Democracy Developers
A Guide to Risk Limiting Audits with RAIRE

Michelle Blom, Peter Stuckey and Vanessa Teague

Date: July 8, 2023
Version 1.0 DRAFT

Contents

Executive Summary

This document is intended to help election administrators, candidates and interested members of the public understand how to conduct Risk Limiting Audits (RLAs) for Instant Runoff Voting (IRV) using RAIRE.

Risk limiting audits (RLAs) are a post-election activity, performed to provide a certain level of confidence in the correctness of the reported outcome, or to correct a wrong outcome by manual recount. These audits involve randomly sampling paper ballots that have been cast by voters. After collecting and analysing such a sample, statistical computations are performed to ascertain a current level of *risk*. A Risk Limiting Audit guarantees a *Risk Limit:* the maximum probability that it will mistakenly confirm a reported outcome that was in fact wrong. The audit proceeds by sampling ballots until this risk falls below an acceptable level, or election administrators determine that a full manual count is required.

This guide describes how RAIRE operates, the software currently available to support RAIRE audits, and the tools required to perform an audit using RAIRE.

Chapter 1 gives a general overview of the audit process incorporating RAIRE. Chapter 2 reviews instant runoff voting and explains how we visualize and analyze IRV elections. Chapter 3 explains the main concept in RAIRE audits: *assertions*. Chapter 4 reviews Risk Limiting Audits. Chapter 5 brings all of this together and explains how to conduct a complete RLA for IRV elections using assertions generated by RAIRE. Finally, Chapter 6 gives the details on how RAIRE generates the assertions.

Chapter 1

Introduction: IRV RLAs with RAIRE

Risk limiting **A**udits for **IR**V **E**lections (RAIRE) allows election administrators to conduct Risk Limiting Audits (RLAs) for Instant Runoff Voting (IRV) elections. IRV elections are widely used across Australia, and are increasingly being introduced in the United States. They are also used in non-governmental elections including the Best Picture Academy Award. IRV RLAs with RAIRE were successfully piloted for vote-by-mail ballots in the San Fransicso District Attorney's contest of 2019 (Blom *et al.* [2020]), using the SHANGRLA RLA toolkit (Stark [2020]).

RAIRE takes an assertion-based view of auditing. Each assertion is a condition – a comparison between two categories of ballots – that we want to check. With the assertion-based view, each of these comparisons is an *assertion* that needs to be checked or verified during the audit. For any assertion, ballots are divided into three categories: those that support the assertion, those that counteract the assertion, and those that do not affect it.

The easiest way to understand assertions is to think about familiar plurality elections, where voters simply choose their favourite candidate. The assertions are so simple they do not need to be written explicitly: the candidate with the majority of votes wins. When auditing this election, we want to check that the reported winner had more votes than each of the reported losers. The assertion "Alice has more votes than Bob" compares "Votes for Alice" with "Votes for Bob". Votes for Alice support the assertion, votes for Bob counteract it, and blank ballots do not affect it.

IRV elections are more complicated because many different assertions can be required to test one election outcome. IRV involves ranked choice voting. Voters indicate their preferences over a set of candidates by ranking them in order from most to least preferred. To determine the winner of an IRV election, each candidate is initially awarded all ballots on which they are ranked first. The candidate with the least number of votes is eliminated. All ballots in their tally pile are then given to the next most preferred candidate on the ballot who has not yet been eliminated. This process of eliminating the candidate

with the smallest tally continues until a single candidate has the majority of votes. This candidate is declared the winner.

What does RAIRE do?

RAIRE generates a set of *Assertions* that imply that the announced winner won. These assertions are tested with an RLA, hence making an RLA of the IRV election result.

RAIRE is a tool for finding a set of assertions to check in an RLA of an IRV election. These assertions, if verified in an audit, will confirm that the reported election outcome is correct, with a certain degree of confidence based on the achieved risk. RAIRE aims to find a least-cost set of assertions, using a rule of thumb to estimate the sample sizes required to verify them in an audit. To find these assertions, RAIRE requires the cast vote records (CVRs) corresponding to the paper ballots cast by voters. RAIRE can be used for ballot polling audits, provided CVRs are available, but is much more efficient with ballot-level comparison audits, where paper ballots are compared to their associated CVR for discrepancies.

What does RAIRE *not* do?

RAIRE can be used to verify that the announced winner won. It does *not* check whether they won by the announced elimination order. This is a deliberate design feature: RAIRE does not waste auditing effort on details that do not affect who won.

Figure 1.1 shows where RAIRE sits within an audit infrastructure. The same mathematical and statistical procedures used to compute risks for the "Alice versus Bob" comparisons in an audit for a plurality election can be applied to compute risks for each RAIRE assertion. Auditing tools designed for plurality elections can be extended to conduct audits for IRV with the integration of RAIRE. Figure 1.1 shows existing (plurality) RLA computations in purple, while new (IRV-specific) modules are green. New software is required to:

1. generate the assertions;
2. visualize the assertions and validate that they imply that the announced winner won;
3. enter the preference rankings observed on the sampled ballots; and
4. compute the discrepancies between the CVRs and the ballots, for each assertion.

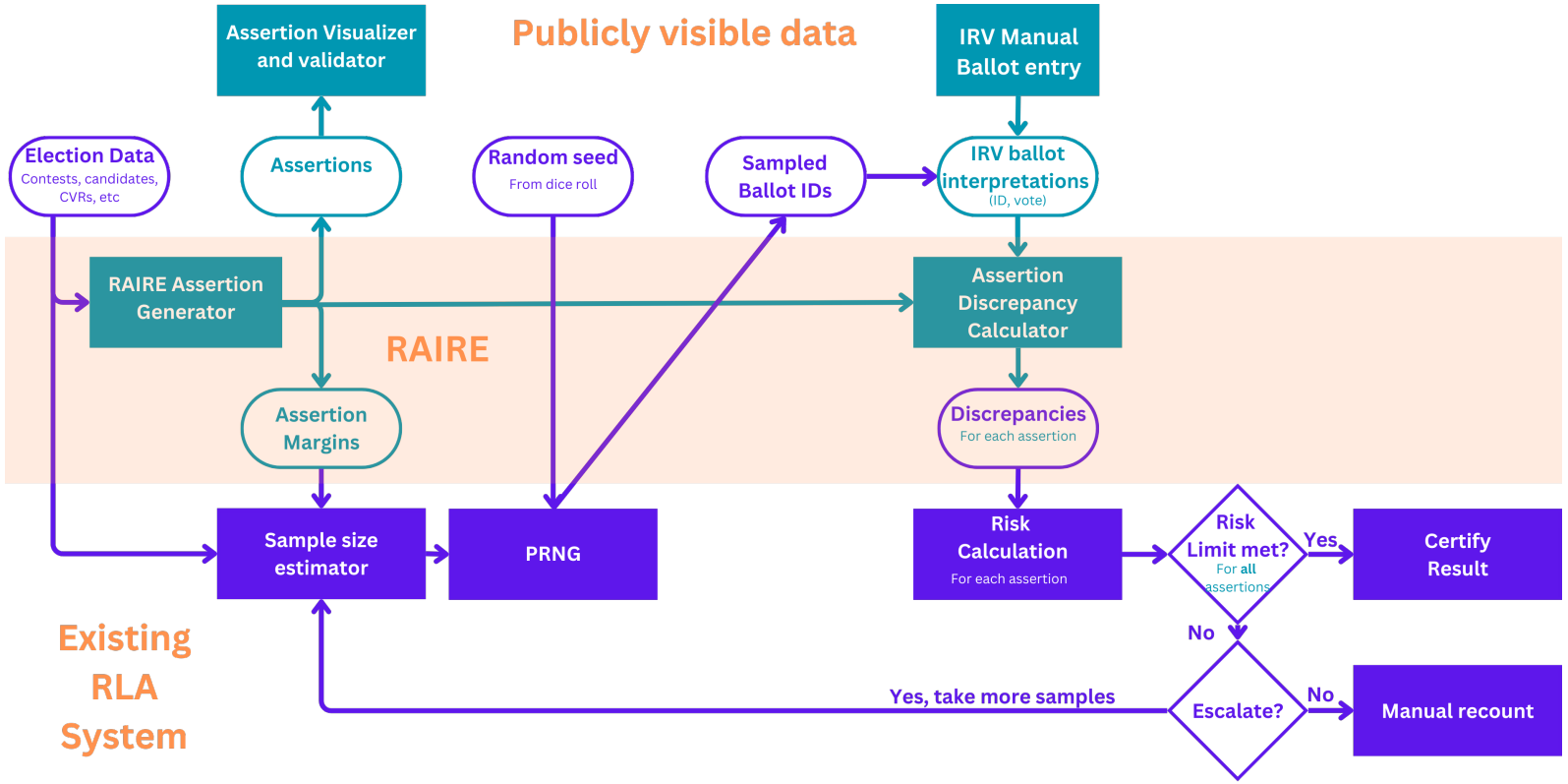

Figure 1.1: A high-level overview of an IRV RLA using RAIRE. Purple elements are existing RLA code. Green indicates new code and data structures for IRV.

The audit process from beginning to end

The basic process of an RLA remains the same as for plurality elections. The main difference is that an IRV outcome depends on multiple assertions, and all the assertions have to be accepted before the audit can confirm the result. The main parts of the workflow are summarised here. New steps for IRV are in bold.

