# Recitation 11 - Homework 5

John Chilton

June 8, 2007

- Homework 5 Problems
- Related Examples

Problem 1 (Exercise 4.6). Let $\mathcal{B}$ be the set of all infinite sequences of 0s and 1s. Show $\mathcal{B}$ is not countable using a proof by diagnolization.

As an example of the method, consider the similar problem of $A = [0, 1) \in \mathcal{R}$. Assume $A$ is countable and let $f$ be some one-to-one mapping from $A$ to $\mathcal{N}$.

| $n$ | $f(n)$ |
|---|---|
| 1 | 0.123010201020302132 84... |
| 2 | 0.314159265833432342 21... |
| 3 | 0.723139342732348232 93... |
| 4 | 0.239342934239323234 92... |
| 5 | 0.120932394234293423 42... |
| 6 | 0.193802348029348203 12... |
| ⋮ | ⋮ |

As an example of the method, consider the similar problem of $A = [0, 1) \in \mathcal{R}$. Assume $A$ is countable and let $f$ be some one-to-one mapping from $A$ to $\mathcal{N}$.

| $n$ | $f(n)$ |
|---|---|
| 1 | 0.123010201020302132 84... |
| 2 | 0.314159265833432342 21... |
| 3 | 0.723139342732348232 93... |
| 4 | 0.239342934239323234 92... |
| 5 | 0.120932394234293423 42... |
| 6 | 0.193802348029348203 12... |
| ⋮ | ⋮ |

Since this is a one-to-one mapping each number in $[0, 1)$ should appear on the right. We derive a contradiction by showing there is a real number on $[0, 1)$ that doesn't appear on the right.

| $n$ | $f(n)$ |
|---|---|
| 1 | 0.$\underline{\mathbf{1}}$2301020102030213284... |
| 2 | 0.3$\underline{\mathbf{2}}$41592658334323422 1... |
| 3 | 0.72$\underline{\mathbf{6}}$13934273234823293... |
| 4 | 0.239$\underline{\mathbf{9}}$4293423932323492... |
| 5 | 0.1209$\underline{\mathbf{3}}$239423429342342... |
| 6 | 0.19380$\underline{\mathbf{1}}$34802934820312... |
| $\vdots$ | $\vdots$ |

Construct a new number $r$ where the *ith* digit of $r$ differs from the *ith* digit of the ith number. For instance add one mod 10 to the digit.

$$r = 0.237042...$$

$r$ differs from every number in the above list even though it is on $[0, 1)$, hence no such one-to-one correspondence could exist. Therefore $[0, 1)$ is uncountable.

Attack this problem in a similar way. Assume some $f$ exists that is a one-to-one mapping between $\mathcal{B}$ and $\mathcal{N}$.

| $n$ | $f(n)$ |
|---|---|
| 1 | $\underline{\mathbf{0}}01010101\ldots$ |
| 2 | $1\underline{\mathbf{0}}0101001\ldots$ |
| 3 | $10\underline{\mathbf{1}}111011\ldots$ |
| 4 | $001\underline{\mathbf{0}}01010\ldots$ |
| 5 | $1010\underline{\mathbf{1}}1101\ldots$ |
| 6 | $11100\underline{\mathbf{1}}010\ldots$ |
| $\vdots$ | $\vdots$ |

Then construct a new infinite sequence in $\mathcal{B}$ that is not mapped to by any natural number.

Problem 2. (Exercise 4.7) Show the following set $T$ is countable.

$$T = \{(i, j, k) \mid i, j, k \in \mathcal{N}\}$$

Problem 3. (Problem 4.10)

$INFINITE_{PDA} = \{M \mid M \text{ is a PDA and } L(M) \text{ is an infinite language } \}$

Show $INFINITE_{PDA}$ is decidable.

Problem 3. (Problem 4.10)

$INFINITE_{PDA} = \{M \mid M \text{ is a PDA and } L(M) \text{ is an infinite language } \}$

Show $INFINITE_{PDA}$ is decidable.

- Would accept a PDA that recognizes $\{0^n 1^n\}$

Problem 3. (Problem 4.10)

$INFINITE_{PDA} = \{M \mid M \text{ is a PDA and } L(M) \text{ is an infinite language } \}$

Show $INFINITE_{PDA}$ is decidable.

- Would accept a PDA that recognizes $\{0^n1^n\}$
- Would reject a PDA that recognizes $\{0^n1^n \mid n \leq 100\}$

Problem 3. (Problem 4.10)

$INFINITE_{PDA} = \{M \mid M \text{ is a PDA and } L(M) \text{ is an infinite language } \}$

Show $INFINITE_{PDA}$ is decidable.

- Would accept a PDA that recognizes $\{0^n1^n\}$
- Would reject a PDA that recognizes $\{0^n1^n \mid n \leq 100\}$
- Would accept a PDA that recognizes syntactically correct C programs.

