ions, inside one that's practically empty—is not sustainable. The ions repel each other, and a process called **diffusion** pushes them to move to an area where there aren't so many of them. Unfortunately, they can't cross the membrane on their own. Their only option is to pass through a protein complex called **ATP synthase**, where, just like water flowing over a wooden wheel, the flow of ions literally causes an element of the protein to revolve (a little like the bacterial flagellum of Fig. 1).

That revolution, in turn, attaches an extra phosphate group to a molecule called **adenosine diphosphate (ADP)**, creating our "battery," **adenosine triphosphate (ATP)**.

ATP disperses across the cell, interacting with all sorts of proteins. Some of them—many enzymes, for instance—will "grab" onto the battery and pull that third phosphate group right back off, releasing energy once again. At the same time, they carry out a second, energy-*dependent* reaction. When the dust settles, the enzyme releases both the newly altered substrate and a molecule of ADP—the uncharged version of our battery.

More properly, ATP is combined with water to produce ADP plus inorganic phosphate, yielding energy: ATP + H_20 → ADP + Pi. This reaction is called **hydrolysis**.

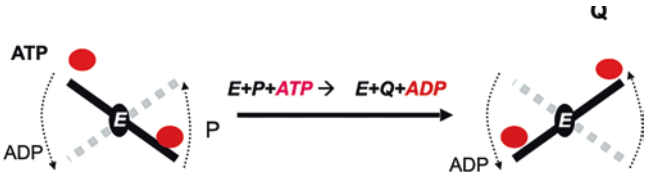


Fig. 8 A coupled reaction

This process, exploiting an exothermic reaction to "power" an endothermic one, is known as **coupling**. Think of a seesaw; pushing down on one side (the energy-releasing reaction) causes the other side (the energy-dependent one) to rise. Coupling is also used throughout the sugar-digesting and ATP-charging process. (Coupling is shown in Fig. 8, with dotted lines around a shape to indicate the high-energy form of that molecule.)

Enzymatic Pathways Have Many Steps

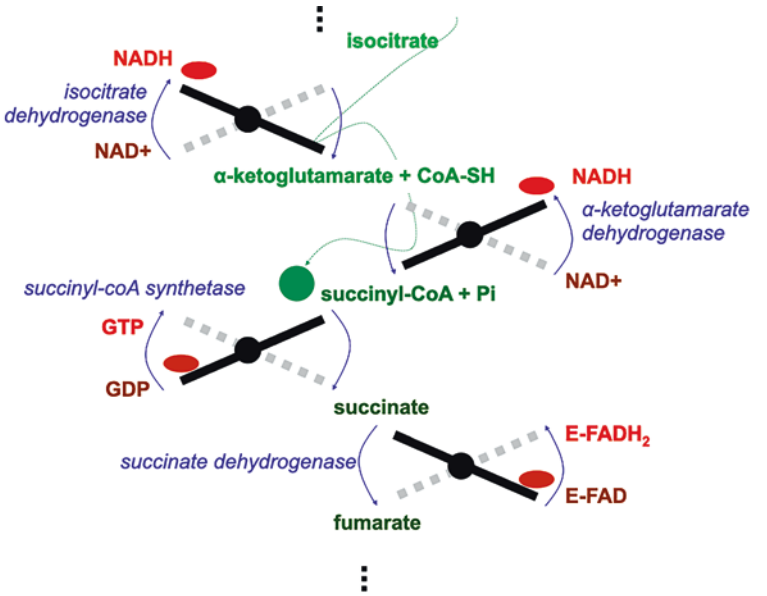
ATP is a very common energy source, but there's only so much power you can get out of one ATP. This means a pathway that produces or consumes lots of energy will need a lot of coupled reactions, involving many different enzymes, again contributing to complexity. Figure 9 shows an example of an **enzymatic pathway**.

This sort of pathway is hard to understand, but the complexity is a consequence of energy usage, and other constraints on chemistry—a single enzyme can only alter its substrate so much, but cells can build a *lot* of different enzymes, and if enzyme A can only carry out half the reaction it needs, it can just add a different enzyme B to finish the job. String together enough of these interactions, where the product of one enzyme becomes the input of the next, and you can carry out some truly impressive cellular alchemy.

Amplification and Pathways

An additional benefit of these long biological pathways is that they can serve as signal amplifiers—at any or all steps of the pathway, a single newly activated enzyme can go on to catalyze thousands of reactions and unleash a flood of signals to downstream proteins.

For an example of this amplification in action, look at **rhodopsin**, one of the key proteins in human vision. In its normal state, its only job is to sit in the cell membrane and hang onto a particular derivative of vitamin A called



Part of the TCA cycle (also called the citric acid cycle or the Krebs cycle) in action. A high-energy molecule of **isocitrate** has been converted to a lower-energy molecule called **α-ketoglutamarate** and then to a still lower-energy molecule, **succinyl-CoA** (as shown by the path taken by the green circle). In the process two low-energy **NAD+** molecules have been converted to high-energy **NADH** molecules. Each "see-saw" is an enzyme (named in *italics*) that couples the two reactions. The next steps in the cycle will convert the **succinyl-CoA** to **succinate** and then **fumarate**, producing two more high-energy molecules, **GTP** and **E-FADH$_2$**.

Fig. 9 Part of an energy-producing pathway

11-cis-retinal. Retinal is unusually good at absorbing light—but in the process, it changes shape into **all-trans-retinal**. Rhodopsin isn't nearly as good at holding the new form, so the molecule pops free. This, in turn, prompts rhodopsin to shift into its "active" form, which is capable of activating a second protein called **transducin**. Transducin activates a third protein called **cGMP phosphodiesterase (PDE)**, which begins to break down the titular cGMP, or **cyclic guanine monophosphate** (**cGMP**). As more and more of the cell's supply of cGMP is activated, ion channels begin to close. Positively charged sodium and calcium ions can no longer flow freely into the cell, causing its electric charge to change, which ultimately sends an electrical signal to your optic nerve and lets your brain know that you've "seen" something.

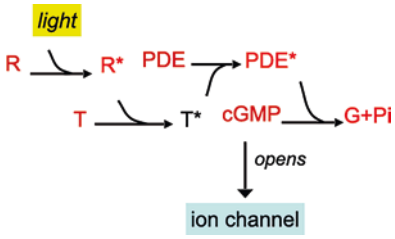


Fig. 10 How light is detected by rhodopsin

Figure 10 lays the process out visually. When you look at it, just remember that **R** is rhodopsin, **T** is transducin, and a* denotes the active form of the protein.

In simple terms, protein R turns on protein T, which turns on protein P, which breaks down molecule C, which closes ion channels.

The key here is that activating transducin doesn't "turn off" the original rhodopsin protein. It remains active, allowing it to continue its job of activating transducin. PDE works the same way, with each individual protein going on to break down thousands of cGMP molecules. The result is a very fast change—quadratic, in fact—with that single original photon ultimately affecting the entire cell. This is shown in Fig. 11.

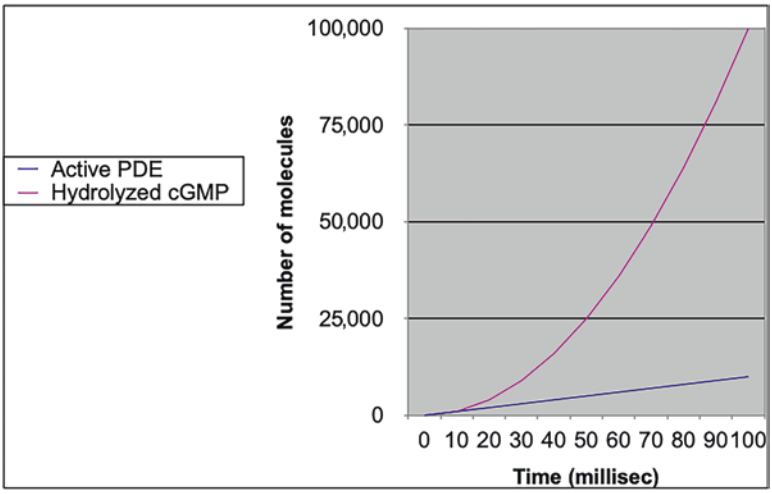
Modularity and Locality Is Limited

How Things That Interact Find Each Other

There are some things about the way the world works that are so intuitive that we don't even think about them. Time moves forward. Objects (when there's gravity) fall. Things can only affect other things when they're close together.

The last principle—known to physicists as **locality**—isn't something computer scientists have to deal with. In a computer program, anything can affect anything else. In response, to make programs comprehensible, we've been forced to invent elaborate schemes to provide the same sort of constraints on behavior that are provided by locality—inventions like interfaces, packages, and private variables are all ways to "localize" interactions and prevent bits of software from interacting in unexpected ways.

To a large degree, cellular processes also lack locality. Even though living cells require millions of copies of thousands of different proteins and



Number of molecules affected over time, assuming that each R* activates 100 transducin per second and each PDE hydrolyses 100 cGMP per second. (The actual numbers are larger).

The number of hydrolyzed cGMP molecules grows rapidly—at a quadratic rate —because it is product of two stages of "linear amplification". More stages of amplification would produce even steeper response curves.

Fig. 11 Amplification rates of two biological processes

chemicals to function, they are—at least in prokaryotes—often not tied to any specific location. All of these components just sort of float around, drifting through the cytoplasm via **diffusion** (random movement) until they run into something they can interact with.

Membranes and Locality

This seems like an inefficient way of handling things, and it is. Molecules move faster than you might think (a molecule of air at room temperature moves at hundreds of miles per hour), but even so, the sheer randomness of the process limits how much ground they can cover. If we take a moment to model an object moving in a random walk or diffusion process—that is to say, one that moves a fixed distance in a random direction with every "tick" of time—we quickly find that the time it takes to get from point A to point B by a random walk depends on the square of the distance.

The consequence of this is that if a cell is small enough—say, prokaryote-sized—diffusion is a fine way of moving things around. In effect, the distances are so short that everything is constantly bumping into everything else. But

when things are larger, diffusion is too slow, and eukaryotic cells are just too large to rely just on random walks to orchestrate interactions. In order to reach the size of eukaryotic cells, it was necessary for life to evolve new tools to move things around so that they could interact when they need to.

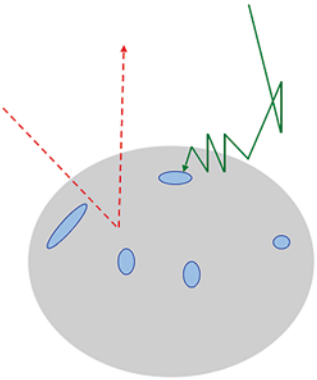
Improving over diffusion is one of the uses of membranes in eukaryotes. Of course, cells use membranes to "wall off" an area, and once enclosed, the movement of the molecules within the enclosed bubble is restricted, and interactions become more common. A less obvious way to make diffusion more efficient is by attaching proteins, or other molecules that need to interact, to a membrane. For instance, the enzymes used by cells to digest sugar are all localized to the inner membrane of the mitochondria.

An organelle is a discrete component of a cell. Some but not all organelles are membrane-enclosed areas.

Intuitively, membranes should help because while things still move by diffusion, they move in a limited, two-dimensional area.[1] However, attaching things to a membrane helps more than one might expect, due to probabilistic reasons explored in Fig. 12. As the dynamics work out, molecules that come close to an organelle tend to remain close, and brush against it multiple times, getting another chance to "get lucky" and running into a relevant receptor each time.

Between this unintuitive dynamic and the limited movement that comes from attaching proteins to a membrane, receptors on organelles can be surprisingly efficient. As an example, if only 0.02% of a typical eukaryotic cell's surface has a receptor for *p*, the cell will be about half as efficient as if the *entire* surface were coated with receptors for *p*. This means that at the moderate cost of decreasing *p*'s efficiency by half, the cell could make room for many, many more receptors. The surfaces of cells and organelles can thus be said to have a high "bandwidth"—they can recognize or absorb hundreds of different chemical signals at the same time.

[1] Cell membranes are, like blood, thicker than water—a free-floating protein can diffuse approximately one hundred times faster than a membrane-bound one. The speed reduction is more than made up for by the efficiency of diffusing along two dimensions instead of three.



It can be shown that if a particle is released at distance δ from the surface of a sphere of radius R, it will touch the sphere before diffusing away with probability $p = R/(R+\delta)$. (See the book by Berg, 1983, cited in the last section, equations 3.1-3.5.) If the particle hits the sphere, bounces off, and returns to distance δ again, it has *another* chance to hit the sphere, again with probability p, so the expected number of times n it hits the sphere before diffusing away is

$$E[n] = \sum_{n=0}^{0} n \cdot \Pr(\text{exactly } n \text{ hits})$$

$$= \sum_{n=0}^{0} n \cdot p^n (1-p) = \frac{p}{1-p} = \frac{R}{\delta}$$

This means that a protein nearing a relatively large membrane-enclosed object (like a cell or organelle) is more likely to follow a path like the solid line than the dashed line—it will typically hit the cell many times before diffusing away, giving it many chances to "find" a receptor.

Fig. 12 The behavior of particles moving by diffusion

Transport by Vesicles and Along Microtubules

Another trick is to use a little membrane bubble to move lots of molecules in the same direction at the same time. When these bubbles, or **vesicles**, reach the desired location—as determined by factors such as membrane-bound proteins—they typically merge with another membrane-bound space, releasing their contents directly into the interior.

Not all **transport** methods involve membranes, though. Eukaryotic cells also build themselves a sort of protein scaffold (or **cytoskeleton**, if you prefer) to help them maintain their shape. It's easy enough for protein "engines" to attach themselves to the individual fibers—the **microtubules**—and use them as highways, dragging other molecules along with them.

Transport in eukaryotic cells leads to locality, and hence to some degree of modularity, which can be used to help understand cellular processes: often if you can work out a protein's typical **subcellular location**, you can make a good guess about its function.

Biological Processes Can Cross Membranes

It's helpful to know where a protein is located, but that will never be the complete story of what it does, because biological processes don't necessarily stop at membrane boundaries. Plasma membranes aren't like brick walls—some small particles, like calcium ions, can slip right past the "barrier."

And although larger molecules, such as proteins and sugars, might not be able to pass through a membrane on their own, cells use a dizzying variety of **channels** to ease their passage. Channels are often highly selective, only opening for very specific molecules; some are even capable of forcing other molecules across a membrane, against their tendency to diffuse away from areas of high concentration (although doing so requires energy). There are also ways that information can cross a barrier without actually sending any substances through it, by using **receptors**.

Receptors

Receptors extend through the membrane on both sides, and allow signals to be sent through the membrane. As an example, one well-studied class of receptors are **G-protein coupled receptor proteins (GPCRs)**. These receptors extend through the membrane on both sides. After the outside end of a GPCR binds to its target ligand, it changes shape in a way that a partner protein, inside the membrane, can detect. Typically, the partner G protein is a small collection of proteins bound together, some of which are released after the receptor detects the ligand. This process is shown in Fig. 13.

Another well-studied example of such a receptor protein is **rhodopsin** (also mentioned above), a protein found in our retina. Rhodopsin is somewhat atypical in that it responds to light, rather than a chemical stimulus.

Receptor proteins (and signaling pathways in general) are important clinically, because they provide a convenient way for drugs to affect an organism. In general, cells make it difficult for outsiders to move chemicals across the plasma membrane; if you want to make them behave, it is often easiest to exploit existing signaling mechanisms.

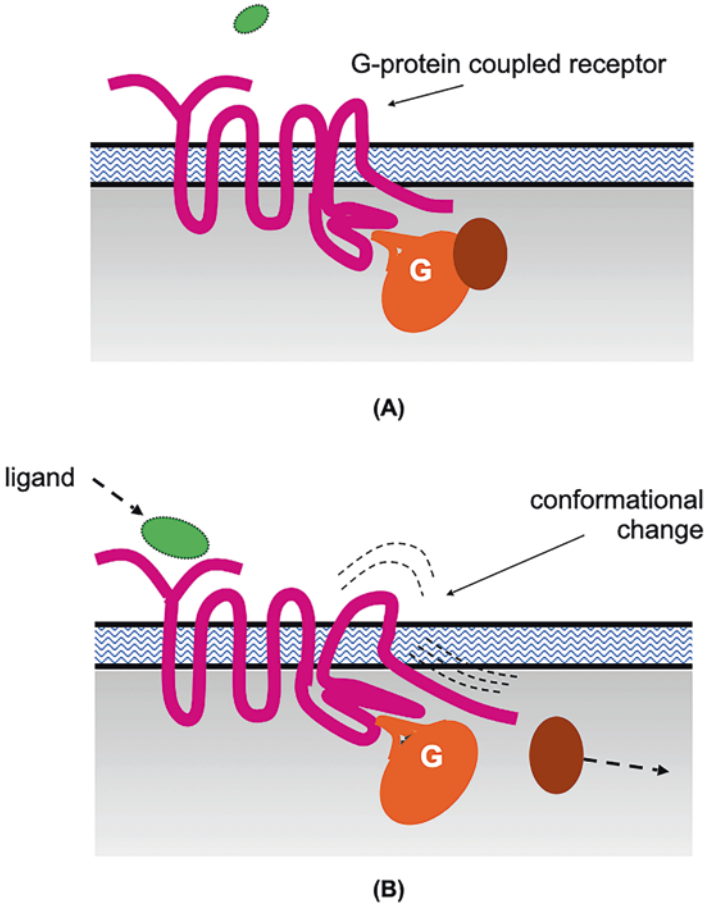


Fig. 13 How a G-protein-coupled receptor protein works. (**a**) A G-protein complex is bound to the G-protein coupled receptor on the inside of the cell. (There are many different types of G-proteins, and many types of receptors.) (**b**) When the receptor binds to the **ligand** molecule, then the entire receptor changes shape. As a consequence, the G-protein complex is altered: part of it is released, to propagate the signal elsewhere in the cell

Ion Channels and Neural Signals: How Nerves Talk

While we're on the subject of intercellular communication, we might as well take a brief detour to discuss biology's most famous type of information processing: the nervous system.

The cells that make up the nervous system—**neurons**—are kind of weird, even by cellular standards, featuring two unique structures.

The second thing you'd probably notice, looking at a picture of a neuron, is that the thing has branches. Lots and lots of thin, branching protrusions jut out in every direction, just like a tree. These **dendrites** are the neuron's ears—the places where it can detect incoming signals from fellow neurons.

The *first* thing you might notice, though, is that one of those dendrites is really, really long. Ridiculously so. The cell's **soma**—the blobby bit in the middle where it keeps its nucleus and organelles—might be anywhere from 4 to 100 micrometers in diameter. This particular protrusion, known as an **axon**, might be anywhere from a thousand micrometers to several *meters* long.

