CS 3100 – Models of Computation – Fall 2010
Frequently Asked/Answered Questions (FAQs)

Contents

1 FAQ for Asg9

1.1 DDcal – don't call "dot" – Parsing Error

```
I'm having some difficulty forcing DDcal to accept a
specific variable
order.  I've written the following into a file called sum_bad.ddc
(without the --start-- and --end-- lines).
--start--
var = ...
...
dot sum_bad.dot
[s1 s2 s3]
--end--
When I load this with DDcal I get an error message saying,
"A syntax
error was detected in the input statement.
 Click on the "OK" button to continue."
...

===
```

Answer: Please don't call dot within ddcal. Remove that line. Then later you can save the drawing as a postscript file and submit.

1.2 What to submit?

```
> What do you want handed in for assignment 9 in regards to DDCal?
>
> Would it be acceptable to submit the script as part of my pdf file or
> would you like the script separate from the pdf doc?

Just for the Lewis Carroll problem, I'd like a script,
a BDD showing all the variables,
the final contradiction output,
and an explanation. For the others,
simply a script and an explanation will do (no PDF needed).
```

1.3 Degenerate output in DDcal

Question:

```
Hi,

for the following script:
var = i...
```

```
A1 = ...
...
[contra]

all it's outputting is (in two separate boxes with no connecting lines):

contra

(nil)

What does this mean?
```

Answer:

It is outputting something degenerate (a 'croc' ?). The way to overcome this bug is to display something else also – try [A1 proofGoal contra] and then you'll see a fuller diagram.

I recommend that you display all the variables except for var. Then you can study and understand each function's BDD (more educational for a smaller example).

1.4 Empty Tape Reduction

The author un-necessarily confuses the description. Here is a better description:

- Take a $\langle M, w \rangle$ pair
- Produce a single machine M_w
- M_w will have a linear sequence of initial steps, with M's code hanging at the end—like this:

  ```
  ->O--a1-->O--a2-->O--a3-->O--a4-->O--..-->O--an-->O->Code-of-M-HERE.
  ```

- What will `a1`, `a2`, etc. be? They will be just instructions to write w on the tape! `a1` will say: write the first symbol of w on the tape; move head right; go to the next state where `a2` gets done; etc.
- Call M_w a "lollipop machine" because it looks like a lollipop. It has a long string of boring steps `a1` etc. and ending in a big lump—which M's code.
- Now if you face up M_w to an empty tape (really it is asking if M_w accepts epsilon), it will chug along the stick of the lollipop. That will populate w on the tape. Then the lollipop head is engaged in, and that runs M on w.
- There is no "erase the tape, grease it, apply Armor-all to the tape, spit on the tape and wipe it with Chamois leather" that you need to do (that part of the author's writing is very confusing).

1.5 DDCal Usage Tips

If files have windows format newline endings, if you want to know how to ssh -Y into cade, etc.

```
GIST: DDcal can only read files with unix newline endings.

>> I'm quite sure you're using a file in dos format. On cade run this
>> command on your file to convert the line endings to unix format:

 vim  {file} -c 'se ff=unix|wq'


>>>> Okay first thank you for your help, I was able to get the program to run.
>>>> However it won't open my file, I get this error message:

>>>>> Have you followed the directions in
>>>>> http://www.eng.utah.edu/~cs3100/notes26/final-notes-cs3100-f10.pdf on
>>>>> how to connect with ssh and use DDcal? Since you didn't specify, the
>>>>> only problem I can imagine is that you don't have x-forwarding enabled
>>>>> (the -Y flag to ssh) or you're using windows. I don't use windows but
>>>>> I think, in addition to x-forwarding, you need to install and run an
>>>>> xserver on top of windows. Try these directions to setup putty and
>>>>> xwin32 on windows:
>>>>> http://www.math.umn.edu/systems_guide/putty_xwin32.html.
>>>>>
>>>>> Let me know if the putty/xwin32 solution is relevant. Then I can try
>>>>> and help you install DDcal locally but I'm a bit clueless about
>>>>> installing unix applications on windows. Let me know what operating
>>>>> system you're using.
>>>>>

>>>>>> You can get it [DDCal] by Googling Fabio Somenzi Colorado. You will need to
>>>>>> install Perl/Tk

>>>>>>> I'm having a bit of trouble running the BDD program through ssh at home.
>>>>>>> Is there anyplace that we can download the program directly?
```