1. Commit to the ballot manifest and CVRs.
2. Choose contest(s) for audit.
3. **Run RAIRE to generate assertions for audit.**
4. **Use the RAIRE assertion validation and visualization module to check that the assertions imply the announced winner won.**[1]
5. Generate a trustworthy random seed, e.g. by public dice rolling.
6. Estimate the required sample size, based on the margin, **for each assertion.**
7. Use the seed to generate the list of sampled ballots.
8. Retrieve the required ballots, compare them to their CVRs, and calculate the discrepancies **for each assertion.**
9. Update the risk **for each assertion** based on the observed discrepancies.
10. For each contest under audit, if the measured risk is below the risk limit **for each assertion**, stop the audit and accept the result.
11. If some results have not yet been confirmed, decide whether to escalate (sample more ballots) or conduct a full manual count.

[1] This can be done at any time, including after the audit is complete.

Chapter 2

IRV elections and Visualizing Outcomes

How IRV counts work

In an IRV election there are a set of candidates, C, who are ranked by voters. Each voter specifies their first preference (just like in plurality elections) and may then, optionally, specify second, third and later preferences. An IRV count proceeds in a series of *elimination* steps.

Initially, every candidate's first preferences are counted. This provides the first set of tallies for *IRV elimination*. We call the tallies of candidates at this stage their *first preference tallies*.

IRV elimination proceeds as described below:

While there is more than one continuing candidate:

- From the continuing candidates, select the candidate l with the lowest tally.
- *Eliminate* l: give each vote in l's tally to the next-preferred continuing candidate on that ballot.

An *elimination order* is a list of candidates in the order they were eliminated. By convention, we write the winner (who is never eliminated) last. There can be many different elimination orders that lead to the same winner. In practice, IRV counts often stop when one candidate w has more than 50% of the votes—at that stage, there is no chance that anyone other than w will remain at the end, so we can cut the algorithm short and declare that w is guaranteed to win.

Example 1. *Suppose there are 4 candidates: Alice, Bob, Chuan and Diego. The votes are as follows:*

Preferences	***Count***
(A, B, C, D)	5
(B, A, C)	1
(B, D, A)	1
(C)	2
(C, D)	2
(D, C)	4

We first count the first preference tallies: Alice has 5 votes, Bob has 2, Chuan has 4 and Diego has 4. This means that Bob is eliminated first and his votes distributed to the next-preferred candidate on each ballot—Diego gets the (B, D, A) vote and Alice gets the (B, A, C) vote. The new tallies are Alice: 6, Bob: eliminated, Chuan: 4, Diego: 5. Now Chuan is eliminated. Diego gains two more votes–the (C, D) ballots–and wins with 7 votes compared to Alice's 6.

The full sequence of tallies is shown below.

Count	**Remaining**	**Alice's tally**	**Bob's tally**	**Chuan's tally**	**Diego's tally**	**Action**
1	A, B, C, D	5	2	4	4	Eliminate B
2	A, C, D	6	-	4	5	Eliminate C
3	A, D	6	-	-	7	Eliminate A; D wins

Visualizing all possible IRV outcomes

Let us represent the number of candidates in the set C as $|C|$. An IRV election with $|C|$ candidates has $|C|! = |C| \times (|C| - 1) \times (|C| - 2) \times \cdots \times 3 \times 2$ possible (complete) elimination sequences. We can visualize these as a collection of *trees*. A *tree*, in computer science parlance, is a way of organizing data hierarchically. Figure 2.1 depicts an example of a tree data structure. A tree contains a number of *nodes*, where each node can be thought of as a container of data. These nodes are organised hierarchically, and connected by *edges*. In the example shown in Figure 2.1, there are five nodes, each labelled from one to five. Node 1 is connected to node 2 by an edge, and to node 3 by an edge.

When describing how we visualize the space of possible outcomes of an election, and how RAIRE works, we will use some additional terminology relating to trees.

Child A node, c, is a *child* of another node, p, if c sits underneath p and c is connected to p by a single edge. In our example, nodes 2 and 3 are children of node 1. Nodes 4 and 5 are children of node 2.

An example of a *tree*

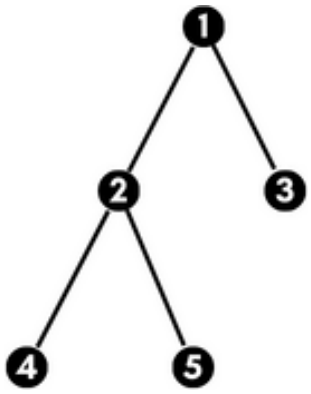

Figure 2.1: A tree data structure.

Parent If node c is a child of node p, then node p is the *parent* of c. In our example, node 1 is the parent of nodes 2 and 3, while node 2 is the parent of nodes 4 and 5.

Ancestor The ancestors of a node n are the set of all nodes that we can reach from n by moving up the tree from child node to parent. In our example, nodes 2 and 1 are the ancestors of nodes 4 and 5. Node 1 is the only ancestor of nodes 2 and 3.

Descendant The descendants of a node n are the set of all nodes that we can reach from n by moving down the tree from parent node to child. In our example, nodes 4 and 5 are the descendants of node 2. Nodes 2, 3, 4 and 5 are the descendants of node 1.

Leaf A *leaf* is a node without children.

Branch A *branch* is a path of connected nodes starting at the very top of the tree, and ending at a leaf. In our example, the sequences of nodes 1–2–4, 1–2–5 and 1–3 represent the branches of our tree.

We visualize the space of possible elimination orders for an election as a *collection* of trees, one for each possible winner (including the reported winner). We call each of these trees an *elimination tree*. For an election between candidates Alice, Bob, Chuan, and Diego, the top level of each of these elimination trees is shown in Figure 2.2.

Consider the first node depicted in Figure 2.2. This node represents all outcomes that *end* with Alice as the winner. Similarly, the second depicted node represents all outcomes that *end* with Bob as the

winner. Each node in an elimination tree represents either a *complete* or a *partial* outcome. Nodes 1 to 4 in Figure 2.2 represent partial outcomes as they do not express a complete elimination order.

RAIRE visualizes all elimination orders that end with Alice as the winner as shown in Figure 2.3. At the second level of the tree, we add a candidate as the runner-up. (The runner-up is the last candidate eliminated, though in IRV this is not necessarily the losing candidate who came closest to winning.)

Elimination Trees – First Level

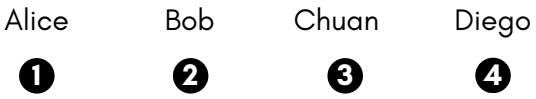

Figure 2.2: Top level of each elimination tree for a contest between Alice, Bob, Chuan, and Diego.

Elimination Tree with Winner Alice

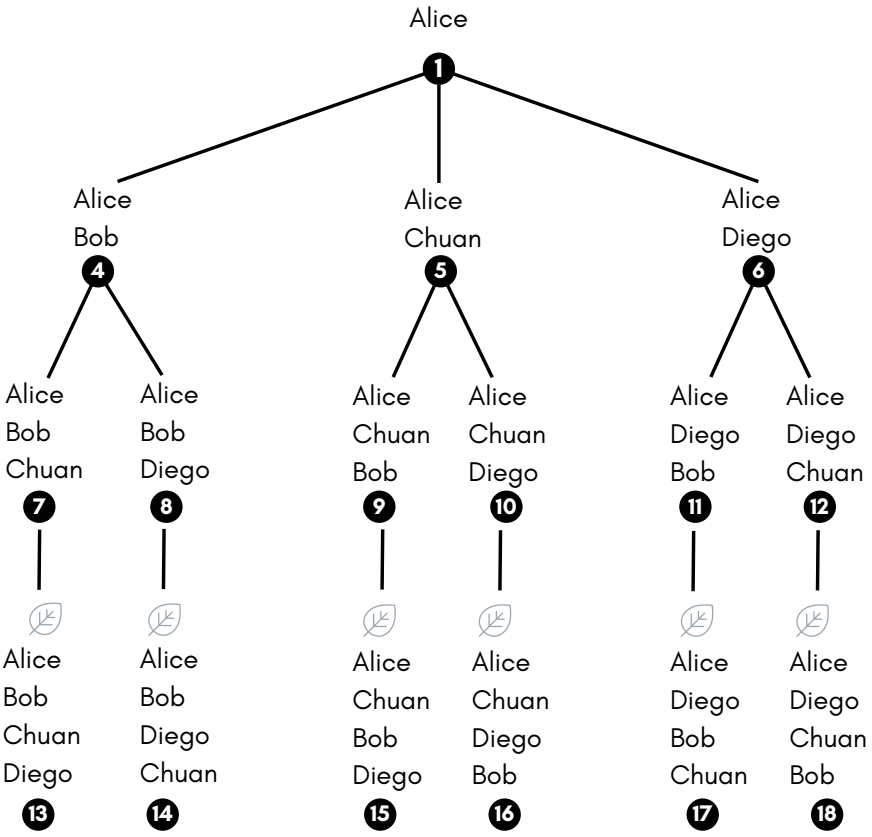

Figure 2.3: All possible elimination orders where Alice is the ultimate winner. The leaves of this elimination tree represent *complete* elimination orders.

Node 4 represents all outcomes that end with Bob as the runner-up and Alice as the winner. In nodes 5 and 6, Chuan and Diego, respectively, are the runner-up candidates.