Over that kind of distance, diffusion simply isn't fast enough to pass on a signal. And so, like a nineteenth-century engineer setting up a telegraph, evolution got electric.

How Nerves Talk

The interior of a cell has a relatively low concentration of positively charged sodium and potassium ions; the extracellular fluid surrounding them is packed with such ions. Normally, these particles can't pass through the cell membrane. Open an **ion channel**, though, and they'll come pouring in, creating a spike in the local electric potential.

That voltage spike signals nearby **voltage-gated ion channels** to open as well. More positively charged ions rush in, the high-voltage area expands, triggering more voltage-gated ion channels, and on and on until you reach the end of the axon.

The resulting signal is known as an **action potential**, and it works kind of like a "wave" at a football game (Fig. 14).

(If you're wondering why the action potential only travels in one direction, it's because voltage-gated ion channels have a refractory period. Once opened, they close again quickly, and can't open again for some time. Meanwhile, other transmembrane proteins work to "pump" the newly arrived ions back out of the cell, maintaining the necessary concentration gradient as shown in Fig. 15).

When the action potential reaches the end of the axon, it triggers the release of a chemical signal called a **neurotransmitter**. These signals diffuse across the narrow **synaptic cleft** between the sending axon and the receiving dendrite on the other end. Once there, they trigger **transmitter-gated ion channels** to open, and the action potential continues to travel from neuron to neuron, as shown in Fig. 16.

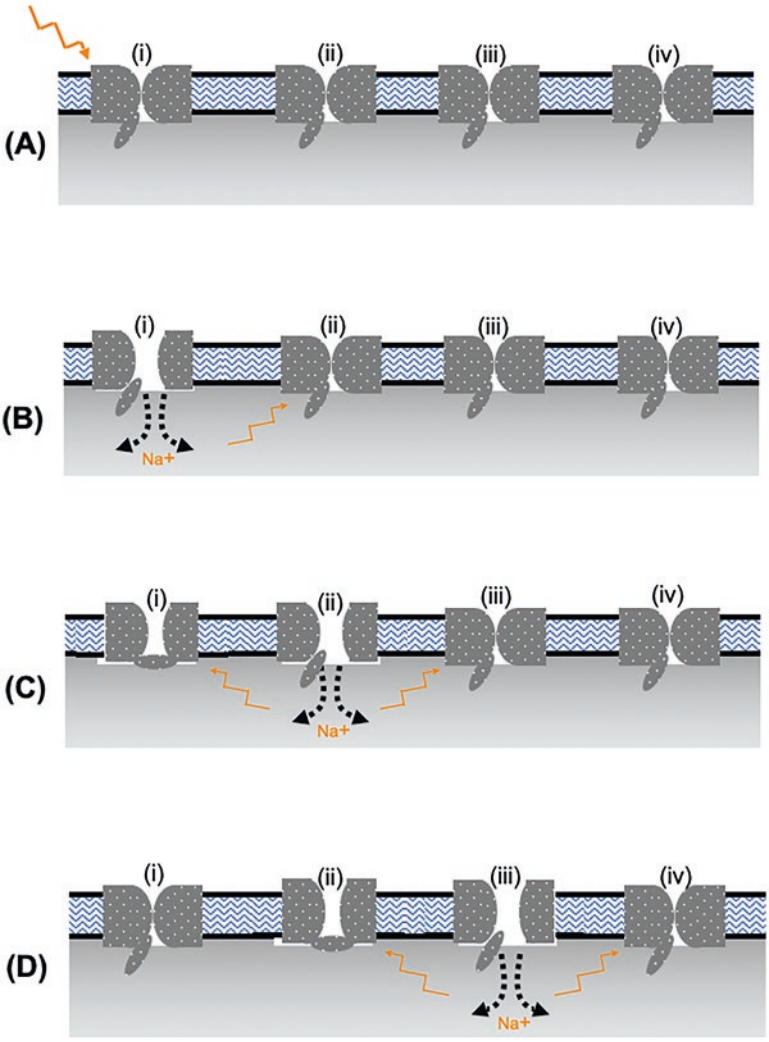
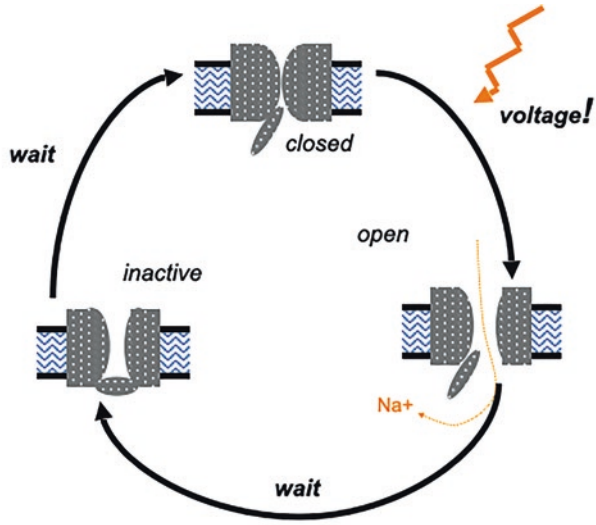


Fig. 14 How signals propagate in a neuron. (a) How a voltage signal travels down a neuron like a wave. First, a voltage signal hits channel (i), as shown in (**a**). Then channel (i) opens, and ions rush in, causing a voltage spike that opens channel (ii), as shown in (**b**). Then channel (ii) opens, sending voltage spikes to channels (i) and (iii), as shown in (**c**). Next, channel (iii) opens, as shown in (**d**). Because (i) is inactive, it cannot open. Ion-produced voltage spikes are now sent to the inactive channel (ii) and the closed channel (iv). Channel (iv) will open next



A voltage-gated ion channel with three states: **closed**, which opens in response to voltage; **open**, which allows ions to pass through; and **inactive**, which blocks ions, and does not respond to voltage. The open and inactive states are temporary.

Fig. 15 A voltage-gated ion channel

Any leftover neurotransmitter molecules are reabsorbed by the axon, through a process known as **reuptake**.

Wrap-Up

If you only remember one thing from this chapter, let it be this: cells are *complicated*, and they are complicated for a variety of complicated reasons. Even the simplest organisms rely on hundreds of intricate systems, and more complex cells exceed that by entire orders of magnitude. The study of how these components interact is known as **systems biology**.

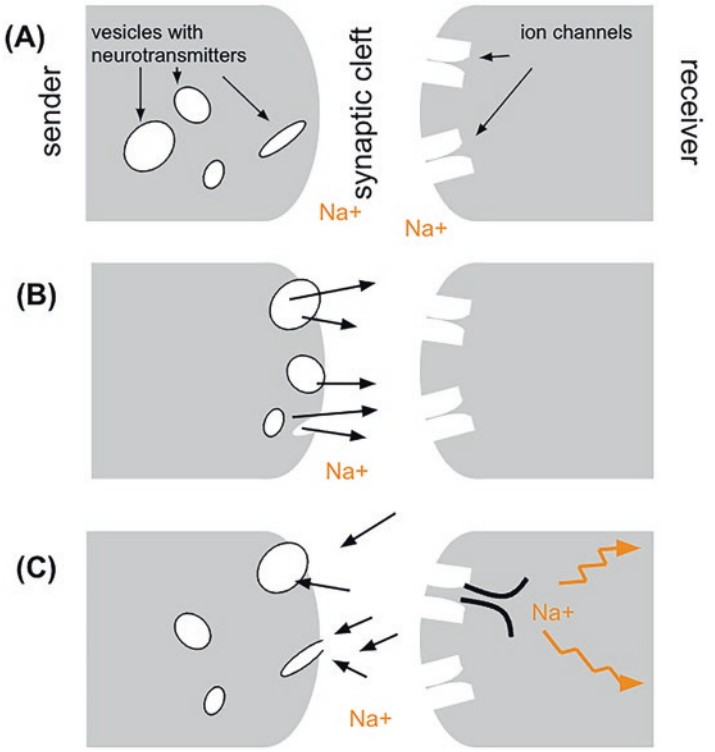


Fig. 16 A transmitter-gated ion channel. (**a**) Shows the initial state. A substance used for signaling (for neurons, this is called a **neurotransmitter**) is held in vesicles by the sender cell. (**b**) In response to some internal change, the neurotransmitter is released. (**c**) Some of the neurotransmitter binds to ion channels on the receiver cell, and causes the channels to open. Most of the remainder of the neurotransmitter is reabsorbed by the sender cell, in a process called **re-uptake**A common neurotransmitter is **serotonin** (which is chemically related to the amino acid tryptophan). Many widely-used **antidepressants** (Prozac, Zoloft, and others) inhibit the reuptake step for serotonin, and are thus called **selective serotonin re-uptake inhibitors (SSRIs)**. They cause serotonin to accumulate in the synaptic cleft, making it more likely that signals will propagate from cell to cell

Looking at Very Small Things

Limitations of Optical Microscopes

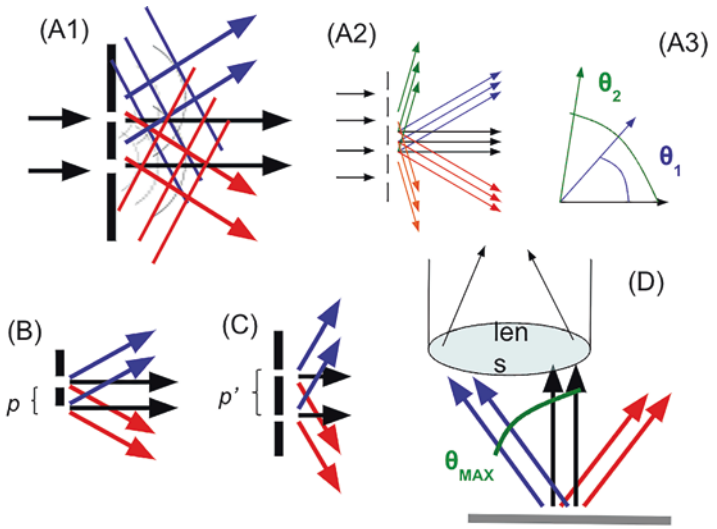
You can't understand a system as complicated as a living being without first gathering information about it. Over the centuries, biologists have come up with any number of ways to probe what's happening inside the body, or inside the cell, but the oldest—and perhaps still most important—is the humble microscope.

The relationship between biology and microscopes—specifically **optical microscopes**, those that work with visible light—goes beyond just making small things big. Unlike X-rays and other types of high-energy radiation, visible light doesn't damage the cells you're looking at. Better yet, since cells are mostly filled with water, that same light can pass right through them and let us see their inner workings in action.

(The idea of a transparent cell might seem strange to those that think of themselves as largely opaque, but remember—even something as small as a mouse is made up of hundreds of millions of cells. The small amount of light scattered by each cell quickly adds up.)

But while optical microscopes remain a critical tool for biological research, they have their limits. In particular, the wavelength of visible light imposes a hard limit on the resolution of optical microscopes. At very small scales, individual light waves begin to interfere with one another, distorting images—a single point might appear to be a series of concentric circles, for instance. For some simple objects, math can help us analyze the result of interference: Fig. 1 summarizes one such result, which shows how the wavelength λ of light and the **aperture** of a microscope—the width of the entry pupil—limits the

W. W. Cohen, C. K. Cohen, *A Computer Scientist's Guide to Cell Biology*,
https://doi.org/10.1007/978-3-031-55907-5_4



Abbe model of resolution: (A) Light passing through two pinholes propagates outward beyond the pinholes much as waves in water would (arcs in A1). Constructive interference between these waves (suggested by dotted lines) causes light to emerge only at certain angles (grey rays) called **diffraction orders**. A "perfect storm" for constructive interference of light with wavelength λ occurs when many pinholes are placed at a uniform distance p (A2); then the diffraction orders (A3) are at angles θ_1, θ_2, θ_3, etc, such that

$$p \sin \theta_N = N\lambda$$

Different spacings p,p' between the pinhole will lead to different diffraction angles (B), (C). To get enough information to determine the separation between pinholes, a microscope needs to capture rays from at least two diffraction orders. The **aperture** (width) of the microscope limits the angle between these to some θ_{MAX} and solving the equation above implies

$$p > \lambda / \sin \theta_{MAX}$$

Unless this holds, the two pinholes cannot be resolved.

Fig. 1 The Abbe model of resolution

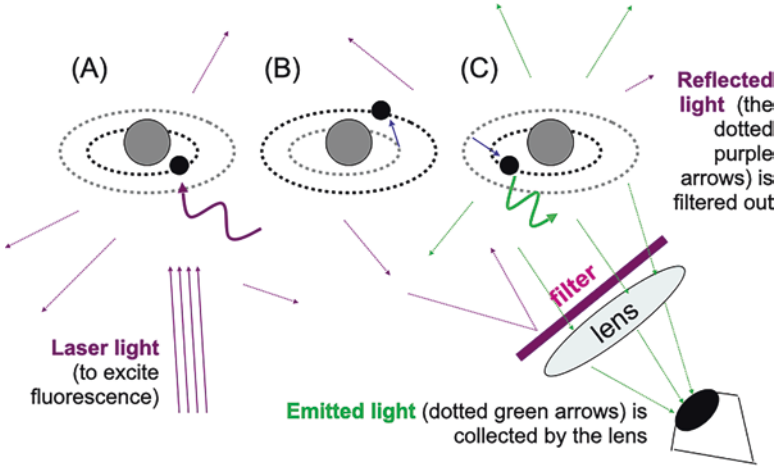
amount of detail that can be distinguished for one class of simple objects. Since visible light has a wavelength of around 0.5 micrometers (μm), objects smaller than around 2 μm are impossible to resolve with an optical microscope, no matter how powerful.

You can see the movement of individual yeast or HeLa cells with a basic high-school-science-lab microscope, and maybe even pick out the nucleus, but that's only scratching the surface of what's really going on: many of the things we're interested in studying are *really* small. A laboratory-grade microscope might be enough to let you examine some of the organelles, but biologists are usually trying to figure out what's going on with individual genes and proteins. Not even the best microscope will be much help there, especially if you want to see them in action.

Fluorescent Microscopes

Fluorescent molecules, or **fluorophores**, process light a little differently than their nonfluorescent kin. When struck by a photon of the right frequency (*f*), one of the molecule's electrons will be excited, or forced into a higher energy state than it wants to be. This change is inherently unstable, and within a nanosecond the electron falls back to its previous position—but in the process, it releases a photon of a *different* frequency (*g*) than the one that first excited it. The result is that the molecule "glows" or **fluoresces** in a different color, as illustrated in Fig. 2.

The fact that *f* and *g* are different wavelengths is very useful, because we can take advantage of that property by filtering out light of frequency *f*, so that the final image only shows those molecules that are actively fluorescing. There will be light at frequency *f* scattering around everywhere, but since that light can be blocked, we can pick out even extremely small amounts of fluorescence at frequency *g*. The result is that we can *detect* fluorescent molecules even when they are too small to image.



1. A photon is **absorbed** (A), pushing an electron to a higher-energy orbit (B).
2. The atom remains in an **excited state** (B) for a short time.
3. The atom **emits** a photon when the electron returns to the low-energy orbit (C). The wavelength of the emitted light is different from the wavelength of the laser light, so the emitted light can be easily separated from reflected light by a filter.

Fig. 2 How fluorescence works

Fluorescent dyes are fluorescent molecules that have been designed to stick to only certain targets. In the past, that's been limited to broad categories such as lipid membranes or nucleic acids, but over the past few decades, biologists have created increasingly specific dyes. These days it's possible to tag a single specific protein, letting us visualize highly specific cellular activity. We also have dyes that fluoresce at different wavelengths, allowing us to effectively combine multiple stains into a single image.

This is an accomplishment, because it's quite hard to find a good fluorescent dye. It's not just a matter of finding a material that glows (which is hard enough); a good dye must also:

- The dye must be easy to excite, so that making it light up doesn't cook the cell.
- The light it gives off must be of a very different wavelength (i.e., color) than the light you used to excite it, letting you filter out the background "noise" from your original light source.
- Anytime a fluorescent molecule becomes excited, it has a chance of changing shape in such a way that it won't glow anymore. So, the dye should be stable enough that it **photobleaches** as slowly as possible.

Figure 3 shows the kind of images that we can gather via fluorescent microscopy. In this experiment, researchers were studying a protein called the HT-52A receptor, which is sensitive to a number of familiar substances, notably including LSD, psilocin, and mescaline.

Confocal Microscopes

If we get even more aggressive with the way we filter the light passing through our sample, we can limit our field of view to a single narrow "slice" of the total image. By scanning slowly through the sample, a **confocal microscope** using this technique can produce fully three-dimensional images.

Surprisingly, the technique was first patented by Marvin Minsky, one of the founders of artificial intelligence. (However, it was just a theory when he introduced it in 1957—confocal microscopy only became practical years later, with the development of lasers.)

Electron Microscopes

Florescent and confocal microscopes are useful, but if you *really* want to get serious about microscopy, you're going to need an **electron microscope**. Unlike optical microscopes, electron microscopes use beams of high-energy

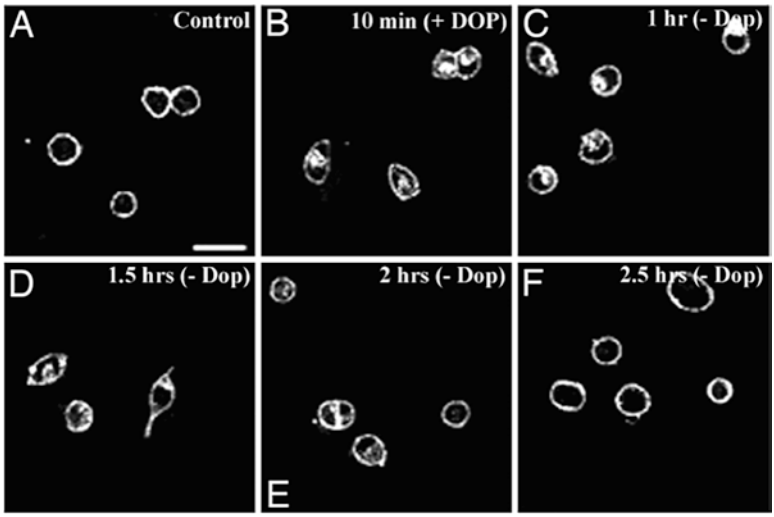


Fig. 3 Fluorescent microscope images. These cells are cultured human cells, in which one of the G-couple protein receptors for serotonin has been made fluorescent. Panel (**a**) shows control cells, in which the fluorescence is all at the surface of the cell. Panel (**b**) shows cells that have been incubated with dopamine, a neurotransmitter, for 10 minutes. After exposure to dopamine, some of the receptors have moved to the interior of the cell—which suggests that the cell will be harder to stimulate with serotonin. Panels (**c**)–(**f**) show cells at various times after the dopamine has been removed: 1, 1.5, 2, and 2.5 hours. After 2.5 hours, most of the receptors have once more moved to the surface of the cells. (From "Activation, internalization, and recycling of the serotonin 2A receptor by dopamine", by Samarjit Bhattacharyya, Ishier Raote, Aditi Bhattacharya, Ricardo Miledi, and Mitradas M. Panicker, *PNAS*, 2006; volume 103; pp. 15248–15253)

electrons instead of light, allowing them to achieve much, *much* higher resolutions. A good optical microscope can magnify an image around 1500×; the best electron microscopes have resolutions of approximately 100,000×, making them powerful enough to see viruses, proteins, and even large molecules like glucose. Figure 4 shows electron microscope images of a number of very small objects.

