MA/CSSE 474

Theory of Computation

Decision Problems

Quiz questions referenced here are on the Day 3 quiz

Total Orders

A ***total order*** $R \subseteq A \times A$ is a partial order that has the additional property that:

$$\forall x, y \in A\ ((x, y) \in R \vee (y, x) \in R).$$

Example: $\leq$ on the rational numbers

If R is a total order defined on a set A, then the pair (A, R) is a ***totally ordered set***.

Q7-8

Infinite Descending Chain

- A partially ordered set (S, <) has an infinite descending chain if there is an infinite set of elements $x_0, x_1, x_2, \ldots \in S$ such that $\forall i \in \mathbb{N}(x_{i+1} < x_i)$
- Example:
  In the rational numbers with <,
  $1/2 > 1/3 > 1/4 > 1/5 > 1/6 > \ldots$
  is an infinite descending chain

Well-Founded and Well-Ordered Sets

Given a partially ordered set (*A*, *R*), an ***infinite descending chain*** is a totally ordered, with respect to *R*, subset *B* of *A* that has no minimal element.

If (*A*, *R*) contains no infinite descending chains then it is called a ***well-founded set***.

- Used for halting proofs.

If (*A*, *R*) is a well-founded set and *R* is a total order, then (*A*, *R*) is called a ***well-ordered set***.

- Used in induction proofs.
- The positive integers are well-ordered
- The positive rational numbers are not well-ordered (with respect to normal <)

Q9

Mathematical Induction

Because the integers ≥ *b are well-ordered:*

The ***principle of mathematical induction***:

If: $P(b)$ is true for some integer base case b, and
For all integers $n \geq b$, $P(n) \rightarrow P(n+1)$

Then: For all integers $n \geq b$, $P(n)$

An induction proof has three parts:

1. A clear statement of the assertion P.
2. A proof that that P holds for some base case b, the smallest value with which we are concerned.
3. A proof that, for all integers $n \geq b$, if $P(n)$ then it is also true that $P(n+1)$. We'll call the claim $P(n)$ the ***induction hypothesis***.

Sum of First *n* Positive Odd Integers

The sum of the first n odd positive integers is n^2. We first check for plausibility:

$(n = 1)$ $1 = 1 = 1^2$.
$(n = 2)$ $1 + 3 = 4 = 2^2$.
$(n = 3)$ $1 + 3 + 5 = 9 = 3^2$.
$(n = 4)$ $1 + 3 + 5 + 7 = 16 = 4^2$, and so forth.

The claim appears to be true, so we should prove it.

Sum of First *n* Positive Odd Integers

Let $Odd_i = 2(i - 1) + 1$ denote the ith odd positive integer. Then we can rewrite the claim as:

$$\forall n \geq 1 \quad (\sum_{i=1}^{n} Odd_i = n^2)$$

For reference; we will not do this in class

The proof of the claim is by induction on *n*:
Base case: take 1 as the base case. $1 = 1^2$.

Prove: $\forall n \geq 1((\sum_{i=1}^{n} Odd_i = n^2) \rightarrow (\sum_{i=1}^{n+1} Odd_i = (n+1)^2))$

$$\sum_{i=1}^{n+1} Odd_i = \sum_{i=1}^{n} Odd_i + Odd_{n+1}$$

$= n^2 + Odd_{n+1}$. (Induction hypothesis.)
$= n^2 + 2n + 1$. ($Odd_{n+1} = 2(n+1-1) + 1 = 2n + 1$.)
$= (n + 1)^2$.

Note that we start with one side of the equation we are trying to prove, and transform to get the other side. We do **not** treat it like solving an equation, where we transform both sides in the same way.

Strong induction

- To prove that predicate P(n) is true for all n≥b:
  - Show that P(b) is true [and perhaps P(b+1) *]
  - Show that for all j>b, if P(k) is true for all k with b≤ k<j, then P(j) is true. In symbols:

$$\forall j > b\ ((\forall k\ (b \leq k < j \rightarrow P(k)) \rightarrow P(j))$$

* We may have to show it directly for more than one or two values, but there should always be a finite number of base cases.

