CS 145 – Foundations of Computer Science

Professor Eric Aaron

Lecture – T R 3:10pm
Lab – M 3:10pm

Lecture Meeting Location: OH 162
Lab Meeting Location: SP 309

Business

• HW4 available from course website, due Apr. 4 / Apr. 5 (as always, see assignment sheet for exact deadlines)

• Reading: Ch.4.1-4.6
  – Our coverage of the material will be different from that in the textbook, but it’s good to see the textbook’s presentation, as well

• Reading: Prof. Hunsberger’s document “The Natural Numbers, Induction, and Numeric Recursion”
  – Posted on the Additional Notes / Readings page of the CS145 website

• Oh, and in case anyone forgot: Exam in class Thursday
  – Recall, Coaches have no information about the exam—please direct any questions about the exam to me!
A General Note about Proving Stuff

• As we’ve discussed in class, proofs are important, and understanding how to structure a proof is important

**Generally Good Idea:** When proving a claim about something, let the definition of that thing guide your proof.

• Example: Proof about transitivity of a relation R.
  – The definition of transitivity is of the form: For all a,b,c, if (a,b) in R and (b,c) in R then (a,c) in R
  – So, the proof might begin with considering arbitrarily chosen a,b,c (because of the *For all* in the definition)
  – Then, the proof might proceed by assuming (a,b) in R and (b,c) in R (because of the *if—then* in the definition) …

A General Note about Proving Stuff

• As we’ve discussed in class, proofs are important, and understanding how to structure a proof is important

**Generally Good Idea:** When proving a claim about something, let the definition of that thing guide your proof.

• Example: Proof about equality of sets S=T.
  – The definition of set equality is of the form: S=T iff For all x, x ∈ S exactly when x ∈ T
  – One way of understanding this meaning is that S = T iff S ⊆ T and T ⊆ S
  – So, the proof might begin with a proof of S ⊆ T
  – Then, the proof might proceed with a proof of T ⊆ S …
A General Note about Proving Stuff

• As we’ve discussed in class, proofs are important, and understanding how to structure a proof is important

**Generally Good Idea: When proving a claim about something, let the definition of that thing guide your proof.**

• Example: Proving something of the form $P \iff Q$
  – One way to understand the meaning of $\iff$ is that $P \iff Q$ is true exactly when both $if \ P \ then \ Q \ is \ true$ and $if \ Q \ then \ P \ is \ true$
  – So, the proof might begin with a proof of $if \ P \ then \ Q$
  – Then, the proof might proceed with a proof of $if \ Q \ then \ P \ ...$

A General Note about Proving Stuff

• As we’ve discussed in class, proofs are important, and understanding how to structure a proof is important

**Generally Good Idea: When proving a claim about something, let the definition of that thing guide your proof.**

• For anything defined by recursion, the preferred proof technique is typically *induction*.
• This is because of the same Generally Good Idea that guides our proofs of transitivity, set equality, etc.—follow the relevant definitions!
  – (But we’ve proved things about numbers by induction… hm….)

**Important Observation:** When we prove something by induction, we are typically being guided to do so by an underlying recursive definition.
Recursive (or Inductive) Definition of a Set

- A recursive definition of a set $S$ consists of three components:
  - **Base**: One or more foundational elements of $S$
  - **Induction**: One or more rules to construct new elements of $S$ from existing elements of $S$
  - **Closure**: The condition that $S$ consists of all and exactly the elements derived from the base elements and induction rules.
    (In the context of a definition, this is often assumed rather than explicitly stated—the fact that it is a definition means that the set is exactly the elements thus specified.)

- What are some recursively defined sets we’ve already seen, and what are their definitions?

A Recursively Defined Set: Propositional Logic Expressions

- Recall our expressions for propositional logic: **propositional letters** (i.e., variables), negations, conjunctions, etc.
- A recursive definition of the set of all propositional logic expressions:
  - **Base**: Given an initial set $A$ of propositional letters (e.g., $p$, $q$, $r$, …), all elements of $A$ are propositional logic expressions
  - **Induction**: If $P$, $Q$ are propositional logic expressions, then the following are also propositional logic expressions (note that the parentheses are part of the expressions)
    * ($\neg P$)
    * ($P \land Q$)
    * ($P \lor Q$)
    * ($P \rightarrow Q$)
    * ($P \leftrightarrow Q$)
  - (implicit, when unstated) **Closure**: The set of propositional logic expressions is all and only these expressions

Two Very Important Questions:
1. Is this a good recursive definition for the set of propositional logic expressions?
2. What are the trade-offs for such a definition?
A Recursively Defined Set: Propositional Logic Expressions

- Recall our expressions for propositional logic: *propositional letters* (i.e., variables), negations, conjunctions, etc.
- A recursive definition of the set of all propositional logic expressions:
  - **Base:** Given an initial set \( A \) of propositional letters (e.g., \( p, q, r, \ldots \)), all elements of \( A \) are propositional logic expressions
  - **Induction:** If \( P, Q \) are propositional logic expressions, then the following are also propositional logic expressions (note that the parentheses are part of the expressions)
    - \( (\neg P) \)
    - \( (P \land Q) \)
    - \( (P \lor Q) \)
  - (implicit, when unstated) **Closure:** The set of propositional logic expressions is all and only these expressions

Two Very Important Questions:

1. Is this a *better* recursive definition for the set of propositional logic expressions? (Is it equivalent to the previous one?)
2. What are the trade-offs for such a definition?

A Recursively Defined Set: The Natural Numbers

- As suggested by all of our inductive proofs about numbers, there is also a recursive definition of the natural numbers
- The *Peano axioms* are conventionally taken as a definition of the natural numbers (here, let \( N \) stand for the natural numbers):
  1. There exists a number 0 s.t. 0 ∈ \( N \)
  2. Every natural number \( n \) has a natural number successor, denoted by \( S(n) \)
  3. There is no \( n \) in \( N \) s.t. \( S(n) = 0 \)
  4. Distinct natural numbers have distinct successors: if \( a \neq b \), then \( S(a) \neq S(b) \)
  5. Let \( P \) be a property of the natural numbers such that:
    - \( P(0) \) holds
    - For every \( a \) in \( N \), if \( P(a) \) holds, then \( P(S(a)) \) holds
  
  If both of those conditions are true, then \( P(n) \) holds for all \( n \) in \( N \).

Axioms 1 and 2 give the recursive construction of the elements of \( N \). The other Axioms are properties of \( N \). Axiom 5 is sometimes called the Axiom of Induction.
Peano Examples

(Like, etudes?)

• The Peano axioms:
  1. There exists a number 0 s.t. 0 ∈ N
  2. Every natural number has a natural number successor, denoted by S(n)
  3. There is no n in N s.t. S(n) = 0
  4. Distinct natural numbers have distinct successors: if a ≠ b, then S(a) ≠ S(b)
  5. Let P be a property of the natural numbers such that:
     • P(0) holds
     • For every a in N, if P(a) holds, then P(S(a)) holds
   If both of those conditions are true, then P(n) holds for all n in N.

• Exercises and examples:
  – How would we write the number 2 in this notation? The number 5?
  – Could we write the number -1 in this notation?