Theory of Computation

Calendar

			October 2011			»
Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

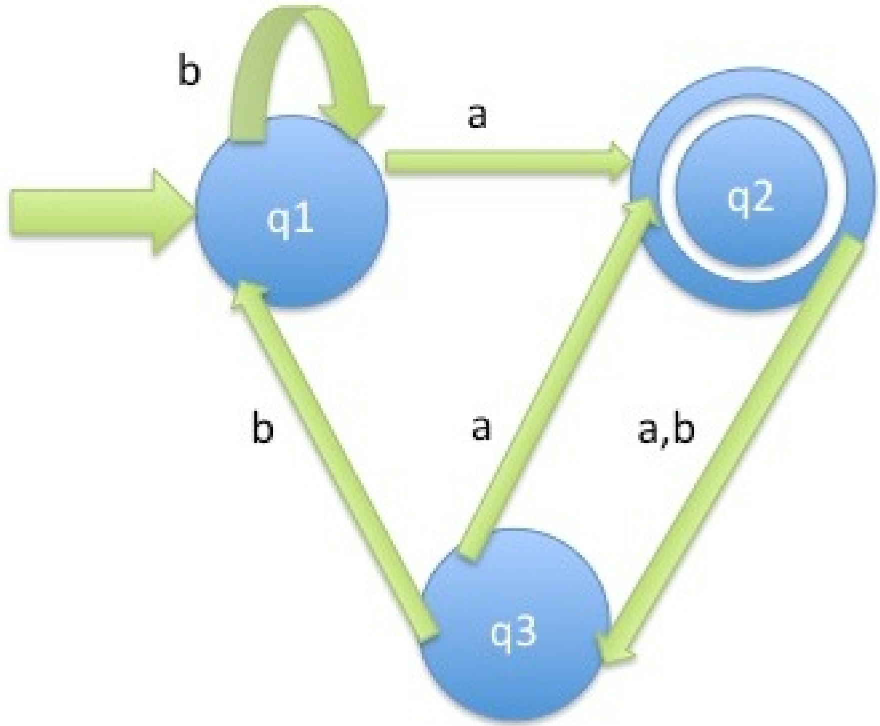
12/18/2011

Random Entry

Sipser 4.4

Sipser 4.4

Sunday, October 30, 2011, 04:20 PM
Posted by Administrator



Emptiness Testing

view entry (1 view) | permalink | print article | ○○○○○ (0 / 0)

Archives

View Archives
2011
October
Sipser 4.4
10/30/11

Search

Search

Go

Counter Totals

Total: **13,507**
Today: **1**
Yesterday: **5**

Most Recent Entries

Sipser 4.4

Bookmark Sharing

SHARE

No Footer - Page Generated in 0.274 seconds | Site Views: 13507

Theory of Computation

Links

Home
Contact
Stats
Message Board
About

Login

Calendar

«		December 2011				
Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

12/18/2011

Random Entry

Sipser 4.4

Archives

About

What is this site about?

This site discusses different theories of computation. With a focus on common books on the subject, people can read the books and then come here to discuss the topic. Many times the books pack a lot of information into a small amount of space. This site can expand that space by providing a discussion mechanism.

Please post and post often!

View Archives
2011
October

Search

Search

Go

Counter Totals

Total: **13,507**
Today: **1**
Yesterday: **5**

Most Recent Entries

Sipser 4.4

Bookmark Sharing

SHARE

No Footer - Page Generated in 0.0612 seconds | Site Views: 13507

Theory of Computation

Discuss Theory of Computation

Search... Search
Advanced search

It is currently December 18th, 2011, 7:35 pm

View unanswered posts • View active topics

FORUM	TOPICS	POSTS	LAST POST
Theory of Computation Discuss topics related to Theory of Computation	9	31	by **tmorelli** December 18th, 2011, 6:06 pm

LOGIN • REGISTER

Username: Password: | Log me on automatically each visit ☐ Login

WHO IS ONLINE

In total there is **1** user online :: 0 registered, 0 hidden and 1 guest (based on users active over the past 5 minutes)
Most users ever online was **5** on December 18th, 2011, 7:26 pm