The third level of the tree in Figure 2.3 identifies a candidate to be eliminated just prior to our runner-up. The leaves, on the fourth level, represent complete elimination orders. Node 13, for example, represents an elimination order in which Diego is eliminated first and Chuan second, leaving Bob as the runner-up, and Alice as the winner. Nodes 13–18 in Figure 2.3 represent *complete* outcomes.

The tree in Figure 2.3 captures all elimination orders that end with Alice as the ultimate winner. The complete set of **alternate outcome trees** that RAIRE considers is formed by the collection of such trees, one *for each reported loser*. To save space we often label each node with only the candidate eliminated at that step, rather than the whole elimination order. The whole order can be read by tracing up to the top. So each path up a tree represents an elimination order, with the first-eliminated candidate at the leaf and the winner at the top. We demonstrate this idea in the next example.

2019 San Franciso DA's Race

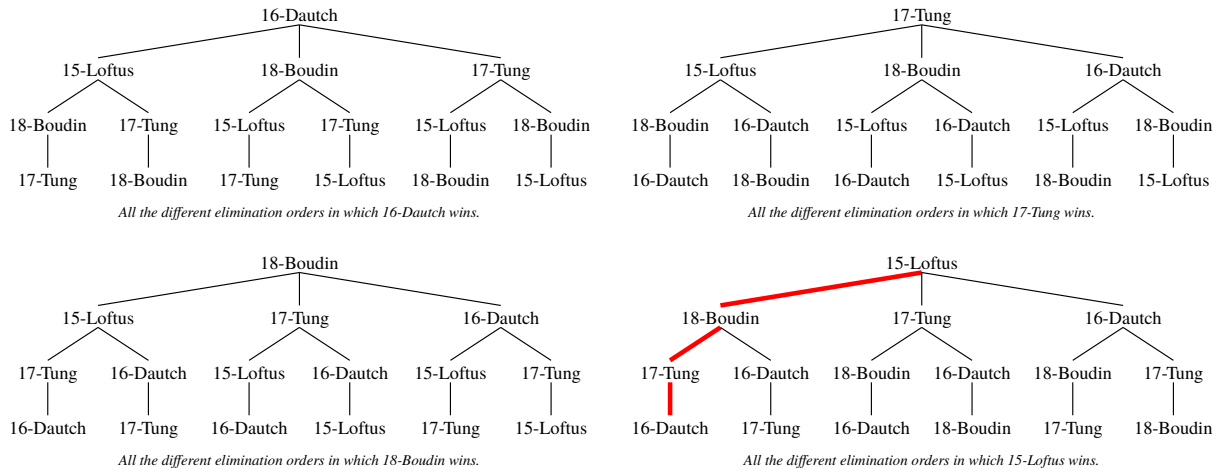

Figure 2.4: Complete elimination trees for the 2019 San Francisco DA race.

Example 2. *San Francisco District Attorney's contest, 2019.*

Figure 2.4 shows the complete set of elimination trees for the San Francisco 2019 DA's contest, including the (vote-by-mail) winner. Each tree shows all the different ways that a particular candidate can win. The bright red line shows the official elimination order: Dautch, Tung, Boudin, Loftus, leading to a win by Loftus. The rest of that tree shows other elimination orders that lead to a Loftus win. The other trees show wins by other candidates, and the elimination orders that lead to those wins.

Exercises

Exercise 1. *Tally the following IRV example election.*

Preferences	***Count***
(A, B, C, D)	*50*
(A, C)	*40*
(B, C, A)	*25*
(B, D, A)	*25*
(C, A, B)	*30*
(C, D, B)	*45*
(D)	*100*

Exercise 2. *Draw the elimination trees for Exercise 1. You can ignore the tree for the apparent winner.*

Exercise 3. *Suppose there are 6 candidates.*

a) *How many different complete elimination sequences are there?*

b) *How many leaves are there in each elimination tree?*

c) *How many leaves are there altogether in all the apparent losers' elimination trees?*

Chapter 3

Assertions for IRV winners

We can verify the outcome of an IRV election with only two types of assertions. These are called Not Eliminated Before (NEB) assertions and Not Eliminated Next (NEN) assertions. To explain these assertion types, we will consider an election with four candidates: Alice, Bob, Chuan and Diego.

Not Eliminated Before (NEB) Assertions

Alice NEB Bob is an assertion saying that Alice cannot be eliminated before Bob, irrespective of which other candidates are continuing. In other words, no outcome is possible in which Alice is eliminated before Bob. When expressed as a comparison of tallies, this assertion says that the *smallest* number of votes Alice can have, at any point in counting, is greater than the *largest* number of votes Bob can ever have while Alice is continuing. Alice's smallest tally is equal to her first preference count – the number of ballots on which she is ranked first. The largest number of votes Bob can have while Alice is continuing is the number of ballots on which he is ranked higher than Alice, or he is ranked and Alice is not.

Example 3. *In the following example, Alice NEB Bob is true because Bob is ranked a total of 80 times without being preceded by Alice, which is less than Alice's first-preference tally of 100. However, Alice NEB Diego is not true, because Diego is ranked 125 times without being preceded by Alice, which is more than Alice's first preference tally.*

Preferences	***Count***
(A, B, C, D)	*100*
(B, D, C)	*40*
(C, B, D)	*40*
(C, D)	*45*

Not Eliminated Next (NEN) Assertions

NEN assertions compare the tallies of two candidates under the assumption that a specific set of candidates have been eliminated. An instance of this kind of assertion could look like this: *NEN: Alice > Bob if only {Alice, Bob, Diego} remain*. This means that in the context where Chuan has been eliminated, Alice cannot be eliminated next, because Bob has a lower tally. When expressed as a comparison of tallies, this assertion says that the number of ballots in Alice's tally pile, in the context where only Alice, Bob and Diego are continuing, is greater than the number of ballots in Bob's tally pile in this context. This example assumes one eliminated candidate – Chuan – however, NEN assertions can be constructed with contexts involving no eliminated candidates, or more than one eliminated candidate. The assertion *NEN: Alice > Chuan if only {Alice,Bob,Chuan,Diego} remain* says that Alice cannot be the first eliminated candidate, as she has more votes than Chuan when no candidates have yet been eliminated. The assertion *NEN: Diego > Bob if only {Bob,Diego} remain* says that Diego has more votes than Bob in the context where those two are the only continuing candidates.

Simple assertions sometimes work

RAIRE works by generating a set of assertions which, together, imply a particular winner. In this section, we introduce some common patterns that those sets of assertions might use. We aim to make it obvious why certain sets of assertions are enough to imply a particular winner, and to match a person's intuition about why a certain candidate won an IRV election.

One candidate dominates

Sometimes one candidate happens to be so strongly ahead of all the others that NEB assertions hold with all other candidates.

Example 4. *Suppose that for the four candidates Alice, Bob, Chuan and Diego, we have:*

Alice NEB Bob,
Alice NEB Chuan and
Alice NEB Diego.

Then Alice must be the winner—it is not possible for her to be eliminated at any stage of the IRV count.

Although this might sometimes be enough, real elections are usually closer. We need to look for other patterns of assertions that are enough to imply that one candidate won.

Two leading candidates

Now suppose there are two candidates who accumulate most of the votes: Alice and Bob.

Example 5. *Suppose Alice NEB Bob is not true, but the following weaker fact is true:*

NEN: Alice > Bob if only {Alice, Bob} remain.

This says that, after Chuan and Diego are eliminated, Alice's tally is higher than Bob's.

Assume we still have two NEB assertions:

Alice NEB Chuan and
Alice NEB Diego.

This, again, is enough to prove that Alice won. To see why, consider the last elimination step. Alice must reach this step, because she cannot have been eliminated before Chuan or Diego. If Chuan or Diego is the other remaining candidate, Alice beats them (by the NEB assertion). The only other possibility is Bob—for this case, the NEN assertion shows that, in the last round, Alice beats Bob.

Visualizing assertions

This reasoning can be visualized using elimination trees. For an audit, we need to disprove all elimination orders that result in a winner other than the announced winner.

The assertion *Alice NEB Chuan* is enough to disprove every elimination order in which Alice is eliminated before Chuan, and hence to disprove the entire tree in which Chuan wins. It also allows us to disprove the orders in Bob's tree and Diego's tree in which Alice is eliminated before Chuan.

The consequences of *Alice NEB Chuan* in Bob's tree and Chuan's tree are shown in Figure 3.1. It still allows the possibility that Bob might win via elimination orders Diego, Chuan, Alice, Bob, or Chuan, Diego, Alice, Bob, or Chuan, Alice, Diego, Bob.