1.6 Worst and Best BDD Variable Ordering

```
> from the chapter it says that finding the best variable ordering for a BDD is NP complete correct? does that mean that there i
No, it means any algo is no better than a brute-force search
> The example in the book gives:
>
> better: abcdef
> worse acebdf
>
> but does not say if they are the best or worst per se.
The one that postpones the decisions maximally causing the BDDs to grow in size is worse.
> it also does not say how these specific orders were determined? For out problem would the best be: order in which the variable
Yes for this problem interleaved is better. Think of the BDD size. The BDD size is also the minimal DFA size for the related lar

Think of "best" and "worst" as "that var order that gives nearly the smallest BDD" and "that order that gives the largest"
```

2 FAQ for Asg8

2.1 Question 4

```
Build a machine similar to the one built in Lecture 24 for RegularTM .
```

```
Only one change needed : make the pattern something other than 0^n 1^n

But the idea is similar. We define a new machine M'

* If M does not accept w, M's language must be non context-free.

* If M accepts w, M's language must be context-free or regular
   (being regular implies being context-free - you may find it
    easier to make M's language regular in this case)
```

2.2 CFLs decidable; TMs having CF languages not!

```
Ganesh,
I am looking at what you wrote as the "StateUsage" or "Deadcode" problem,
but I  am having a tough time following it. (lec notes 24)
Why are we able to suppose there is a decider for that problem?
We suppose -- precisely to run into a contradiction
On problem 4 it
seems we would not be able to do this because we would be proving something is
-not- decidable.  Is there a property to CFGs that would make them not
decidable?  In class you suggested we should use proof by contradiction, but I
am not sure what would be contradictory about CFGs and decidability.

>> You are mixing up things
>> Two problems discussed above

>> * whether a TM will use a state q

>> * whether a TM has a language which happens to be a CFL
>> (this is possible, right? A TM can have any language you wish it to have.
>> What if you arrange a TM to have a CFL as its language?)

>> The trouble is , ... by looking at TMs, smelling TMs,
>> measuring the number of "left move" commands in them, etc. etc.
>> one has NO WAY of knowing
>> whether the TMs themselves have a CFL as their language.


If I remember correctly, the book mentioned that not all CFGs are decidable,
is that true?

>> Are you asking if all CFLs are decidable? You can't ask if a CFG is
>> decidable.
>> Decidability is an attribute of a language, not a grammar.

>> You have to ask "all CFLs are decidable"? Yes they are.
>> CFLs are decidable. All you need is: provide a PDA.

>> BUT WHAT IS NOT DECIDABLE IS WHETHER A TM THAT IS
>> SLIPPED UNDER YOUR NOSE has a language that is a CFL...



It is not a touring machine. But Turing may have had a touring machine such as
```

```
a touring bike. Can Turing's touring bike go into an infinite loop?


>   Also, how would I go about proving something is decidable?

Provide an algorithm (means always halts).  Generally we propose an
algorithm in English and argue that it halts on all inputs. There are
other ways of doing this also.

> For 1c (If there is a GCD(X, Y) = Z, then (GCD(X+Y, Y) = Z) I mean that I think
> there exists some specific mathematical proof that I am supposed to use, but I
> don't know it.  Even if I show a few examples, that is not 'proving' it.

You are right.

Here is how you prove

Let "|" means "is divisible by"

so X|Z and Y|Z and further if some X|p then Z >= p... right?

That is how you go!  Then use some logic



If M accepts w, does the authors' "fixin' job" (mapping reduction)
produce a machine M' that qualifies for being in Abt ?

> For qn 4 ...

I wrote a proof for regular_TM . This problem merely asks you to

1) understand that proof
2) change the construction of the machine constructed in that proof
 to suit this problem's requirements.
```

3 Dealing with IFF

For proving A IFF B, one approach is : prove A Implies B and then prove B Implies A. Another approach that works handier often is: prove A Implies B and then prove not(A) Implies not(B). This is because A Implies B is completely equivalent to not(B) Implies not(A). If you don't believe me, check the truth values.

So for mapping reductions, x in A iff f(x) in B can be broken down as: (1) x in A implies f(x) in B (2) x not in A implies f(x) not in B. Hey, ain't that easier?! This will help you with Asg8. This is on the FAQ in fact.

4 Notes on Cardinality and Infinities

The TAs pointed out that some of you are having trouble with 'finite' vs 'infinite'. Here is a simple explanation.

Something is finite if there is a natural number associated with it that measures its "size".