Problem 3. (Problem 4.10)

$INFINITE_{PDA} = \{M \mid M \text{ is a PDA and } L(M) \text{ is an infinite language } \}$

Show $INFINITE_{PDA}$ is decidable.

- Would accept a PDA that recognizes $\{0^n1^n\}$
- Would reject a PDA that recognizes $\{0^n1^n \mid n \leq 100\}$
- Would accept a PDA that recognizes syntactically correct C programs.
- Would reject a PDA that recognizes syntactically correct C programs limitted to 100 Kb.

Problem 4. (Problem 4.25)

$E = \{M \mid M \text{ is a DFA that accepts some string with more 0s than 1s}\}$

Show $E$ is decidable.

Problem 5. (Problem 4.26) $C = \{(G, x) \mid G$ is CFG that generates some string $w$ where $x$ is a substring of $w\}$

Problem 5. (Problem 4.26) $C = \{(G, x) \mid G$ is CFG that generates some string $w$ where $x$ is a substring of $w\}$
You are given a CFG and a string $x$. You must decide if the CFG generates any string that has $x$ as a substring.

Problem 6.

- Remember an enumerator for an infinite language, will never stop and will eventually print each string in the language.
- Use an enumerator and try an argument by diagonalization.

Problem 7. (Extra Credit) Remember this is an iff, and one direction is fairly simple and everything you need is right there. Try to get at least one direction.

4011 Standard Library - dfa.h contents

- $A_{DFA}$: Given DFA and string, decides if the string is accepted.

4011 Standard Library - dfa.h contents

- $A_{DFA}$: Given DFA and string, decides if the string is accepted.
- $E_{DFA}$: Given DFA, decides if it accepts the empty language.

4011 Standard Library - dfa.h contents

- $A_{DFA}$: Given DFA and string, decides if the string is accepted.
- $E_{DFA}$: Given DFA, decides if it accepts the empty language.
- $EQ_{DFA}$: Given two DFAs, decides if accept same language.

4011 Standard Library - dfa.h contents

- $A_{DFA}$: Given DFA and string, decides if the string is accepted.
- $E_{DFA}$: Given DFA, decides if it accepts the empty language.
- $EQ_{DFA}$: Given two DFAs, decides if accept same language.
- Using given DFAs can construct new ones that recognize the union, intersection, star, concatenation, compelement, reverse, drop-out, perfect shuffle.

4011 Standard Library - dfa.h contents

- $A_{DFA}$: Given DFA and string, decides if the string is accepted.
- $E_{DFA}$: Given DFA, decides if it accepts the empty language.
- $EQ_{DFA}$: Given two DFAs, decides if accept same language.
- Using given DFAs can construct new ones that recognize the union, intersection, star, concatenation, compelement, reverse, drop-out, perfect shuffle.
- Can convert between DFAs, NFAs, and regular expressions.

4011 Standard Library - dfa.h contents

- $A_{DFA}$: Given DFA and string, decides if the string is accepted.
- $E_{DFA}$: Given DFA, decides if it accepts the empty language.
- $EQ_{DFA}$: Given two DFAs, decides if accept same language.
- Using given DFAs can construct new ones that recognize the union, intersection, star, concatenation, compelement, reverse, drop-out, perfect shuffle.
- Can convert between DFAs, NFAs, and regular expressions.
- Can also create TMs, CFGs, and PDAs that recognize the same language as a DFA.
  - Cannot go in the other direction.

4011 Standard Library - dfa.h contents

- $A_{DFA}$: Given DFA and string, decides if the string is accepted.
- $E_{DFA}$: Given DFA, decides if it accepts the empty language.
- $EQ_{DFA}$: Given two DFAs, decides if accept same language.
- Using given DFAs can construct new ones that recognize the union, intersection, star, concatenation, compelement, reverse, drop-out, perfect shuffle.
- Can convert between DFAs, NFAs, and regular expressions.
- Can also create TMs, CFGs, and PDAs that recognize the same language as a DFA.
  - Cannot go in the other direction.
- Can determine the pumping length of a DFA.

4011 Standard Libray - cfg.h contents

- $A_{CFG}$: Given CFG and string, decides if the string can be generated.

4011 Standard Libray - cfg.h contents

- $A_{CFG}$: Given CFG and string, decides if the string can be generated.
- $E_{CFG}$: Given CFG and decides if it generates empty language.

4011 Standard Libray - cfg.h contents

- $A_{CFG}$: Given CFG and string, decides if the string can be generated.
- $E_{CFG}$: Given CFG and decides if it generates empty language.
- Can form new CFGs from union, star, concatenation of given CFGs.