Registered users: No registered users
Legend: Administrators, Global moderators

STATISTICS

Total posts **31** • Total topics **9** • Total members **5** • Our newest member **ksimpson**

Michael Sipser

NEWTOPIC Search this forum... Search

9 topics • Page 1 of 1

TOPICS	REPLIES	VIEWS	LAST POST
Theorem 4.23 by ksimpson » December 18th, 2011, 6:04 pm	1	2	by **tmorelli** December 18th, 2011, 6:06 pm
Theorem 4.22 by sgreen » December 18th, 2011, 5:44 pm	1	3	by ksimpson December 18th, 2011, 5:49 pm
Theorem 4.8 by ksimpson » December 18th, 2011, 5:13 pm	1	4	by jwilliams December 18th, 2011, 5:26 pm
Theorem 4.6 by **tmorelli** » December 18th, 2011, 4:42 pm	5	7	by **tmorelli** December 18th, 2011, 4:56 pm
Theorem 4.5 by ksimpson » December 18th, 2011, 4:13 pm	4	6	by ksimpson December 18th, 2011, 4:18 pm
Theorem 4.3 by **tmorelli** » December 16th, 2011, 9:05 pm	1	5	by ksimpson December 17th, 2011, 8:18 am
Theorem 4.2 by sgreen » December 16th, 2011, 8:11 pm	2	6	by sgreen December 16th, 2011, 8:28 pm

Need help with Theorem 4.1 by sgreen » December 16th, 2011, 7:29 pm	4	10	by sgreen December 16th, 2011, 7:54 pm
Theorem 4.4 by **tmorelli** » October 30th, 2011, 1:17 pm	3	14	by jwilliams December 16th, 2011, 8:45 am

Display topics from previous: All Topics Sort by Post time Descending Go

NEWTOPIC

9 topics • Page 1 of 1

Return to Board index

Jump to: Michael Sipser Go

WHO IS ONLINE

Users browsing this forum: No registered users and 2 guests

FORUM PERMISSIONS

You **cannot** post new topics in this forum
You **cannot** reply to topics in this forum
You **cannot** edit your posts in this forum
You **cannot** delete your posts in this forum
You **cannot** post attachments in this forum

Board index The team • Delete all board cookies • All times are UTC - 8 hours [DST]

Need help with Theorem 4.1

POSTREPLY Search this topic... Search

5 posts • Page 1 of 1

Need help with Theorem 4.1

by **sgreen** » December 16th, 2011, 7:29 pm

Theorem 4.1 states:

ADFA is a decidable language. It seems easy, but I really dont understand it. Can anyone help?

sgreen

Posts: 8
Joined: December 16th, 2011, 9:01 am

Re: Need help with Theorem 4.1

by **tmorelli** » December 16th, 2011, 7:31 pm

If I were you, I would define the terms in the theory, then walk through the steps.

For example, what is ADFA?

What is a decidable language? Do you know the answer to those?

tmorelli
Site Admin

Posts: 10
Joined: October 26th, 2011, 2:28 pm

Re: Need help with Theorem 4.1

by **sgreen** » December 16th, 2011, 7:48 pm

OK Thanks.

I will show you what I know

ADFA:
A: acceptance.

DFA: Deterministic Finite Automaton
That means there is exactly one exiting transition for each symbol in the alphabet

But what is the acceptance problem?

sgreen

Posts: 8
Joined: December 16th, 2011, 9:01 am

Re: Need help with Theorem 4.1

by **tmorelli** » December 16th, 2011, 7:53 pm

OK, that is good..
So the acceptance problem tests whether a particular deterministic finite automaton accepts a given string.

So he presents a Turing Machine, M, that decides ADFA

This is a two step turing machine:
1. Simulate B on input w
2. If the simulation ends in an accept state, accept, if it ends in a nonaccepting state reject

The DFA is represented by B and the input is w

tmorelli
Site Admin

Posts: 10
Joined: October 26th, 2011, 2:28 pm

So basically run the input w through the dfa that is defined by B

make sense?

