1. The Coherence-Based Genealogical Method, or CBGM, has a reputation of being a “black box” in Münster whose inner workings are only divulged to those initiated into the mystery. My goal today is to show you that this isn’t true, but the easiest way to start is to admit why I think people see the CBGM this way. [0:20]

2. As Peter Gurry has pointed out in the beginning of his thesis, Gerd Mink’s work on the method started in the late 70s as part of an effort to apply computers to New Testament textual criticism. Even as the method matured, this aim did not change. The CBGM was developed to be programmed on a computer, and I think the easiest way to understand it is from a programming perspective. The books and papers listed here are great for learning about the method, but for me, the best way to learn how it actually worked was to program it myself. A few years ago, I reverse-engineered the CBGM and made the results available as open-source software. Most of you probably don’t have the free time I had to do this, but my hope is that I can walk you through what I discovered in an hour or less. [0:45]

3. The problem the CBGM was designed to solve is *contamination*, or mixture, in the manuscript tradition. Here we have a classical stemma representing a reconstruction of the tradition. The nodes with solid borders are the manuscripts we still have, the ones with dashed borders are their reconstructed lost ancestors, and the edges represent lines of descent. With just the black edges, we have a simple family tree going all the way back to the latest ancestor of the surviving tradition (little *omega*) and from there to the text of the author, as best as we can reconstruct it (big *omega*).

But as often happens, real life isn’t so simple, and at different points, scribes copying the New Testament must have combined more than one source. They might have done this by remembering something from a source other than their exemplar, by copying words from the margin of their exemplar into the text, or by reading two exemplars side-by-side and copying the readings they liked. The red edge here illustrates this secondary influence. Unfortunately, this throws a wrench into the classical process of reconstructing the family tree, and the CBGM proceeds from the assumption that this kind of reconstruction is impossible. [1:23]

4. Consequently, the CBGM starts by assuming that we are just going to work with the witnesses we have. This means that a family tree constructed with the CBGM will consist entirely of familiar faces. There’s only one exception—the initial text. It’s conventionally denoted by A, for the German *Ausgangstext*, but if you like the family tree metaphor, you can remember it as A for [click to next slide] “Adam.” Functionally, it’s just a placeholder for all the readings that we are confident were the earliest, and through the process of the CBGM, we will attempt to fill it out as best as possible. In other words, much like Michelangelo, we must sketch out our best approximation of Adam in light of the observable features of his descendants.

The CBGM also has other methodological assumptions, which are so foundational that they get their own numbered list. There’s no quiz at the end of this talk, but if there were, these would be on it. One: we assume that scribes preserved the text more often than they changed it. (Otherwise, the whole tradition is plunged into chaos.) Two: we assume that every scribe’s change came from somewhere; new readings don’t come from thin air. Three: we assume that scribes used fewer sources rather than many. (This is sometimes called *parsimony*). And four: we assume that scribes used sources similar to what they produced, because there’s no reason to assume errors where they are not necessary. [1:37]

5. Some of these assumptions set the CBGM apart from previous methods. But it’s also important to clarify what the CBGM is *not*. It’s not really a new method of textual criticism unto itself, because it doesn’t tell us how we should evaluate readings or manuscripts. Instead, it is a meta-method that we can fit on top of our preferred approach for weighing the evidence. Consequently, the CBGM is also not a way for computers to do the critical work for us; it simply uses them to aggregate our judgments across the whole text for all witnesses, so we can better see the big picture that our textual judgments end up painting and adjust those judgments if the picture doesn’t look right. [click to next slide] Looking at you, buddy. [0:45]

6. Anyway, with the preliminaries out of the way, here’s our map for the rest of the talk. I’m going to detail the functional elements of the CBGM, starting from the top with the collation that we feed to the method and working my way down to the global stemma, which is essentially the CBGM’s version of a textual family tree.