A Quick Reminder

Mitochondria are organelles that produce energy from glucose and oxygen. Actin is a protein that forms microfilaments, long filaments which help give a cell its shape.

That kind of power comes with a cost. Unlike light, electrons don't penetrate far into a cell. If you want to look at something like the nucleus, you must first slice the cell into very thin sections—not the easiest task. Once

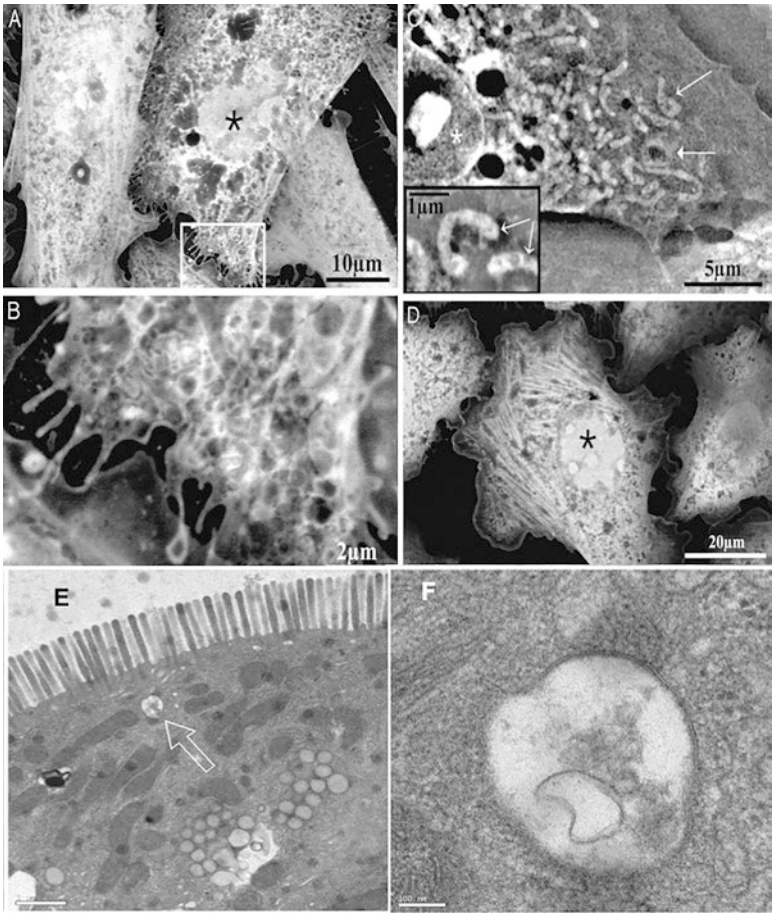


Fig. 4 Electron microscope images. (**a**) Human HeLa cells (**b**) the inset in (**a**), further magnified. (**c**) Hamster CHO cells, with some mitochondria shown in the inset. (**d**) Actin filaments. (**e**) Part of the intestinal call of a 4-day old rat. (**f**) The vesicle indicated with an arrow in (**e**). Scale bars are 1 μm in (**e**), 100 nm in (**f**)

prepared (typically by either rapid freezing or dehydration and infusion with resin), samples have to be stained with a heavy metal such as gold and placed in a vacuum.

Needless to say, this makes it difficult to look at cells in anything close to normal conditions.

Manipulation of the Very Small

Taking Small Things Apart

Imagine, if you will, that you've found an old desktop computer, and you want to figure out how it was made. The more detailed the better—ideally, you want to be able to go online and order the exact same parts and assemble them the exact same way, kludges and all. Unfortunately, you're locked in a cage a hundred feet away, and all you have to work with are a bunch of remote-controlled robots with hammers and chisels.

The scenario is, of course, absurd. How would you even begin going about such a task? Is it even possible to learn anything useful? What kind of results could you possibly publish?

It's a question biologists have been wrestling with for generations. It's impossible to simply dissect a cell and examine the individual components—everything's too small to see without advanced microscopes, and the tools that we have to manipulate them are often about as subtle as our hammer-wielding robots up above.

Absurd as it may sound, the protocol from Fig. 1 isn't that far off from how biologists often have to work. Think about what the (imaginary) authors are doing: they're breaking their subject down into the smallest possible components, then separating them based on how they respond to a force. Lighter bits will fly farther, bits with jagged edges will stick to the carpet and move more slowly, and you wind up with the pieces of the computer (roughly)

W. W. Cohen, C. K. Cohen, *A Computer Scientist's Guide to Cell Biology*,
https://doi.org/10.1007/978-3-031-55907-5_5

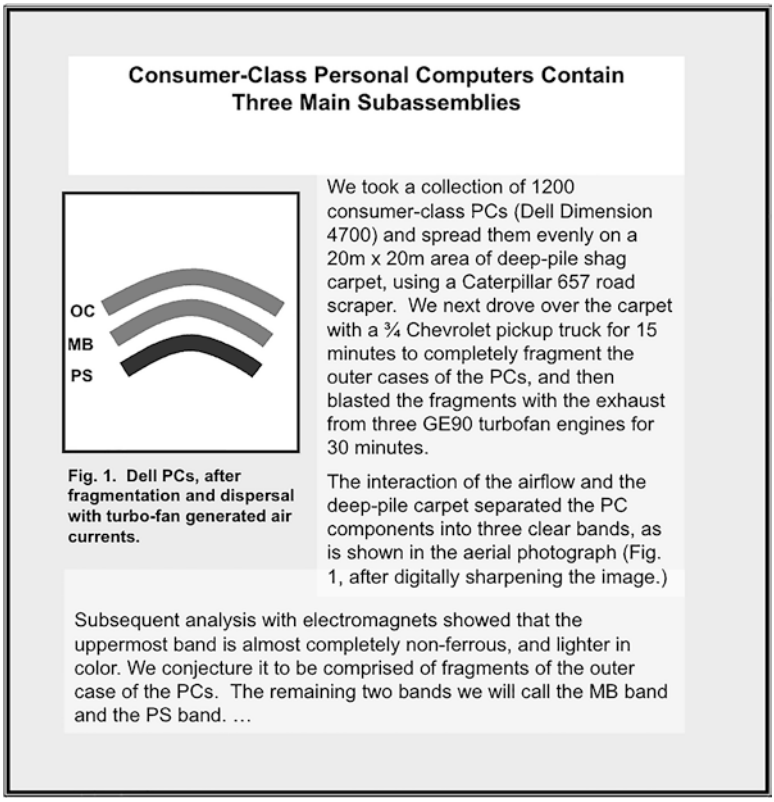


Fig. 1 An imaginary article on reverse engineering PCs

sorted by category. It's not *great* information, but it might be enough to guess that the computer is—to a first approximation—made of a plastic outer casing, a power supply, and a silicon motherboard.

Splitting a mixture into components is called **fractionation** (if you're thinking about the input to the splitter) or purification (if you're using fractionation to collect one particular mixture element, and you're thinking about the output).

Let's look at a few examples of how biologists separate out the different bits of the cell.

Sorting Small Things

Centrifugation: Separation by Weight

As we mentioned, the technique our hypothetical computer scientists used is pretty similar to one of the simplest tools in the biologist's arsenal: separating out the desired parts of the cell by their size and weight.

The first step is to break the cell—or cells (most of the time biologists work with large populations of identical cells)—into tiny pieces. Depending on what you're looking for, you might do so by either mechanical or chemical means. You can use something as sophisticated as an ultrasound emitter, or as simple as a blender. (Just remember to put your samples in an appropriate buffer first).

Once you've reduced everything to rubble, the next step is to use a **centrifuge** to separate components by size and weight. As your sample is spun at incredible velocities, centrifugal force pushes everything to the bottom of the test tube. Smaller fragments will quickly find their way down; larger fragments will move more slowly, due to their greater inertia and mass. At one point, centrifugation was only good at sorting large components like nuclei and mitochondria, but modern methods, such as **velocity sedimentation** and **equilibrium sedimentation**, use forces of up to half a million gravities to separate out individual molecules by mass.

Chromatography: Separation by Charge or Other Properties

There's a simple experiment you might remember from elementary-school science classes: take a strip of paper towel, color one end with a marker, and stick that end of the paper in a cup of water. Over the next hour or two, you'll see that single band of ink turn into a broad smear of color, shifting hues as it travels farther and farther from its original location.

The same idea underlies **column chromatography**. Once they've fractionated their cells, biologists pour the resulting mixture through a solid—but porous—column called a **matrix** (chromatography). As the mixture filters downward, proteins capable of interacting with the matrix find their progress slowed to a crawl as they find themselves repeatedly sticking to the sides. Other proteins find their way to the bottom much more quickly, where they can be collected and categorized based on how long it took them to work their way through the column.

The simplest forms of column chromatography only care about gross physical features like size and mass, but that doesn't have to be the case. It's possible to design matrixes that bind tightly to certain elements—you can, for example, attach antibodies to microbeads, so that the highly specific "heads" face out; another technique might be to attach DNA strands onto which the protein you're interested in is known to bind.

This practice, known as **affinity chromatography** to distinguish it from less specific methods, makes isolating specific elements simple. As the sample passes through the layers of beads, your desired protein (or other compound) sticks to the microbeads, while everything else passes through with no interference. Once all the excess is removed, you only need to add a solvent to dissolve the bonds between matrix and sample, and your chosen protein will collect neatly at the bottom of the tube.

In the old days, chromatography took hours, as biologists waited patiently for gravity to do its job of pulling samples through the matrix. Thankfully, newer systems use tiny beads and high pressures to force a mixture through the matrix in a matter of minutes.

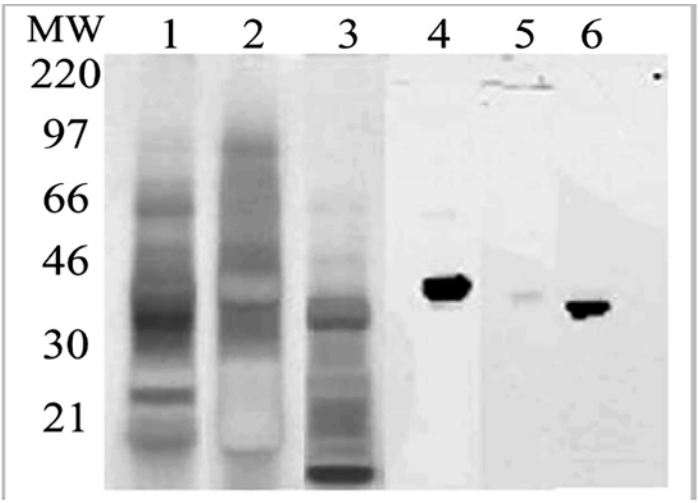
Electrophoresis: Separation by Size or Shape

Column chromatography depends on gravity to get the job done, but there are other options. If you lay your matrix out on a flat surface and apply an electric charge to one end, you'll still be forcing the contents to separate according to size, just using a different motive force. This process is known as **electrophoresis**; the matrix, in turn, is called a **gel**. (Probably because they're typically made of Jello-like substances, but more research is needed to know for sure.)

One extremely common variety of electrophoresis is sodium dodecyl sulfate polyacrylamide-gel electrophoresis—or, if you'd rather avoid such a mouthful, **SDS-PAGE**.

Sodium dodecyl sulfate—the "SDS"—is a detergent; when added to a cellular extract, it causes proteins to **denature** and unfold back into simple linear shapes. The treatment lets us ignore the effects of protein shape, and separate them purely by length. Typically, biologists will place several mixtures on the same gel in order to compare them, each in their own **lane**. Another lane is usually filled by a **marker**, a premade mixture of proteins of known sizes. Figure 2 shows what this looks like.

Once the gel has run for long enough, you can recover proteins of a specific size range by literally cutting out that section of the gel. Alternately, you can



An example of an SDS-PAGE gel. Lanes 1-3 are a complex mixture of several substances, and lanes 4-6 show the corresponding mixture components after purification (via Western blotting, described in the text). The leftmost column is provided by the authors, and shows the molecular weights of substances that migrate to each level. Here the authors are demonstrating the effectiveness of the purification method used.

From *Mass spectrometric analysis of electrophoretically separated allergens and proteases in grass pollen diffusates,* by Mark J Raftery , Rohit G Saldanha , Carolyn L Geczy and Rakesh K Kumar, *Respiratory Research 2003, **4**:10*

Fig. 2 Using SDS-PAGE to separate components of a mixture

transfer all of the proteins to a thinner membrane and add fluorescently tagged antibodies to highlight specific proteins—this process is known as a W**estern blot**.

Another variant is **2D gel electrophoresis**, which—as the name suggests—involves two dimensions of separation. Before using SDS to denature the proteins, samples in a single well are separated purely by charge. At the conclusion of this first run, the proteins will be spread out vertically in a narrow column.

The technique for separating by charge used in 2D gels is called **isoelectric focusing**; it causes proteins to migrate to their **isoelectric point** (i.e., the point at which the protein has no net charge).

We then take that column of separated proteins and put it on one side of an SDS gel before using a second electric force to separate them by size, just as we do in a normal SDS-PAGE gel. The end result is a sort of two-dimensional

Table 1 Different ways of sorting mixtures

Method	What is typically sorted	(Numeric) Property sorted by
Centrifugation	Whole cell extract	Weight
Column chromatography	Mixture of proteins	Size, weight, charge, or hydrophobicity
Gel electrophoresis	Mixture of proteins or nucleic acids	Size (folded) or electric charge
SDS-PAGE	Mixture of denatured proteins	Size (after denaturing)
2D gel	Mixture of proteins	Size in one dimension, then electric charge in the second dimension

graph, where every protein is sorted to a specific spot based on its charge and size.

2D gel electrophoresis can typically be used to sort up to a thousand different proteins, but it's also one of those protocols where skill plays a major role. An inexperienced lab tech can easily turn the whole thing into a useless smear; a master, by contrast, can resolve as many as 10,000 proteins on a single gel (Table 1).

The Many Approaches to Sorting and Selection

To summarize, the types of fractionations that we've seen so far are the biologist's version of a common computer science operation: they **sort** mixture components according to a numeric function. Centrifugation sorts components by weight, gel electrophoresis sorts by size or charge, 2D electrophoresis by both size *and* charge (Table 1 gives an overview of these). In contrast, affinity chromatography extracts components according to a "user-defined predicate"—components that pass a certain experiment-specific test are separated from those that don't. In biology this kind of operation is called **selection**.

> Biologists often use the term **selection** for a "user predicate that can be applied quickly, in parallel." For instance, one can select for antibiotic-resistant bacteria by treating a group of them with the antibiotic. A test that requires manual effort for each item is usually called a **screen**. To a first approximation, a screen is an O(n) operation, and a selection is an O(1) operation.

The Western blot described above is another example of a selection operation, but there are a number of related methods. One Western-blot variant uses a one-dimensional gel containing RNA molecules, to see which ones **hybridize** to some DNA molecule *X*—this is called a **Northern blot**. A

Table 2 Selecting components of mixture that satisfy some property

Method	What is selected	(Boolean) Property selected for
Affinity chromatography	Mixture, e.g., of proteins	Does a mixture component bind to a user-selected substance?
Western blot or proteome chip	Mixture of proteins	Does a protein bind to one of a set of user-selected proteins fixed on a substrate?
Northern blot, microarray, or gene chip	Mixture of RNAs	Does an RNA hybridize to one of a set of user-selected DNAs fixed on a substrate?
Southern blot, microarray, or gene chip	Mixture of DNAs	Does a DNA hybridize to one of a set of user-selected DNAs fixed on a substrate?

Northern blot with DNA instead of RNA is called a **Southern blot**. (Historically, Southern blots came first—they were invented by a biologist named Ed Southern in 1975.)

Some widely used high-tech descendants of the Northern blot are **gene chips** and **microarrays**, which we will talk about next. Table 2 gives an overview of these selection methods.

Measuring Proteins at Scale

Just because two cells share the same DNA, it doesn't mean that they make the exact same set of proteins—after all, if cells couldn't pick and choose which genes to express, there wouldn't be any multicellular life to read this sentence. But single-celled organisms can get in on the act as well. They might not be as wildly varied as the cells in a human body, but they're still capable of adjusting their manufacturing process to better match a new environment. We refer to the overall set of proteins being produced as the cell's **proteome**, and studying such changes is a common goal in experimental biology.

One of the easiest ways to survey the proteome is by looking at mRNA. Cells can't build proteins without the directions, after all. It might be a more indirect method of looking at protein production, but the upside is that it's very easy (at least when graded on a molecular biology curve) to isolate mRNA using tools like affinity chromatography.

Regardless of what protein they're encoding, nearly every strand of mRNA in a cell has a long "**polyA tail**"—a long string of more than a hundred adenine nucleotides, "AAAAAAAAAAAAA." Under normal conditions, the polyA tail helps determine how long the strand will stick around before being degraded by the cell's own enzymes; under experimental conditions, it means

that we don't need to know a strand's sequence to catch it. A nice long string of thymine bases ("TTTTTTTT") is all you need to bind any mRNA molecules that happen to be floating around.

Microarrays and Gene Chips

But this is the twenty-first century, the age of easy genetic sequencing and microengineering. Simple strands of thymine are so passé; the fashionable (or at least well-funded) molecular biologist usually prefers to use **microarrays** to probe the proteome with far more precision.

