Gmail by Google

Theron Stanford <shixilun@gmail.com>

Fwd: Analogical Modeling

23 messages

Nathan Glenn <garfieldnate@gmail.com> Wed, Dec 14, 2011 at 4:47 PM
To: shixilun@yahoo.com

Hello Theron,
My name is Nathan Glenn and I work with Deryle Lonsdale. I'd like to do some work with the AM code; however, I am having a lot of trouble understanding it. I was wondering if you had a commented version of the code somewhere? I assume that all of the relevant stuff for calculating the output is in Parallel.xs. Also, I am very curious about this little note on lines 66-68 that says "ask me for details some time". Why would adding {} make the loop not work?
Regards,
Nate

Theron Stanford <shixilun@gmail.com> Thu, Dec 15, 2011 at 5:24 PM
To: Nathan Glenn <garfieldnate@gmail.com>

Hello, Nate.

Glad to see someone looking at the code again.

Assuming you have the complete tarball AM-Parallel.tar.gz, there should be two documents, doc/AM.pod and doc/lattice.pod, that give an overview to the AM algorithm and the background behind some of the tricks I use; they can be viewed using perldoc. (I use some higher mathematics that even most undergraduate math majors don't see.) There are also comments in lib/AM/Parallel.pm and Parallel.xs, of course.

There is an underlying assumption that you know how to write C code and that you have some familiarity with writing XS code to interface with Perl. The documentation pages perlxs, perlguts, and perlapi, which can be found at perldoc.perl.org, are the ultimate source; the book "Extending and Embedding Perl" (http://www.manning.com/jenness/) is a good resource to learn from.

Other than that, since I myself have a copy of the code, I can always give hints on what is going on if you point me to the line numbers.

As for lines 66-68, I don't remember exactly why putting the block inside braces makes it not work. All I know is that when I did so, I did not get the correct results. I don't remember if this was confined to the Microsoft compiler cl, or if it broke with gcc as well. To save stress on future coders, I put that comment in just to let them know that it broke for me when I did it, so they wouldn't waste their time.

Actually, seeing that you say that comment is at lines 66-68, while I see it at lines 230-232, it's clear that you do not have the complete tarball, so I am attaching it to this message.

Happy coding!

Theron
[Quoted text hidden]

AM-Parallel.tar.gz
38K

Theron Stanford <shixilun@gmail.com> Thu, Dec 15, 2011 at 5:35 PM

To: Nathan Glenn <garfieldnate@gmail.com>

Also, you might want to see my first attempt at combining Perl and C by reading the implementation explained in Skousen's "red book", which Dr Lonsdale can point you to.

On Wed, Dec 14, 2011 at 4:47 PM, Nathan Glenn <garfieldnate@gmail.com> wrote:
[Quoted text hidden]

Nathan Glenn <garfieldnate@gmail.com> Fri, Dec 16, 2011 at 3:03 PM
To: Theron Stanford <shixilun@gmail.com>

Thank you for this! Maybe I haven't navigated the site well enough; I thought that AM-Parallel-Unix was the full version.
[Quoted text hidden]

Theron Stanford <shixilun@gmail.com> Fri, Jan 20, 2012 at 3:37 PM
To: Nathan Glenn <garfieldnate@gmail.com>

So, how goes it?

Nathan Glenn <garfieldnate@gmail.com> Fri, Jan 20, 2012 at 4:25 PM
To: Theron Stanford <shixilun@gmail.com>

Pretty good, I think I understand most of the algorithms pretty well. I started by updating the Wikipedia page by finishing off the example section. I think it should also have a section on implementation because it's so neat.
I'm just trying to implement it all in Java right now, tiny piece by tiny piece (I know Java's slower, but it's my best language and the goal is understandable code and a Weka plugin, not lightning speed). I'm at the part where I need to fill the lattice, which seems rather tricky and will require more reading. Your chapter in the 'red book' is very helpful. My periodic questions have mostly been answered by that. Do you happen to know the license that the current AM distribution is released under? All it says is (C) Royall Skousen. If you would like to see my incomplete, junky code I can send you a link to it.
Nate

On Fri, Jan 20, 2012 at 4:37 PM, Theron Stanford <shixilun@gmail.com> wrote:
> So, how goes it?

Theron Stanford <shixilun@gmail.com> Fri, Jan 20, 2012 at 6:09 PM
To: Nathan Glenn <garfieldnate@gmail.com>

For filling the lattice, you might want to look at the earlier pure Perl implementations first to get a handle on it. The latest C code uses all sorts of tricks for speeding things up and minimizing memory usage (e.g., direct products of lattices -- my one and only application of graduate school knowledge), and since you're not currently interested in either... :) The last pure Perl implementation does a pretty good job at representing the entire lattice while at the same time not duplicating memory unnecessarily. I might even have an old handout explaining it somewhere...