Implications of NEB Assertions

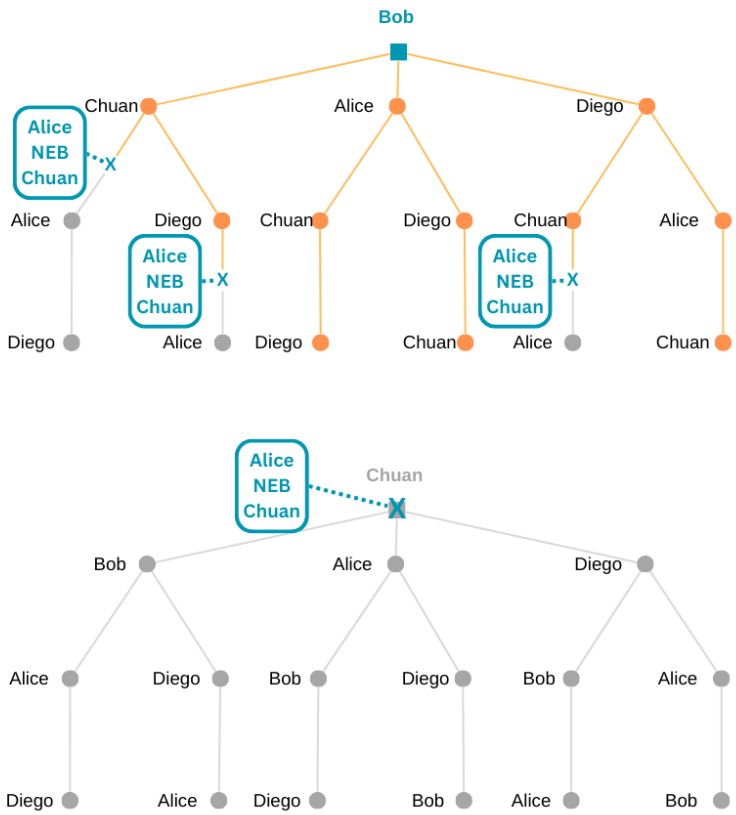

Figure 3.1: Implications of *Alice NEB Chuan* in the trees where Bob and Chuan win. Chuan cannot win, but Bob might.

Suppose we add another other NEB assertion: *Alice NEB Diego*. Now the only possible alternate winner is Bob. Figure 3.2 adds this assertion and the final assertion, *NEN: Alice > Bob if only {Alice, Bob} remain*, leaving Alice as the only possible winner.

This pattern, with two leading candidates and several others who are easily excluded, is common in IRV elections. In the Australian state of New South Wales, this set of assertions is true in most cases.

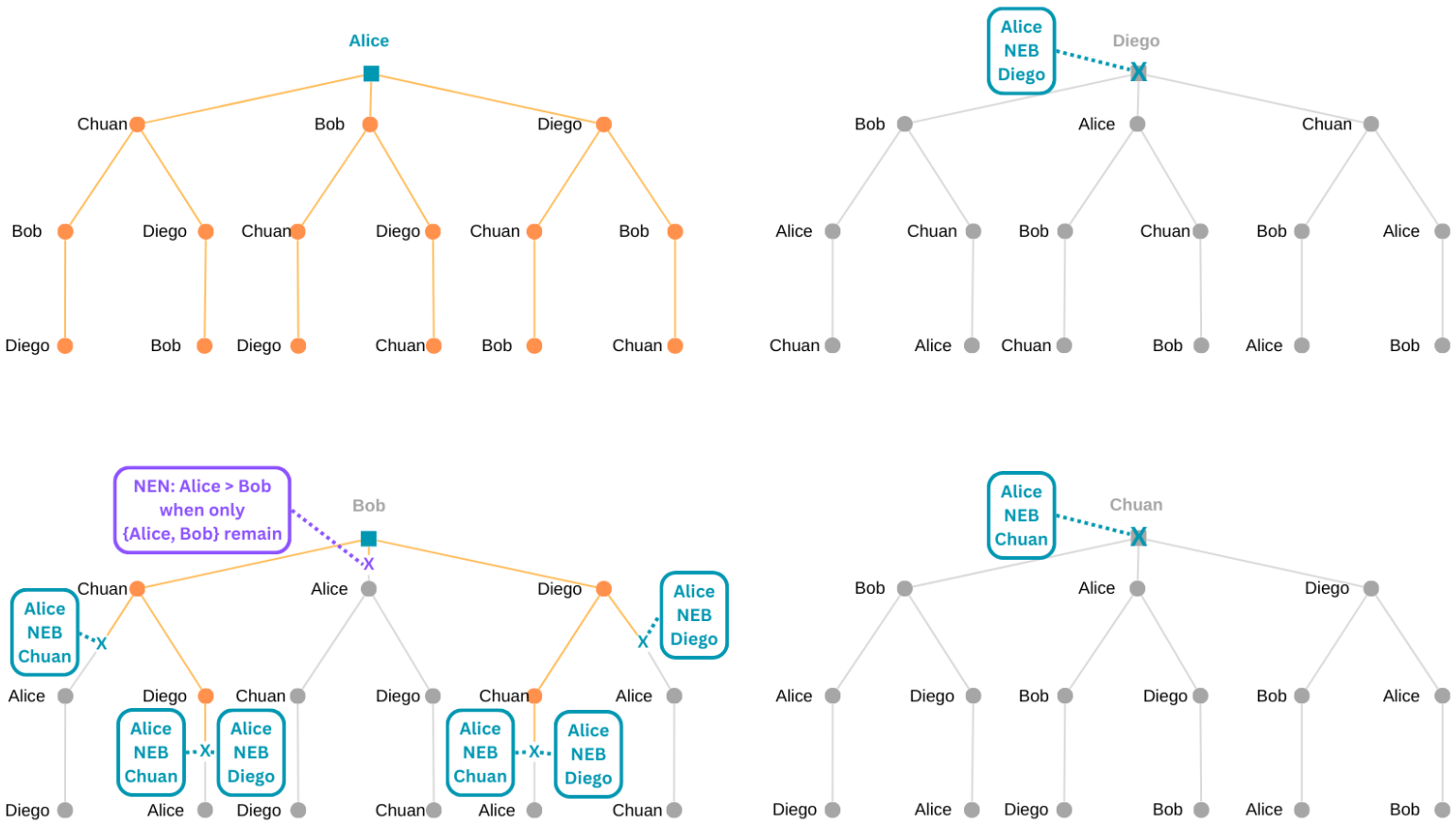

Figure 3.2: Implications of *Alice NEB Chuan* combined with *Alice NEB Diego* and *NEN: Alice > Bob if only {Alice,Bob} remain*. Alice is the only possible winner.

Why not audit every elimination step?

A common suggestion is to conduct an IRV RLA by testing every elimination step. This is valid, but it can waste a lot of effort verifying comparisons that do not alter who wins.

Example 6. *Suppose there are 4 candidates, Alice, Bob, Chuan and Diego. The votes are as follows:*

Preferences	***Count***
(A)	*10,000*
(B, A, C)	*5,000*
(C, A, B)	*4,999*
(D, C, B, A)	*16,000*

The full sequence of tallies is shown below.

Count	***Standing***	***Alice's tally***	***Bob's tally***	***Chuan's tally***	***Diego's tally***	***Action***
1	A, B, C, D	*10,000*	*5,000*	*4,999*	*16,000*	*Eliminate* C
2	A, C, D	*14,999*	*5,000*	*-*	*16,000*	*Eliminate* B
3	A, D	*19,999*	*-*	*-*	*16,000*	A *wins*

Auditing the first elimination step would need a lot of samples, because the difference between the two lowest candidates, Bob and Chuan, is only one vote. However, it is easy to see that it does not matter whether Bob or Chuan is eliminated first. If the CVRs are wrong, and the truth is instead that there are 4,999 (B, A, C) *votes and 5,000* (C, A, B) *votes, then the correct tally sequence is the following.*

Count	***Continuing***	***Alice's tally***	***Bob's tally***	***Chuan's tally***	***Diego's tally***	***Action***
1	A, B, C, D	*10,000*	*4,999*	*5,000*	*16,000*	*Eliminate* B
2	A, C, D	*14,999*	*-*	*5,000*	*16,000*	*Eliminate* C
3	A, D	*19,999*	*-*	*-*	*16,000*	A *wins*

This produces the same winner.

Auditing effort—the number of ballots that need to be sampled in order to be confident that the announced winner won—tends to be higher for small margins. Auditing every elimination step would waste effort, not because there are many assertions, but because the very close comparison between Bob and Chuan would require a large sample to confirm. This effort does not need to be expended in this example, because the comparison between Bob and Chuan is irrelevant to who wins.

Of course, for some IRV elections, early eliminations do matter. RAIRE examines all the CVRs and determines which comparisons can be ignored and which need to be tested with an RLA.

More complicated sets of assertions

Sometimes the simple assertion sets above are not enough, or do not lead to an efficient audit. In these cases, we need to use RAIRE to generate a set of assertions that is sufficient to prove that the announced winner won. This is described in Chapter 6.

Verifying that a set of assertions implies a winner

Each assertion (if true) rules out certain elimination orders, removing the corresponding paths in the trees that visualize all IRV outcomes. If some candidate's tree has all paths down to its leaves removed, then that candidate cannot win: all the elimination orders that let them win have been ruled out.

How can we verify that a certain set of assertions truly implies that the announced winner won? There are three ways:

- Using the visualization trees to check that, for all other candidates, all the paths down to their leaves are cut by an assertion, or
- Using a logical argument like the example in the section Two Leading Candidates on p. 16, or
- Using software to do the validation.

This is a crucial part of the RAIRE audit, especially when the assertions are generated by RAIRE and do not fit into one of the simple, familiar patterns above.

Exercises

Exercise 4. *Suppose there are three candidates: Alice, Bob and Chuan.*

a) *Draw out all the elimination trees.*

b) *Suppose that the announced elimination order is (Bob, Alice, Chuan), so Chuan is the announced winner. Identify that elimination order in your trees.*

c) *Add the assertion Chuan NEB Alice. Show which elimination orders are removed from your trees.*

d) *Think of one more assertion which, if combined with Chuan NEB Alice, implies that Chuan wins. Show its implications on your trees.*

e) *Is your answer to Part d) the only possible answer?*

Exercise 5. *Consider the San Francisco DA's contest with four candidates: Tung, Loftus, Dautch and Boudin. Suppose you are given a set of three assertions:*

- *Loftus NEB Tung*
- *Loftus NEB Dautch*
- *NEN: Loftus > Boudin if only {Loftus, Boudin, Dautch} remain*

Does this imply that Loftus won? Either argue that it does, or provide an alternative winner via an elimination order that is consistent with all these assertions.