Re: Need help with Theorem 4.1

by **sgreen** » December 16th, 2011, 7:54 pm

I think that helps! Thanks!

So in general:

A DFA, B, starts at state q0, and then increments through the input defined by w one at a time. At each state, it will take the transition based on the next input from w. Once all the inputs defined in w have been run through, the final state is the return value.

sgreen

Posts: 8
Joined: December 16th, 2011, 9:01 am

Display posts from previous: All posts Sort by Post time Ascending Go

POSTREPLY

5 posts • Page 1 of 1

Return to Michael Sipser

Jump to: Michael Sipser Go

WHO IS ONLINE

Users browsing this forum: No registered users and 2 guests

Theorem 4.2

Theorem 4.2

by **sgreen** » December 16th, 2011, 8:11 pm

Now that I think I understand theorem 4.1, I am going to attempt to explain theorem 4.2 (Which is similar)

Theorem 4.2 states: ANFA is a decidable language

This is the same as 4.1 except instead of looking at deterministic finite automaton, it is looking at non-deterministic finite automaton

And the steps to prove this is to first convert the NFA to a DFA, then just run the new DFA on the string and get the answer!

That sounds too easy. Can you just convert an NFA to a DFA? Is it magic?

sgreen

Posts: 8
Joined: December 16th, 2011, 9:01 am

Re: Theorem 4.2

by **tmorelli** » December 16th, 2011, 8:27 pm

If you look at the proof in the book, he says to look at theorem 1.39 to convert an NFA to a DFA

We can define an NFA in a little more detail:
An NFA can have multiple paths for the same input at a given state. For example, state Q0 may go to state Q1 and Q10 for an input of 1.
In this case, the NFA replicates itself into 2 copies, one that goes to state Q1 and one that goes to state Q10
And this replication can continue until all copies of itself are completed.
Another aspect is the transition defined by "e". When this is encountered, it copies itself for all of the exit points that contain an "e" and also keeps a copy of itself at the current state.
Once all of these copies finish, if any one of them finishes in an accept state, then the entire computation accepts.

So it is like a bunch of threads running at once.

In order to make this into a DFA, we need to make sure all of our copies are represented in one big DFA. If you read through Theorem 1.39, that is all that is happening. It is taking the union of all of the possible paths, including copies, and that ends up in a big DFA with only 1 action for each symbol in the alphabet at each state.

Make a little more sense?

tmorelli
Site Admin

Posts: 10
Joined: October 26th, 2011, 2:28 pm

Re: Theorem 4.2

by **sgreen** » December 16th, 2011, 8:28 pm

yes it does thanks!

sgreen

Posts: 8
Joined: December 16th,

2011, 9:01 am

Display posts from previous: All posts Sort by Post time Ascending Go

POSTREPLY

3 posts • Page 1 of 1

Return to Michael Sipser

Jump to: Michael Sipser Go

WHO IS ONLINE

Users browsing this forum: No registered users and 2 guests

Board index

The team • Delete all board cookies • All times are UTC - 8 hours [DST]

Theorem 4.3

Theorem 4.3

by **tmorelli** » December 16th, 2011, 9:05 pm

And just to complete the trifecta, I will attempt to explain a little bit about Theorem 4.3 since it follows very closely 4.1 and 4.2

Theorem 4.3: AREX is a decidable language

This theorem tests whether or not a regular expression generates a given string.

The proof once again looks simple, because it refers to an earlier proof that converts a REX to an NFA. Since we already discussed the proof that ANFA is a decidable language (4.2), it is easy to see that once we convert the REX to an NFA we are good to go, but how is that done?

The book refers back to Theorem 1.54.