AM::Parallel isn't released under any license at the moment. BYU paid me as a research assistant to Royal Skousen when I wrote it, so I don't know what license the BYU lawyers would want. We put (C) Royal Skousen mostly because we didn't know what else to put. (Of course, I would vote for Perl Artistic License.) If you really need to choose a license, I'd approach Deryle Lonsdale about it.

I tend to avoid Java code if I can help it, so I'll let you keep it to yourself for now :)

Theron

[Quoted text hidden]

Nathan Glenn <garfieldnate@gmail.com> Fri, Jan 20, 2012 at 7:29 PM
To: Theron Stanford <shixilun@gmail.com>

Yes the pure perl one may be easier to read, thank you. Just to make sure, the one on the site is AM-2.0-unix and am.pl is 385 lines long. Do you have a longer more commented version? Also, if this is pure perl then why are there separate versions for Unix, Windows, and Mac? Also, thanks for the book recommendation from earlier, I definitely plan to read it.
Nate
[Quoted text hidden]

Theron Stanford <shixilun@gmail.com> Sat, Jan 21, 2012 at 7:10 AM
To: Nathan Glenn <garfieldnate@gmail.com>

I do not have a longer commented version, but I can look around to see if I saved my separate handouts from when I explained the program to other members of the group.

There are separate versions to account for the different end-of-line conventions on the three operating systems and to archive the files according to the preferences of those operating systems.

Theron
[Quoted text hidden]

Nathan Glenn <garfieldnate@gmail.com> Tue, Jan 24, 2012 at 8:22 PM
To: Theron Stanford <shixilun@gmail.com>

Yes, if you find that pamphlet please send it to me.
[Quoted text hidden]

Nathan Glenn <garfieldnate@gmail.com> Thu, Feb 23, 2012 at 4:35 PM
To: Theron Stanford <shixilun@gmail.com>

Well, I'm having a lot of trouble figuring out how the lattice filling works. I'd like to implement it the way you wrote in the red book chapter; it seems like it would simplify pointer counting at the end.
Here's what I understand:
AM_SUPRA
[Quoted text hidden]

Nathan Glenn <garfieldnate@gmail.com> Thu, Feb 23, 2012 at 5:10 PM
To: Theron Stanford <shixilun@gmail.com>

-AM_SUPRA forms a linked list node.
-You store all of the AM_SUPRA instances in one circular linked list, starting with *supralist.
-The lattice is a 2^n-1 length array of AM_SUPRA*.
-When you add a subcontext to the lattice, you skip contexts that are heterogeneous, and you check for introduced heterogeneity and mark the corresponding AM_SUPRA accordingly.
-If there was no heterogeneity, the item is added to the AM_CONTEXT* in the corresponding AM_SUPRA

What I still don't get: why are all of the AM_SUPRA in a circular linked list? When do you increment baseindex (after an item is added?)? Do the pointers in the lattice point to a small number of AM_SUPRA that are being updated, or to separate AM_SUPRA?

In general I am just not understanding the structures and functions involved in adding a subcontext to the lattice.
Can you offer any help?
Nate

[Quoted text hidden]

Theron Stanford <shixilun@gmail.com> Fri, Feb 24, 2012 at 5:49 PM
To: Nathan Glenn <garfieldnate@gmail.com>

Nate:

OK, the problem might be this fundamental misunderstanding: The lattice is a 2^n-1 length array of *AM_SUPRA, but that doesn't mean I store 2^n-1 individual AM_SUPRA. That would take up too much room.

Look at the diagram on page 391. There are eight supracontexts, but only five *distinct* supracontexts. One is empty; one represents heterogeneity; the other three are homogeneous. So I have to store only *five* copies of AM_SUPRA, not one for each element of the lattice. (I'll have to look at the code in more detail later -- there might not be an actual AM_SUPRA for the common heterogeneous supracontext, which means only four copies along with a "bogus" AM_SUPRA pointer for flagging heterogeneity.)

The elements of lattice[] are then pointers to the AM_SUPRA. If two supracontexts in the lattice would be identical, I just make the pointers identical and create only one AM_SUPRA. Saves memory, no? It also makes counting at the end quick, because AM_SUPRA keeps track of how many times it appears in the lattice.

The AM_SUPRA are in a circular linked list for a number of reasons.

1. They are in a linked list so that I can traverse all of them. (It would be wasteful to traverse the lattice to find them all, because I would run into so many duplicates.)
2. They are in a linked list so that I can insert and delete supracontexts at will.
3. They are in a linked list so that when I create new supracontexts from old supracontexts by adding a new subcontext, I can place the new one immediately after the old one; this, coupled with the use of baseindex, helps me move the pointers in lattice[] around very easily.