The advantage of this functional outline is that it doubles as a process chart for using the CBGM. The edges highlighted in blue are the steps of what’s called the “iterative workflow,” which is the central process by which we refine our initial judgments about the text. More on that later. [0:37]

7. To start, you’re probably already familiar with the idea of a collation. If not, it’s very simple: you take the texts of all your manuscripts, line them up as best as possible, and identify the places where they differ, which we call *variation units*. This process is illustrated nicely in Reuben Swanson’s horizontal apparatus of Luke. I’ve highlighted and numbered the variation units for convenience. By definition, each variation unit consists of at least two variant readings, with the manuscript evidence divided between them. [0:35]

8. The CBGM is not concerned with places where the entire tradition agrees, so variation units are its basic points of comparison for the texts found in manuscripts. From the perspective of the CBGM, collation data can be thought of as a table, with witnesses as the row labels and variation units as the column labels. [0:23]

9. On the programming side, encoding a collation so it can be parsed by a computer is easy; the Text Encoding Initiative, or TEI, has developed an XML schema that allows us to capture virtually all of the features we would need to include in a collation, such as information about the published source of the data, lists of textual witnesses, variation units, and variant readings. [0:25]

10. Let’s zoom in on a single variation unit for more detail. Reading 1 is the majority reading ελεγεν, but for the sake of space, the only witness I’ve included in its list is the Family 13 group. Reading 1-f1 is a subreading of this reading, because it’s a “defective” form of that reading; specifically, there’s damage to the page, but we can still tell what the reading must have been. The lacuna, or gap, at the start of the word and the uncertainty about letters resulting from this damage are encoded with the appropriate TEI XML elements. Finally, reading 2 is ειπεν, found only in Family 1.

The CBGM would parse each variation unit like this one into a data structure with several components. One such component is a *support dictionary* like the one depicted here in pseudocode, which allows us to look up the reading of a witness by the witness’s unique ID efficiently. [1:00]

11. An even more crucial component of a variation unit is the *local stemma* of its variant readings. The local stemma is an elementary piece of the CBGM, as it is the method’s most basic unit of comparison. It organizes variant readings according to our critical judgments of their origins, and thus it gives us a measure of the historical priority between readings. In short, the local stemma is a visual representation of how we view the history of readings at a given variation unit. It is “local” in the sense that it does not concern what happens in any other unit and because it reflects the approach of Kurt Aland’s “local genealogical” principle. So in this first example on the left, we might judge reading *b* to be posterior to *a*, either on the preponderance of manuscript evidence or due to the transcriptional argument that scribes could easily drop a single-letter word. In the example on the right, we might say that reading *a* is most likely original because the other two readings are best explained as independent attempts to strengthen its connection to the preceding context. [1:13]

12. Naturally, some local stemmata get more complicated. In addition to having more substantive readings to judge between, we can also have multiple types of readings. In the example from Swanson’s collation, P75 had a *defective* reading. Other types of defective readings include obvious misspellings and errors, like reading *af* in this example. Typically, when these readings can be explained as having their source in a substantive reading, there is no reason to treat them as distinct; in practice, we would treat them as agreeing with their parent reading, as the dashed line from *a* to *af* indicates. *Orthographic* readings, which involve regional differences in spelling that are common enough not to be considered misspellings, also occur, and they may or may not be treated as substantive. The CBGM also supports *splitting the attestation* of a reading like *d* into two or more copies when the manuscript evidence suggests that the reading arose on different occasions independently. But to keep things simple, we might prefer to collapse split attestations into the same reading instead. Finally, *ambiguous* readings like *zw-a/b*, which typically result from damage so extensive that we can’t tell which reading was originally present, can be treated as full lacunae, in which case they’re just ignored, or they can be treated as agreeing with any of their potential readings. [1:37]

13. But at its heart, a local stemma is really simple to encode. It’s just a directed graph. Readings are the nodes and their relationships are one-directional edges. Such a structure is easily accommodated in TEI XML using the elements shown on the right. [0:20]

14. Determining the relationship between two given readings can be reduced to a few computational tasks, all of which can be done with quick lookups if we preprocess the graph ahead of time. First, we check if there is a path of length 0 from either reading to the other. If so, this means that the readings are identical or that one is a subreading that we’ve collapsed into the other; either way, the readings are considered to agree. If there is a path of length greater than 0 from one reading to the other, then we say that the first reading is *prior* to the second one, or conversely, that the second is *posterior* to the first. If no path exists between the readings, we check if there is some other reading that has a path to both. If such a reading exists between our readings, then we know that they developed independently and we can classify them as having no relationship.