A string is finite if its length is a Natural number (0, 1, 2.. etc). Even Avogadro's number (sutably rounded) is allowed, but it has to be a number (Avogadro's number is roughly 6.022 * 1023). A string that is this long is also finite. A string is infinite if its length can't be expressed by any natural number. In other words, an infinite string has, for any position i that is a natural number, a well defined position i+1 in the string. A finite string has a "last" position beyond which it has no positions.

A finite set is the same thing. Its cardinality or size is expressible as a natural number. You can also count the elements in some fashion. There is a "last" element. An infinite set is one where after removing a finite subset out of it, we still have elements left.

There are many cardinalities for infinite sets. Nat is a proper subset of Real. But they have different cardinalities. There can't be a 1-1 correspondence between them. Odd is a proper subset of Nat. But Odd and Nat have the same cardinality. There is a 1-1 correspondence between them. So our gradations are really

Finite , Countably infinite , Uncountably infinite

Uncountable infinities are many too. The first uncountable infinity is that of Reals, or $\aleph_1$. Reals are equal in size to the powerset of Nat which is $\aleph_0$. If you take a powerset of Reals, you come to $\aleph_2$, the second uncountable infinity .. and so on. There are an infinite number of infinities.

Imagine the set of all squiggly lines you can draw on a piece of paper. Each squiggly line is a subset of real coordinates. Each coordinate pair can be mushed into one Real number (by weaving the digits - first coordinate digits go into odd positions, second coordinate digits into even positions). Thus each curve is a subset of Reals. So the set of all curves is $\aleph_2$ big. It is the second uncountable infinity - bigger in size than Reals.

Now that is the Real story of infinities.

The word cardinality (of sets) stands for the "size" of sets.

- The cardinality of $\emptyset$ is 0;
- The cardinality of $\{1, 2, 3\}$ is 3, and so on...

For any two **finite** sets A and B such that A is a proper subset of B, clearly, the cardinality of A is smaller than that for B.

This is not necessarily true for infinite sets!

- *Odd* is a proper subset of Nat, yet set *Odd* is as big as set Nat in the sense of cardinality
- *Odd* and Nat have the same cardinality because we can find a bijection from *Odd* to Nat and vice-versa (one-to-one, onto, and total map, as was explained in lectures). Consider the function that takes an odd number x and produces $(x-1)/2$; this produces every Nat. Conversely, given Nat y, the function $2y+1$ produces every *Odd* bijectively.
- For Nat and *Real*, while Nat is a proper subset of *Real*, there exists no such bijection. This is why the cardinality of *Real* is higher than that of Nat.
- The absence of bijection is proven through diagonalization.

5 RE sets being closed under Kleene-star

How do we in general show that some set S is closed under Kleene star?
First, start with a definition:
S^* = union (for every k greater than equal to 0) of S^k
What we have to show is: if S is RE, then so is S^*.

- We have to show S^0 is RE. This is trivial: the TM checks if the given string is empty.
- Assuming S^k is RE, show S^{k+1} is RE. This is easy. TM1 for S^k and TM2 for S are used together. The given string s is split in all possible ways a, b such that $s = ab$, and TM1 checks for membership of a and TM2 for b. Both TM1 and TM2 must accept; then s is a member of S^{k+1}.
- This shows by induction that for all k, if S is RE, so is S^k
- What we need to show is that union (for every k greater than equal to 0) of S^k is RE
- This is easy: each string, if in S^*, is in some S^k. A master TM that keeps generating successive S^k membership checking "subroutine" calls is what is needed. If at all some given string s is in some k iterate of S, it will be found.

6 TM Questions on Asg7

```
> What do you mean by show the first six configurations

this word is defined in Sec 13.4

> in the Trace View? What I take that to mean is to take one step at a
> time and for each step, take a screenshot of each step, and then put
> the first 6 steps like so in to a document?

Yes the first six steps taken by the TM.

> And also, is is basically the same thing for number 3 or is there
> something else we have to do in order to deal with the non-determinism

The NDTM would go thru different configurations
```

7 The Post Correspondence Problem (PCP)

7.1 Usage of the PCP solver in Asg7

```
Where is the PCP solver that the extra credit document talks about as
well? Thanks!

---
PCP
```

The sources are in ~cs3100/pcp

In that dir, there is a linux binary in file pcp and a README file

Also "pcp -h" will give you help info

You may run it while situated in your directory as follows

- goto your directory

- ln -s ~cs3100/pcp/pcp (assuming you are on a linux machine)

- Make an input file , say "in1", containing for instance

```
4 3
1    01    0   001
101  0   001   1
```