This has a lot of cross references. First what defines a regular expression?

tmorelli
Site Admin

Posts: 10
Joined: October 26th, 2011, 2:28 pm

According to definition 1.52, R is a regular expression if:
1. a for some a in the alphabet sigma
2. e
3. 0
4. (R1 U R2) where R1 and R2 are Regular expressions
5. (R1 o R2) where R1 and R2 are Regular expressions, or
6. (R1*) where R1 is a regular expression

And to define them in more detail:

2. e -> The language containing a single string - the empty string
3. 0 -> a language that doesnt contain any strings at all
4. (R1 U R2) -> The union of R1 and R2
5. (R1 o R2) -> The concatenation of R1 and R2 (the combination of R1 and R2)
6. (R1*) -> The star operator takes the combination of strings within R1 and combines them.

To convert a REX R into an NFA, you will need to convert each of the 6 components of an REX

1. R = a for some a in E. Then L(R) = {a}
2. R = e
3. R = 0
4. R = R1 U R2
5. R = R1 o R2
6. R = R1*

For some nice pictures, you can look at example 1.56 and example 1.58 They illustrate the above a little bit better than words.

And once you construct the NFA, then you can follow the proof in 4.2 to finish the proof of 4.3

Re: Theorem 4.3

by **ksimpson** » December 17th, 2011, 8:18 am

Wow! That is a long explanation for 3 lines out of the book!

Thanks!

ksimpson

Posts: 8
Joined: December 16th, 2011, 9:06 am

WHO IS ONLINE

Users browsing this forum: No registered users and 2 guests

Theorem 4.4

POSTREPLY Search this topic... Search

4 posts • Page 1 of 1

Theorem 4.4

by **tmorelli** » October 30th, 2011, 1:17 pm

tmorelli
Site Admin

Posts: 10
Joined: October 26th, 2011, 2:28 pm

Theorem 4.4 Introduction to the Theory of Computation 2nd Ed:
E DFA is a decidable language.

This is the theory that tests if an empty DFA is a decidable language. Also known as *emptiness testing*. It is used to determine whether a finite automaton accepts any strings at all.

First define some terms:

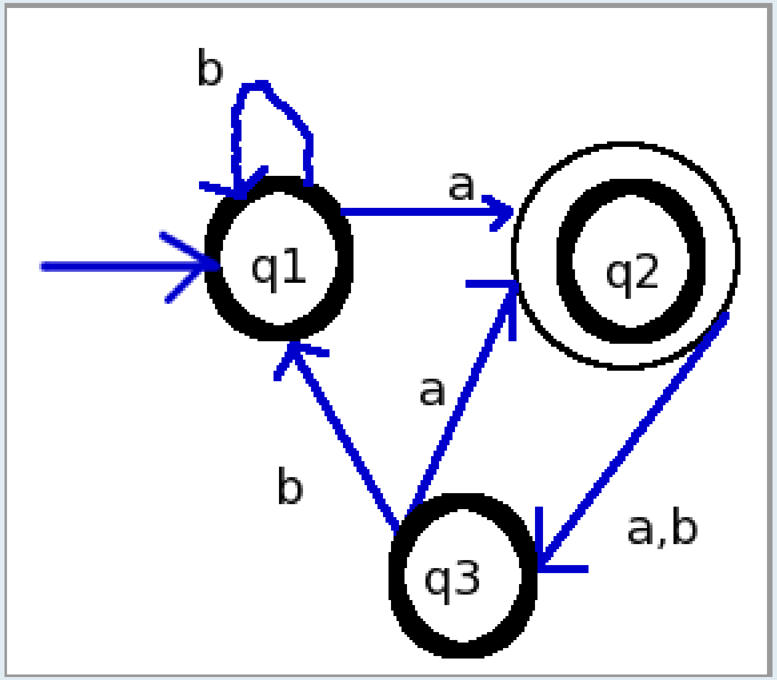
E DFA
Empty Deterministic Finite Automaton

decidable language
A decidable language is a language that will always halt. This is different than a recognizable language that may halt or loop.

First, construct a Turing Machine that views the given DFA A as a directed graph with vertices(states) and edges (transitions) and explore all paths from the start state in this graph. If it finds a path connecting the start state to some accepting state it accepts, otherwise if no such path exists, it rejects.