Exercise 6. *Write all the assertions for checking every elimination step in Example 6 to show that Alice won. Hint: you will need a lot of NEN assertions, with sets that match the set of candidates still standing in the table.*

Exercise 7. *Write a simpler set of assertions to check that Alice won in Example 6. Hint: try some NEB assertions.*

Exercise 8. *Make up an example IRV election in which the comparison between the lowest two candidates in the first elimination round does matter.*

Chapter 4

Risk Limiting Audits

This section reviews Risk Limiting Audits. Although RAIRE can work with any RLA, we concentrate on *ballot-level comparison RLAs* because these are currently used by the State of Colorado. Our presentation follows Lindeman and Stark's *Super-simple simultaneous Risk Limiting Audits* (Stark [2010]). Skip this chapter if you are already familiar with RLAs.

The aim of an RLA is to test an *apparent election outcome*—usually a single winning candidate w—by examining a random sample of ballot papers. An RLA may either accept the outcome, or fall back to a full manual recount in order to find the true winner. The *Risk Limit* α is chosen in advance, often by legislation. If the apparent outcome is wrong, the audit is guaranteed not to accept it except with probability at most α.

For plurality elections, the *apparent margin* between an apparent winner w and and apparent loser l is simply the winner's tally minus the loser's, according to the CVRs. The *actual margin* is $w's$ tally minus $l's$ according to the ballots. If the actual margin is negative, w is not the true winner. The *diluted margin* is the apparent margin as a fraction of total votes cast in the contest. Stark [2010] defines the diluted margin as the apparent margin divided by the total number of *ballots cast in contests under audit*, but since CO audits each contest separately, our simpler definition is equivalent.

First, administrators commit to a *ballot manifest*—a trustworthy list of the paper ballots—and a *cast vote record* (CVR) for each ballot. Having seen the CVR commitments, the public can then participate in a dice rolling ceremony to seed a *pseudorandom number generator* (PRNG) which generates a series of ballot IDs for sampling. The number of ballots to be sampled is a function of the risk limit, the margin, and the expected number (and type) of errors.

Ballot-level comparison RLAs proceed by taking a random sample of ballot papers and checking each one against its corresponding CVR. An error that advantages the apparent winner is an *overstatement*; an error that advantages an apparent loser is an *understatement*. The *discrepancy* is the difference

between the CVR and the actual ballot—by convention, overstatements are positive discrepancies, while understatements are negative.

Example 7. *Suppose Diego is the reported winner and Chuan is a reported loser in a plurality contest.*

- *If the CVR records a vote for Diego but the ballot paper has a vote for Chuan, that is a discrepancy of 2, i.e. a* 2-vote overstatement.
- *If the CVR records an invalid vote, but the ballot paper has a vote for Chuan, that is a discrepancy of 1, i.e. a* 1-vote overstatement.
- *If the CVR and the ballot paper are identical, that is a discrepancy of 0.*
- *If the CVR records an invalid vote, but the ballot paper has a vote for Diego, that is a discrepancy of -1, a* 1-vote understatement.
- *If the CVR records a vote for Chuan but the ballot paper has a vote for Diego, that is a discrepancy of -2, i.e. a* 2-vote understatement.

Obviously overstatements are far more important than understatements—if there are enough overstatements, the apparent winner may be wrong.

The discrepancies for all the sampled ballots are input into to a *risk measuring function*. If the measured risk is less than the Risk Limit α, the audit can stop and accept the announced election result. Otherwise, officials can decide to either *escalate*—take more samples and continue the audit—or perform a full hand count to establish who really won.

For more careful explanations of how to run an RLA, see the Democracy Fund's RLA Standard Operating Procedures 2022[1] or their excellent "Knowing it's right" series[2].

[1] https://static1.squarespace.com/static/61da19f9986c0c0a9fd8435f/t/620e599548201f38d44ada3d/1645107606630/RLA+SOP.pdf

[2] https://democracyfund.org/idea/knowing-its-right-limiting-the-risk-of-certifying-elections/

Chapter 5

Using assertions to audit IRV outcomes

Now for the audit! The main insight of RAIRE is *we can verify an IRV outcome using assertions, without repeating the elimination steps.*

We have a set of assertions which imply that the announced winner won. We could (but we won't) verify each assertion by manually examining all the ballot papers. For example, *Diego NEB Chuan* could be verified by manually counting all the first preferences for Diego and checking that that tally was greater than the (manually counted) total number of times Chuan was preferenced without being preceded by Diego. Similarly, NEN assertions could be verified by counting the tallies ignoring candidates not specified in the assertion. For example, we could verify *NEN: Bob $>$ Chuan if only {Bob,Chuan,Diego} remain* by sorting every ballot into a tally pile according to which of Bob, Chuan, and Diego was the highest preference, ignoring any preference for Alice, and then checking that Bob's total was higher than Chuan's. If we did this for every assertion in the set, it would be a logically sound way to verify the election result, but it would be very inefficient.

Instead, we test each of the assertions using an RLA at some risk limit α. If the audit accepts them all, we conclude the audit and accept the IRV election result.

If the announced winner is wrong, then at least one of the assertions must be false. Since we test each assertion with an RLA at risk limit α, the RLA for the wrong assertion will mistakenly accept it with probability at most α. Hence the overall process is a valid Risk Limiting Audit— it will mistakenly accept the wrong outcome with probability at most α.

Both types of assertions—NEB and NEN—can be tested with standard RLA systems, but they need to be carefully transformed into the right form.

NEB Assertions

Scoring NEB assertions

An NEB assertion, for example *Alice NEB Bob*, says that Alice's first preferences exceed the total number of mentions of Bob that are not preceded by a higher preference for Alice.

We start with a set of CVRs and count them as follows:

Ballot contents	**Counted for**	**Example**
First preference for Alice	Alice 1st preference	(A, B, C, D)
Mentions Bob *without* higher preference for Alice	Bob mention	(C, B, D, A)
Mentions Bob *with* higher preference for Alice (not first)	Neither	(C, A, B)
Other	Neither	(C, A, D)

This fits naturally into any existing Risk-limiting audit process, except that our two candidates are "first preferences for Alice" and "mentions of Bob not preceded by Alice."

Auditing NEB assertions

Consider again the assertion *Alice NEB Bob*. To conduct a ballot-level comparison audit of this assertion, the process for randomly selecting ballots for audit is the same as any other RLA. When a ballot is selected, overstatements are errors that advantage the "first preferences for Alice" candidate, while understatements are errors that advantage the "mentions of Bob (not preceded by a higher preference for Alice)" candidate. An *overstatement* is an error that either mistakenly records a first preference for Alice, or mistakenly omits a mention of Bob not preceded by Alice.

For example, if a CVR says that a vote is a first-preference for Alice, but the ballot paper shows only a second preference for her (and a first preference for some other candidate, say Diego), then this is a one-vote overstatement. If the ballot paper actually shows a mention of Bob, not preceded by a higher preference for Alice, then the error is a two-vote overstatement.

NEN Assertions

Scoring NEN assertions

An NEN assertion, for example *NEN: Alice $>$ Bob if only {Alice,Bob,Chuan} remain*, says that Alice beats Bob when Alice, Bob, and Chuan are the only continuing candidates.

This is also easy to fit into any existing Risk-limiting audit process, except that our two candidates are "Alice's tally when only Alice, Bob, Chuan remain" and "Bob's tally when only Alice, Bob, Chuan remain."

Scoring vote $\mathbf{b} = (\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_n})$		
Assertion A	SCORE$_A(b)$	**Notes**
c_1 *NEB* c_k where $k > 1$	1	Supports 1st prefs for c_1
c_j *NEB* c_k where $k > j > 1$	0	c_j precedes c_k but is not first
c_j *NEB* c_k where $k < j$	-1	a mention of c_k preceding c_j
NEN: $c_i > c_k$ *if only* $\{\mathcal{S}\}$ *remain*		
where c_i is b's first-ranked candidate in S	1	counts for c_i (expected)
where neither c_i nor c_k is b's first-ranked candidate in S	0	counts for neither c_j nor c_k
where c_k is b's first-ranked candidate in S	-1	counts for c_k (unexpected)

Table 5.1: Complete scoring for vote $(c_1, c_2, \ldots, c_n)$ for all assertions. Over-statements are counted by subtracting the ballot paper score from the CVR score (and vice versa for under-statements).

We start with a set of CVRs and count them as follows:

Ballot contents	**Counted for**	**Example**
Alice, not preceded by Bob or Chuan	Alice	(A, B, C, D)
Bob, not preceded by Alice or Chuan	Bob	(D, B, C, A)
Chuan, not preceded by Alice or Bob	Neither	(D, C)
Anything else	Neither	(D)

We simply allocate the vote as if Diego has been eliminated. The same works when sets of more than one candidate have been eliminated, or when there are more than 4 candidates: simply score the vote for the first-ranked candidate among those continuing.

Auditing NEN assertions

For an NEN assertion *NEN: Alice $>$ Bob if only $\{\mathcal{S}\}$ remain*, an *overstatement* is an error that advantages Alice by mistakenly listing her as the highest preference in set $\mathcal{S}$, or disadvantages Bob by mistakenly not listing him first among $\mathcal{S}$. An understatement is the opposite.