- then run ./pcp -i in1 -o out1

- Go look into out1.
  It prints a solution, if there is a solution, it may look like this:

```
Instance 1:

4 3 10 1 0.005301
1    01   0    001
101  0    001  1

Solvable!

 1 1 1 0
 1 0 0 1
Gcd: 1
Visited node in original direction: 4604
Visited node in reverse direction: 46
Choose the reverse direction:
Find the solution in depth: 10 (depth threshold: 10)
  1  3  1  4  2  1  2  2  4  2
Time spent: 0.005301 seconds.


visited node: 46, last iteration: 46
```

```
cutoff node : 0, last iteration: 0
time: 0.005301, last iteration: 0.000024
Search speed: 0.009M/s, laster iteration: 1.917M/s


---

The solution is the reverse of 1 3 1 4 2 1 2 2 4 2
(see my lecture notes for lecture 17).

Sometimes the solver chooses the forward direction in which case the
solution is printed normally (without reversal)
```

7.2 General Facts

The "puzzle" using tiles that I showed today is called the Post Correspondence Problem (or PCP) – named after Emil Post, famous logician who formulated this puzzle.

- There is a nice Wikipedia page on this puzzle at

  `http://en.wikipedia.org/wiki/Post_correspondence_problem`

- There is a website that shows how the PCP solver works.

  `http://www.theory.informatik.uni-kassel.de/~stamer/pcp/pcpcontest_en.html`

  I have not downloaded and built the solvers on this site. However one cool animation at this site under "Puzzle" of the above site. Here you can type the "Hint" button and see it solving a puzzle of colored tiles.

- Another PCP solver in PHP is at

  `http://jamesvanboxtel.com/projects/pcp-solver/`

  This does not appear to be a fast solver, but does solve simple puzzles thru the web.

- The most powerful PCP "solver" that I am aware of (i.e. solves when possible) is by Zhao. I compiled and kept a copy in `~cs3100/pcp`.

  I'll teach you its usage soon, but the README file in this directory together with the help messages obtained by typing "pcp -h" will give you a pretty complete picture.

  There are examples in file sol-s4-w4 which are pretty amazing (small sets of tiles resulting in long solutions)

8 CKY Parsing Illustrated

```
(This will go onto the FAQ.)

On question 6, I did not mean you to be doing a full parse using the CKY algorithm. You can inspect the grammar and see how "ab"

But since some of you may want to get practice on using the CKY algorithm, I can provide some tips in that regard also.
```

```
S -> EF | AF | EB | AB
X -> AY | BY | a | b
Y -> AY | BY | a | b | c
E -> AX
F -> BX
A -> a
B -> b
C -> c

Positions and string:
0 1 2
 a b

Dyn. programming tables

  0

 {X,Y,A}    1

 {S,E,X,Y}  {X,Y,B}      2

Since S is in the position 0,2, we can derive ab.
```

9 Unix Shell, Script File, a.out, and paths

1) ./a.out is needed for some people who don't have "." (current directory) in search path. That is "d

2)

> I don't really understand how to "submit my work as a script file" can you explain what is going on w
A good approach:

Let % be your unix prompt

% script asg6-recording.txt

% .. then do your work here... for example

% echo " I yam gonna show ya how yacc works "

% lex ...

% yacc ...

```
% cc ...

% ./a.out

  type calculator expressions here

% echo " see, my calculator worked fine - all associativity and such A-Ok "

% exit

.. Here the system will have a recording of all your work from the line "script .." till "exit"

mail that file (asg6-recording.txt) to us.

The "echo " commands allow you to document your work and make that part of your recording.

Do a "man script" for more info.
```

10 Flex Syntax

```
Flex is finicky
The syntax is as below - i.e. the rule must be on the same line !!
At least it must begin on the same line.

THIS IS OK:

 Your-Reg-Exp {
  printf("Found a string %s", yytext);
 }

THIS IS NOT OK:

 Your-Reg-Exp
 {
  printf("Found a string %s", yytext);
 }
```

11 Assignment 4, Extra Credit

```
> For JFLAP does a parse mean a single string?

That is usually the meaning.
```

```
> JFLAP offers two options for brute force single or multiple.
Which option should we use?

Single is enough. Brute-force parse is very slow. That said,
if you think that multiple brute-force parse will finish reasonably quickly,
it makes sense to show it because it nicely shows what is accepted and what
is not.  You may want to test as 'single brute-force'
and then for those strings that are quick to parse,
perhaps list it one after the other as multiple brute-force.
Don't spend too much time on this part though.

> If single, do you want the graph printed out for the successful string?

It will be natural to print out the parse tree ("graph" as you put it)
for the successful parse. At least for one successful parse.
```

12 Pumping Lemma Clarified

```
> I don't quite understand this.  How do you choose u v and w?
Making the string sufficiently long, ie  0^k  and 1^k ,  you are FORCING  u and v to lie within the 0s.

The w is then the "catch all the remaining" string. We never care about what w has.
>    I see the point of
> it, though, to prove an RE has finite states, but I don't quite see how it does
> prove it.
>
> For example, the 0^n, 1^n, n>0 thing.  This is not an RE because it would have
> to remember n, but how does the lemma prove that?

See below.
```

Conversation between U and Y (two completely fictitious characters; I switch U and Y each time I tell this..).