In other words -

1. Take a DFA
2. Find the start point
3. Traverse the DFA
4. When finished, if no accept states are marked, accept (this means it is empty), otherwise reject (not empty)



4.4.png (6.29 KiB) Viewed 15 times

This represents a modified version of example 3.23 where the graph is a directed graph.

In the above simple example, if the input is a, q1 and q2 are marked which means an except state is marked, so it would be rejected.
In the above simple example, if the input is b, only q1 is marked which means an except state is not marked, so it would be accepted.

Because it will either be accepted or rejected, it is a **decider**

Re: Theorem 4.4

by **jwilliams** » December 16th, 2011, 8:31 am

That is a really horrible picture. Did you draw that in paint?

jwilliams

Posts: 5
Joined: December 16th, 2011, 8:28 am

Re: Theorem 4.4

by **tmorelli** » December 16th, 2011, 8:32 am

Yes I did.. I am more concerned about the content. Sorry that I cant draw. I have good news for you! You can re-draw it and post it for everyone to see!

tmorelli
Site Admin

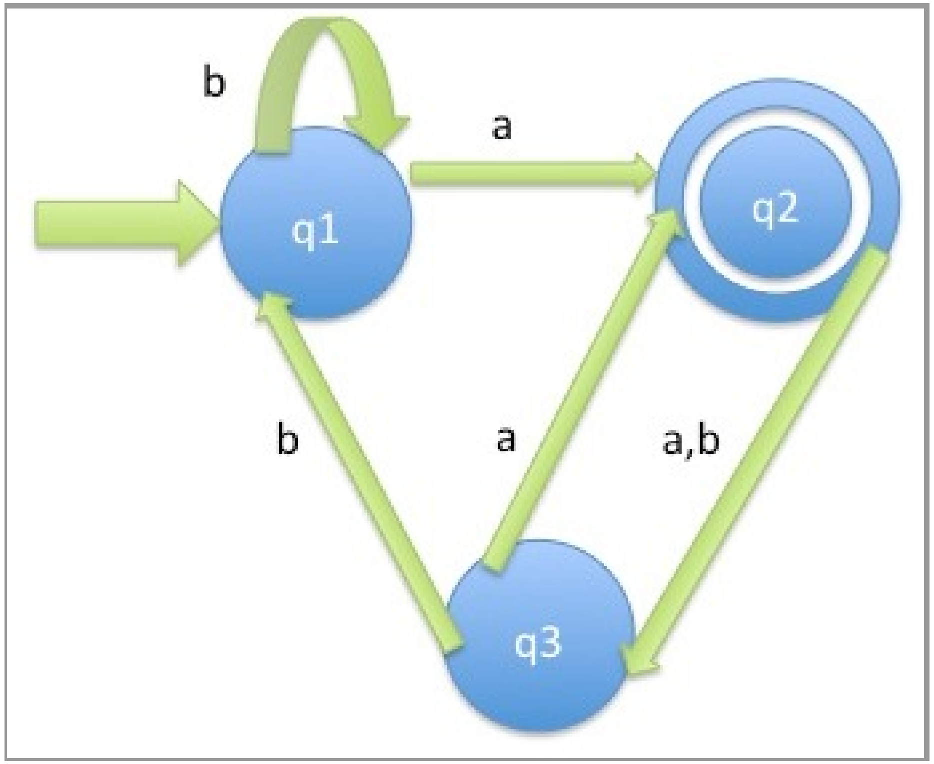
Posts: 10
Joined: October 26th, 2011, 2:28 pm

Re: Theorem 4.4

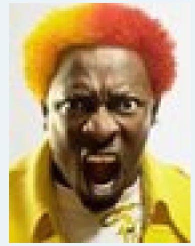
by **jwilliams** » December 16th, 2011, 8:45 am

OK, here is a better picture.