For example, for the assertion *NEN: Alice $>$ Bob if only {Alice,Bob,Chuan} remain*, if the CVR was (Diego, Alice, Bob, Chuan), but the ballot paper actually contained (Diego, Bob, Alice, Chuan), that would be a two-vote overstatement. (The first preference for Diego is ignored because we consider only {Alice, Bob, Chuan} as continuing.)

Assertion scoring summary

Table 5.1 shows how to score each ballot for each possible kind of assertion. Note that the score is a function of the assertion and the vote only - it does not depend on the apparent outcome.

The over-statement counts (discrepancies) are derived by simply subtracting the ballot paper score from the CVR score (and vice versa for understatements). For example, if a CVR says that a vote contained a first preference for w, but the actual ballot contains a mention of l that precedes w, then it overstates the *w NEB l* assertion by $1 - -1 = 2$. By convention, overstatements are written as a positive discrepancy, while understatements are negative.

Examples

Returning to Example 5 on p. 16, some example scores for the CVR and (actual) ballots are given below, for each assertion. The difference between the actual and apparent score is the overstatement.

Example 8. *Suppose the CVR is (A, B, C, D) and the ballot is (B, A, C, D). Scores for the CVR and ballot are given below, with the corresponding overstatement.*

Assertion	*CVR Score*	*Ballot Score*	*Overstatement*
Alice NEB Chuan	*1*	*0*	*1*
Alice NEB Diego	*1*	*0*	*1*
NEN: Alice > Bob if only {Alice, Bob} remain	*1*	*-1*	*2*

This error overstates Alice's win—too many errors like this may mean that the announced election outcome is wrong.

Example 9. *Suppose the CVR is (D, C, B, A) and the ballot is (C, D, B, A). Scores for the CVR and ballot are given below, with the corresponding overstatement.*

Assertion	*CVR Score*	*Ballot Score*	*Overstatement*
Alice NEB Chuan	*-1*	*-1*	*0*
Alice NEB Diego	*-1*	*-1*	*0*
NEN: Alice > Bob if only {Alice, Bob} remain	*-1*	*-1*	*0*

Although there is an error, it causes no discrepancies for any of the assertions. This means the error does not affect the logic for proving who wins.

Doing the audit

Now the audit can proceed exactly like a plurality audit, using existing plurality audit software, except that an IRV outcome should be confirmed only if *all* of its assertions are confirmed.

In order to estimate the required sample size, we need to compute an *Assertion Margin* for each assertion—this is simply the sum of the CVR scores for that assertion. This is the number of overstatements that would need have occurred for that assertion to be truly false (according to the paper ballots). Just like

plurality contests, IRV assertions with smaller margins tend to require larger samples to confirm. The overall sample size is approximately the sample size required by the closest margin.

As ballots are sampled, their discrepancies are computed as described above. This can be done by hand, but it is easier for people if they can simply enter the ballot they see into some software which calculates the discrepancies for each assertion. The discrepancies are then entered into the risk calculation, which is the same as for plurality.

The audit of an IRV contest can stop and confirm the result when all of its assertions are confirmed. If there are some assertions which remain above the risk limit after the whole sample is taken, officials can decide whether to take a larger sample or do a manual recount.

Assertions, audits and outcomes: logical summary

- If all of the assertions are true, and the set of assertions implies that the announced winner won, then the announced winner won.
- The announced winner may be the true winner, but not via the announced elimination order. RAIRE does not verify the elimination order, just the winner.
- The announced winner may be the true winner, but some of the assertions may be false if they won via a different elimination order from the announced one. In this case (with probability at least $1 - \alpha$) RAIRE will resort to a full hand count, which will confirm the announced winner.
- If the announced winner is wrong, then at least one of the assertions must be false. In this case, with probability at least $1 - \alpha$, RAIRE will fall back to a full hand count (with IRV elimination steps) and thus find the true winner.
- The audit may pass even if the announced winner is wrong, but this happens with probability at most α, the risk limit.

Exercises

Exercise 9. *Suppose the CVR is (D, C, B, A) and the ballot is (C, D, B, A). Fill in the table below with the CVR score, ballot score and overstatement.*

Assertion	*CVR Score*	*Ballot Score*	*Overstatement*
Chuan NEB Alice			
Chuan NEB Diego			
NEN: Chuan > Bob if only {Alice, Bob, Chuan} remain			

Exercise 10. *Suppose the CVR is (B) and the ballot is (B, A, C, D). Fill in the table below with the CVR score, ballot score and overstatement.*

Assertion	*CVR Score*	*Ballot Score*	*Overstatement*
Bob NEB Chuan			
NEN: Alice $>$ Chuan if only {Alice, Bob, Chuan} remain			
NEN: Alice $>$ Chuan if only {Alice, Chuan, Diego} remain			
NEN: Chuan $>$ Bob if only {Alice, Bob, Chuan} remain			

Chapter 6

Using RAIRE to generate assertions

This chapter describes the *RAIRE assertion generation* module in Figure 1.1.

Sometimes the very simple assertions that we have seen thus far can be picked by a human, but sometimes an IRV election is complex enough to need automated analysis to derive a set of assertions that are true and imply that the announced winner won. To explain how RAIRE does this, let's revisit our earlier example election between Alice, Bob, Chuan, and Diego. Previously, our example involved only a handful of ballots. We now consider a larger election with thousands of ballots. Chuan is the winner of this IRV election, with Bob, Diego, and Alice eliminated in that order. We will start by outlining how the unoptimized version of RAIRE generates assertions for this election. We will then explain how a key optimization allows RAIRE to find an appropriate set of assertions more efficiently.

Example 10. *Suppose there are 4 candidates: Alice, Bob, Chuan and Diego. The votes are as follows:*

Preferences	***Count***
(C, B, A)	*5000*
(B, C, D)	*1000*
(D, A)	*1500*
(A, D)	*4000*
(D)	*2000*

Alice, Bob, Chuan and Diego initially have 4000, 1000, 5000, and 3500 votes, respectively. Bob has the smallest tally and is eliminated first, with his 1000 ballots transferred to Chuan. Alice, Chuan and Diego now have 4000, 6000, and 3500 votes, respectively. Diego is eliminated, giving 1500 votes to Alice. Chuan, still on 6000 votes, has more votes than Alice, on 5500, and is declared the winner.

To determine what assertions we need in order to to audit this IRV election, RAIRE visualizes the space of alternate election outcomes–in which a candidate other than the reported winner wins–as a collections of trees, as described on p. 9. Then, RAIRE generates assertions that rule out all outcomes in these

'alternate outcomes trees' by attacking each of their branches at their *weakest point*. To do so, RAIRE uses the election's CVRs to assess which branches are most likely to be ruled out with a small sample.

For elections with a small number of candidates, we can easily create a complete set of elimination trees, visualizing all possible elimination orders ending with an alternate winner. For large elections, it would be computationally intractable to build and analyse full elimination trees. RAIRE applies a method that allows us to partially construct these trees, while at the same time deriving assertions that demonstrate that none of the alternative outcomes is the true one.

RAIRE starts by considering only the first level of the full set of alternate outcomes, visualized in Figure 6.1. Recall that the reported winner is Chuan. We do not consider any outcome that ends with Chuan winning. Our goal is to rule out all outcomes in which Alice, Bob, or Diego win.

First Level of Alternate Outcomes Trees

Figure 6.1: First level of nodes in our alternate outcome tree set for the election in Example 10.

We can attack nodes at the first level of our tree set with NEB assertions. If we can show that there is a candidate that *cannot be eliminated before* Alice, for example, this rules out any outcome that has Alice as the winner. For each node at this first level, we find the easiest to audit assertion, requiring the smallest sample size, that: holds on the basis of the CVRs; and that attacks the associated outcome. Each assertion is assigned a numeric 'difficulty', representing how 'easy' it will be to audit. We expect that assertions with a higher difficulty will require a larger ballot sample to audit. For each node, we take the easiest to audit assertion that attacks it, and assign it to the node along with its difficulty. If we cannot find an assertion to attack a node, we assign the node a difficulty of infinity (∞).

RAIRE can be used with any supplied function to compute assertion difficulty–or 'ease of auditing'. For ballot level comparison audits, one option is to use *1 divided by the diluted margin* of the assertion to estimate its difficulty. For ballot polling audits, one option is to use the formula *1 divided by the square of the diluted margin* of the assertion to estimate its difficulty.

Given a set of assertions that attacks an outcome, the 'easiest to audit' assertion is the one we expect would require the smallest sample of ballots to audit. Given two assertions, we assume that the assertion with the larger diluted margin will be the easier one to audit. For the purposes of our running example, we will define the diluted margin of an assertion with winner w and loser l as the difference in tallies of the winner and loser divided by the total number of ballots cast. Recall that when we talk about the winner and loser of an assertion, we are not referring to the ultimate winners and losers of the election – we are just referring to the candidates being compared by the assertion itself. Also recall

Assertion	Winner	Loser	Minimum Winner Tally	Maximum Loser Tally	Difficulty
Bob NEB Alice	Bob	Alice	1000	5500	–
Chuan NEB Alice	Chuan	Alice	5000	5500	–
Diego NEB Alice	Diego	Alice	3500	9000	–

(a) NEB assertions to show that Alice cannot be the winner.