In this example, reading *a* is prior to readings *b*, *c*, and *d*. Readings *b* and *c* have no relationship, and likewise for readings *c* and *d*. [1:10]

15. In some cases, like the example from 3 John 13 below, we may not know the source of some variant readings at first, so we leave them disconnected from the rest of the local stemma. In these cases, such readings—*e* and *f* in this case—have no common ancestor with any other reading, and thus they are said to have an *unclear* relationship to all the others.

The last potential relationship two readings can share, and perhaps the most important one, is when one reading *explains* another. This can occur in one of two ways: if the first reading agrees with the second one, or if it is directly prior to it, meaning that it has one edge or a path of length 1 to it. This relationship will be very important later, so keep it in the back of your mind.

The final note here is probably obvious, but it’s an easy detail to forget in implementation, so I’ll mention it: since lacunae are not treated as readings in the local stemma, they can’t relate to other readings, and other readings can’t relate to them. [1:15]

16. Now we can turn to the other side of a collation: witnesses. For the purposes of the CBGM, a *witness* is nothing more than a sequence of variant readings, like any row in this collation table. Remember when I told you earlier that a key assumption of the CBGM was that we use extant witnesses instead of hypothetical ancestors to reconstruct a family tree? Well, there are many times when a younger manuscript preserves an earlier text than an older one. When this happens, it’s probably because that manuscript copied the text of an earlier exemplar that is now lost. But we can’t reconstruct that exemplar, so the best thing to do is treat the younger manuscript as a proxy for it. And for this to work, we have to divorce texts from the physical artifacts that bear them. [0:49]

17. This separation of texts and artifacts has always been tacitly assumed for non-manuscript witnesses. A version, or translation, has its own tradition potentially consisting of many manuscripts in a different language, but when it is cited as a witness, the whole tradition is usually abstracted to a singular source, with its readings translated as best as possible back into Greek. The same situation holds for citations from church fathers. A patristic writing may be preserved in hundreds of manuscripts, but what we care about it is what that church father originally wrote, so a patristic witness usually consists only of the text quoted by that father, to the extent that we can confidently infer it.

In theory, then, there is no reason that versions or patristic works could not be treated as witnesses in the CBGM. But in practice, they have not been used this way, for different reasons. Versions, for instance, often cannot be retroverted precisely. Sometimes this is due to the language of the translation itself: in Latin, there is no article, so we have no way of knowing whether its Greek source had the article or not at the beginning of 3 John. Other times, it has more to do with translation method: the Philoxenian Syriac version translates too loosely for us to know for sure how it handled minor things like this, while the Harklean Syriac version, which was like a seventh-century Syriac NASB, is wooden enough for us to know what its Greek source read. As a result, versional witnesses will likely have ambiguous readings at many variation units. For patristic citations, the problem is usually that most comments are not precise enough to count as quotations, and precise quotations are so few and far between that a patristic witness would effectively be too fragmentary to be useful. So in what follow, we’ll focus primarily on manuscript witnesses. [2:10]

18. Much of the CBGM’s automated work consists in calculating *genealogical relationships* between pairs of witnesses. This is important, so I’m going to repeat it verbatim from the slide: The relationship of two witnesses is the overall pattern *of the relationships of their readings* where both are extant. So if we have two witnesses W1 and W2, and either one of these has a lacuna in this first variation unit on the left, then we would skip that unit when we add everything up. But if that’s not the case, then W1 would have one agreement with W2, one prior reading, one posterior reading, one unrelated reading, and one unclear relationship. The first and third readings would be ones where W2’s reading explains W1’s reading—in the first, by agreement; in the third, by descent.