4011 Standard Libray - cfg.h contents

- $A_{CFG}$: Given CFG and string, decides if the string can be generated.
- $E_{CFG}$: Given CFG and decides if it generates empty language.
- Can form new CFGs from union, star, concatenation of given CFGs.
  - In general, CANNOT create CFGs by intersecting or complementing other CFGs.

4011 Standard Libray - cfg.h contents

- $A_{CFG}$: Given CFG and string, decides if the string can be generated.
- $E_{CFG}$: Given CFG and decides if it generates empty language.
- Can form new CFGs from union, star, concatenation of given CFGs.
  - In general, CANNOT create CFGs by intersecting or complementing other CFGs.
- Can convert between CFGs and PDAs.

4011 Standard Libray - cfg.h contents

- $A_{CFG}$: Given CFG and string, decides if the string can be generated.
- $E_{CFG}$: Given CFG and decides if it generates empty language.
- Can form new CFGs from union, star, concatenation of given CFGs.
  - In general, CANNOT create CFGs by intersecting or complementing other CFGs.
- Can convert between CFGs and PDAs.
- Can determine the pumping length of a *CFG*.

More tools for DFAs than CFGs.

- $EQ_{CFG}$ is undecidable.

More tools for DFAs than CFGs.

- $EQ_{CFG}$ is undecidable.
- Cannot construct intersection or complement from CFGs in general.

$$INFINITE_{DFA} = \{M \mid M \text{ is a DFA and } L(M) \text{ is an infinite language}\}$$

$$INFINITE_{DFA} = \{M \mid M \text{ is a DFA and } L(M) \text{ is an infinite language}\}$$

Solution Idea: If a DFA doesn't accept any strings of length at least $p$, where $p$ is the pumping length, then it must be finite, and if it accepts even one string of length $p$ then it can be pumped, so the language is infinite. So we just need to test if the DFA accepts even one string of length at least $p$.

$$INFINITE_{DFA} = \{M \mid M \text{ is a DFA and } L(M) \text{ is an infinite language}\}$$

$T=$ On input $M$ where $M$ is a DFA.

1. Calculate the pumping length of $L(M)$ call this $p$.
2. Construct a DFA, $A$, which accepts a given $w$ iff $|w| \geq p$.
3. Construct a DFA, $B$, which recognizes $L(M) \cap L(A)$
4. Run $E_{DFA}$ with $B$ as input. If it rejects, *accept*, else *reject*.

Problem 3 asks you to show the same thing PDAs. This is a bit more tricky, but you can use a solution along the same lines. Some problems:

Problem 3 asks you to show the same thing PDAs. This is a bit more tricky, but you can use a solution along the same lines. Some problems:

- 1) Cannot calculate the pumping length from a PDA directly.

Problem 3 asks you to show the same thing PDAs. This is a bit more tricky, but you can use a solution along the same lines. Some problems:

- 1) Cannot calculate the pumping length from a PDA directly.
- 2) Can construct a PDA that accepts all strings of length at least $p$, but cannot form a PDA by intersecting two.

Problem 3 asks you to show the same thing PDAs. This is a bit more tricky, but you can use a solution along the same lines. Some problems:

- 1) Cannot calculate the pumping length from a PDA directly.
- 2) Can construct a PDA that accepts all strings of length at least $p$, but cannot form a PDA by intersecting two.

Use results from cfg.h to address 1. Hopefully next example will illuminate how to deal with 2.

$$PAL_{DFA} = \{M \mid M \text{ is a DFA and accepts some palindrome.}\}$$

Show $PAL_{DFA}$ is decidable.

Some possible solutions?

$T_1=$ On input $M$ where $M$ is a DFA.

1. Construct DFA $R$ that recognizes $L(M)^R$
2. Construct DFA $N$ that recognizes $L(M) \cap L(R)$
3. Run $E_{DFA}$ against $N$. If it rejects, *accept*, else *reject*.

Some possible solutions?

$T_1$= On input $M$ where $M$ is a DFA.

1. Construct DFA $R$ that recognizes $L(M)^R$
2. Construct DFA $N$ that recognizes $L(M) \cap L(R)$
3. Run $E_{DFA}$ against $N$. If it rejects, *accept*, else *reject*.

$T_2$= On input $M$ where $M$ is a DFA.

1. Construct DFA $R$ that recognizes $\{w \mid w = w^R\}$.
2. Construct DFA $N$ that accepts $L(M) \cap L(R)$
3. Run $E_{DFA}$ against $N$. If it rejects, *accept*, else *reject*.

Some possible solutions?

$T_1$= On input $M$ where $M$ is a DFA.

1. Construct DFA $R$ that recognizes $L(M)^R$
2. Construct DFA $N$ that recognizes $L(M) \cap L(R)$
3. Run $E_{DFA}$ against $N$. If it rejects, *accept*, else *reject*.

$T_2$= On input $M$ where $M$ is a DFA.