ATTACHMENTS



Presentation1.jpg (19.41 KiB) Viewed 5 times



jwilliams

Posts: 5
Joined: December 16th, 2011, 8:28 am

Display posts from previous: All posts Sort by Post time Ascending Go

POSTREPLY

4 posts • Page 1 of 1

WHO IS ONLINE

Users browsing this forum: No registered users and 2 guests

Theorem 4.5

Theorem 4.5

by **ksimpson** » December 18th, 2011, 4:13 pm

ksimpson

Posts: 8
Joined: December 16th, 2011, 9:06 am

Thanks for this forum! I'd like to contribute an explanation of Theorem 4.5 which can be found on page 169.

This is the proof that determines whether or not two DFA's can recognize the same language.

Specifically:

EQ DFA = {<A,B> | A and B are DFA's and L(A) = L(B)}

So that defines the equivalence problem, and the proof is to show that that is a decidable language.

The way the proof works is this:

The above statements refer to the two DFA's as A and B. You create a new DFA C. C is a DFA that accepts strings recognized by A and B, but not both. So it is everything that is not in the union of them

Then, if A and B recognize the same language, C will accept nothing. Since C only takes stuff in A or B but not both, if A=B there will be nothing left in C

L(C) = (L(A) intersect L(B)') U (L(A)' intersect L(B))

The L(A) intersect L(B)' is known as the symmetric difference
And looking closer, (L(A) intersect L(B)') - Everything in A combined with everything not in B. If A=B, this should produce nothing.
Same goes for (L(A)' intersect L(B))

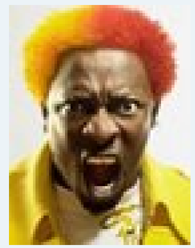
And if both are empty, the whole thing will be empty

Once C is constructed, you can use theorem 4.4 which was explained earlier to see if it is empty or not..

Re: Theorem 4.5

by **jwilliams** » December 18th, 2011, 4:14 pm

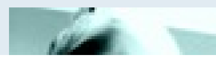
Close but not quite. Only post correct interpretations



jwilliams

Posts: 5
Joined: December 16th, 2011, 8:28 am

Re: Theorem 4.5



by **tmorelli** » December 18th, 2011, 4:15 pm

jwilliams -

This is for interpretations. I am not sure if there is such a thing as a correct or incorrect interpretation. And by the way, it looks correct to me

tmorelli
Site Admin

Posts: 10
Joined: October 26th, 2011, 2:28 pm

Re: Theorem 4.5

by **sgreen** » December 18th, 2011, 4:17 pm

Actually, I think I found a small issue with the explanation

> The above statements refer to the two DFA's as A and B. You create a new DFA C. C is a DFA that accepts strings recognized by A and B, but not both. So it is everything that is not in the union of them

I think that should read - "So it is everything that is not in the INTERSECTION of them". Everything not in the union is stuff that isnt in either, which really isnt useful!

sgreen

Posts: 8
Joined: December 16th, 2011, 9:01 am

Re: Theorem 4.5

by **ksimpson** » December 18th, 2011, 4:18 pm

Yes you are correct. That was either a typo or my fingers moving faster than my brain!

Thanks for POLITELY pointing that out!

ksimpson

Posts: 8
Joined: December 16th, 2011, 9:06 am

Display posts from previous: All posts Sort by Post time Ascending Go

POSTREPLY

5 posts • Page 1 of 1

Return to Michael Sipser

Jump to: Michael Sipser Go

WHO IS ONLINE

Users browsing this forum: No registered users and 2 guests

Board index

The team • Delete all board cookies • All times are UTC - 8 hours [DST]

Theorem 4.6

Theorem 4.6

by **tmorelli** » December 18th, 2011, 4:42 pm

It is exciting to see this board getting some use!

I am going to briefly describe Theorem 4.7

First, ACFG is defined as:
ACFG {<G,w>|G is a CFG that generates string w}

Theorem 4.7 states:
ACFG is a decidable language.

Decidable means that it will always halt.