When we tally up the different kinds of reading relationships, we also need to keep track of what we’ll call the *cost* of the relationship between W1 and W2. This is calculated as the number of explained readings that are explained by descent instead of agreement. So in the example here, the relationship of W2 relative to W1 and vice-versa would both have a cost of 1. This concept will be important later on, as it will help us gauge how well one witness serves as a parent of another in the family tree. [1:40]

19. For the purposes of implementation, it is extremely useful to encode these different types of relationships using *bitmaps*, or arrays of zeroes and ones with a position for each variation unit. Computer hardware is optimized for combining such arrays and counting the number of ones they contain, which will allow us to accelerate one of the most complicated procedures of the CBGM later. [0:28]

20. Comparing all collated witnesses, on the other hand, is quite simple, but it takes time. Say we have *n* witnesses and *d* variation units in our collation. (The *d*, if you’re wondering, stands for the *dimension* of our data.) Then to calculate all of the genealogical relationships between these witnesses, we have to do something on the order of *n*2*d* comparisons of readings. If we have a hundred witnesses and a hundred variation units, then this is already on the order of one million comparisons, which, even for a computer, takes a minute or two. The good news is that this only has to be done once each time we load up our collation and local stemmata into the CBGM, and it can easily be parallelized so that different cores of a computer can divide up the task and finish it faster.

To give you an idea of what genealogical comparisons look like with real data, I’ve included a screenshot of the compare\_witnesses module, which prints out a table of genealogical relationship data relative to a given witness. In this example, the base witness is GA 5, which has readings in 116 units in 3 John. The witnesses being compared to it appear in the leftmost column. Their general relationship to 5 appears in the second column (so, for example, GA 623 is generally prior to 5). The NR column gives the ranks of other witness as potential ancestors of 5; I’ll talk about what this means in the next slide. The PASS column tells us the number of units where both 5 and the other witness are extant. The EQ column gives the number of agreements. The next two columns count the number of readings where 5 is prior and posterior, respectively. The next two columns count the number of places where 5 has no relationship to the other witness or the relationship is unclear. The second-to-last column gives the number of units where 5’s reading is explained by the other witness’s reading. The last column gives the cost of these explained readings. Note that the cost is only calculated for potential ancestors, because it only needs to be known for these witnesses. [2:17]

21. So what are potential ancestors? Simply put, one witness is a *potential ancestor* to another if it has more prior readings than posterior readings relative to the other witness. The importance of this concept is that it allows us to arrange witnesses in a strict ordering, so we don’t have branches on our family tree that go in circles. We can picture this ordering of witnesses as a spectrum from early to late texts. The dashed lines connecting potential ancestors to potential descendants are very numerous, but that is simply because they are *potential* relationships; in reality, witnesses will typically have no more than a few ancestors in the family tree. [0:47]

22. Now that we know how to determine genealogical relationships between witnesses and identify a given witness’s potential ancestors, we can take advantage of the CBGM’s namesake—its ability to evaluate *coherence*. The tool we use to do this is called a *textual flow diagram*. A textual flow diagram operates at the level of a single variation unit, just like a local stemma, but it relates witnesses instead of readings. An example for a variation unit in 3 John 13 is pictured at the bottom here. If you’ve read anything on the CBGM before, chances are that you’ve already seen one of these.

You probably can’t make out all of the details in this example textual flow graph. This is because I’m using a 4-3 aspect ratio for these slides and don’t have room in them for all my pictures. But more importantly, it’s because textual flow relates *all* of the witnesses in our collation.

This is important because, as I mentioned earlier, witnesses are the sums of their readings. And when we’re not sure about the priority or sources of readings like *e* and *f* on their own merits, textual flow helps us figure that out based on the overall relationships of the witnesses containing these readings.

Before we get into the details, there’s one other point that needs mentioning, because others have gotten this wrong before: *a textual flow diagram is not the same thing as a global stemma*. What we ultimately want is a family tree of witnesses, and this may look just like a family tree, but I repeat: *this ain’t it*. I’ll explain why when we talk about the global stemma. [1:38]

23. As it does for most things, the CBGM has a procedure for constructing a textual flow diagram. The nice thing is that it works one witness at a time; you get a piece of the graph for each witness, in whatever order you want, and then you combine all the pieces to get the full graph. The piece that we want for each witness is its *textual flow ancestor*. In visual terms, it’s the witness in the textual flow graph that connects to the current witness.