Assertion	Winner	Loser	Minimum Winner Tally	Maximum Loser Tally	Difficulty
Alice NEB Bob	Alice	Bob	4000	6000	–
Chuan NEB Bob	Chuan	Bob	5000	1000	3.375
Diego NEB Bob	Diego	Bob	3500	6000	–

(b) NEB assertions to show that Bob cannot be the winner.

Assertion	Winner	Loser	Minimum Winner Tally	Maximum Loser Tally	Difficulty
Alice NEB Diego	Alice	Diego	4000	4500	–
Bob NEB Diego	Bob	Diego	1000	3500	–
Chuan NEB Diego	Chuan	Diego	5000	7500	–

(c) NEB assertions to show that Diego cannot be the winner.

Table 6.1: Potential NEB assertions to show that (a) Alice, (b) Bob, and (c) Diego cannot be the last candidate standing. An NEB assertion between a winner w and a loser l can be formed when the maximum tally of the loser (l) is greater than the minimum tally of the winner (w). A '–' in the Difficulty column means that this property does not hold for the associated assertion, and it *cannot* be formed.

that what ballots sit in the respective tallies of this winner and loser when performing this comparison depends on the rules of the assertion. In our example, the total number of cast ballots is 13500. Let's also assume that we are undertaking a ballot level comparison audit, and use *1 divided by the diluted margin* to estimate the auditing difficulty of a given assertion.

RAIRE first considers node 1 (Alice) in Figure 6.1, and looks for NEB assertions to show that there is a candidate that could never be eliminated while Alice remains standing, as shown in Table 6.1a.

Table 6.1a looks at all NEB assertions that could possibly be formed with Alice as the loser. Recall that NEB assertions compare the smallest possible tally the winner (in the context of the assertion) could have against the largest possible tally that the loser could ever attain. In the first entry of Table 6.1a, we have chosen Bob as the winner. Bob's minimum tally is his first preference count of 1000 votes. Alice's maximum tally counts all ballots where Alice is ranked higher than Bob, or where Alice is ranked while Bob has not been preferenced on the ballot. Alice's maximum tally in this context is 5500 ballots. This includes the 4000 (A, D) ballots and the 1500 (D, A) ballots. All other ballots either rank Bob higher than Alice, or do not preference Alice at all. It is clear that we can not form any NEB assertions to rule

out Alice as a winner, as Alice's maximum tally in all three possibilities is greater than the minimum tallies of Bob, Chuan, and Diego. So, at this point, RAIRE will assign node 1 (Alice) a difficulty of infinity.

RAIRE will now move its attention to node 2 (Bob). Table 6.1b shows that we can form *one* NEB assertion stating that *Chuan NEB Bob* (i.e., Chuan will always have more votes than Bob, irrespective of who is eliminated prior). The maximum tally Bob can have while Chuan is still standing is 1000 votes. Bob will have the 1000 (B, C, D) ballots. Chuan's minimum tally is his fist preference tally of 5000 votes. RAIRE will assign the assertion *Chuan NEB Bob* a difficulty of 1/((5000-1000)/13500) = 3.375.

Based on the above, RAIRE will assign node 2 (Bob) a difficulty of 3.375, and keep a record of the best assertion it found to rule out the outcome where Bob wins (*Chuan NEB Bob*).

RAIRE will now move its attention to node 3 (Diego). Table 6.1c shows that RAIRE cannot form any NEB assertions at this stage to show that Diego could not be the eventual winner. RAIRE will assign node 3 (Diego) a difficulty of infinity. Figure 6.2 shows the current state of our alternate outcomes trees.

First Level of Alternate Outcomes Tree Set – Evaluated

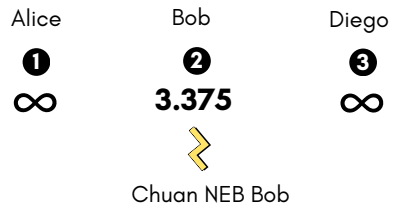

Figure 6.2: Evaluation of the first level of nodes in our alternate outcomes tree set.

RAIRE now considers all nodes in our trees that do not currently have any children, but are not complete outcomes. We call these nodes *expandable*. In Figure 6.2, nodes 1, 2, and 3 are all expandable. Of these expandable nodes, RAIRE selects the node with the highest difficultly estimate. In this case, RAIRE could select either node 1 (Alice) or node 3 (Diego), but will select node 1 (Alice) as it was 'created' first. RAIRE will add a child under Alice in our tree for every possible candidate that could be her runner-up. We call this process *expansion*. Bob, Chuan, and Diego are the three candidates that could be Alice's runner-up. Three new nodes are added to our tree (Figure 6.3), one for each of these candidates. Node 4 (Bob, Alice), for example, represents an outcome that ends with Bob as the runner-up and Alice as the winner.

RAIRE will now look at nodes 4, 5, and 6, in turn, and search for assertions that would rule out the outcomes that they represent. When we talk about a node that is not at the top level of one of our alternate outcomes trees, we will often refer to the *elimination order* or set of orders that it represents. Node 4, for example, represents the set of outcomes in which Bob is the runner-up, and Alice is the winner. We will often describe such nodes with a sequence, e.g., (Bob, Alice). When these sequences do

Expansion of Node 1

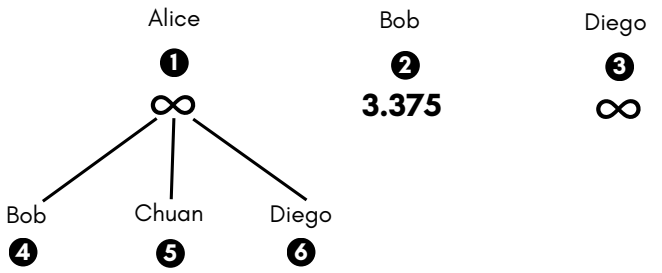

Figure 6.3: 'Expansion' of node 1, creating three new nodes in our alternate outcomes tree set.

not contain all candidates, they represent the *tail end* of an elimination order. Here, the sequence (Bob, Alice) does not mean that Bob is eliminated first, and Alice is eliminated next – it means that Alice is the last candidate standing, and Bob is the runner-up. The sequence (Chuan, Bob, Alice) refers to the set of outcomes where Chuan is eliminated second, Bob is eliminated next, and Alice is the ultimate winner. Finally, the sequence (Diego, Chuan, Bob, Alice) represents one specific elimination order where Diego is eliminated first, Chuan next, Bob as the runner-up, and Alice as the winner.

Now that we are looking at nodes that make statements about multiple candidates (i.e., they describe an elimination order 'tail' sequence with multiple candidates in it) RAIRE will consider both NEB and NEN assertions when trying to rule out the set of outcomes the nodes represent. For node 4, RAIRE will first consider whether it can show that one of the candidates not present in the branch between node 1 and 4, Chuan and Diego, could not be eliminated before Bob. If so, then RAIRE may form the assertions *Chuan NEB Bob* or *Diego NEB Bob*. Either would rule out the outcome where Bob is the runner-up and Alice is the winner. Table 6.1b tells us that we can form *Chuan NEB Bob*, of cost 3.375, but not *Diego NEB Bob*. Next, RAIRE will consider whether, at the point where only Alice and Bob remain, Bob actually has more votes than Alice. In this context, Alice has 5500 votes and Bob has 6000. So, we can form *NEN: Bob > Alice if only {Bob, Alice} remain* with a cost of $1/(500/13500) = 27$. The NEB assertion is a cheaper way of attacking the outcome, and so we assign it, and its cost, to node 4. This is shown in Figure 6.4.

This process of looking for NEB and NEN assertions to attack a node is called *evaluation*. To summarise, evaluation involves: looking for NEB assertions to attack the node's outcome; looking for NEN assertions to attack the node's outcome; selecting the least cost of these assertions (if any were found); and assigning the chosen assertion to the node with its associated difficulty, or assigning a difficulty of infinity if no assertions were found. The process of evaluating a node is specified more formally in Listing 6.1.

RAIRE will repeat this evaluation process, as described in Listing 6.1, for nodes 5 (Chuan, Alice), and 6 (Diego, Alice). For node 5, RAIRE first considers whether it can show that Bob or Diego cannot be eliminated before Chuan (Step 1 of Listing 6.1). Neither *Bob NEB Chuan* or *Diego NEB Chuan* can be

Listing 6.1: Process RAIRE uses to evaluate a node.

Let N denote an arbitrary node representing the tail end of a possible elimination order:

$$\pi = [c,\ldots,w]$$

where $c,\ldots,w$ is a sequence of candidates, and w the eventual winner.

RAIRE applies a specific process for generating assertions to attack this node.

1. RAIRE will look at the last candidate that was added to the front of the node's elimination sequence, c. For all candidates o, that are *not mentioned* in this elimination sequence, RAIRE will determine if it can form the assertion o *NEB* c. If this assertion can be formed, then it provides one way of ruling out the outcome that N represents, as it is not possible to have an outcome where o appears before c in the elimination sequence. RAIRE will record the difficulty of these NEB assertions.

2. Next, RAIRE will look at how many votes each candidate in π has in the context where all candidates *not mentioned* in π have been eliminated (i.e., π represents the set of candidates still standing or active). If c has *more* votes than some other active candidate, o', in π, then we can form the assertion *NEN:* $c > o'$ *if only* $\{\pi\}$ *remain*.