Explanation:
Does G generate w?

tmorelli
Site Admin

Posts: 10
Joined: October 26th, 2011, 2:28 pm

-> Go through all the derivations to determine if any is a derivation of w.
- This takes too long
- There could be infinitely many derivations
- if G does not generate w, this algorithm would never halt
- A turing machine is a recognizer (halt or loop) but not a decider (halt only) for ACFG

-> To make the Turing Machine a decider we need to ensure that the algorithm tries only finitely many derivations
- This can be done by limiting the search space to the length of w

Problem 2.26 shows that if G were in Chomsky Normal Form any derivation of w has 2n-1 steps
- n is the length of w
- Since this is a finite amount, if we could do this conversion we would have a Turing Machine that could go through a finite amount of steps and thus will always halt.
- List out all the derivations. If any generate w, accept, otherwise reject

Here is how to convert ACFG to Chomsky Normal Form:
A->BC <- any variable except for start variables
A->a <- Terminal
S->e <- Start variable (optional)

Re: Theorem 4.6

by **ksimpson** » December 18th, 2011, 4:43 pm

What is a CFG, and can you explain what the steps to convert to Chmosky mean?

ksimpson

Posts: 8
Joined: December 16th,

2011, 9:06 am

Re: Theorem 4.6

by **sgreen** » December 18th, 2011, 4:53 pm

A CFG is:
Context Free Grammar

An example of a context free grammar is
A -> 0A1
A -> B
B -> #

A grammar has substitution rules, variables, and terminals. One variable is the start variable

A grammer describes a language by generating each string of the grammer.

So the a Grammer that is described above can generate the string
000#111

Can you see that?
First start with 0A1
Then you can substitute A with 0A1
so you end up with 00A11
And do it again..
000A111
Then we see that B can substitute for A

So we have 000B111
Then we see that be can be substituted with the terminator #
and we end up with 000#111

sgreen

Posts: 8
Joined: December 16th, 2011, 9:01 am

Each of these steps is called a derivation

So if you follow these rules you can convert to Chomsky normal form as defined above..

The specific steps may vary based on the specific grammar, but this is how you would solve the general case.

Re: Theorem 4.6

by **ksimpson** » December 18th, 2011, 4:54 pm

I think i get it. Thanks!

ksimpson

Posts: 8
Joined: December 16th, 2011, 9:06 am

Re: Theorem 4.6

by **jwilliams** » December 18th, 2011, 4:55 pm

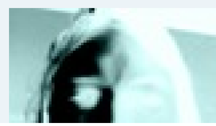
Why is the subject of this thread 4.6 when you are describing 4.7?

jwilliams

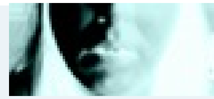
Posts: 5
Joined: December 16th, 2011, 8:28 am

Re: Theorem 4.6

by **tmorelli** » December 18th, 2011, 4:56 pm



I guess that was a typo. I am not sure how to change it. But to clarify, this is about 4.7



tmorelli
Site Admin

Posts: 10
Joined: October 26th, 2011, 2:28 pm

Display posts from previous: All posts Sort by Post time Ascending Go

POSTREPLY

6 posts • Page 1 of 1

Return to Michael Sipser

Jump to: Michael Sipser Go

WHO IS ONLINE

Users browsing this forum: No registered users and 2 guests

Board index The team • Delete all board cookies • All times are UTC - 8 hours [DST]

Powered by phpBB © 2000, 2002, 2005, 2007 phpBB Group

Theorem 4.8

Theorem 4.8

by **ksimpson** » December 18th, 2011, 5:13 pm

ksimpson

Posts: 8
Joined: December 16th, 2011, 9:06 am

Let me try this one out! Let me know if you interpret it differently!

Theorem 4.8 is the emptiness testing for the language of a CFG.

Is defined as:

ECFG = {<G> | G is a CFG and L(G) = 0}

It is trying to determine if whether or not the CFG can generate any strings at all is decidable (need to halt)

The way this one works is pretty neat. You create a Turing Machine (the exact steps are in the book). What this Turing Machine does is to work backwards. It first marks all symbols that can create the terminating symbol. Then it marks all symbols that can substitute with the symbols already marked off. It continues in this loop, essentially working backwards to the start symbols.

