Lecture 3- Strong Induction

Strong Induction

Let P(n) be any predicate.If P(0) is true and $\forall n(P(0) \wedge P(1) \wedge ... \wedge P(n)) \implies P(n+1)$ is true,
then every P(n) is true.

Hint

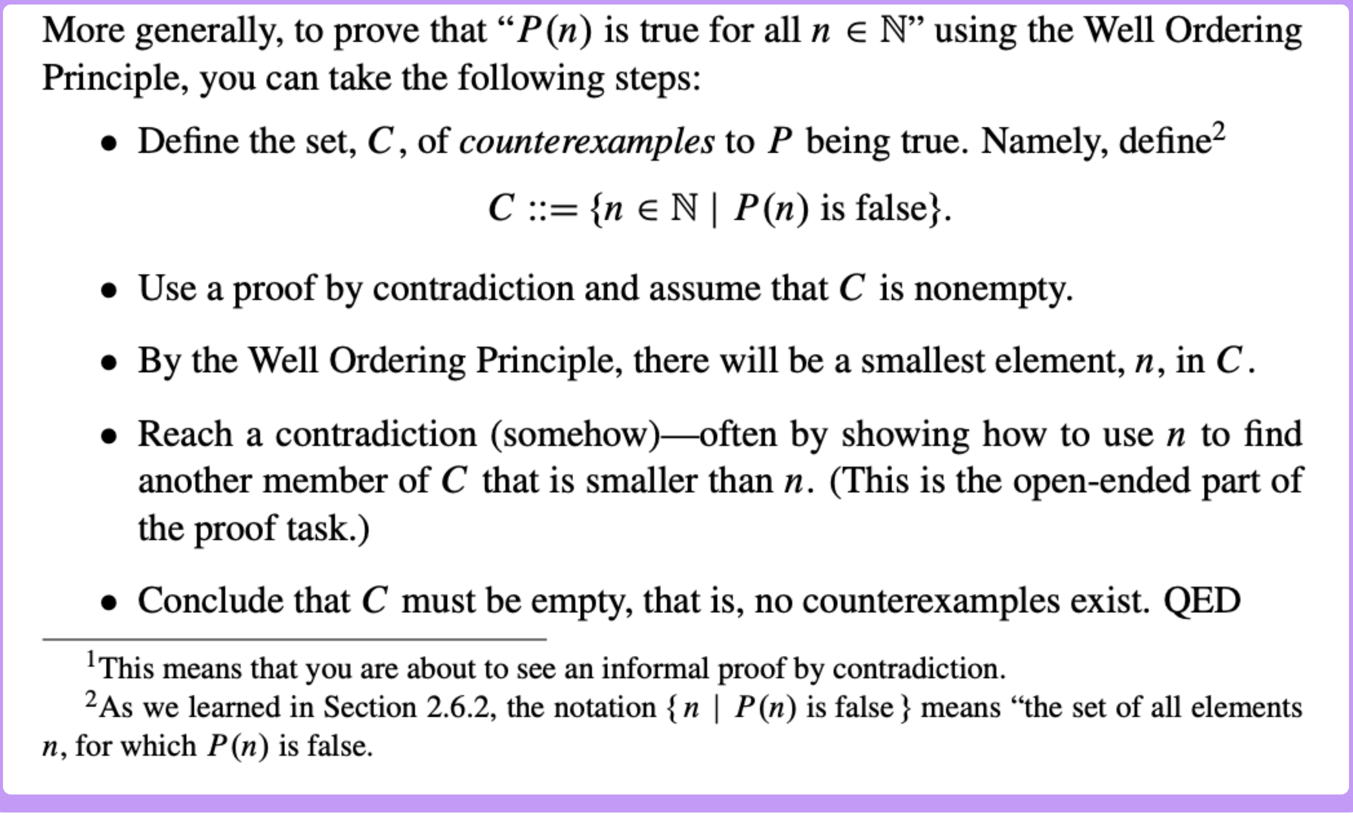
Any proof can be done by strong induction can be done by ordinary induction.But strong induction make it easier to prove

The Well Ordering Principle

Every nonempty set of nonnegative integers has a smallest element.