To find it, we first have to set what’s called a *connectivity limit*, usually denoted by the letter *kappa* or *k*. This parameter is just a way of saying how similar we want two witnesses to be for one to count as the other’s textual flow ancestor. For each witness, the pool of candidates is its list of potential ancestors, which I talked about a couple slides ago. We sort this list in order of similarity to the current witness. Then we start from the top at the closest potential ancestor and work our way down the list until one of two things happens: either we find a potential ancestor with the same reading as the current witness, or we get to the *k*th-closest potential ancestor without finding one with the same reading. In this second case, we just pick the closest potential ancestor with any reading in this unit. In the textual flow diagram, the edge from the ancestor to the descendant will be blue, implying a change in reading. These are the most important edges in the graph, because they give us an idea of where the witness’s reading might have come from.

If you remember from a few slides ago, the genealogical relationships that determine a witness’s potential ancestors are *global*; they are determined based on all the variation units. In a textual flow diagram, we’re using global patterns of similarity and priority to better understand what’s happening *locally*, between the readings in a single variation unit. [2:15]

24. As you’ve seen, the full textual flow diagram can be pretty bulky. Sometimes, you may only care about seeing how the witnesses to a specific reading are related. In this example, we have the subgraph of the textual flow diagram with just the witnesses that have reading *a*. In this case, all of the witnesses with this reading trace their origin back to a single ancestor, so we would say that reading *a* has *perfect coherence*. If, within the connectivity limit, some of them could only descend from witnesses with a different reading, then we would say that reading *a* has *imperfect coherence*.

The labels on the edges here show why the connectivity limit matters. The edge from the initial text to 180 is labeled 7, meaning that none of 180’s six closest potential ancestors shares its reading here. If we set the connectivity limit to, say, 5, then this connection would be severed and 180 would have to descend from a witness with a different reading.

The connectivity limit is not one-size-fits-all; it depends on the nature of the variant readings. If a secondary variant reading was easy for multiple scribes to reach independently, then imperfect coherence would not be unusual, and a narrow connectivity limit may be acceptable. For a less accidental secondary reading, we would expect perfect coherence, so we might set the connectivity limit high to accommodate readings obtained by mixture. [1:40]

25. This offers us a way to check whether certain readings are more or less likely to be the initial one. As an example, let’s say we set a new reading—reading *b* in this example—as the initial reading in the local stemma. If we get imperfect coherence in its textual flow diagram, then this would suggest that at some point, the initial reading got lost and a later scribe independently stumbled back onto it. This is generally considered unlikely, unless the initial reading happens to be intrinsically really smooth or easy to stumble upon. [0:33]

26. On the other hand, if we get perfect coherence for reading *b*, as we do here, then the reading may be suitable as the initial one on the grounds of coherence. The next thing we would want to check in this case is how good or bad the coherence of other readings becomes under the assumption that *b* is the initial one. [0:22]

27. Here’s the textual flow diagram for reading *a* under the assumption that *b* is initial. This, too, has perfect coherence, so it doesn’t pose a problem. In practice, we would want to do this for every other reading to see if it has worse coherence with *b* as the initial reading than it does with *a* as the initial reading. [0:22]

28. Textual flow has other uses, too. An even more important one is that it can help us explain an unexplained reading based on its witnesses’ textual flow ancestry. Here’s our familiar example again. To help us find the readings that may have given rise to readings *e* and *f*, we can use another variation on the textual flow diagram called the *coherence in variant readings* graph. [0:30]

29. This graph consists only of the edges in the textual flow diagram where the reading changes from ancestor to descendant. For the descendants in these cases, the graph also includes any other potential ancestors within the connectivity limit that have other readings, so we can compare different hypotheses about which readings gave rise to which.