Y claims: "Hey, I've built a DFA for $\{0^n1^n..\}$ but I won't show it to you."

U : "Impossible! Would you at least tell me how many states the DFA has?"

Y : "k"

U : "OK, here is a string – 0^k1^k – go feed it to your DFA and tell me what it did."

Y : "It accepts!" :-)

U : "Ahha, while your DFA was processing 0^k, did it go thru a loop? It must!"

Y : (reluctantly) "Yes it did."

U : "OK, call the portion till the loop state "u", call the portion within the loop "v", and call the rest of the string w."

Y : "OK."

U : "Now, "v" must be purely made up of 0's, right? Also v is non-empty, right?"

Y : "Yes (growing concerned..)"

U : "OK, tell me what the string reads like if I bypass the loop AND go to the final state." (Going to the final state means it accepts!)

Y : "u w"

U : "OK, tell me what string if I spin the loop twice AND THEN go to the final state."

Y : “u v v w”
U : “OK, your DFA is not quite right, because u w has fewer 0s than 1s and u v v w has more 0s than 1s.”
U : “So your first claim (that you have a DFA for EXACTLY THIS LANGUAGE) is wrong.”
Y : “Thanks for the correction. I withdraw my DFA from consideration.”
Like that, every claimed DFA will be refuted. This means such a DFA can’t exist.

13 Distinguishable Strings Clarified

Some of you have questions on distinguishable strings. I decided to postpone a full in-depth discussion till after the exam.

Here is the question I received on the topic:

```
>  For distinguishable strings say in language L,
> do x,y have to be elements of L themselves to concatenate
> suffix z s.t. xz is in L but yz is not?
>  The book is not clear. I would think not.
> For instance L={00,11} there is no suffix z
> other than epsilon to concatenate s.t. xz is in L if x is in L as well.
```

The idea is to give any claimed DFA for a language L “royal trouble.” I.e. so much trouble that it can’t exist !!

How do we give the claimed DFA for L so much trouble? Here is how.

We have to be creative enough to pick a BIG set of strings - ANY set of strings S such that

1. S has infinitely many strings in it
2. S must be such that if we pull out ANY TWO strings x and y out of S, we must be able to find a z – ANY z – that distinguishes x and y with respect to the DFA (or the language of interest, L).

If such a big set S is found, then we can tell our fictitious character “Y” (the purveyor of impossible DFAs !!) that the DFA can’t exist :

1. Look, I have this big set S. Your DFA will have to distinguish every pair of strings in S with respect to the z’s that I shall provide
2. Hence the DFA must have infinite number of states (to distinguish xz and yz, the DFA must be in different states after ingesting xz as opposed to ingesting yz
3. Hence what you have is a DA not a DFA – the ”F” word has to be dropped :-) (i.e. you have a DIA - Deterministic Infinite-state Automaton)

How to do this in practice?

1. Stare hard at the language L (say 0^n1^n)
2. After a few days of that, you suddenly realize – ahha,

$$\{0^i \mid i \geq 0\}$$

is my big S set.

3. Why? because if I take any two members of my S set... i.e. 023 and 024 for instance, I have ONE z, let us say 123

   such that

   023 123 is in L

   but

   024 123 is NOT in L

The DFA will have to be in two different states after seeing 023 123 and 024 123. This is true for every pair of strings drawn from S.

14 Asg3 and Midterm Practice

- Q: If I have a DFA with 6 states and it is being subject to product construction with a DFA of 60 states, will I get 360 states?
- A: Not always. You have to introduce pairs of states lazily, i.e. as needed. Often the resulting DFA may only have very few states.
- Q: What is the symbol for epsilon in JFLAP?
- A: It is `!` (exclamation mark) – not `e`. If you typed an `e` in an RE, your machine will look for you to type in an `e`.
- Q: Where is the JFLAP tuturial?
- A: `http://www.cs.duke.edu/csed/jflap/tutorial/`