Template for Well Ordering Proofs

Template



More generally, to prove that "$P(n)$ is true for all $n \in \mathbb{N}$" using the Well Ordering Principle, you can take the following steps:

- Define the set, C, of *counterexamples* to P being true. Namely, define[2]

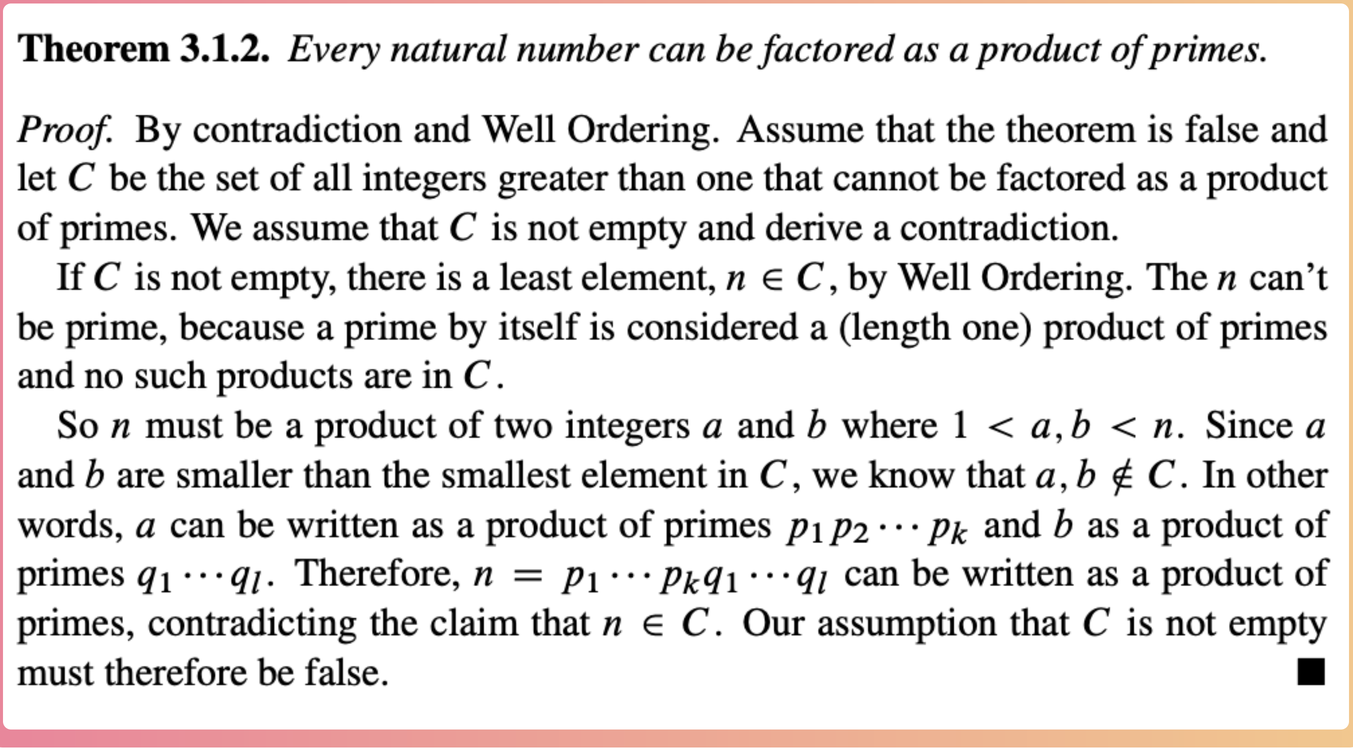
$$C ::= \{n \in \mathbb{N} \mid P(n) \text{ is false}\}.$$

- Use a proof by contradiction and assume that C is nonempty.
- By the Well Ordering Principle, there will be a smallest element, n, in C.
- Reach a contradiction (somehow)—often by showing how to use n to find another member of C that is smaller than n. (This is the open-ended part of the proof task.)
- Conclude that C must be empty, that is, no counterexamples exist. QED

[1]This means that you are about to see an informal proof by contradiction.

[2]As we learned in Section 2.6.2, the notation $\{n \mid P(n) \text{ is false}\}$ means "the set of all elements n, for which $P(n)$ is false.