When making these comparisons, it is often helpful to highlight the *flow strengths* from ancestors to descendants, which is a measure of how stable the priority of one witness over another is. It can be calculated in different ways, but probably the most straightforward way is to take the number of the ancestor’s prior readings, subtract the number of its posterior readings, and divide this difference by the number of its extant readings. So it will be a number between 0 and 1. A flow strength close to 0 (indicated by a dotted line in this graph) warns us that if we change too many local stemmata, we could reverse this ancestor-descendant relationship. A high flow strength means that the direction of the relationship is more secure, but if it’s really high (like in these thick lines), then that means that the ancestor is distant rather than close. Generally, we want something in the middle, like these thin, solid lines. The rank of the textual flow ancestor—the number labeling the edge—is also informative; a closer ancestor with a low rank is better, while a more distant ancestor with a high rank is worse.

With local stemmata like this one, it’s helpful to weigh the evidence of coherence with any internal evidence we have for the readings themselves. Reading *e*, for example, has the aorist infinitive γραψαι, so it is more likely related to another reading that has the aorist than to one that has the present. This elevates readings *c* and *d*. As it turns out, the best witness with reading *e*, 0142, does not have any textual flow ancestor with reading *d*, but its closest ancestor, 025, has reading *c*, so between coherence and internal evidence, reading *c* is looking pretty good as the source of reading *e*.

Reading *f* is harder, because we don’t have the luxury of internal evidence to help us. The representative witness here is 61. Reading *a* is probably out, as the closest potential ancestor is the initial text, and given its flow strength, it’s probably too distant. Readings *cf* and *e* aren’t strong contenders, either, because the closest potential ancestors of 61 with these reading is its tenth closest and eighth closest, respectively. That leaves readings *b*, *c*, and *d*. All three are attested in close potential ancestors, but the ancestor with reading *b* is the closest, so we’ll go with that. [3:30]

30. So here’s the result of our sample round through the CBGM’s iterative refinement process. We did it! We now have a local stemma with all secondary readings explained. If we wanted to build a family tree in the form of a global stemma, then we would repeat this process for every unit where we have an incomplete stemma. [0:25]

31. We now turn to the last and most important part of the CBGM process: the part where we construct the *global stemma*, a family tree of witnesses that’s robust to contamination. Like a textual flow diagram, the global stemma is built up one piece at a time for each witness in any order. The piece of the global stemma for a particular witness is called its *substemma*, and it consists of the witness’s ancestors in the global stemma. Much like textual flow ancestors, stemmatic ancestors must come from the witness’s pool of potential ancestors.

To explain how we find the substemma for a witness, I’m going to need you to think back with me to when I was talking about the different kinds of genealogical relationships witnesses can have—specifically, how one reading *explains* another if it agrees with the other reading or if there’s one edge in the local stemma from it to the other reading. A crucial property of any witness’s substemma is that *every* extant reading in the witness must be explained by a corresponding reading in at least one of its stemmatic ancestors, illustrated here. You can also see here, if we represent the *no*s by zeroes and the *yes*es by ones, why it’s good to use bitmaps for this; combining the explained reading arrays is very fast with bitmaps.

What’s implied here is that a witness can have more than one ancestor in its substemma. And in fact, in many cases, more than one is needed. Because it accounts for all variant readings and allows witnesses to inherit them from different sources, this is precisely how the CBGM accounts for contamination. [1:40]

32. Since most witnesses have many potential ancestors, they will often have multiple valid substemmata. But not all substemmata are created equal.

This is where two of the CBGM’s methodological assumptions are very relevant. Assumption 3, a.k.a. the assumption of *parsimony*, means that it will usually be our best bet to pick a substemma with as few ancestors as possible. Assumption 4 tells us that we should prefer a substemma with ancestors that agree with the descendent as much as possible. Back when I was telling you about genealogical relationships between witnesses, I defined the *cost* of a genealogical relationship as the number of explained readings that aren’t agreements. In terms of this definition, we should prefer a substemma with the lowest total cost from the ancestors to the descendant.