The best advice I can give you is for you to find a very large sheet of paper and execute the code by hand using the 312 example. It shouldn't take long at all. As you create supracontexts and move pointers around, make sure each step is consistent with the corresponding step in section 2.2.

Theron
[Quoted text hidden]

Nathan Glenn <garfieldnate@gmail.com> Fri, Feb 24, 2012 at 7:34 PM
To: Theron Stanford <shixilun@gmail.com>

I think I get it now! Thanks!
Nate
[Quoted text hidden]

Theron Stanford <shixilun@gmail.com> Sat, Feb 25, 2012 at 4:14 PM
To: Nathan Glenn <garfieldnate@gmail.com>

Nate:

Here's a little more insight. (I had to dig into memory to remember why a *circular* list and exactly how baseindex works; I don't use the baseindex trick in AM::Parallel.)

First, to review: the circular linked list contains the minimum number of AM_SUPRA needed to completely describe the supracontextual lattice.

Now, suppose that part of the list looks like this:

[snip]

AM_SUPRA_11 contains subcontexts a b
AM_SUPRA_12 contains subcontexts x
AM_SUPRA_13 contains subcontexts x y
AM_SUPRA_14 contains subcontexts w
[snip]

Assume that 11, 12, 13, 14 are the indices of the supracontexts. Let's also assume that nextindex = 25.

Next, suppose we look at just three elements of the lattice, where for convenience I will use binary to represent the placement in the lattice:

lattice[0010] points to AM_SUPRA_12
lattice[0110] points to AM_SUPRA_12
lattice[0111] points to AM_SUPRA_13

Now, let's suppose I want to add subcontext z to 0010 and everything below it. First, I look at lattice[0010] and see that I have to add it to the list of subcontexts in AM_SUPRA_12. I do this by duplicating AM_SUPRA12, adding subcontext z, and placing the new supracontext immediately after:

[snip]
AM_SUPRA_11 contains contexts a b
AM_SUPRA_12 contains contexts x
AM_SUPRA_25 contains contexts x z
AM_SUPRA_13 contains contexts x y
AM_SUPRA_14 contains contexts w
[snip]

I label the new AM_SUPRA with nextindex. Then I set baseindex to 25; baseindex will now remain fixed while I am in the process of adding subcontext z to the other supracontexts. Then I increment nextindex to 26.

I change the pointers in the lattice:

lattice[0010] points to AM_SUPRA_25
lattice[0110] points to AM_SUPRA_12
lattice[0111] points to AM_SUPRA_13

So far, so good.

Now, I have to add subcontext z to everything below 0010. When I want to add it to 0110, I notice that lattice[0110] points to AM_SUPRA_12. I look at the supracontext after it, AM_SUPRA_25. I notice that its index is greater than or equal to the baseindex. This signals me that I don't have to create a new AM_SUPRA to represent lattice[0110]; it's already there. I just have to move the pointer.

Now the lattice looks like this:

lattice[0010] points to AM_SUPRA_25
lattice[0110] points to AM_SUPRA_25
lattice[0111] points to AM_SUPRA_13

So, let's add subcontext z to lattice[0111]. This points to AM_SUPRA_13. The supracontext after that is AM_SUPRA_14. 14 is not greater than or equal to the baseindex; this signals me that I have to create a new AM_SUPRA. So I duplicate AM_SUPRA_13, label it with nextindex, and add subcontext z:

[snip]
AM_SUPRA_11 contains contexts a b
AM_SUPRA_12 contains contexts x
AM_SUPRA_25 contains contexts x z
AM_SUPRA_13 contains contexts x y

AM_SUPRA_26 contains contexts x y z
AM_SUPRA_14 contains contexts w
[snip]

Then I change the pointer:

lattice[0010] points to AM_SUPRA_25
lattice[0110] points to AM_SUPRA_25
lattice[0111] points to AM_SUPRA_26

Using a *circular* linked list ensures that every AM_SUPRA has a successor with a valid index; I don't ever have to worry about dropping off the end.

Now, I have ignored some other details, like what happens when we introduce heterogeneity, or incrementing and decrementing the AM_SUPRA counts and cleaning out any AM_SUPRA that are no longer pointed to by anybody, but that's the general process.