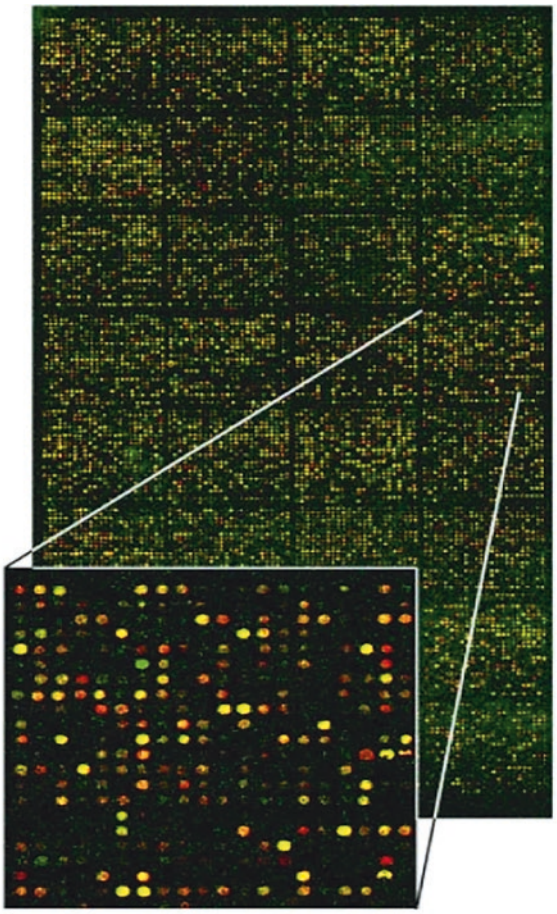
At first glance, a microarray looks like someone glued a computer chip to a microscope slide—but it's nothing of the sort. No logic processes are being computed here. The thing that looks like a chip is actually a microscopic grid covered with DNA strands, each "point" containing the DNA for a different gene. A *lot* of genes: a single microarray can have enough coverage to have a "point" for every single gene in a yeast cell's genome.

Both DNA and RNA can be either single-stranded, or double-stranded. In double-stranded DNA/RNA, each strand is complementary to the other. In the right conditions and at the right temperature, two single strands that are complementary can spontaneously form a double-stranded molecule; this process is called **hybridization** or **base-pairing**. Hybrid strands can be DNA-DNA or DNA-RNA.

A typical microarray experiment involves comparing two different mixtures of mRNA—probably one from a colony of cells grown under normal conditions (the **control**), and one from a colony that's been treated somehow for your experiment (the **treatment**). The first step is to treat each sample with a different color fluorescent label; it just so happens that the lab fridge only has green and red labels, so you'll make the control green and the treatment red. Once that's done, all you need to do is add a tiny amount of each sample to the same microarray and turn on a fluorescent microscope.

At each point on the microarray, you can measure the color and intensity of the resulting light. If the point is blank, neither sample expresses that gene. If it's red, the gene is only being expressed in the treatment sample; if it's green, it's only being expressed in the control. Most will probably show up as yellow (in this case), indicating a mixture of the two colors.

More prolific genes will return a brighter signal; those that are only being expressed at low levels won't produce much fluorescence. The final step in the



Microscope image of a DNA microarray slide. From "Application of microarray technology in pulmonary diseases", Tzouvelekis, Patlakas & Bouros, Respiratory Research volume 5, Article number: 26 (2004)

Fig. 3 Measuring expression levels of many genes with a microarray

experiment is running a computer vision algorithm that reads these levels from the many, many points on the microarray (Fig. 3).

It's worth noting that although the technology is new, the *idea* of using fluorescence to measure gene expression is an old one. Biologists used to have to separate the genes out manually, perhaps by using one of the gel electrophoresis processes we talked about above. A microarray is basically just a high-tech version of a Northern or Southern blot.

Gene chips are an even newer development of the same idea. Instead of creating—or, more likely, buying—a slide that's already been prepared with the full DNA sequences for each gene, a gene chip starts out "blank." The scientist using the chip can customize it, synthesizing the DNA sequences right on the chip. These sequences will, by necessity, be much shorter (around 25 base pairs), but by looking at the sample organism's full genome and carefully selecting your short strands, you can still create an unambiguous mapping between the sequences on your chip and specific genes.

Parallelism, Automation, and Reuse in Biology

One advantage of many of these selective assays is that they lend themselves very well to parallel processing—it's possible to use them on many different samples at the same time.

Take the humble Western blot, for example. All it's doing is looking for the presence of one particular protein—in principle it doesn't matter if you load one sample, two or several, as long as they all fit on the same membrane. If you choose your primary and secondary antibodies carefully and have an imaging system capable of distinguishing between fluorescent colors, you can probe the same samples for multiple proteins at the same time. In practice, however, this sort of parallelism is limited: you can only physically fit so much stuff on one gel, and it's trickier to handle an enormous gel without ruining it.

The high-tech versions of this assay enabled by microarrays enable much more **parallelism**—often *every* mRNA in mixture is tested for compatibility with *every* gene in a genome. (As an aside, this sort of parallel processing is also largely the reason that biologists are currently awash in experimental data.)

In computer science, one of the best ways to gain productivity is by **software reuse:** finding the right high-quality software package and using it correctly is almost always faster than writing (and testing and debugging) your own custom package. The introduction of microarrays and gene chips has given a similar advantage to biologists. Of course, unlike a software package, any given gene chip can only be used once—but manufacturers can easily churn out large volumes of blank chips, so it gets cheaper and cheaper to do experiments.

Beyond the effects of parallelism and the economies of scale, recent high-throughput methods are also valuable because they are often easier to use than older approaches. Using a Western blot, for example, is practically an art form—steps like loading sample into the gel or laying out the transfer membrane require training, practice, natural dexterity, and occasionally blind luck.

It's hard to duplicate that sort of human skill without resorting to very expensive and painful processes, like graduate school.

Thus, throughout biology, more and more experimental procedures are being partially or fully automated. Every time you hand a human a pipette you're creating another opportunity for a slip-up, but modern **liquid-handling robots** can carry out many routine procedures merely by taking over all the pipetting steps; more advanced (and expensive) robots can even handle tasks like incubations, centrifugations, and spectroscopy without ever involving a human. Even traditionally complex tasks like DNA extraction can be automated these days.

Liquid-Handling Robots

Liquid-handling robots are also parallel—most are set up to work with 96- or 384-well plates, allowing for huge amounts of samples to be processed at the same time.

When you first hear the phrase "liquid-handling robot," it's tempting to imagine a Hollywood-style setup with individual robot arms darting to and from across a plate, sucking up fluids here, and squirting them back out there. As usual, the reality is considerably less exciting.

A liquid-handling robot typically features a large, flat grid. Technicians attach plates, boxes of disposable pipette tips, heating blocks, and other necessary items to the grid in carefully preselected positions. A blocky pipettor with 12 or 96 little rods moves back and forth across the grid on a set of tracks, one each for the X, Y, and Z axis. Now it dips down to grab a set of pipette tips; now it descends to suck up a buffer; now it moves back down to add that buffer to a plate of samples. And on, and on, and on.

Many variants exist, of course, to perform different sets of operations on different numbers of samples. Some are highly customizable and can be programmed to the lab's exact needs; others are specialized for a single task such as PCR or DNA extraction. All of them are very easy to mess up if you deviate at all from the program's expectations.

Databases of Prepared Samples

Another increasingly common practice is the building of public databases of samples or genetic information, allowing scientists to build directly on each other's results without first having to replicate the original experiments. Take,

for example, the Yeast GFP Fusion Localization Database[1]—by systematically tagging (almost) every protein expressed in yeast cells with green fluorescent proteins, they've done away with the need for individual researchers to do their own gene-splicing. Instead, if you want to study a particular yeast protein, all you have to do is go online and place an order, saving you from spending days or weeks messing around with cell cultures and plasmids.

From an economic perspective you could look at this as a move toward a "horizontal economy," in which individual labs outsource almost every procedure and concentrate on a small handful of tasks. But you can also take a programming perspective, and think about resources like GFP-tagged yeast proteins as a sort of "subroutine package" for biological experiments. In computer science, you might save time by using some other programmer's machine-learning software package; in biology, you might save just as much time by using some other researcher's library of genetically engineered yeast. Biology is not quite at the level of "software as a service," but it's gotten much closer over the last two or three decades.

Classifying Things by Their Pieces

Histograms and Peptide Maps

Before the advent of large language models, computer scientists used a simple way of determining how similar two documents were. The documents (X and Y) would be converted into "bags of words"—that is, one would count the number of times each word appears in each document. This lets us turn the "bags" into tidy little functions like $h_X(w)$, where w is a word and $h_X(w)$ is the number of times w occurs in X. These functions turn the documents into **histograms** of substrings (individual words), and then we can directly compare the functions $h_X(w)$, and $h_Y(w)$ using histogram-based similarity metrics.

We can use a similar approach when comparing two proteins. After all, proteins are ultimately linear strings of amino acids; we can convert them into "bags of words" by using chemicals to break up the amino acid sequence in a consistent way.[2] Then we can use techniques like chromatography or electrophoresis to separate and sort the fragments into a unique pattern called a **peptide map**—sort of a molecular fingerprint that we can compare to other, previously mapped proteins.

[1] https://yeastgfp.yeastgenome.org/

[2] Cyanogen bromide, for example, will snip the amino acid chain in two after each methionine residue.

We can think of the entire process as a function, $f(z)$, where z is a protein fragment and f represents the process of separating and sorting. The overall peptide map, then, becomes a function $h_P(n)$, in which $h_P(n)$ is the number of fragments z in such that $f(z) = n$. The peptide map is a "fingerprint" for the protein, and it can be used to identify it from a list of candidates that have been previously "fingerprinted" by the same procedure.

Mass Spectrometry

While you can get a serviceable peptide map with basic electrophoresis, if you want a really high resolution one, you'd be better off with a technique such as **mass spectrometry**, which sorts and counts the fragments by both weight *and* charge. At the end of the analysis, you're left with a neat histogram of how many fragments displayed which charge-to-weight functions—a histogram called the **mass spectrum**. Better yet, unlike a gel, mass spectrometry can be performed on extremely small amounts of samples.

Mass spectrometry is hardly limited to proteins, incidentally. The technique can be applied to all sorts of molecules. Nor is it limited purely to analysis—during the Manhattan Project, a type of mass-spectrometry machine called a **calutron** was used to separate uranium isotopes!

DNA Fingerprinting

Another example of "classification by separation" is **DNA fingerprinting**. As with proteins, one begins by cutting the DNA into fragments using a chemical that cuts in a predictable way: for DNA, the chemicals that are used are called **restriction endonucleases** (which will be discussed more below). Since DNA sequences differ slightly from individual to individual, the "bag of fragments" representation of two DNA sequences is likely to be different, so it is possible to compare the resulting fragments using the same sort of histogram-based similarity measures we've already used with peptide maps and mass spectroscopy.

Since no two people's DNA is exactly the same, different people's DNA fingerprints will appear different. Known as **restriction fragment length polymorphism** (or **RFLP**), the process is a classic tool for tasks like paternity testing and matching potential criminals to evidence like strands of hair.

It's worth noting that DNA fingerprinting rarely makes use of the entire genome—after all, even if we have *some* differences in our genetic code, the

vast majority must be the same for us to be, you know, alive. Rather than waste time mapping these highly functional genes, DNA fingerprinting usually focuses on noncoding or "junk" DNA.

A particularly useful set of targets are **minisatellites**, highly repetitive bits of code that are, themselves, typically repeated many times on the genome. Because they are not involved in the production of important proteins or mRNA, they're prone to mutations and tend to vary widely across the population. Minisatellites are easy to find using hybridization, and the resulting histograms tend to be highly distinctive.

Wrap-Up on Classifying Things by Their Parts

About now, you might be feeling a little lost amidst a raging sea of biology jargon, so it's probably worth stopping to take a broad, computer-science-oriented look back at everything we've talked about in this chapter. Essentially, all of these techniques and experiments boil down to different implementations of three basic operations, where X is a known object with an unknown structure:

1. Given an object X, take that object apart into components $W_1 \ldots W_n$, and then sort the components according to some numeric property $F(W_i)$.
2. Given an object X, take that object apart into components $W_1 \ldots W_n$, and then extract all components that satisfy some Boolean property $P(W_i)$.
3. Given an unknown object X and a set of known objects $Y_1, \ldots, Y_n$, determine the Y_i that X is most similar to.

If you take a bunch of cell **lysate** (the mix of proteins, lipids, and nucleic acids that spill out of a cell when their membranes are dissolved, or "lysed") and throw it in a centrifuge, you're performing operation (1)—separating components $W_1 \ldots W_n$ according to their weight. Similarly, using a gene chip to pick out specific strands of mRNA is an example of operation (2)—extracting all compounds that satisfy the property of "matches one of the genes on this chip." (Or, if you're looking for one specific strand, operation (3)—you're comparing the mRNA mixture to a set of known genes.)

Reprogramming Cells

Our Friends, the Microorganisms

Biologists have come up with a lot of clever ways to sort and analyze cellular processes, but even gene chips and mass spectroscopy are basically still just smashing the computer with a hammer and picking through the fragments. You *can* learn a lot that way, but wouldn't you rather have some tools that were actually made for the job?

Well, you can—if you're willing to outsource. Living cells are jam-packed with all sorts of molecular machinery, and over the years we've found many ways to make it work for us.

Restriction Enzymes and Restriction-Methylase Systems

Viruses, as we mentioned in an earlier chapter, hijack a cell by inserting a copy of their own DNA into the host's genome—essentially "tricking" it into making more viruses. Given that cells would much rather that *not* happen (if nothing else, it's a huge waste of resources), evolution has equipped them with a few ways to resist.

One of their most effective types of antiviral weaponry is the **restriction-methylase (R-M) system**. And despite the scientific-sounding name, it's the same basic idea as hiring a bunch of big guys with axes to kill anyone who wanders in without a badge.

To identify its own DNA, cells attach some sort of molecular marker directly to the strands. Typically, this "badge" is a methyl group (one carbon

W. W. Cohen, C. K. Cohen, *A Computer Scientist's Guide to Cell Biology*,
https://doi.org/10.1007/978-3-031-55907-5_6

and three hydrogen molecules); the protein that actually attaches it to the nucleotides, then, is called a **methylase.**

To kill off intruders, cells also have protein complexes called **restriction endonucleases (REs)**. Like a guard checking for ID cards, when these enzymes stumble across a piece of DNA that doesn't have the proper methyl "badge," they act quickly to destroy it.[1] More specifically, they look for a specific sequence of nucleotides, such as "GAATTC" (along with the complementary "CTTAAG" on the other strand) where they can attach themselves to the DNA strand and chop through the sugar-phosphate "backbone" to create a pair of newly shortened (but still double-stranded) nucleic acids. (Remarkably, both the initial binding and the resulting cleavage require no external energy sources.)

As with everything in biology, REs are a lot more complicated in practice.

Luckily, we don't need to understand everything about them to make use of them. You can think of a RE as a sort of black box, a complicated software module that you can drop into a program wholesale as long as you understand its "interface." DNA fingerprinting, for example, doesn't actually care *how* the RE chops up the sample, only that it does so in a consistent way.

And with just a little more knowledge of that "interface" (i.e., what sequence the RE acts on), you can begin synthesizing entirely new DNA molecules by cutting and pasting together strands of existing DNA.

A note on nomenclature: "restriction endonuclease" may be quite a mouthful, but like many terms in biology, there's a twisted sort of logic to it.

- The suffix "-ase" indicates that you're talking about an enzyme.
- The rest of an enzyme's name is usually the substrate it interacts with. REs make changes to nucleic acids—hence, "nuclease."
- "Endo-" and "exo-" are suffixes that mean "within" and "without," respectively. REs attack the middles of nucleic acids—hence, "endonuclease."

You can probably guess where "restriction" comes from.

By convention, specific endonucleases are identified by four letters—the first three identifying the organism where it originated, and the fourth the exact "strain"—and a Roman numeral. The second RE found in strain **d** of the **Haemophilus influenzae** bacterium, for example, is known as HindII (pronounced "hin dee two").

[1] The DNA, not the poor employee that forgot their badge.

Constructing Recombinant DNA with REs and DNA Ligase

Strangely, it turns out that the second half of that operation—the "pasting"—also depends on the actions of restriction endonucleases. Specifically, it hinges on *how* the enzymes make their cuts.

REs, you see, don't make perfectly clean cuts and leave behind perfectly symmetrical, double-stranded DNA strands. They actually make diagonal cuts, bisecting a handful of base pairs in the process and leaving each of the newly divided strands with a little tail of single-stranded DNA—the **sticky end**.

It's probably easier to imagine this visually. Take, for example, the *E. coli*–derived RE EcoRI shown in Table 1. The gray areas show one of the two fragments after the cut; the white areas the other.

Table 1 Small fragment of DNA before being cut by EcoRI

GATTACA	G	AATT	C	CATATTAC
CTAATGT	C	TTAA	G	GTATAATG

Our two sticky ends, then, are AATT and TTAA. You might notice that the two sequences are complementary. They match; surely the fragments will quickly "stick" and glue themselves back together, right?

Well…sort of. Using a restriction endonuclease will leave you with a *lot* of fragments, all of them with the same sticky ends; those ends might still be able to match, but who's to say what nucleotides came *before* the target site (see Table 2). If the original strand has multiple target sites, there's absolutely no guarantee that the fragments will glue themselves back together in the right order—or even wind up the same length.

Take a closer look at the white fragment in Table 2, and its specific sticky ends. They're the same on both sides, meaning that you could totally remove that particular fragment and still get a stable reconstruction. This kind of randomness means that letting a sample reassemble doesn't just give you the original strand—you're left with *all possible* versions of the reassembly process.

Consider two strands of DNA, *xSy* and *wSz*, where "S" is the sequence recognized by a restriction endonuclease and *w, x, y,* and *z* are all different

Table 2 A longer DNA fragment, showing how it is cut by an RE

GATTACA	G	AATT	C	ATTACCAT	G	AATT	C	CATATTAC
CTAATGT	C	TTAA	G	TAATGGTA	C	TTAA	G	GTATAATG

DNA sequences. If you treat the two strands with the RE and allow them to reassemble, you'll still have strands *xSy* and *wSz* in the final mixture, but you'll also have the brand-new (or **recombinant**) strands *xSz* and *wSy*.

CRISPR/Cas9

Useful as they are, restriction enzymes are still limited—there are only so many known examples, targeting only so many specific DNA sequences. Trying to make a cut in exactly the right place can be more like a logic puzzle than anything else, as frustrated biologists try to mix-and-match enzymes to isolate the sequences they're interested in.

What would really be useful, thousands of researchers have grumbled, would be some sort of "programmable" restriction enzyme, one that you can tell exactly when and where to cut.

And as luck would have it, such a thing actually exists. In one of the most important discoveries of the last 20 years, scientists discovered a sort of "modular" restriction enzyme known as **Cas9**. Unlike most such proteins, Cas9 doesn't bind directly to DNA. Instead, it delegates the attachment process to a strand of **guide RNA (gRNA).** The gRNA is responsible for determining where the greater Cas9 complex grabs and cuts, not the intricacies of protein structure—and it's much, *much* easier to synthesize custom gRNA than it is to design entirely new restriction enzymes.

Cas9 doesn't exist solely for the convenience of biologists, of course. Under normal conditions, it serves as part of a bacterial immune system, of sorts—a collection of restriction enzymes that break down viral RNA before it can infect the cell's own DNA. And just like the eukaryotic immune system, this system is capable of "learning" from past experience.