If all the candidate substemmata are equally parsimonious, then the decision is easy: pick the one with the lowest cost. But sometimes we have to choose between a substemma with fewer, but more distant ancestors, like the one with the lone ancestor L938, and a substemma with more ancestors, but ones closer to the witness, like any of the others pictured here. The lesson is that even in this step of the CBGM, where computers do most of the heavy lifting, human judgment is still indispensable. [1:40]

33. That said, I need to explain why and how computers do most of the work here. Let’s start with the *why*. The process of *substemma optimization*, which is just a fancy term for finding the most promising candidates for a witness’s substemma, is technically difficult. It’s actually an instance of the *weighted set cover* problem, which asks us to find a combination of sets—potential ancestors in this case—that cover a series of target elements—that is, explain the witness’s readings at all variation units—such that the total cost of the sets is minimized. In the example here, there are two singleton sets that don’t cover all the readings and are therefore not *feasible* solutions; of the five other feasible combinations, only one, the one highlighted in green, is *optimal*, having the lowest cost.

This is a prime example of what is called an *NP-hard* problem in computer science. I don’t have time to get into the theory here, but a good short description is that NP-hard problems are problems not known—or believed—to admit a fast general-purpose algorithm for solving them.

So how do we do this? You might think that trying every possible combination of the witness’s potential ancestors and picking the one that explains all readings and has the lowest cost would work, but this won’t be practical; the number of substemmata we can make with a given number of potential ancestors grows exponentially with the number of potential ancestors, so for just thirty potential ancestors, there are already more than one billion possibilities. This quickly becomes prohibitive even for a computer.

Peter Gurry has suggested trying all combinations with one potential ancestor, then all combinations with two potential ancestors, and so on, until we find a feasible solution. This is guaranteed to find the most parsimonious solution, but it may not have the lowest possible cost; recall from a few slides ago that a substemma with two close ancestors can be preferable to one with a single more distant ancestor. Also, if a witness requires many stemmatic ancestors to explain its readings, then this procedure will take longer. What we’d like is a practical way to find the lowest-cost solution without having to check all possible solutions. [2:45]

34. One well-known approach is the *branch-and-bound* heuristic, which is illustrated here. This strategy takes advantage of the fact that adding more ancestors to a substemma will never improve its total cost. This has two helpful corollaries: first, we can stop any part of the search as soon as we reach a feasible solution, since adding anything else to it won’t help; and we can save the cost of the best combination we’ve found so far and then cut off any part of the search when it exceeds that cost.

Let’s use the example from the previous slide. We’ll start by trying the potential ancestors in order. If we include ancestor A in the substemma, then we already have a feasible solution with a cost of 4. This is feasible, so we don’t need to add anything else to the substemma. So we backtrack up one level, and this time we exclude A from the substemma.

Obviously, an empty substemma isn’t feasible, so we need to decide whether or not to include ancestor B. If we do, then we still don’t have a feasible substemma, so we have to decide whether or not to include ancestor C. If we do this, then we have a feasible substemma with a cost of only 3. So 3 is the new lowest cost. Now we backtrack and check if we can exclude C. The substemma with only B isn’t feasible, so that doesn’t work. We backtrack again and see if we can exclude B. In this case, even if we include C, the substemma isn’t feasible, so we’re done. The best substemma we found is the one consisting of B and C, which has a total cost of 3.

What’s nice about this approach is that it can also be adapted to check for substemmata with any cost within a given bound; we just use that instead of the best-found cost when we cut off parts of the search. So if we only wanted substemmata with costs up to 3, we would stop the search that starts by including A because its cost is 4. [2:30]

35. So that’s a handy way to find good candidates for a witness’s substemma. Theoretically, we could have an input so bad that this would take as long as the brute-force approach of checking every possible combination, but real-world data is virtually never that pathological. In practice, this process takes within a second or two on a laptop. Here’s the output of the optimize\_substemmata module for witness 5, which has returned all substemmata with costs up to 10. The “AGREE” column gives the number of readings in 5 that are explained by agreement with one or more its ancestors. This is usually higher for substemmata with more ancestors, but it’s a helpful way to compare candidates with the same number of ancestors. As I mentioned earlier, informed text-critical judgment is required of us even in how we choose between our options here. [0:52]