Fibonacci Running Time

- From Weiss, Data Structures and Problem Solving with Java, Section 7.3.4
- Consider this function to recursively calculate Fibonacci numbers:
  $F_0=0$ $F_1=1$ $F_n = F_{n-1}+F_{n-2}$ if $n \geq 2$.
  -
```
def fib(n):
    if n <= 1:
        return n
    return fib(n-1) + fib(n-2)
```
- Let C_N be the number of calls to fib during the computation of fib(N).
- It's easy to see that $C_0=C_1=1$, and if $N \geq 2$, $C_N = C_{N-1} + C_{N-2} + 1$.
- **Prove that** for $N \geq 3$, $C_N = F_{N+2} + F_{N-1} -1$.

Q10

Languages and Problems:

The Big Picture

Chapter 3

Decision Problems

A ***decision problem*** is simply a problem for which the answer is yes or no (True or False). A ***decision procedure*** answers a decision problem.

Examples:

- Given an integer *n*, does *n* have a pair of consecutive integers as factors?

- The language recognition problem: Given a language *L* and a string *w*, is *w* in *L*?

↑

Our focus in this course

The Power of Encoding

Anything can be encoded as a string.
For example, on a computer everything is encoded as a string of bits.

<*X*> is the string encoding of *X*.
<*X*, *Y*> is the string encoding of the pair *X*, *Y*.

Problems that don't look like decision problems about strings and languages can be recast into new problems that do look like that.

Web Pattern Matching

Pattern matching on the web:

- Problem: Given a search string *w* and a web document *d*, do they "match"? In other words, should a search engine, on input *w*, consider returning *d*?

- The language to be decided:

{<*w*, *d*> : *d* is a candidate match for the string *w*}

The Halting Problem

Does a program always halt?

- Problem: Given a program *p*, written in some some standard programming language L, is *p* guaranteed to halt, no matter what input it is given?

- The language to be decided:

$$HP_{ALL} = \{p \in L : p \text{ halts on all inputs}\}$$

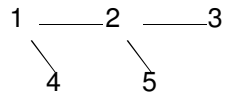
Primality Testing

- Problem: Given a nonnegative integer n, is it prime?
- An instance of the problem: Is 9 prime?
- To encode the problem we need a way to encode each instance: We encode each nonnegative integer as a binary string.
- The language to be decided:

  PRIMES = $\{w : w$ is the binary encoding of a prime integer$\}$.

Graph Connectivity

- Problem: Given an undirected graph G, is it connected?
- Instance of the problem:



- Encoding of the problem: Let V be a set of binary numbers, one for each vertex in G. Then we construct $\langle G \rangle$ as follows:
  - Write $|V|$ as a binary number,
  - Write a list of edges,
    Each pair of binary numbers represents one edge.
  - Separate all such binary numbers by "/".

    1/10/1/100/10/101/10/11
- The language to be decided: CONNECTED = $\{w \in \{0, 1, /\}^* : w = n_1/n_2/\ldots n_i$, where each n_i is a binary string and w encodes a connected graph, as described above$\}$.

Protein Sequence Allignment

- Problem: Given a protein fragment *f* and a complete protein molecule *p*, could *f* be a fragment from *p*?
- Encoding of the problem: Represent each protein molecule or fragment as a sequence of amino acid residues. Assign a letter to each of the 20 possible amino acids. So a protein fragment might be represented as `AGHTYWDNR`.
- The language to be decided:

{<*f*, *p*> : *f* could be a fragment from *p*}.