How does this "learning" process work when a viral DNA (or any foreign DNA) is inserted in a cell? Certain restriction enzymes chop up some of the invasive nucleic acid and produce short fragments known as **protospacers**. Proteins in the cell then take those fragments and add a short sequence called a **protospacer adjacent motif (PAM)** to each end—thus "labeling" the fragment as foreign, and creating a longer sequence called (you guessed it) a **spacer**. These spacers are then saved in "long-term memory" by being spliced into the genome in a particular region known as **CRISPR,** short for "**clustered regularly interspaced short palindromic repeats."** CRISPR is a long series of repeating patterns occasionally interrupted by one of these captured spacer sequences. The repetitive structure is recognizable enough that a single set of enzymes can "read" and transcribe any spacer, recognizing it as some

sort of foreign DNA—even though the exact contents of different spaces are different.

These spacer-containing sequences ultimately become the guide RNA we mentioned earlier. Once transcribed, they're trimmed down to shorter fragments and the rest of the Cas9 protein complex forms around them, like a pearl building up around a speck of dirt. Now the cell has a defense mechanism tuned to cut a particular kind of DNA—DNA from the original virus that was chopped up to form these spacers.

From the cell's point of view, this mechanism is a great defense against reinfection. And because the memory of CRISPR sequences are part of the bacterial genome, they'll be passed down to any daughter cells, so as long as some distant ancestor of a cell stumbled across a virus before the cell is protected.

But this mechanism is also highly convenient for biologists. Like we just saw, normal restriction enzymes are highly specific—they attach to one RNA sequence, and one sequence only, and modifying one to work with something else would be a hideously complicated process. Cas9 is a complex that does a similar thing, and it has already evolved to be modular, as it can accept many RNA sequences as gRNAs. So cutting DNA exactly where you like is as simple building a new Cas9 complex, which—in turn—is as simple as synthesizing a bit of RNA and dropping it in a test tube full of proteins.

From a computer scientist's perspective, this kind of ability is fascinating. If the broader CRISPR/Cas9 machinery is a sort of cellular function, the viral fragment from the spacer is essentially an argument to that function. You can change the DNA sequence—call a different argument—and the process will still function.

Inserting Foreign DNA into a Cell

Let's take a moment to think like computer scientists again. DNA is, basically, the language with which the cell is programmed—that's a common enough analogy, and it generally does work fairly well. By that logic, the genome is a set of rock-bottom commands that, when properly arranged, give rise to vastly more complicated systems.

Following the metaphor a little farther, we could think of each individual gene as its own little program; each codon, as a single specific task ("get protein A, get protein B, get protein C…").

Restriction enzymes—and CRISPR in particular—let us open up that source code and make our own changes, essentially "hacking" the cell. It

opens the door to all sorts of interesting new experiments. What happens when you remove a particular protein from the genome? What about adding one? Or maybe you need a lot of a protein normally produced by a slow-growing bacteria—can you transfer the gene to some nice prolific HeLa cells and harvest the protein from them?

Plasmids as Insertion Vectors

Modifying DNA can be done **in vitro**—in a lab, rather than a living cell (which would be **in vivo**). But if you want to actually run your newly hacked program—sorry, synthesize your set of proteins using the recombinant DNA—in a live cell, to see what it will really do, you have to find a way to upload the code to a living organism, and make the program executable. How do biologists do that?

Plasmids, as we discussed earlier, are little loops of DNA that are commonly absorbed by prokaryotes. Since prokaryotes are already used to picking up new genes from plasmids, inserting a recombinant DNA sequence into, say, a culture of *E. coli* is simple. All you need to do is add your hacked gene to a plasmid, give it a few runs through a PCR machine to create lots of copies, and squirt it into your cell culture.

That said, if you want to get a prokaryote to actually *use* the DNA you'll need to take an extra two steps, both of which also require modifying the DNA sequence. We'll discuss this more below, but for now, what you need to know is that there is a molecular machine that binds to DNA to initiate transcription. To get that machine started, you'll need to include a **promoter**, a special DNA sequence that gives DNA or RNA synthase an attachment point, and an **origin of replication** sequence, to show the machine where to actually begin transcription. Adding these sequences is all that's needed for the natural machinery of the cell to run your "uploaded code."

DNA insertion with plasmids is a little bit more complicated with eukaryotic cells, which are smarter about not running random bits of code.[2] You'll have to give them a helping hand—for example, placing your cells in a salty solution can make their membranes leaky enough for plasmids to slip inside.

[2] Perhaps they know not to open suspicious email attachments?

Markers

Whatever cells you're working with, however you're adding plasmids, the process isn't going to be 100% effective—at the end of the day there are still going to be a bunch of cells left that did not incorporate your plasmids. (In fact, it's worse than that, because the process of recombination you just used to make your plasmids isn't perfect either. You'll actually end up with a mix of altered and unaltered plasmids, and cells that have absorbed either plasmid, both plasmids, or neither). Or, in other words, if you follow the recipe above you'll end up with a mixture of modified and unmodified (or **wild-type**) cells.

If you want to see what your DNA does, you'll have a rather noisy signal—and if you wanted to use some bacteria to synthesize a protein for you, you're going to be culturing a bunch of free-loading wild-type cells that are doing nothing useful. To fix this, you need some way of killing off everything that didn't get a plasmid, or got the wrong kind of plasmid.

Let's imagine that our plasmid originally contained the sequence (xy)(z), where (xy) is a gene that makes the host resistant to a particular antibiotic A, and (z) does the same thing for antibiotic B. By picking the right restriction enzymes (or using CRISPR), you can make your "cut" right in the middle of gene xy—and make sure that (w), your strand of recombinant DNA, was "cut" at the same sites, so that the sticky ends will match. If you did it right, you'll wind up with the sequence (x(w)y)(z),[3] a functional gene (w), and a nonfunctioning gene (xy). You'll also still have a bunch of (xy)(z) plasmids that closed back up without any changes.

Once you've finished introducing your plasmids to your cells, you'll wind up with three intermixed populations.

- Cells that haven't picked up a plasmid at all, and are vulnerable to antibiotics A and B.
- Cells that picked up only nonaltered plasmids, and are thus resistant to both A and B.
- Cells that picked up altered plasmids, which—because (xy) no longer functions—are only resistant to antibiotic A.

Getting rid of the first group is easy—all you need to do is add antibiotic A, and that will kill them. Removing the second group is harder, since the

[3] Actually, you'll also wind up with (x(w)(w)y)(z). And (x(w)(w)(w)y)(z), and (x(w)(w)(w)(w)y)(z), and on and on. This isn't a problem; it just means that some plasmids will tell the cell to make more of protein W than others. For the purposes of this explanation (and to keep the page count down), I'll keep using a single (w).

population you want to kill is the one that *resists* the drug. The simplest way to do so, **replica plating**, takes advantage of bacteria's rapid reproduction. If you seed a petri dish with a relatively small number of cells, the resulting colonies will be widely spaced and **monoclonal**—each one grown from a single cell. That means you can take a tiny bit from a colony, transfer it to a new plate, and be confident that anything that grows on that plate is the same as the original colony.

To break it down then, in replica plating, you first take a single petri dish D1 that contains many cell colonies, and copy the colonies, in parallel. (One way to do this is to touch a blotter to the surface of the dish, and then touch the blotter to a new dish.) After some cell growth, the result is a copy D2 of the colonies in D1, where all colonies have the same relative position in D1 and D2. You then treat D2 with antibody A and see which colonies die off: these are sensitive to A. Finally, you go back to the original disk D1 and pick out the colonies that were the sources of the A-sensitive colonies in D2.

Promoters and Regulating Recombinant Organisms

If you wanted to use some bacteria to synthesize a protein, you might still be in for a disappointment: it might be that the next morning, all your carefully selected cultures are dead. One likely reason is that your poor bacteria drove themselves into exhaustion trying to keep up the kind of protein expression your plasmid demanded.

To keep this kind of thing from happening,[4] you need to make sure that whatever gene or genes you're injecting can be easily **regulated.** That means picking a promoter that only functions under certain conditions, such as a high temperature or different food source. That way, you can give your cells the time they need to grow and reproduce before starting your experiment.

Phages as Insertion Vectors

Plasmids aren't the only **insertion vectors** that can be used to inject recombinant DNA into other cells. They were the first that biologists learned to use, but they're limited to short strands of DNA, around 8000–20,000 base-pairs long. If you want to transfer more, you'll need to use viruses. After all, they've evolved to perform exactly this task—inserting foreign DNA into a living

[4] To say nothing of your creation breaking out of your lab and wreaking havoc on the unsuspecting population.

cell—and nothing else. They can handle much longer DNA sequences, and come pre-armed with the tools to get it inside a cell. The process is basically identical; the only difference is that you replace your custom plasmids with bespoke viruses.

For safety, you'll want to use viruses that only infect single-celled organisms, known as **phages.**

Using Genomic DNA Libraries

The experiments above are feasible for many labs to do in-house. But they can also be scaled up and parallelized by specialist labs. Like most of the techniques we've talked about, you can deploy insertion vectors (or other tools for injecting foreign DNA) in a massively parallel way and create a **genomic DNA library** for other scientists to use.

A genomic DNA library is a large collection of cells that have had another organism's DNA inserted—one gene's worth for each colony of cells. Because those colonies can then continue to grow indefinitely, it's easy to split off a new population to use for a particular experiment without depleting the original stock. In other words, it is never necessary to "return" anything to this library—one can withdraw a copy of every piece, and the entire library will still be available for the next researcher.

Genomic libraries are typically created by randomly fragmenting the subject's genome, inserting *those* pieces into plasmids, and adding multiple plasmids to the same colony of cells. By carefully balancing the number of cells to the number of plasmids, you can make it just hard enough for a single cell to grab multiple plasmids so you can be reasonably confident that cells have only a single plasmid, if they survive the initial screenings.

Genomic DNA libraries are particularly useful finding the DNA code that gave rise to a particular mRNA molecule[5].

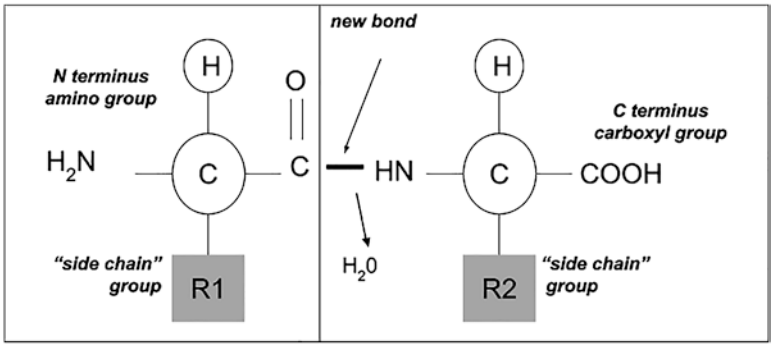
[5] In eukaryotes, mRNA is often spliced and otherwise altered before it's translated; the sequence of the final molecule can be quite different from the original gene—meaning you can't just search the genome for the exact mRNA sequence.

Creating Novel Proteins: Tagging and Phage Display

Adding new genes isn't the only thing that recombinant DNA is good for. It can also be used to *modify* already-existing genes. Sometimes that means messing with a protein to see what happens if you swap out *this* amino acid for *that* one—If something goes dramatically wrong, that was probably important to the protein's function. And sometimes it means **tagging** particular proteins by adding some kind of marker that makes them easier to find later on.

The classic example of this is probably **green fluorescent protein (GFP)**, which we briefly mentioned a few sections back. (Other colors exist, but green was the first to be discovered.) One way to track a protein in a fluorescent microscope is not to use a dye, but to modify the DNA of the protein so that it contains, in addition to the usual amino acid sequence, a sequence for GFP. Originally found in jellyfish, GFP has two properties that make it a popular choice for this:

- It's small—made up of less than 238 amino acids—and thus can be added to the end (the n- or c-terminal; see Fig. 1) of another protein, usually without interfering with its functions. (Exceptions exist, but they're rare.)
- It's very stable and remains fluorescent for a long time before photobleaching. Once produced, GFP creates a new set of covalent bonds that let them absorb and reemit light, then curls up into a narrow tube around those precious bonds, offering extra protection.



A protein, which is a chain of amino acids, has an **N-terminus** (where there is an unlinked nitrogen-containing **amino group**) and a **C-terminus** (with an unlinked carbon-containing **carboxyl group**).

Fig. 1 Structure and nomenclature of proteins

Once altered, the resulting protein is known as a **fusion protein** or **chimeric protein.**

Besides microscopy, another good use for fusion molecules is to mess with a virus' protein "coat." Adding the gene for a particular protein to the right section of the virus' genome results in it becoming part of the coat—the virus "displays" the protein on the outside, where it's free to bind with whatever other proteins it typically interacts with. This is another of those techniques that can be deployed in parallel. By creating **phage displays**—large collections of viruses that have each been altered to "display" a different protein—you can test millions of potential protein partners at the same time.

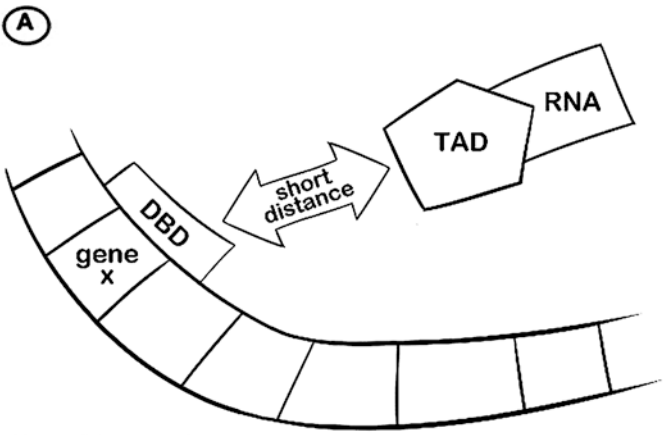
You can test your altered viruses against a target protein or molecule *q* that's been anchored to the bottom of a petri dish. All you need to do is add your phages, each for a different protein *p* to the plate, and give them some time to find partners. Then you can wash away the excess so that only phages "displaying" proteins *p* that bind strongly to the target *q* remain in the plate. This lets you know which proteins *p* interact with *q.* The viruses might not be fluorescent, but they do carry the gene for whatever protein just interacted with the target, and extracting and sequencing the DNA from those leftover viruses lets you precisely identify those proteins.

Yeast Two-Hybrid Assays Using Fusion Proteins

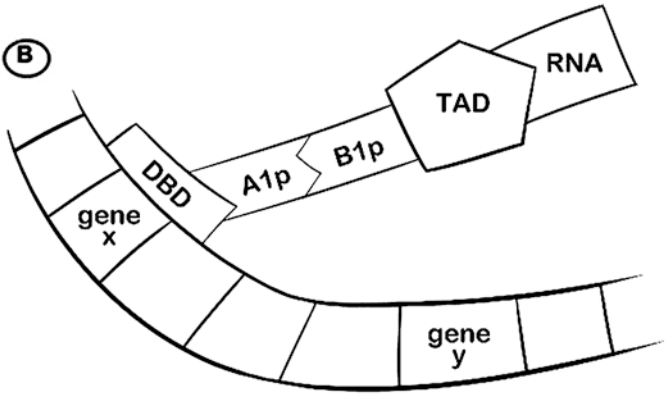
It's even possible to test protein-protein reactions at the genetic level by taking advantage of an interesting quirk. Most eukaryotic **transcription factors**—the proteins that help RNA synthase attach itself to the DNA—are made up of two subunits. The **DNA-binding domain (DBD)** handles the connection with the strand of DNA; the **transcription activation domain (TAD)** calls in transcriptional machinery like RNA synthase.

And the two subunits *don't need to bind together* to act as a functional transcription factor—mere proximity is enough (see Fig. 2A). Conversely, if you stop the two subunits from getting close to each other, they won't encourage any transcription. You might have already guessed where this is going: to test if two proteins (let's call them *p* and *q*) interact, you can combine one with each half of the transcription factor for one of those reporter genes. If *p* and *q* interact with one another, the transcription factor will come together and work normally (Fig. 2B).[6]

[6] For the most part, the two subunits would be physically connected in this case, which may or may not affect things.



(A) In wild yeast, *A* binds *B*, which activates gene *x*. Only the DNA binding domain (DBD) is needed for *A* to find the promoter site, and only the transcription activation domain (TAD) is needed for *B* to activate transcription.



(B) In hybrid yeast, the DNA has a promoter for x near a reporter gene y. *A1p* can bind to the promoter site using the DBD of *A*, and *B1q* will activate transcription—of gene *y*—using the TAD of *x*. But *A1p* will only recruit *B1q* if proteins *p* and *q* bind. So, *y* is expressed iff *p* and *q* bind.

Fig. 2 The yeast two-hybrid system

To perform such an experiment, you'll need to find or introduce is a **reporter gene** that has an obvious effect when expressed, and arrange the DNA so that a reporter gene can only be expressed when activated by *p* plus the DBD and *q* plus the TAD. It's possible to use something like GFP as a reporter gene, but it's usually easier to cut out the middleman and use a vital

metabolic gene—one that the cell has been altered to lack. If the two proteins *p* and *q* interact, then the DBD and TAD domains will be put in proximity, the reporter gene will be expressed, and the cell will live. If they *don't* interact, the subunits will not get close enough to do their job, the vital protein won't get made, and the cell will die.

What makes this process especially useful is how easily it scales up. All you need to do here is assemble two separate cell libraries—one with a wide range of options for protein *p*, and the other with lots of variants on protein *q*. Once prepared, cross-breeding the two libraries results in a massive number of different hybrids, each with a different *p* and *q* pairing. At that point, the mechanics above kick in—cells whose *p* and *q* variants interact will survive the screening, and cells with nonmatching variants will die.

Since such experiments are normally done using yeast cells, the process is known as the **yeast two-hybrid system**.

Other Ways to Use Biology for Biological Experiments

Replicating DNA in a Test Tube

DNA Replication: The Basics

Before a cell can start dividing its cytoplasm, it first must create (or replicate) a second copy of its genome. The process involves two steps: **initiation** and **polymerization**.

During initiation, special proteins attach themselves to the double-stranded DNA and "unzip" it into a pair of components called (not surprisingly) strands. Another protein, **RNA primerase**, attaches to each new strand and uses it as a template to assemble a short strand of RNA. This, in turn, will serve as a scaffold for the rest of the (many) proteins involved, which come together to form a complex called a **replisome**.

During polymerization, each replisome moves down its strand, grabbing free-floating nucleotides and using them to build a complementary strand. When they reach the end, they pop off, and you're left with two perfectly matching double-helixes.