Example Proof



Theorem 3.1.2. *Every natural number can be factored as a product of primes.*

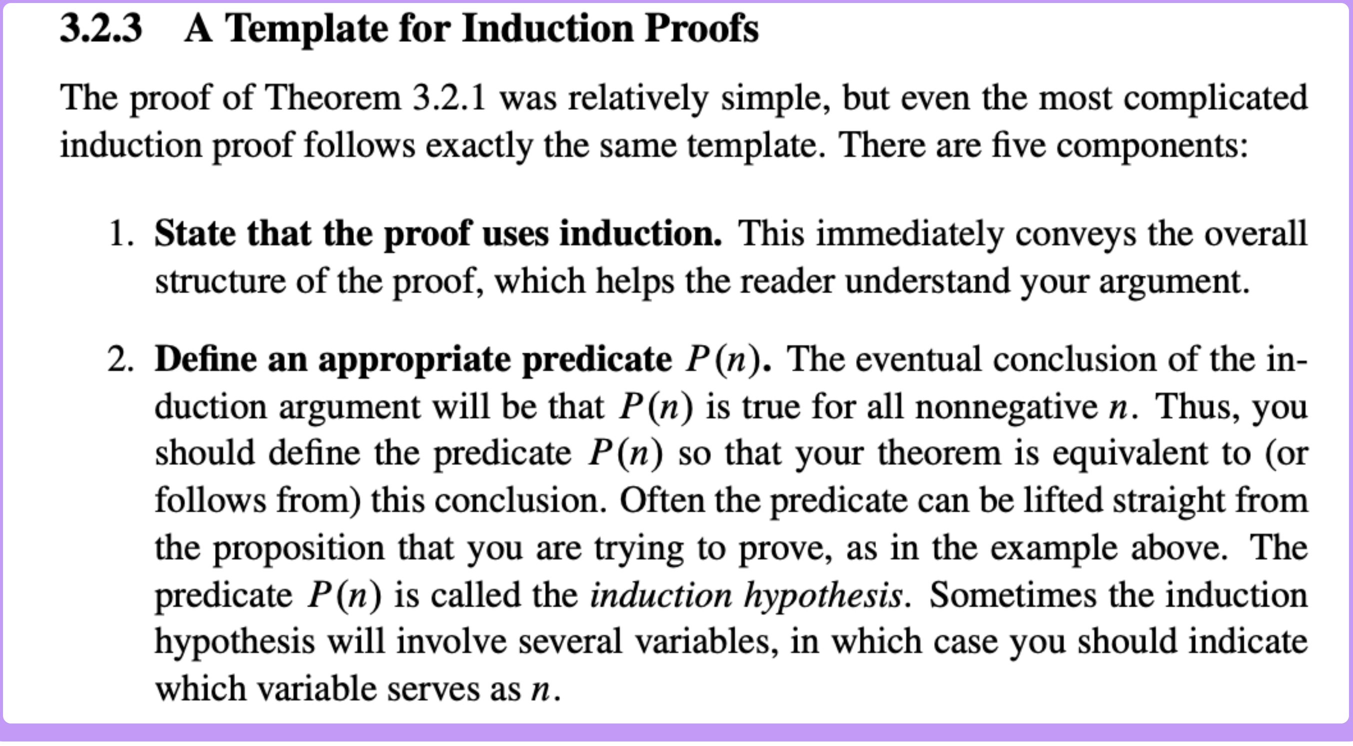
Proof. By contradiction and Well Ordering. Assume that the theorem is false and let C be the set of all integers greater than one that cannot be factored as a product of primes. We assume that C is not empty and derive a contradiction.

If C is not empty, there is a least element, $n \in C$, by Well Ordering. The n can't be prime, because a prime by itself is considered a (length one) product of primes and no such products are in C.

So n must be a product of two integers a and b where $1 < a, b < n$. Since a and b are smaller than the smallest element in C, we know that $a, b \notin C$. In other words, a can be written as a product of primes $p_1 p_2 \cdots p_k$ and b as a product of primes $q_1 \cdots q_l$. Therefore, $n = p_1 \cdots p_k q_1 \cdots q_l$ can be written as a product of primes, contradicting the claim that $n \in C$. Our assumption that C is not empty must therefore be false. ■

Template for Ordinary Induction Proofs

Template



3.2.3 A Template for Induction Proofs

The proof of Theorem 3.2.1 was relatively simple, but even the most complicated induction proof follows exactly the same template. There are five components:

1. **State that the proof uses induction.** This immediately conveys the overall structure of the proof, which helps the reader understand your argument.