36. Now that we know how to do this for each witness, it just remains for us to put it all together into a global stemma. Like I said a few slides ago, this is similar to how we construct a textual flow diagram. It can be done in any order, but it’s easiest to illustrate it as working from the bottom up, starting with the most posterior witnesses. [0:20]

37. Another thing I mentioned before: for this to work, every extant reading in every witness other than the initial text has to be explained. So we can’t have any incomplete local stemmata like the one on the right. If we do, then what we’ll get for the global stemma is something like this one at the bottom.

What happened? Well, whenever we have an unexplained reading in a local stemma, like *e* or *f* in this one, there will always be at least one earliest witness with that reading. Normally, this witness will just be the initial text and the reading will be the initial reading, in which case everything’s fine. But if the witness is a later one, then none of its potential ancestors will be able to explain its unexplained reading. No substemma for it will be feasible, which means that the witness will just become an isolated extra root in the global stemma.

It’s worth noting that this can also happen to super fragmentary witnesses, but for different reasons. Consider the text of an early, highly lacunose papyrus. Where its readings have survived, they might all agree with the initial text. But by definition, this means that it has equal priority to the initial text; neither one has a prior or posterior reading relative to the other, so the papyrus is technically not a descendant of the initial text. This can be addressed in one of two ways; we can add a rule to break such ties in favor of the initial text every time, or we can exclude witnesses that are too fragmentary from consideration. [1:40]

38. But if we address fragmentary witnesses and complete every local stemma so that it forms a proper tree, then everything looks as we would expect. The global stemma now has one proper root at the initial text.

A complete global stemma relating all witnesses for a New Testament book or corpus has been an elusive goal, a sort of Holy Grail, [click to the next slide] for practitioners and developers of the CBGM ever since it became popularized. Gerd Mink produced a small portion of a global stemma for a few key witnesses. Peter Gurry did so for the Harklean group of Greek witnesses in the Catholic Epistles. Using the data structures and algorithms I broadly described in this talk, I was able to construct the first global stemma for all Greek manuscript witnesses of a New Testament book. But that book was 3 John, so the prize here is probably more of a Dixie cup than a grail. [1:00]

39. Anyway, before we wrap up, I wanted to reiterate the point that I tried to drill into your heads earlier: *a global stemma is not the same as a textual flow diagram*. And now that you’ve seen examples of both, it should be easier to spot the differences. For one, a witness will always have one textual flow ancestor, but it can have more than one ancestor in the global stemma. For another, ancestor-descendant relationships in the global stemma are constrained by the requirement for all readings in the descendant to be explained by readings in at least one of its ancestors, but the same relationships in the textual flow diagram can involve a change between any two readings. It’s true that the textual flow diagram will try to connect witnesses by agreement as much as possible, but where it can’t do this, the whole point is to help us change our judgments on how a reading might be explained, rather than confirm them. Finally, the computational process of constructing a textual flow diagram is straightforward and fast, while that of constructing a global stemma is complex and will usually be at least a bit slower. [1:15]

40. And that gets us to the end of our roadmap! Like I said, the software I developed to learn all of this is open-source, and the core library and a command-line interface based on it are available on my GitHub page. If you’re the type who likes to learn things more by looking under the hood yourself, you’re welcome to check out the code. If you shuddered at the phrase “command-line,” I’m happy to say that David Flood, one of my friends at the CSNTM, has expressed interest in incorporating the software into one of his characteristically friendly and convenient user interfaces, so keep an eye out for that.

I’m also happy to share that the team at Münster has come out with an official implementation of the CBGM, one that is already equipped with a graphical user interface. The instructions on how to setup and use that are linked here.

In conclusion, I sincerely hope that I’ve succeeded in making the “black box in Münster” a bit more transparent and intuitive. Thank you for your time and for the honor of speaking at the Greek Paul Project Webinar. [1:15]