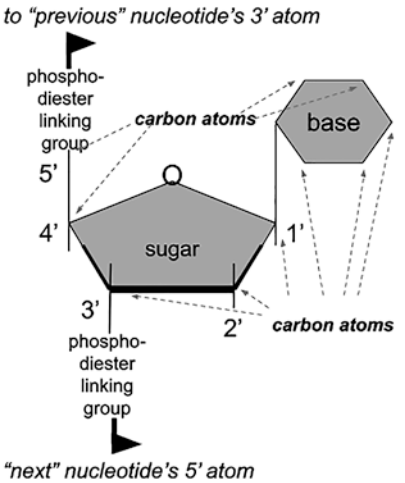
A molecule that contains a number of repeated units arranged linearly is a **polymer**. DNA, RNA, and proteins are all polymers.

W. W. Cohen, C. K. Cohen, *A Computer Scientist's Guide to Cell Biology*,
https://doi.org/10.1007/978-3-031-55907-5_7

DNA Replication: Not So Basic

There's just one little part of that explanation that throws a wrench in things—the word "down." That doesn't really apply to densely coiled, endlessly long strands of DNA floating in cytoplasm. Figure 1 gives names to some of the landmarks in DNA: you can see that a better term might be "from 5′ to 3′." **DNA polymerase III**, the key enzyme at work here, only works in one direction.

Except…DNA is asymmetric. The strands "point" in different directions. When you start to unzip them, one (the **leading strand**) will have its 5′ end pointing out; the other (the **lagging strand**) will present its 3′ end. The leading strand is unrolling in the same direction that DNA polymerase works, but the lagging strand is moving backward. No smooth rides there.



A **nucleoside** consists of a **nucleobase** (e.g., adenosine, thymine, cytosine, guanine) and a **sugar group—ribose** for RNA, and **deoxyribose** for DNA. Normally sugars are linear atoms, and the carbon atoms are numbered 1,2,3,4,5. In nucleic acids they fold into a ring, but the atoms are numbered in the same order; however, they are labeled 1',2',...,5' to distinguish them from the carbon atoms on the ring associated with the nucleobase (which are labeled 1,2,...,6).

A **nucleotide** is a nucleoside plus a phosphate group, which links it to the next nucleotide in the polymer. The phosphate groups link the 3' atom in one nucleotide to the 5' atom in the next. By convention, DNA strands are usually written with the "5' end" (the end with a "dangling" 5' carbon, not attached to any nucleotide) to the left.

Fig. 1 Structure and nomenclature of DNA molecules

Instead, duplication proceeds in a series of jumps. At first, nothing happens. When the lagging strand has around a thousand base pairs exposed, DNA polymerase will jump in and copy those, then detach as it runs out of single-stranded DNA to copy. When the next thousand pairs are revealed, it'll repeat the process. And again, and again, in an endless series of steps.

It's a messy process; although all the base pairs will be in place, the backbone will still be broken up into short chunks. The cell will need to deploy additional protein machinery to patch up those holes. Other proteins will "proofread" the newly generated DNA (from both strands) (Fig. 2 illustrates this process).

DNA Replication in a Tube: PCR

Initiation isn't the only way to unzip DNA. The bonds between nucleotide pairs are much weaker than those of the backbones. Crank up the temperature until it's close to a hundred degrees Celsius, and they'll break apart (or **denature**) and give you a pair of single-stranded DNA molecules, just like that.

That's not a very useful trick for cells, which would burn to death long before getting hot enough for their DNA to denature. On the other hand, it's a wonderfully useful trick when you're working with purified DNA in a test tube. Heat it up until it denatures, then add polymerase once it cools to turn those single strands back into double helixes—twice as many as you started with, in fact. Enough repetitions and you'd have more than enough DNA to play with, even if your initial sample was very small.

It was a time-consuming process, though, until the discovery of variety of DNA polymerase III capable of surviving high temperatures.[1] Once we started using it for DNA replication, we no longer needed to keep adding more polymerase after every heating cycle. The process could be entirely automated using what is, essentially, nothing more than a fancy heating block. Entire platefuls of DNA could be heated until the strands denature, cooled until the polymerase finished, then heated again, then cooled again, on and on and on.

That mostly automated process is known as **polymerase chain reaction,** or **PCR**. And it is *phenomenally* powerful—each cycle doubles the amount of DNA available. After a few dozen cycles, even a single molecule can be **amplified** to the point that it can be further studied or sequenced.

That kind of power makes PCR very useful for forensics, where you have very limited sample to work with. It also makes PCR very sensitive to contamination, since any foreign DNA will *also* be doubled with each cycle. Figure 2 illustrates PCR, and how it differs from natural DNA replication.

[1] To be precise, it was discovered in "extremophile" bacteria living in hot springs.

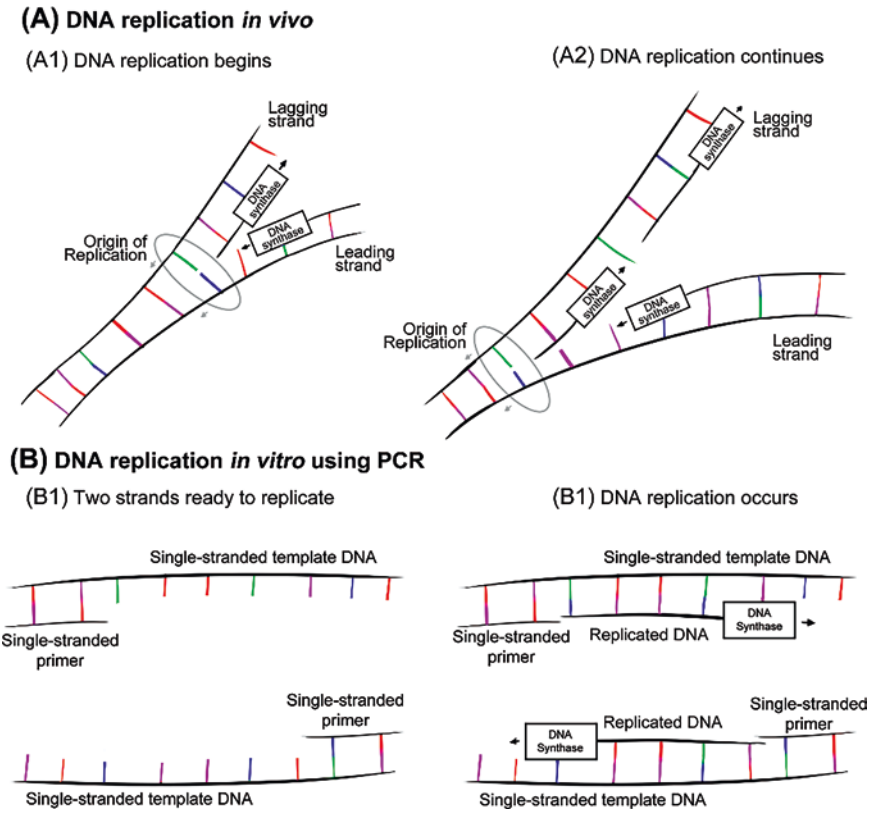


Fig. 2 How DNA is duplicated, naturally and with PCR

Sequencing DNA by Partial Replication and Sorting

Sanger Sequencing

How can we read something as small as DNA?

Suppose, while running PCR, we supplement one of the four base (A, T, G, C) with a variant called a **dideoxynucleotide**, which halts replication once placed. Better yet, let's prepare four versions of this flawed process, supplementing a different nucleotide each time, and run everything through PCR a few times.

We'll wind up with four distinct populations of DNA: one where replication stopped after adding an A nucleotide, one where replication stopped after adding a T nucleotide, one for G, and one for C.

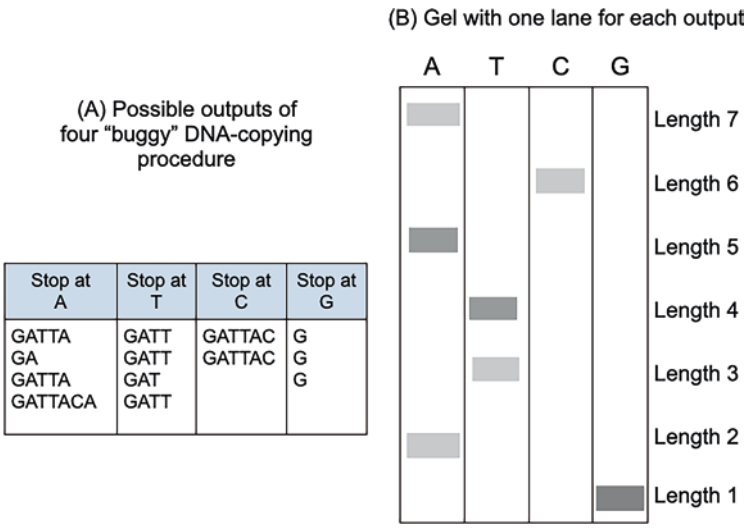
Note the "and" there, though. Because the PCR mixture you use contained both the normal and dideoxynucleotide versions of A, not all of the resulting

fragments will be the same length. Some will have grabbed the variant and stopped the first time they hit an A, some would have gotten one "real" A and a variant on the second, some would have stopped after the third A, and so on. You would, in fact, have every possible sequence of that DNA molecule that ends with A.

Same goes for the other three nucleotides.

The next step is to use run a gel and separate each set of DNA fragments by size. The longer the fragment, the more slowly it will travel through the gel. Comparing the bands to a control lane, prepared using DNA strings of known lengths, we can tell how long each fragment is.

We can, therefore, look at our gel and say "the A mixture has a band at 5 nucleotides; therefore, the fifth letter of the DNA is A." The entire sequence can be read right off the gel, starting at the bottom (i.e., the shortest strand) and working your way up[2] (Fig. 3).



Sanger method for sequencing DNA. (A) The result of using variants of a DNA-copying procedure on many copies of a single strand of DNA. The variants randomly stop at prefixes that end in "A", "T", "C", and "G", respectively. The result of this is four populations of prefixes of the unknown DNA strand. (B) Using a gel to separate the four populations by weight. From the gel, the length-7 prefixes end in "A", the length-6 prefixes end in "C", and so on. Hence the final string can be reconstructed as "GATTACA".

Fig. 3 Sanger method for sequencing DNA

[2] To make the process easier (and easier to automate), most labs these days use fluorescent dideoxynucleotides, with each of the four bases glowing in a different color.

is hour upon the stage and
shadow, a poor player that
and then is heard no more.
but a walking shadow, a poor
struts and frets his hour upon
a poor player that *struts and*
life's **but a walking shadow**
hour upon the stage and then is
the stage and then is heard

Fig. 4 Example of how to reassemble random substrings (see text)

This type of sequencing, known as **Sanger sequencing**, was one of the earliest methods for reading DNA, but it came with a rather significant flaw. Putting the fragments in order means being able to distinguish between, say, a 516-base-pair-long sequence and a 517 one—needless to say, not an easy task. Sanger sequencing is usually limited to around 1000 base pairs, while the human genome is a bit longer—around 3 billion pairs.

This gap is bridged by **shotgun sequencing**, which is a process in which many random overlapping segments of DNA are sequenced, and then computers are used to reconstruct the original DNA strand. The key idea is illustrated in Fig. 4 with some random and overlapping short subsequences from a possibly familiar bit of English text.

It's easy to see the general idea of how to solve this sort of puzzle: for example the overlapping text in bold ("but a walking shadow") suggests we can put together those two fragments to get "…life's but a walking shadow, a poor…", and then underlined common fragment brings us to "…life's but a walking shadow, a poor player that…", and so on. Even with millions of fragments, computers can handle this sort of task well. The main difficulty is doing the right thing with repetitions in the genome—for example, with short enough fragments, it might be impossible to determine if the start of this quote is "tomorrow and tomorrow" or "tomorrow and tomorrow and tomorrow."

Massively Parallel Sequencing

We have, however, come a long way since the days when Sanger sequencing was the pinnacle of biotechnology. New techniques for automation and parallel processing have given rise to the practice known as **massively parallel** or **next-gen sequencing (NGS)**. Like Sanger sequencing, NGS can only read

short fragments of DNA—but it can look at millions at once, and distinguish between dozens of distinct samples.

As in, "you just dump all your (fragmented) samples in the same well and press go."

Once inside the machine, DNA fragments wash over a specially made surface where one end sticks fast, leaving the other to dangle helplessly—and ready to accept new additions. The machine then adds a flood of fluorescently labeled "A" nucleotides (adenine). Because DNA synthesis must begin at one end of the strand and continue in a single direction, without skipping any bases, only fragments whose first base is "T" (thymine) will manage to grab one of the new fluorescent nucleotides. Ultrasensitive cameras take note of each spot where a fluorescent molecule was just added, and the excess is washed away. Then the process repeats—As, Ts, Gs, Cs, As, Ts, Gs, Cs, and on and on—until each DNA fragment is complete, and the machine knows the exact order of the bases attaching to each of the fragments.

Four-Channel Sequencing

More sophisticated machines can actually add multiple nucleotides at the same time, using different fluorescent "colors" to figure out which nucleotide was just added.

How Fast, Cheap Sequencing Has Changed Biology

Twenty years ago, when the first human genome was sequenced, it took years and cost billions of dollars. The advent of faster and better DNA sequencing has led to many new applications.

One is **precision medicine,** where decisions about how to treat a patient are made based on looking at that individual's own genome. One part of this is **pharmacogenomics,** the study of which drugs to use based on genetic information. The potential benefits of this are obvious to anyone that has had to try one therapeutic drug after another to treat a common problem, waiting days or weeks before being able to tell if the latest medicine is really working. But there are huge research challenges involved in many aspects of this, not least the ethical issues involved in collecting and distributing databases of individual's medical and genomic information, and the legal and regulatory challenges in ensuring highly targeted therapies are safe.

Another fascinating research field that is enabled by advances in sequencing is **metagenomics**. Culturing a bacterial colony so you can sequence its DNA

is slow work, but sometimes it is possible to collect a sample from an environment of interest—say seawater—that contains many microorganisms, and then amplify the DNA from that sample and shotgun sequence all of it, and finally put the pieces back together. The end result is a DNA sample from many species, sometimes hundreds or thousands of species (usually of bacteria or viruses), all of which are present in that sample. Sometimes this process reveals the existence of unknown species, whose characteristics can be inferred from their DNA.

Metagenomics involves many difficult challenges—for computer science, challenges include assembling DNA sequences from an unknown number of organisms simultaneously, and also doing data analysis with very large-scale datasets that are created: samples from the human gut, for instance, can lead to hundreds of billions of base pairs of DNA.

Other In Vitro Systems: Translation and Reverse Transcription

Translation in a Tube

These processes—and a surprisingly large number of others—can be carried out in vitro, outside of living cells. That means we can play with them in ways that would be impossible in a living organism.

Take, for example, translating an mRNA molecule into a protein. Ideally, you'd only want three things in your test tube—the mRNA, a supply of amino acids, and the finished protein. But the process of translation involves a lot of other proteins, which can make it difficult to find the one you're looking for on a gel.

But if you add radioactively or fluorescently labeled amino acids to the tube, you can be confident that whatever protein the mRNA codes for will be built out of those labeled amino acids; any other proteins will be dark, making it very easy to pick out your target on a gel.

Reverse Transcription

Normally, genetic information in a cell only flows one way: from DNA to mRNA to protein. But some viruses have "learned" how to reverse that process, and use **reverse transcriptase** proteins to turn their RNA into **complementary DNA (cDNA)** that can be spliced into the host cell's genome.

This, naturally, means that *we* can use those viral proteins to do the same thing to create **cDNA libraries.** By isolating all of the mRNA in a cell, then using reverse transcriptase to turn it into cDNA, you wind up with a DNA library that reflects the *activity* of cell rather than its genetics. In the cDNA library, genes that are being highly expressed will appear in the library many times; genes that aren't being used won't appear at all. You can do this sort of counting by sequencing the cDNA, or with microarrays, or other techniques discussed below.

Antibodies: Exploiting the Natural Defenses of a Cell

One of the most useful tools in modern molecular biology is the **antibody**. These Y-shaped proteins are part of the immune system and are designed very carefully to bind to one random foreign substance, their **antigen**. Just one, and no other.

In the body, antibodies make it harder for pathogens to operate (since they're covered in big Y-shaped proteins) and signals other immune cells to attack their target. In the lab, the specificity of antibodies makes them perfect for picking a given substance or molecule out of a mix—just add the right antibodies. (Or, more likely, attach the right antibodies to a series of microbeads and add the mixture.)

Making Antibodies

Antibodies are very easy to produce. If you need antibodies for substance X, all you need to do is inject a small amount of it into a mouse or rabbit and wait for the immune system to do its job. After a few days, the animal's bloodstream should be filled with antibodies targeting X (or "anti-X" antibodies) and getting your hands on the one you want is just a matter of filtering and purifying a blood sample.

If you want to get really serious about antibody production, you can instead isolate some of the mouse's B-lymphocyte cells—cells whose only job is to pump out antibodies on demand—and screen out the ones making anti-X. (Ironically, culturing these important immune cells requires cancer cells. B-lymphocytes are extremely difficult to grow in vitro, and so are often crossed with cancerous B-lymphocytes to create hybrid cells, or hybridomas.)

Immunofluorescence

One of the most common uses of antibodies in the lab is to create highly specific dyes, by attaching a fluorescent molecule to one end. Add the labeled antibodies to a sample, wash away the ones that don't bind, and you'll have a glowing map of every part of your sample containing that antigen—a process known as **immunofluorescence.**

First you add a specific **primary antibody** that binds to the target you're looking for, then you add a **secondary antibody** that targets the primary ones.

One important application of antibodies is to construct highly specific fluorescent dyes—dyes that affect only a specific protein X. This is typically done in a modular way with two types of antibodies: Ab1, a primary **antibody** against X, which is produced in (say) rabbits; and Ab2, a general-purpose fluorescently tagged **secondary antibody** that binds to all rabbit antibodies. In the cell, Ab1 will bind to X, and Ab2 will bind to Ab1, thus tagging X. This scheme means that you can deploy several different primary antibodies at the same time. As long as they're from different species, you can tag each one with a differently colored secondary antibody.

Antibodies in Action: COVID Tests

Since the COVID-19 pandemic began, we've come to rely on **rapid antigen tests** to monitor its spread and let us know if it's safe to come out of our holes and see our friends. Despite their simplicity, they're a great example of how different types of biological techniques can be combined.

When you stick the swab in your nose, you're collecting more than just a nasal sample, you're also sampling all of the bacteria, fungi, viruses, pollen, and dust that have gotten stuck there. And if you're unlucky enough to be infected, some of those viruses will be COVID-19.

The next step, the "extraction solution," breaks everything apart and leaves you with a soup of random proteins and lipids—an example of fractionation.

When you drip some of that soup onto the designated spot in the plastic test cassette or card, you're soaking one end of a paper strip. There, it mixes with a bunch of custom-made antibodies. The "business end" has binding sites for one of the proteins that make up the COVID-19 viral particle (or **antigen**); the other end has been attached to a molecule of dye.