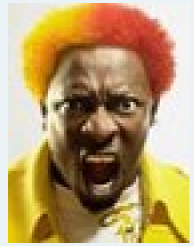
When it is complete, if any of the start symbols has been marked, then you accept. If not reject.

How does that sound?

Re: Theorem 4.8

by **jwilliams** » December 18th, 2011, 5:26 pm

Good luck finding more than one start variable



jwilliams

Posts: 5
Joined: December 16th, 2011, 8:28 am

Display posts from previous: All posts Sort by Post time Ascending Go

POSTREPLY

2 posts • Page 1 of 1

Return to Michael Sipser

Jump to: Michael Sipser Go

WHO IS ONLINE

Users browsing this forum: No registered users and 2 guests

Board index

The team • Delete all board cookies • All times are UTC - 8 hours [DST]

Theorem 4.22

Theorem 4.22

by **sgreen** » December 18th, 2011, 5:44 pm

sgreen

Posts: 8
Joined: December 16th, 2011, 9:01 am

I guess it is my turn now!

Theorem 4.22 investigates a Turing Co-Recognizable Language

The theorem states:
A language is decidable iff (if and only if) it is Turing Recognizable and co-Turing Recognizable

So basically, a language is decidable only when it and its compliment are Turing Recognizable

In order to prove this we will do the following:

1. Prove that if A is decidable, both A and A' are Turing Recognizable
This is proven by definition - Any decidable language is Turing-recognizable.
The complement of a decidable language is decidable

2. If both A and A' are turing recognizable, we let be M1 be the recognizer for A and M2 be the recognizer for A'

M is the decider for A

So then you run the input W on M, where M contains both M1 and M2
You run the input on M1 and M2 at the same time

If M1 accepts, accept. If M2 accepts, reject

M will accept all strings in A and reject all strings in A' so it is a decider!

Re: Theorem 4.22

by **ksimpson** » December 18th, 2011, 5:49 pm

I get it. So essentially you run the input on both a machine that accepts the language and another one that accepts its compliment

If the machine that accepts the language accepts, then we know its good. If the machine that accepts the compliment of the language accepts, then we know its bad. Since they are running in parallel the first one that gets to either of these states is correct and the other one can be stopped.

Pretty neat!

ksimpson

Posts: 8
Joined: December 16th, 2011, 9:06 am

WHO IS ONLINE

Users browsing this forum: No registered users and 2 guests

Theorem 4.23

POSTREPLY Search this topic... Search

2 posts • Page 1 of 1

Theorem 4.23

by **ksimpson** » December 18th, 2011, 6:04 pm

ksimpson

Posts: 8
Joined: December 16th, 2011, 9:06 am

This is actually a corollary -

ATM' is not Turing-Recognizable

We know that ATM is turing-recognizable, but what about its compliment?

The book refers us back to Theorem 4.11 which shows that ATM is undecidable

It is important because restricting ATMs to be deciders will limit the types of problems that can be fed into Turing Machines. This means that sometimes Turning machines may not halt, but that is OK

The point is that if ATM and its compliment ATM' are both Turing-Recognizable, then ATM would be decidable. Which is not correct, the Turing Machine must recognize inputs, not decide them.

Since that is the case, ATM' must not be Turing-Recognizable

Re: Theorem 4.23

by **tmorelli** » December 18th, 2011, 6:06 pm

It is also important to look at Theorem 4.22.

Sometimes it may be better to run multiple copies of the Turing Machines where some may not halt and others may halt. We can always kill off a turing machine that does not halt, but preventing it from running would be very bad. Just another reason why Turning Machines are Recognizers and Not Deciders.

tmorelli
Site Admin

Posts: 10
Joined: October 26th, 2011, 2:28 pm

WHO IS ONLINE

Users browsing this forum: No registered users and 2 guests