Turning Problems Into Decision Problems

Casting multiplication as decision:

- Problem: Given two nonnegative integers, compute their product.
- Encoding of the problem: Transform computing into verification.
- The language to be decided:

L = {*w* of the form:
<$integer_1$>x<$integer_2$>=<$integer_3$>, where:
<$integer_n$> is any well formed integer, and
$integer_3 = integer_1 * integer_2$}

`12x9=108` $\in L$
`12=12` $\notin L$
`12x8=108` $\notin L$

Turning Problems Into Decision Problems

Casting sorting as decision:

- Problem: Given a list of integers, sort it.
- Encoding of the problem: Transform the sorting problem into one of examining a pair of lists.
- The language to be decided:

$L = \{w_1 \# w_2 : \exists n \geq 1$
(w_1 is of the form $<int_1, int_2, \ldots int_n>$,
w_2 is of the form $<int_1, int_2, \ldots int_n>$, and
w_2 contains the same objects as w_1 and
w_2 is sorted)$\}$

Examples:

`<1,5,3,9,6>#<1,3,5,6,9>` $\in L$
`<1,5,3,9,6>#<1,2,3,4,5,6,7>` $\notin L$

Turning Problems Into Decision Problems

Casting database querying as decision:

- Problem: Given a database and a query, execute the query.
- Encoding of the problem: Transform the query execution problem into evaluating a reply for correctness.
- The language to be decided:

$L = \{d \# q \# a$:
d is an encoding of a database,
q is a string representing a query, and
a is the correct result of applying q to $d\}$

Example:

```
(name, age, phone), (John, 23, 567-1234)
(Mary, 24, 234-9876)#(select name age=23)#
(John)
```
$\in L$

The Traditional Problems and their Language Formulations are Equivalent

By equivalent we mean that either problem can be ***reduced to*** the other.

If we have a machine to solve one, we can use it to build a machine to do the other, using only the starting machine and other functions that can be built using machines of equal or lesser power.

Reduction does not always preserve efficiency!

An Example

Consider the multiplication example:

$L = \{w$ of the form:

$<integer_1> \times <integer_2> = <integer_3>$, where:

$<integer_n>$ is any well formed integer, and

$integer_3 = integer_1 * integer_2\}$

Given a multiplication machine, we can build the language recognition machine:

Given the language recognition machine, we can build a multiplication machine:

Student Solution Presentations

(as many as we have time for)

- A4
- Consider the English sentence, "If some bakery sells stale bread and some hotel sells flat soda, then the only thing everyone likes is tea." This sentence has at least two meanings. Write two (logically different) first-order-logic sentences that correspond to meanings that could be assigned to this sentence.
- Use the following predicates:
  $P(x)$ is *True* iff x is a person;
  $B(x)$ is *True* iff x is a bakery;
  $S_B(x)$ is *True* iff x sells stale bread;
  $H(x)$ is *True* iff x is a hotel; $S_S(x)$ is *True* iff x sells flat soda;
  $L(x, y)$ is *True* iff x likes y; and
  $T(x)$ is *True* iff x is tea.

Student Solution Presentations

(as many as we have time for)

- A11(c)
- Using the definition of $\equiv_p$ (equivalence modulo p) that is given in Example A.4, let R_p be a binary relation on $\mathbb{N}$, defined as follows, for any $p \geq 1$:

$$R_p = \{(a, b): a \equiv_p b\}$$

  So, for example, R_3 contains (0, 0), (0, 3), (6, 9), (1, 4), etc., but does not contain (0, 1), (3, 4), etc.
- Is R_p a partial order? A total order? Prove your answer.

Student Solution Presentations

(as many as we have time for)

- 2.5a
- Consider the language L of all strings drawn from the alphabet {a, b} with at least two different substrings of length 2.
  - Describe L by writing a sentence of the form $L = \{w \in \Sigma^* : P(w)\}$, where Σ is a set of symbols and P is a first-order logic formula.
  - You may use the function $|s|$ to return the length of s.
  - You may use all the standard relational symbols (e.g., $=$, $\neq$, $<$, etc.), plus the predicate
    - *Substr*(s, t), which is *True* iff s is a substring of t.

Student Solution Presentations

(as many as we have time for)

- 2.8
- For each of the following statements, state whether it is *True* or *False*. Prove your answer.
- (a) $\forall L_1, L_2\ (L_1 = L_2 \text{ iff } L_1^* = L_2^*)$.
- (c)Every infinite language is the complement of a finite language.
- (g) $\forall L_1, L_2\ ((L_1 \cup L_2)^* = L_1^* \cup L_2^*)$.
- (l) $\forall L\ (\varnothing \cup L^+ = L^*)$.