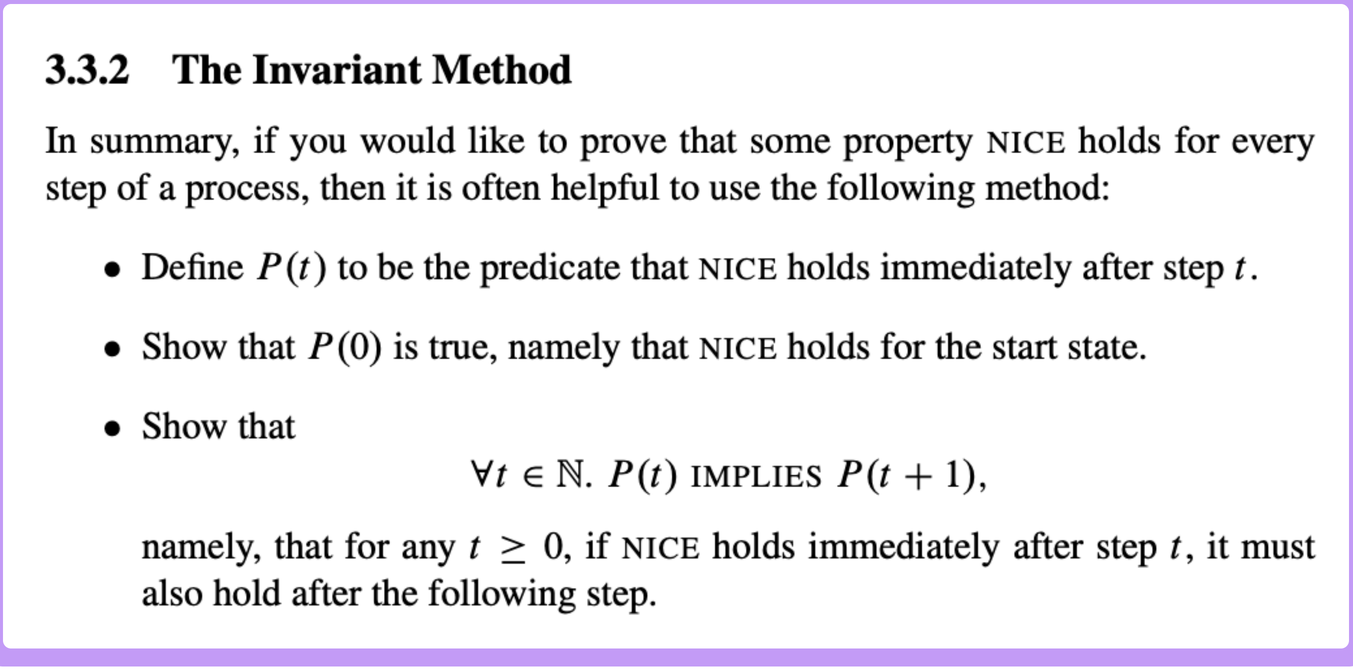
2. **Define an appropriate predicate $P(n)$.** The eventual conclusion of the induction argument will be that $P(n)$ is true for all nonnegative n. Thus, you should define the predicate $P(n)$ so that your theorem is equivalent to (or follows from) this conclusion. Often the predicate can be lifted straight from the proposition that you are trying to prove, as in the example above. The predicate $P(n)$ is called the *induction hypothesis*. Sometimes the induction hypothesis will involve several variables, in which case you should indicate which variable serves as n.

Invariant

One of the most important uses of induction in computer science involves proving that a program or process preserves one or more desirable properties as it proceeds.

A property that is preserved through a series of operations or steps is known as an invariant.

Template



3.3.2 The Invariant Method

In summary, if you would like to prove that some property NICE holds for every step of a process, then it is often helpful to use the following method:

- Define $P(t)$ to be the predicate that NICE holds immediately after step t.
- Show that $P(0)$ is true, namely that NICE holds for the start state.
- Show that
$$\forall t \in \mathbb{N}.\ P(t) \text{ IMPLIES } P(t+1),$$
namely, that for any $t \geq 0$, if NICE holds immediately after step t, it must also hold after the following step.

Structural Induction

Recursive Data Types

```
Recursive data types are specified by recursive definitions.
```

Recursive definitions:

- Base case(s) that dont depend on anything.
- Constructor case(s) that depend on previos cases.

Structural Induction on Recursive Data Types

- Prove P for the base cases of the definition.
- Prove P for the constructor cases of the definition, assuming that is true for the component data items.

Lecture 6 - Graph Theory and Coloring