1. Construct DFA $R$ that recognizes $\{w \mid w = w^R\}$.
2. Construct DFA $N$ that accepts $L(M) \cap L(R)$
3. Run $E_{DFA}$ against $N$. If it rejects, *accept*, else *reject*.

What is wrong with them?

For any alphabet $\Sigma$, can construct CFG that recognizes the set of all palindromes. Start with rule $S \longrightarrow \epsilon$, and for each $a \in \Sigma$ add rules $S \longrightarrow aSa$ and $S \longrightarrow a$.

For any alphabet $\Sigma$, can construct CFG that recognizes the set of all palindromes. Start with rule $S \longrightarrow \epsilon$, and for each $a \in \Sigma$ add rules $S \longrightarrow aSa$ and $S \longrightarrow a$.

$T_3 =$ On input $M$ where $M$ is a DFA.

1. Convert $M$ into PDA $M'$ that recognizes same language.
2. Construct PDA $P$ from palindrome CFG created this $\Sigma$
3. Construct PDA $Q$ to recognize $L(M') \cap L(P)$
4. Convert PDA $Q$ into CFG $T$
5. Run $E_{CFG}$ on $T$, if it rejects, *accepts*, else *reject*.

For any alphabet $\Sigma$, can construct CFG that recognizes the set of all palindromes. Start with rule $S \longrightarrow \epsilon$, and for each $a \in \Sigma$ add rules $S \longrightarrow aSa$ and $S \longrightarrow a$.

$T_3=$ On input $M$ where $M$ is a DFA.

1. Convert $M$ into PDA $M'$ that recognizes same language.
2. Construct PDA $P$ from palindrome CFG created this $\Sigma$
3. Construct PDA $Q$ to recognize $L(M') \cap L(P)$
4. Convert PDA $Q$ into CFG $T$
5. Run $E_{CFG}$ on $T$, if it rejects, *accepts*, else *reject*.

What is wrong with this?

Homework 5 Lemma: Given a PDA $P$ and DFA $D$, one can construct a PDA that recognizes $L(P) \cap L(D)$.

- Can "intersect" two DFAs, by simulating both DFAs with states. In new DFA, each state represents a pair of states.

Homework 5 Lemma: Given a PDA $P$ and DFA $D$, one can construct a PDA that recognizes $L(P) \cap L(D)$.

- Can "intersect" two DFAs, by simulating both DFAs with states. In new DFA, each state represents a pair of states.
- Can't "intersect" two PDAs, because cannot simulate two stacks with one.

Homework 5 Lemma: Given a PDA $P$ and DFA $D$, one can construct a PDA that recognizes $L(P) \cap L(D)$.

- Can "intersect" two DFAs, by simulating both DFAs with states. In new DFA, each state represents a pair of states.
- Can't "intersect" two PDAs, because cannot simulate two stacks with one.
- Can "intersect" a PDA and a DFA, by simulating pairs of states and using the new PDA's stack to simulate the original PDAs stack.

Homework 5 Lemma: Given a PDA $P$ and DFA $D$, one can construct a PDA that recognizes $L(P) \cap L(D)$.

- Can "intersect" two DFAs, by simulating both DFAs with states. In new DFA, each state represents a pair of states.
- Can't "intersect" two PDAs, because cannot simulate two stacks with one.
- Can "intersect" a PDA and a DFA, by simulating pairs of states and using the new PDA's stack to simulate the original PDAs stack.
- Add this to your toolkit.

$T =$ On input $M$ where $M$ is a DFA.

1. Construct a PDA, P, from the palindrome CFG for this $\Sigma$.
2. Use above lemma to build PDA $Q$ to recognizes $L(P) \cap L(M)$
3. Run $E_{CFG}$ on $Q$, if it rejects, *accept*, else *reject*.

Revisiting Problem 4. Show the following language is decidable.

$E = \{M \mid M \text{ is a DFA that accepts some string with more 0s than 1s}\}$

This problem is somewhat analogous to the above problem. If you get stuck review the above problem thoroughly.

Problem 5. $C = \{(G, x) \mid G$ is CFG that generates some string $w$ where $x$ is a substring of $w\}$

Problem 5. $C = \{(G, x) \mid G$ is CFG that generates some string $w$ where $x$ is a substring of $w\}$

Start with some simpler problems and let me know if you need advice on these.

- $\{D \mid D$ is DFA that generates some string $w$ where 111 is a substring of $w\}$

Problem 5. $C = \{(G, x) \mid G$ is CFG that generates some string $w$ where $x$ is a substring of $w\}$

Start with some simpler problems and let me know if you need advice on these.

- $\{D \mid D$ is DFA that generates some string $w$ where 111 is a substring of $w\}$
- $\{(D, x) \mid D$ is DFA that generates some string $w$ where $x$ is a substring of $w\}$
- Now extend this to the CFG case.