And then, as with any chromatography test, it takes a little time for the liquid to migrate its way up the paper, carrying the antibodies along for the trip. There, they'll encounter two obstacles—lines of antibodies that have been fixed to the paper.

The first line they cross is also made up of anti-COVID antibodies, fixed to the paper so they won't be swept along with the liquid. The second line, the control line, is made of anti-antibody antibodies[3] that will grab onto any antibody that comes their way.

If you're lucky, there won't be any COVID proteins in your system, and all of the dye-labeled antibodies will get stuck to the second, or, control line. With millions of them in the same place, the dye they were labeled with becomes concentrated enough that we can see a nice blue line.

If you do have COVID-19, though, there won't just be unencumbered dye-labeled antibodies—many of them will have attached themselves to a viral protein. When they reach the first detection line, the fixed antibodies will *also* grab hold of that protein, stopping the antibodies in their tracks. If enough are trapped, a blue line becomes visible, and you have your unfortunate answer.

All without leaving home.

Costs and Benefits

Antigen tests are far from the only way to detect COVID-19 infection, of course. **Molecular testing**, which detects the presence of viral mRNA, is both more reliable and more sensitive—if your infection is still in the earliest phases, there might not be enough protein in your snot to give you a visible line. Molecular tests, on the other hand, run your sample through PCR to amplify even tiny traces of the virus.

But such tests *also* require a lab full of trained scientists (or, at the very least, one harried lab tech). The advantage of rapid antigen tests is that they don't require any special training or equipment to perform. All you need is basic literacy and a functional knowledge of where your nostrils are.

[3] Say that three times fast.

mRNA Vaccines

In recent years, a revolution in public health has taken place—perhaps the greatest since wide-scale DNA sequencing became practical. Even if you've never heard of it, you're probably already intimately familiar with it. You've probably lined up to have it injected into your body, in fact.

mRNA vaccines have been in development for almost 20 years, but they exploded into prominence during the COVID-19 pandemic. In some ways, they're not much different from any other vaccine; in others, they represent a quantum leap in biotechnology.

Traditional Vaccines

Before we can understand what's so cool about mRNA vaccines, let's take a moment to remember how "normal" vaccines work. And to do *that,* we're going to need to talk about the immune system a bit. Bear with us.

The Immune System

The human immune system is deeply fascinating, and almost unbelievably intricate. The body has dozens of mechanisms for fighting off invaders—fever, inflammation, phagocytes, natural killer cells, B cells, killer T cells, helper T cells, antibodies, complement cascades, and that's only scratching the surface. If you're at all interested, it's worth diving into, but for now we're going to have to simplify some things even more than usual.

The human immune system has two components—the innate immune system and the adaptive immune system.

The cells and proteins of the **innate immune system** respond first, killing invaders and breaking them down into smaller fragments. These fragments are then "presented" to the adaptive immune system. The cells of the **adaptive immune system** vary wildly, but *some* of them will be able to recognize the fragment and leap into action. And once the threat is clear, certain adaptive immune cells—**memory cells**—live on, ready to spring into action the moment they see their target again. It's those memory cells that give us long-lasting resistance.

Vaccines, then, are an attempt to "trick" the adaptive immune system into producing memory cells—without getting you sick in the process. Sometimes

that means a weaker (or **attenuated**) form of the disease; sometimes it means dead disease-causing cells; sometimes it means bits and pieces of the pathogen. Whatever its exact contents, the immune response is triggered—innate immune cells track down the invader and show it to the adaptive system, which kicks off and produces memory cells (in addition to large numbers of "active" cells). That way, if you run into the disease in the wild, the adaptive immune system is ready and waiting to spring into action.

mRNA Vaccines

"Normal" vaccines use a dead or weakened form of the pathogen to provoke a response—but if you think about it, the adaptive immune system doesn't need an entire bacteria or virus to create a response. All it needs is a fragment of the whole, a piece important enough that it's unlikely to change as the pathogen evolves.

We've taken advantage of that fact before. **Subunit vaccines**, such as the ones for HPV or shingles, only contain isolated proteins or sugars from the pathogen—but those proteins or sugars still have to be made in a lab. Depending on their complexity, that can be a difficult, time-consuming process. They also provoke weaker immune responses, requiring more frequent boosters.

"But hang on," an early researcher presumably said. "Why bother making the protein beforehand? The body's great at making proteins; why not just give it a blueprint and let it do the work for us?"

And thus was born the mRNA vaccine.

Instead of providing dead pathogens, or bits of pathogens, the vaccine would contain mRNA *coding* for a bit of the pathogen. (In the case of the COVID-19 vaccines, one of the spike proteins that help the virus enter host cells.) Once injected—and picked up by a cell—that mRNA strand would be translated like any other, producing that pathogen-specific protein without the body having to get anywhere close to an actual disease-causing agent.

It's not quite that simple, of course. The body naturally tries to digest mRNA, especially foreign mRNA, and nucleic acids are pretty bad at slipping through cell membranes. To get the idea to work, the mRNA needs to be enclosed in a **lipid nanoparticle** before being injected. Unlike free-floating mRNA, the nanoparticle—a tiny sphere of phospholipids—*can* easily make it into a cell. The lipids of the nanoparticle simply merge into the cell membrane, leaving its contents—the mRNA—to float away into the cytoplasm.

RNA vaccines have three major advantages compared to "traditional" vaccines.

- Making large quantities of mRNA is fast and easy; producing lots of pathogens (or their subunits) is slow and requires huge cultures of disease-causing agents.
- Because it uses the body's own machinery, an mRNA vaccine can lead to more of the protein in question being produced than would normally be used in a subunit vaccine, creating a stronger immune response.
- Since the only significant difference between an mRNA vaccine for COVID-19 and one for, say, influenza, is the mRNA sequence itself, it's very easy for factories to switch production to accommodate new variants or diseases.

The COVID-19 vaccine is living proof of their efficiency. The virus first appeared in early 2020, and it took less than 6 months for mRNA vaccines to reach large-scale testing. By the time 2021 rolled around, the first doses were reaching first responders; by the end of the year, more than six billion doses had been distributed. Since then, several updated vaccines and boosters have been developed to keep up with the continually mutating virus.

RNA Interference

One of the best ways to figure out a gene's function is to get rid of it and see what changes. (Your cultures all die? Guess it's more important than you thought!) The most direct way to do so is via genetic editing techniques, such as recombinant DNA, and is known as **knocking out** the gene.

Alternately, you could turn to **post transcriptional gene silencing**—interrupting the normal mechanisms of transcription and translation so that a gene never makes it into protein form.

The most straightforward method of doing so is **RNA interference**, or **RNAi**, and exploits one of the key differences between mRNA and genomic DNA—mRNA is single-stranded. RNAi involves adding a second strand of RNA that's complementary to the one you want to silence, sending the molecule off on a completely different trajectory.

Instead of heading to a ribosome to be translated into protein, the newly double-stranded (and, thus, alien) RNA is attacked by an enzyme called **dicer**.

As the name suggests, dicer proceeds to chop the RNA strand into tiny pieces, no more than 25 or so base pairs long.

But the effect goes beyond preventing that particular mRNA strand from being translated. The fragments, now known as **small interfering RNAs (siRNAs)**, combine with other RNA-degrading proteins to form an **RNA-induced silencing complex (RISC)**. Once it forms, RISC is highly specific—it will only destroy mRNA strands that match its siRNA. The rest of the cell's functions are completely untouched.

Why does this happen? That's a good question. We know that some viruses use double-stranded RNA as their genetic material, which would suggest RNAi evolved as a defensive measure. But long pieces of double-stranded RNA already produce a strong antiviral response (hence the use of tiny siRNAs instead). We know that some species use RNAi to regulate the expression of their own genes; we know that many others don't. Regardless of their origins, today we can use siRNAs to quickly and easily silence any gene we want. And in the future, they might become a medical tool as well as a biochemical one.

Serial Analysis of Gene Expression

To finish off this chapter, we're going to take a look at one more experimental process in some detail: **serial analysis of gene expression** (**SAGE**), which lets us "summarize" the cDNA by snipping out short sequences, long enough to identify the gene, but short enough that sequencing is still easy.

SAGE is useful because while microarrays are useful, they are not necessarily efficient. You have to extract mRNA, use reverse transcriptase to turn it into cDNA, and use PCR to make lots of copies before you can actually run the microarray. You're also limited by the availability of microarrays—if vendors don't make them for the particular organism or gene you're interested in, you're out of luck. You could also directly sequence the cDNA to study gene expression levels. That would be a neat way of getting around the limits of microarrays. The only problem is that the procedure isn't cheap, and sequencing large volumes of cDNA is beyond the means of many labs.

An alternative is to use SAGE. The first step of SAGE is, as with microarrays, isolating mRNA and using reverse transcriptase to turn it into cDNA. This

time, though, you'll make sure that one end (specifically, the one corresponding to the mRNA's polyA tail) is marked with a molecule called **biotin.**[4]

A protein called **avidin** binds strongly to biotin, so mixing your cDNA solution with avidin-coated microbeads is a quick way to anchor it—specifically, to anchor it to the *polyA tail* side of the cDNA. Unlike most filtering methods, this method guarantees you'll wind up with all of the cDNA strands pointing the same way.

The next step is to add a restriction enzyme with a short enough binding sequence that it'll find lots of places on the cDNA molecules to make cuts. Wash away the excess and you'll still have cDNA bound to your microbeads, but it will be much shorter. And, crucially, each strand will have the same "sticky end." That means you can use DNA ligase to attach the same sequence to each cDNA strand. Specifically, you'll attach a marker sequence that contains a binding site for a restriction enzyme called **Fok1**.

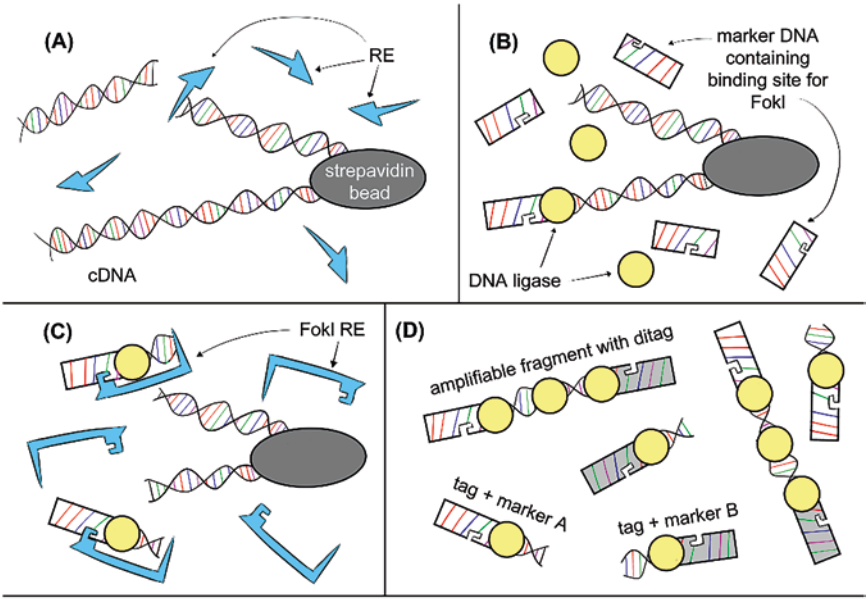
Why do we want to make use of this particular RE? What's special about it? Well, unlike many other REs, Fokl makes its cut a short distance "downstream" of its binding site. Mixing it with your microbeads results in very short free-floating fragments of DNA. One end will still have the marker sequence and Fok1 binding site; the other will have a sticky end. The middle, the short sequence clipped from the original cDNA, is known as a **tag.**

Because the initial restriction enzyme had so many different sites where it could have made cuts, the cDNA strands attached to the microbead were of all different lengths. Thus, although Fok1 will chop off the same number of base pairs each time, you'll get a wide variety of tags—all taken from the *middle* of the original cDNA strand.

Repeating the process with a different marker sequence creates a second flood of tiny fragments with a marker on one end, a tag in the middle, and a sticky end on the other. Because all of the fragments have now been cut by the same restriction enzyme, all of their sticky ends will match, and you can mix the two populations to create **ditag** molecules, with a marker on each end and a tag in the middle.

Finally, you'll amplify the ditag molecules using PCR, and it's those ditag molecules that you'll sequence. Fok1 cuts off 20 nucleotides at a time, meaning that each ditag molecule will only have 40 nucleotides worth of tag—short enough to easily read using Sanger sequencing (Fig. 5).

[4] You might also know it as vitamin B_7.



The process used in SAGE for "summarizing" cDNA.
(A) A RE is used to randomly cut cDNA bound to a strepavidin bead.
(B) Marker DNA fragments are ligated to the ends of the remaining cDNA fragments.
(C) The RE FokI is added. FokI binds to a site in the marker DNA, and cuts 20bp downstream of the binding site. Each cut releases a fragment of DNA containing the marker and a small part of the original cDNA, called a **tag.**
(D) Two populations of marker-tag fragments, with different markers, are mixed and ligated together. PCR can be used to amplify those cDNA fragments containing both markers, which must also contain at least two tags. These **ditag**-containing DNAs are then sequenced.

Fig. 5 How serial analysis of gene expression (SAGE) works

To recap, the original goal of SAGE was to see what proteins are active, which we do by converting mRNA to cDNA with reverse transcriptase. To see what cDNA sequences are in our sample, we want to turn cDNA into easily sequenced short "tags." The steps for doing this are to perform the following:

1. Bind the polyA end of a cDNA strand to biotin.
2. Use the biotin to anchor the strands onto avidin-coated microbeads.
3. Cut the anchored strands lots and lots of places with a RE, leaving sticky ends.
4. Glue a marker for FokI on to the sticky ends with DNA ligase.
5. Chop off the last 20 base pairs of the strand with FokI, leaving you with a fragment of the cDNA that starts after the first RE made its cut.
6. Sequence these random substrings.

Like many of the other techniques we just discussed, SAGE is just a sequence of relatively low-tech techniques like PCR and restriction enzymes strung together in a particular way, and computer scientists probably find it hard to understand why this particular sequence is interesting. (At least, it's interesting enough to be cited thousands of times!) Part of the art of designing biology experiments is understanding how to develop these sorts of plans. Doing this requires understanding many different techniques well—not only their inputs and outputs, but their cost and reliability—and using this understanding to create new, cost-effective sequences to solve a particular problem.

Bioinformatics

Biologists are drowning in data. Increasingly sophisticated techniques for automation and parallelism are generating ever-greater quantities of raw information—much more than they could ever hope to analyze by hand. Almost out of self-defense, the field of bioinformatics has evolved to find ways to deal with the deluge (and other problems in biology) by adapting and inventing computational methods to make sense of increasingly massive datasets.

DNA Sequence Analysis

In 1976, scientists announced that they had completely sequenced the genome of the bacteriophage MS2. Although less than 4000 base pairs in length, it was the first time a genome had been "read" from start to finish.

In 2013, the Human Genome Project announced that, after more than a decade of hard work, they had successfully sequenced and sorted all three billion base pairs found in human DNA.

In 2023, the National Library of Medicine contained more than 78,000 completed genomes, covering everything from individual plasmids to blue whales.[1]

Needless to say, searching through such a huge database is far from simple. Scientists would like to be able to search through this entire library to find

[1] Or, if you want to be pedantic, you could say "from the smallest plasmid to the Japanese canopy plant," which currently holds the world record for longest genome—150 billion base pairs.

W. W. Cohen, C. K. Cohen, *A Computer Scientist's Guide to Cell Biology*,
https://doi.org/10.1007/978-3-031-55907-5_8

items of interest, such as groups of **homologs**—genes from different organisms that have highly similar sequences. Or, in programming terms, they'd like to search the very long string S and find all substrings T that are similar to a "query string" Q. (Where S is the sequence database, Q is the gene of interest, and the T's are homologs.)

Homologs

Two genes from different organisms that are highly similar in sequence are homologous. Homologs from the same organism are called paralogs, and homologs from different organisms are orthologs.

Before you can look at any results, though, you have to define exactly what "similar" means. One simple criterion is the **minimal edit distance** between Q and T—the number of times you would need to perform **edit operations** (i.e., genetic mutations) to transform one sequence into the other. Typically, those operations are as follows:

- **Delete**: removing a single "letter," or base pair, from the sequence.
- **Insert**: adding a single "letter" to the sequence.
- **Substitute**: replacing one "letter" with another.

Levenshtein Distance

Take, for example, the strings "will cohen" and "walt chen." Transforming one into the other only takes three operations—substituting the *i* with an *a*, the second *l* with a *t*, and deleting the *o*. The minimal edit distance between the strings, then, is three; that measure is known as the **Levenshtein distance.**

There are other ways of measuring distance. When aligning amino acid sequences, you usually assign different costs to different substitutions, resulting instead in the sequences' Needleman-Wunch distance.

There's an elegant method for calculating the minimal edit distance between Q and T in time $O(|Q|*|T|)$. The key is taking advantage of the following recursive definition for the minimal distance between the first m letters of Q and the first n letters of T:

$$\text{distance}\ (Q,T,m,n) = \min \begin{cases} \text{distance}\ (Q,T,m-1,n-1) & \%\ \text{if}\ Q_{m-1} = T_{n-1} \\ \text{distance}\ (Q,T,m-1,n)+1 & \%\ \text{insert} \\ \text{distance}\ (Q,T,m,n-1)+1 & \%\ \text{delete} \\ \text{distance}\ (Q,T,m-1,n-1)+1 & \%\ \text{substitute} \end{cases}$$

It's not hard to see why this definition works. The third line, for instance, is the result of recursively finding the minimal edit distance between the first *m - 1* letters of *Q* and the first *n - 1* letters of *T*, and then substituting T_{n-1} for Q_{m-1} at an additional cost of one edit operation.

It's also not hard to see the potential problem with the definition—you would find yourself calling "distance" with the same *m* and *n* arguments over and over again, slowing things to a crawl. The trick is to use a technique called **dynamic programming**. Rather than redoing the calculation every time you reuse the argument, you can create a matrix (linear algebra) to store the result of every distance computation between Q and T and simply fill it in starting with the upper-left corner.[2]

You can see an example of such a computation in Fig. 1. Each entry in the matrix can be found by looking only at the entries above and to the left of it; the final distance between the two strings appears in the bottom-right corner of the matrix. (Which, since we're still looking at "will cohen to walt chen," will be 3).