Theron
[Quoted text hidden]

Nathan Glenn <garfieldnate@gmail.com> Sun, Feb 26, 2012 at 9:44 AM
To: Theron Stanford <shixilun@gmail.com>

Interesting. Thanks, this helps.
So, checking baseindex on the following AM_SUPRA is basically the same as checking data[data[0]] on the following AM_SUPRA? If it matches the subcontext you're adding, then baseindex should match, right?
[Quoted text hidden]

Theron Stanford <shixilun@gmail.com> Mon, Feb 27, 2012 at 9:41 AM
To: Nathan Glenn <garfieldnate@gmail.com>

Yes, that sounds right. I hadn't thought of checking it that way, but it would appear to work and reduce the size of AM_SUPRA to boot. Using baseindex at first seemed clever, then too clever and overblown; in AM::Parallel I use a field called touched, but then I have to remember to clear it after each addition of a subcontext. (Since I go through and clear out unused AM_SUPRA anyway, it's not a big deal.)

Just be careful about one thing: the empty supracontext has data[0] = 0 and data[data[0]] = 0, so you have to make sure you don't trip over yourself if you add a subcontext 0 (which could happen if the given is in the data set).
[Quoted text hidden]

Nathan Glenn <garfieldnate@gmail.com> Thu, Mar 1, 2012 at 7:17 PM
To: Theron Stanford <shixilun@gmail.com>

Thanks for all your help! I got a version working. There were some tweaks here and there but mostly it follows the 2.1 implementation.
When I'm done configuring, cleaning, modularizing, etc., I'd like to do the 2.2 implementation with the lattice intersection. I was wondering if you had a reference on lattice intersection; I think I understand how it works, but it would be nice to be able to cite something.
Nate
[Quoted text hidden]

Theron Stanford <shixilun@gmail.com> Fri, Mar 2, 2012 at 4:31 PM
To: Nathan Glenn <garfieldnate@gmail.com>

Nate:

Congratulations!

Here are some random references from a quick Google search:

https://www.google.com/search?q=direct+product+of+lattices&ie=utf-8&oe=utf-8&aq=t&rls=org.mozilla:en-US:official&client=firefox-a

http://books.google.com/books?id=nIlklfHgySQC&pg=PA165&lpg=PA165&dq=direct+product+of+lattices&source=bl&ots=WuRwtqXh3L&sig=Pq68H_bnJ8GGQMA2dWDL1QuQKG8&hl=en&sa=X&ei=P2NRT-TcDYjzggeD5-jiDQ&ved=0CDQQ6AEwAg

http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cts=1330734467467&ved=0CEgQFjAE&url=http%3A%2F%2Fwww.uv.es%2FEBRIT%2Fmacro%2Fmacro_5000_11_44.html&ei=P2NRT-TcDYjzggeD5-jiDQ&usg=AFQjCNF47wK2gQo7ur2NQROg89t48yqE-g&sig2=CNBsWVJblCkYGWgmabYvKg

http://en.wikipedia.org/wiki/Lattice_%28order%29

http://www.math.uwaterloo.ca/~snburris/htdocs/ualg.html

Remember, this was math from grad school I was using...not easy to find basic materials! And don't forget the POD in the AM-Parallel distribution.

Theron
[Quoted text hidden]

Nathan Glenn <garfieldnate@gmail.com> Thu, Jul 5, 2012 at 9:18 AM
To: Theron Stanford <shixilun@gmail.com>

Hi Theron,
I was wondering if you could tell me anything about outputting gang effects? I know what a a gang effect is (lots of words with a little effect put together have a big effect), and I believe that you wrote code to output gang effects, but I haven't seen much documentation on it, except a note that says the total number of possible outcomes is used to compute them. How does it work?
Nate
[Quoted text hidden]

Theron Stanford <shixilun@gmail.com> Mon, Jul 9, 2012 at 7:00 PM
To: Nathan Glenn <garfieldnate@gmail.com>

Nate:

Printing gang effects is simple; when you create the job, you just specify -gangs => 'yes' for the full printout and -gangs => 'summary' for the summary (same way you specify -commas). Try printing it to see what it looks like...I worked very hard to get it all lined up!

Gang effects are computed at the very end of Parallel.xs. They end up in the hash %gang, whose keys are contexts and whose values are the size of the gang effect. This variable is described (briefly) near the top of Parallel.pm.

Gang effects are printed by the code in Parallel.pm between the lines

begin gang

and

end gang

Let me know what else I can help with.

Theron
[Quoted text hidden]

Nathan Glenn <garfieldnate@gmail.com> Sat, Jul 14, 2012 at 5:55 AM
To: Theron Stanford <shixilun@gmail.com>

Thanks! Looks like a gang effect equals the number of pointers in a subcontext (in the quadratic case) times the number of exemplars in that context. This is pretty similar to outputting the analogical set, which does the same thing except on a per-exemplar basis (right?).
Nate
[Quoted text hidden]

Theron Stanford <shixilun@gmail.com> Sat, Jul 14, 2012 at 8:36 AM
To: Nathan Glenn <garfieldnate@gmail.com>

Yes, that sounds right.
[Quoted text hidden]