Smith-Waterman Similarity

Another way of quantifying the relationship between two strings is **Smith-Waterman similarity.** (A distance function assigns low scores to similar strings, and a similarity function assigns high scores to similar strings.) Smith-Waterman is defined by this recursive function:

[2] Alternatively, you could "memo-ize" the function—that is, cache the results for each *m*, *n* pair as they are computed, for example, using Python's functools.cache or it's equivalent.

		w	a	l	t		c	h	e	n
		0	1	2	3	4	5	6	7	8
w		1	1	2	3	4	5	6	7	8
i		2	2	1	2	3	4	5	6	7
l		3	3	2	2	3	4	5	6	7
l		4	4	3	3	2	3	4	5	6
		5	5	4	4	3	2	3	4	5
c		6	6	5	5	4	3	3	4	5
o		7	7	6	6	5	4	3	4	5
h		8	8	7	7	6	5	4	3	4
e		9	9	8	8	7	6	5	4	3
n										

An example of how to compute the Levenshtein distance between two strings. The element on row *i* and column *j* of the matrix stores distance(*Q*, *T*, *i* ,*j*), and the value of the lower right-hand corner entry (i.e., 3) is the distance between the two strings. The shaded entries are those that were used in the computation of the minimal cost (i.e., the cases of the *min* computation that were used to find the final score).

Fig. 1 Computing the Levenshtein edit distance

$$\text{score}\left(Q,T,m,n\right)=\max\begin{cases}\text{score}\left(Q,T,m-1,n-1\right)+2 & \%\,\text{if } Q_{m-1}=T_{n-1}\\ \text{score}\left(Q,T,m-1,n\right)-1 & \%\,\text{insert}\\ \text{score}\left(Q,T,m,n-1\right)-1 & \%\,\text{delete}\\ \text{score}\left(Q,T,m-1,n-1\right)-1 & \%\,\text{substitute}\\ 0 & \%\,\text{restart}\end{cases}$$

The function adds two "points" for every matching base pair, and subtracts one "point" for every insertion, deletion, or substitution. The final value used for score (*Q, T*) is the maximum value for "score" (*Q, T, m, n)* over all *m* and *n*.

The important distinction from Levenshtein distance is that Smith-Waterman scores allow a new option: the score can be "reset" to zero at any point, meaning that high "scores" can reflect a *partial* match between the overall strings.

Figure 2 shows an example of how Smith-Waterman distance works in practice, again using "will cohen" and "walt chen" as the partial match and the sequence "will walt chen come" as a larger string containing it. The shaded areas represent locally maximized scores (i.e., those with no higher-scoring neighbors), and the values that were used in the series of "max" computations that gave you those high scores.

	w	i	l	l		w	a	l	t		c	h	e	n		c	o	m	e
w|	2	1	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0
i|	1	4	3	2	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
l|	0	3	6	5	4	3	2	3	2	1	0	0	0	0	0	0	0	0	0
l|	0	2	5	8	7	6	5	4	3	2	1	0	0	0	0	0	0	0	0
|	0	1	4	7	10	9	8	7	6	5	4	3	2	1	2	1	0	0	0
c|	0	0	3	6	9	9	8	7	6	5	7	6	5	4	3	4	3	2	1
o|	0	0	2	5	8	8	8	7	6	5	6	6	5	4	3	3	6	5	4
h|	0	0	1	4	7	7	7	7	6	5	5	8	7	6	5	4	5	5	4
e|	0	0	0	3	6	6	6	6	6	5	4	7	10	9	8	7	6	5	7
n|	0	0	0	2	5	5	5	5	5	5	4	6	9	12	11	10	9	8	7

Computing the Smith-Waterman similarity between two strings. The largest element of the matrix (i.e., 12) is the similarity. The long grey-shaded area is associated with the score 12, and the substrings "will cohen" and "will walt chen". The other grey-shaded area corresponds to an approximate match of "_cohe" to "_come" (with a score of 7).

Fig. 2 The Smith-Waterman edit distance method

You can see that for the shaded cells corresponding to matching characters, the similarity score will increase by 2 relative to the neighbor above and to the left. If you look at the strings directly above or below the shaded areas, you can identify the strings participating in the partial matches, and if you look for cases where the expected increase of 2 does not occur, you can determine exactly where substitutions and deletions took place.[3]

In this particular example, the highest score is 12, which shows where "will cohen" partially matches "will walt chen." By tracing through score changes in the large gray-shaded area, you can also reconstruct the choices made to get to a score. For example, if you remember that each score is based on neighboring scores to the left and above, and that the only way to get a two-point gain is to match a character, it's clear that the last steps of the algorithm decided to match the "hen" parts of "cohen" and "chen," which ran up the score to 12 from 6. To get to the 6, the only possible move was to go from the 7 above it, and pay the one-point cost for an edit, namely inserting the "o" in "cohen" (or if you prefer, deleting it from "chen.") Before that, there is a long horizontal run where the scores decrease going left to right, which correspond to inserting all the letters in "walt." This pair of partially matching substrings is called an **alignment**.

There is, of course, more than just one possible match leading to any score. Here, for example, it doesn't matter whether you match the blank in "will cohen" with the first or second blank in the other string (the blue-shaded area shows the variant path through the matrix).

[3] Or, at least, where the edits took place during the optimal sequence.

Instead of using scores above and to the left to build on, another choice the algorithm can make is to restart a score at zero. Intuitively, this is like deciding to ignore everything up to this point in each string—so when looking for matches, you don't have to start at the very beginning of each string. We can, for example, skip the first few words entirely and compare "_cohen" and "_come." The first three characters, _co, match perfectly, which we can see in the bottom right of the matrix—the green-shaded cells show another steady score progression. The fourth character is different—"h" verses "m"—resulting in a one-point loss of similarity, which then begins to tick back up with the matching "e."

We can also use a matrix like this to think about how one string "mutated" into another. There are almost always going to be different ways of getting from one string to another. In this particular example, two routes jump out at us:

1. One-point deletion that changes "will cohen" to "will chen," plus long insertion of "walt" (or "walt") in the middle of the sequence.
2. One-point mutation and one deletion that change "will cohen" into "will come," plus a longer insertion to add "walt chen" in the middle.

With Smith-Waterman, route 1 comes out with a higher score, meaning that it "costs" less to change one string to another following that particular order, and that this route is more evolutionarily likely than route 2.

But we're not just computer scientists anymore, and the above result doesn't necessarily line up with what we've just learned about genetics. It's easy to imagine how a point mutation (swapping an "h" for an "m") could happen—DNA polymerase only needs to make one mistake. Such a change is also unlikely to have any downstream effects, most likely resulting in the substitution of one amino acid for another. The resulting mutant protein might not work as well—or at all—but the rest of the cell's functions are unaffected.

A deletion (losing the "o") represents a similarly small change—again, it's just one singular error—but the consequence might be much more severe, depending on what these letters actually mean. Remember that protein structure is dictated by in three-base-long codons, with each codon corresponding to a particular amino acid. Deleting one base means shifting everything that follows up a step, creating totally different codons. Our algorithm should *probably* take that kind of extra cost into account, and count a single deletion of a base pair very unlikely. (Perhaps for this reason, most of the sequence-similarity methods in biology are used on the alphabet of amino acids rather than base pairs.)

Still, these are fairly basic processes. Can we find a way to make better suited to biological sequence data? The answer is "yes," and there are essentially two ways of going about the task. The first way is to inject some prior knowledge by tweaking the costs associated with the different edit operations. For example, some pairs of amino acids are chemically more similar than others: maybe that information could be encoded by specifying lower substitution costs for similar pairs, and higher costs for dissimilar pairs. Maybe inserting a long string (as a virus might do) should be scored as lower in cost than inserting each of its letters one by one. Maybe a different representation would be better as a starting point for designing similarity functions—for instance probabilistic approaches like **hidden Markov models** are widely used. Bioinformaticians have explored all of these methods and more to improve simplistic edit distances for biological tasks.

A point accepted mutation matrix (PAM matrix) is a 20-by-20 matrix that estimates probabilities of pointwise amino acid mutations "sticking": specifically M(i, j) is the probability that if amino acid i is replaced by amino acid j, that mutation will be accepted by the evolutionary process. PAM matrixes are widely used to define substitution costs in a sequence-similarity function based on edit distance.

The second general approach to improving sequence alignment is a little more radical, and it is discussed below.

Multiple Sequence Alignment

For those of us that have studied machine learning, the general rule is that more data is better data. When it comes to DNA, we can't make these strings longer, but we *can* add data another way: instead of comparing just two strings, compare many at once. This leads to a harder problem—one that needs more tricks than just dynamic programming to solve quickly on a computer—but reducing the task to pairwise alignments and doing the most-similar pairs first lead to good heuristic methods.

Figure 3 shows part of a **multiple sequence alignment** (MSA) between eight genes from different organisms that are hypothesized to be related. With eight strings, we can't show the alignment the same way we did above: instead, the colors in the figure (which were manually added here) indicate substrings that likely align with each other, and the dashes indicate deletions. (Notice

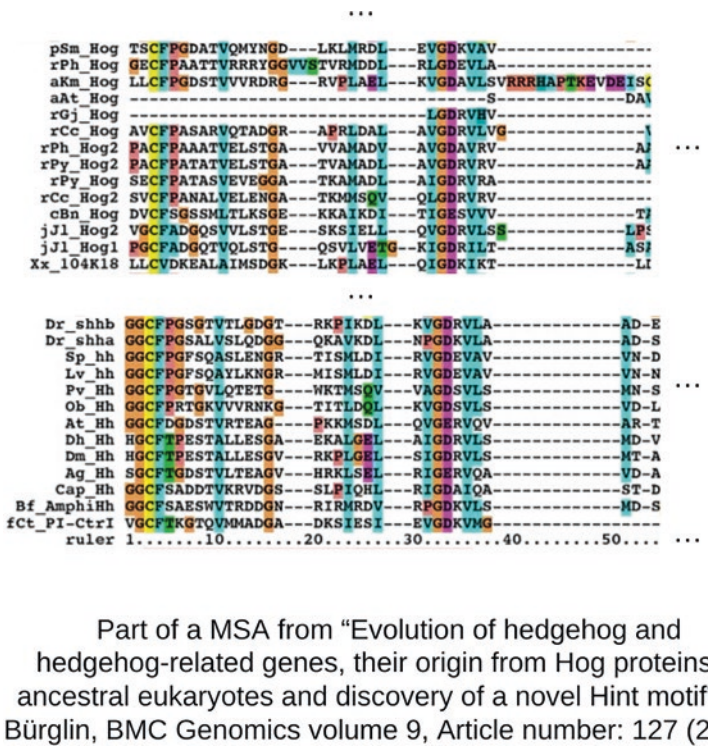


Fig. 3 Part of a multiple sequence alignment

that this alignment method considers it ok to align different letters, like "M" and "O," since it "knows" the amino acids involved are chemically similar.)

Analyzing Similarities of Species

The more similarities there are between two genes, the less **evolutionary distance** exists (or is likely to exist) between those two species. Human hemoglobin, for example, is more similar to mouse hemoglobin than sparrow, and much more similar than shark. Even if that's all you know about the four organisms, you can guess that tree (A) in the figure below is much more likely than (B).

The trees shown in Fig. 4 are more known as phylogenetic trees, and the study of evolutionary history as a whole is called phylogeny.

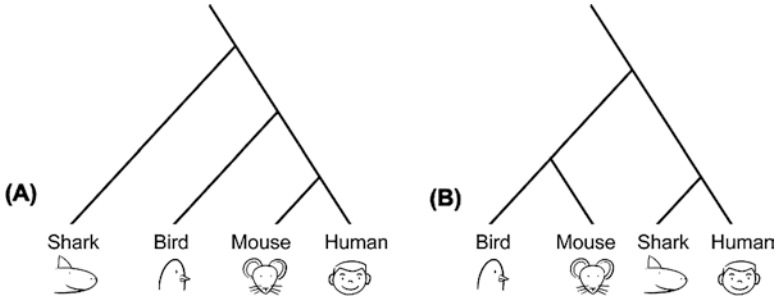


Fig. 4 Two possible evolutionary trees

Biologists and computer scientists are still working on ways to formalize that guess, and how to use said formalization to search for the most likely evolutionary tree.

Molecular Clocks

It's often possible to determine the rate at which proteins—and, thus, the underlying genome—change over time by comparing evolutionary trees to the fossil record. Doing so gives you a good idea of where other species would have diverged, even if they don't leave behind many fossils.

The key to estimating evolutionary rates is picking the right sort of protein (your **molecular clock**) to examine for changes. Quickly mutating genes are well suited to fast evolutionary processes; genes that tend to be conserved between species are better for following slow evolutions.

Data Mining DNA Sequence Databases

Another common bioinformatic experiment is to go "data mining" in some of these huge databases of genetic and biochemical information, looking for anomalies and regularities.

Take, for example, protein composition. Many of the largest examples are "modular," for lack of a better term, assembled from multiple subunits (remember the two-hybrid assays from the last chapter?). The genetic code for these **domains**, as the subunits are called, often appears in multiple different genes; the domains themselves often appear as subunits in many different protein complexes.

Think of domains as blocks of Lego—small pieces that can be combined in many ways to create much more complicated structures. Some of them are so

simple and straightforward that they're used in hundreds of different proteins; others are complex and entirely unique.

If you have a bunch of completed genomes, you can computationally "mine" them to find regularities, and hopefully to uncover repeated domains, domains that are the same in unrelated species, and other such regularities.

Mining Proteins

The same techniques can also be used to look at structures smaller than domains—protein motifs. Motifs are the Lego blocks that make up domains; each one represents a basic "shape" of amino acids, such as a loop or bend.

A crucial part of such research is deciding how to *represent* a discovered domain. A string is not appropriate, because often a protein domain isn't always made up of exactly the same amino acids—sometimes you can switch around one or two amino acids without affecting the protein structure. One popular solution is to borrow a page from machine learning and define the domain probabilistically, via a probability distribution over the amino acids that can appear at each position (e.g., "the first amino acid is X 80% of the time, Y 15% of the time, and Z 5% of the time; the second amino acid is M 98% of the time and N 2% of the time," and so on and so on.) Another popular representation for biological sequences that can vary slightly is a hidden Markov model (aka HMM).

High-Throughput Experiments

Probabilities and statistics can also be vital to understanding the results of high-throughput experiments.

A single microarray, for example, might produce tens of thousands of separate data points, each one summarizing the expression level of a particular gene under a particular condition. You can be fairly confident that changing conditions will change gene expression, but it can be difficult to pick such changes out of the "noise" of a data set, and to tell whether or not they're actually caused by a biological feature and not random chance.

Using computational methods to sort through the data not only reduces the cognitive load on unfortunate biologists who went into the field to avoid dealing with complex math; it's also a lot faster and more accurate than trying to do the same thing by eye.

Search Engines

The "scientific literature" is a vast collection of data. Millions of scientific articles have already been published in various journals, and the rate at which new ones are being added to the pile is getting faster and faster. If you run into a problem or unexpected result during an experiment, there's a very real chance that someone else has already worked on it—if you can only find their paper. An important subfield of bioinformatics revolves around building tools to help biologists search and monitor the literature.

Many projects seek to distill information from multiple sources into a single readily accessible format—think of a database of known protein-protein interactions. But such **curated** databases require constant human input, slowing their pace and driving up the cost. Bringing natural language processing and machine learning techniques to bear and at least partially automating the process would make such databases much easier (and cheaper) to maintain.

Where to Go From Here?

What Now?

Even after finishing this book, you can't really say you "know" biology. It's a massive field with unbelievable depth; following any single thread can take you down a rabbit hole of increasingly complicated subtopics. But hopefully you'll have built up a solid-enough base to understand what's going on in experiments and work your way through research papers without getting completely lost.

If you're interested in learning more, there's a vast range of options, we'd like to suggest two places to start.

- If you want something approachable, check out *The Cartoon Guide to Genetics*,[1] by Larry Gonick and Mark Wheelis, and their 2019 follow-up book, *The Cartoon Guide to Biology*.
- If you want something ridiculously comprehensive, try *Molecular Biology of the Cell*.[2]

[1] *The Cartoon Guide to Genetics* (1991) by Larry Gonick and Mark Wheelis. Published by HarperCollins.

[2] *Molecular Biology of the Cell* (2002) by Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. Published by Garland Publishing, a member of the Taylor & Francis Group.

W. W. Cohen, C. K. Cohen, *A Computer Scientist's Guide to Cell Biology*,
https://doi.org/10.1007/978-3-031-55907-5

Sources

In preparing the first edition of the book, the textbooks and/or web sites below were used as references. For the second edition, we've updated our descriptions and added a few others.

Biochemistry, Mary K. Campbell, and Shawn O. Farrell. Published by Thomson-Brooks/Cole. A good introductory textbook on biochemistry. In preparing this book, I used the 2002 edition, but it has been updated many times since: the most recent edition is the Ninth Edition, from 2017.

Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids (1998), by R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. Published by Cambridge University Press. This classic text is an excellent introduction to the many aspects of sequence modeling, including hidden Markov models, edit distances, multiple alignment, and phylogenetic trees, this text has uses beyond biology as well.

An Introduction to the Genetics and Molecular Biology of the Yeast Saccharomyces cerevisiae (1998), by Fred Sherman. Another classic text, which was widely available online when the first edition was prepared (and still can be found easily). It is a version of: F. Sherman, *Yeast Genetics*, in *The Encyclopedia of Molecular Biology and Molecular Medicine*, pp. 302–325, Vol. 6. Edited by R. A. Meyers, VCH Pub., Weinheim, Germany, 1997. A text focusing on yeast biology has also been released since the first edition was published: *Yeast: Molecular and Cell Biology*, 2nd Edition, by Horst Feldmann, 2012.

Molecular Biology, by Robert F. Weaver. Published by McGraw-Hill. This book contains many in-depth discussions of the research, results, and reasoning processes behind our understanding of biology, illustrated by detailed analysis of specific

W. W. Cohen, C. K. Cohen, *A Computer Scientist's Guide to Cell Biology*,
https://doi.org/10.1007/978-3-031-55907-5

research papers. It is a good resource for those wanting to obtain a "reading knowledge" of biology—that is, for those that want to be able to read and understand recent publications in biology. In preparing the first edition of this book, I used the Third Edition (2005); the most recent version is the Fifth Edition, from 2025.

Random Walks in Biology (1983), by Howard Berg. Published by Princeton University Press. This is a short book with some very accessible discussions of diffusion in biological systems. An expanded version from 1993 is also available.

Transport Phenomena in Biological Systems (2004), by George Truskey, Fan Yuan, and David Katz. Published by Pearson Prentice Hall. An in-depth treatment of transport and diffusion.

Biological Physics: Energy, Information, Life (2003), by Philip Nelson. Published by W.H. Freeman. A beautiful and very readable treatment of the mathematics behind a number of biologically important processes, including diffusion, energy transfer, self-assembly, and "molecular machines."

Index

W. W. Cohen, C. K. Cohen, *A Computer Scientist's Guide to Cell Biology*,
https://doi.org/10.1007/978-3-031-55907-5