3. RAIRE will choose the least difficult of the NEB and NEN assertions it was able to form, and assign this difficulty, and the assertion itself, to N. If RAIRE was not able to form any assertions, N will be assigned a difficulty of infinity.

formed. For *Bob NEB Chuan*, Bob's minimum tally is 1000, and Chuan's maximum tally, while Bob remains as a continuing candidate, is 5000. For *Diego NEB Chuan*, Diego's minimum tally is 1500, and Chuan's maximum tally while Diego remains is 6000. RAIRE next considers whether Chuan has more votes than Alice in the context where only Alice and Chuan remain (Step 2 of Listing 6.1). In fact, Chuan would have 6000 votes and Alice 5500. So, we can form the assertion *NEN: Chuan > Alice if only {Chuan,Alice} remain*, of cost $1/(500/13500) = 27$. Node 5 will be assigned a difficulty of 27 (Step 3 of Listing 6.1).

Repeating this process for node 6 (Diego, Alice), RAIRE considers whether it can show that either Chuan or Bob cannot be eliminated before Diego. We know from Table 6.1c that neither *Chuan NEB Diego* or *Bob NEB Diego* can be formed. RAIRE next considers whether Diego has more votes than Alice in the context where only they remain standing. In this context, Alice has 9000 votes and Diego has 4500. So, we cannot form the assertion *NEN: Diego > Alice if only {Diego,Alice} remain* and node 6 is assigned a difficulty of infinity. Figure 6.4 shows the current state of our set of alternate outcomes trees.

Expansion of Node 1 – Evaluation

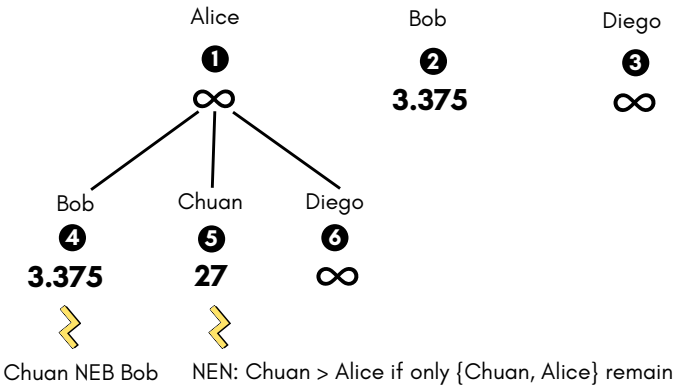

Figure 6.4: Evaluation of nodes 4-6 in our alternate outcomes tree set.

Expansion of Node 3 – Evaluated

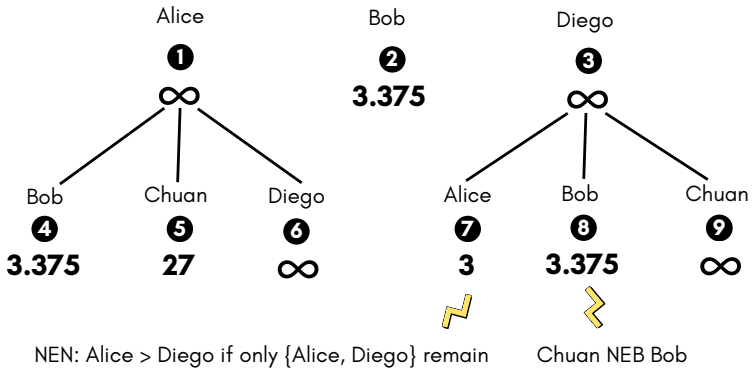

Figure 6.5: Alternate outcomes trees after the expansion of node 3 (Diego) and evaluation.

We are again in a context where all childless nodes in our trees have been evaluated. RAIRE now looks at all nodes in our tree set that do not currently have children, but are not complete outcomes–the set of *expandable* nodes–and selects a node with the highest difficultly estimate. RAIRE will select node 3 (Diego) with its difficulty of infinity. It could have selected node 6 (Diego, Alice), but node 3 was created earlier and RAIRE uses this as its tie-breaking rule. RAIRE will add a child under Diego for every possible candidate that could be his runner-up. These candidates are Alice, Bob, and Chuan. RAIRE will evaluate these new nodes (7-9) following the process in Listing 6.1. The result is shown in Figure 6.5.

RAIRE continues its process of selecting a node with the highest assigned difficulty, expanding it by adding a child for every candidate not mentioned in the node's elimination sequence, and evaluation

Expansion of Node 6 – Evaluated

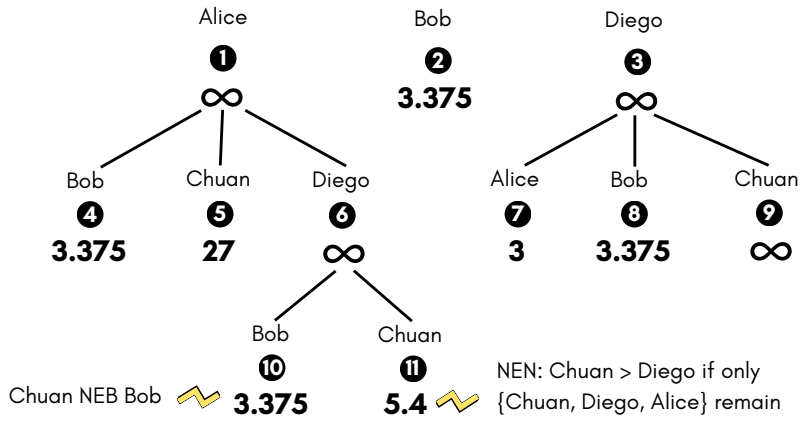

Figure 6.6: Alternate outcomes tree set after expansion of node 6.

Expansion of Node 9 – Evaluated

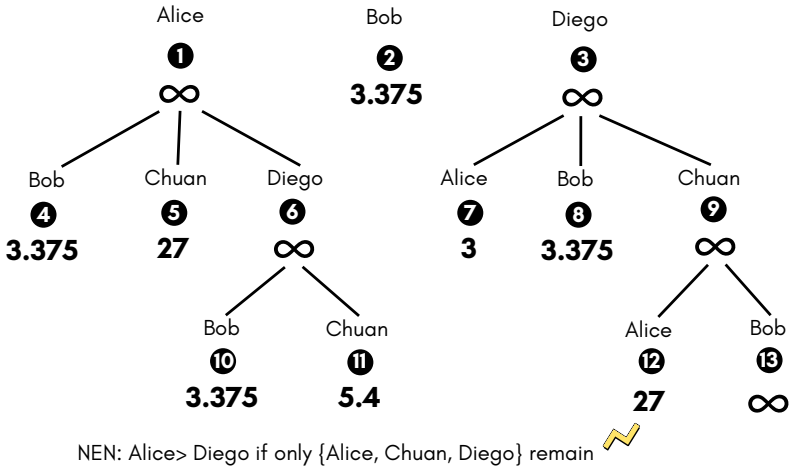

Figure 6.7: Alternate outcomes tree set after expansion of node 9.

of the newly created children. Figure 6.6 shows our set of alternate outcomes trees after RAIRE has expanded node 6 (Diego, Alice), with its difficulty of infinity, and has evaluated each of its children.

RAIRE will next expand node 9 (Chuan, Diego), with its difficulty of infinity, and evaluate its children. Figure 6.7 shows the resulting state of out set of alternate outcome trees.

RAIRE shifts its attention to node 13 (Bob, Chuan, Diego), as this node has the highest difficulty. This node has only one child, which is added to our trees as node 14 (Alice, Bob, Chuan, Diego). Only one

Expansion of Node 13 – Evaluated

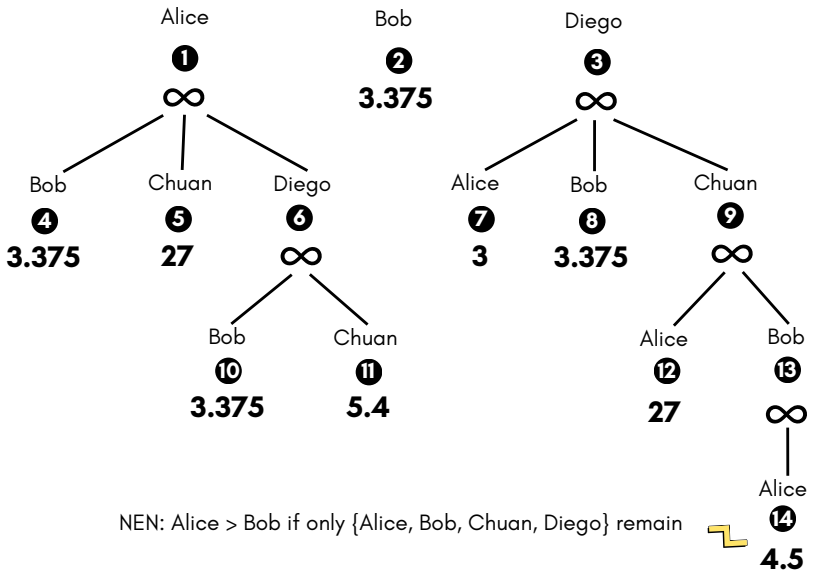

Figure 6.8: Alternate outcomes tree set after expansion of node 13.

assertion can be formed to attack this node: *NEN: Alice > Bob if only {Alice,Bob,Chuan,Diego} remain*. Alice has a tally of 4000 ballots in the context where no candidate has yet been eliminated, and Bob has a tally of 1000 ballots. The assertion is assigned a difficulty of $1/(3000/13500) = 4.5$. Figure 6.8 shows the current state of our set of alternate outcomes trees. Node 14 (Alice, Bob, Chuan, Diego) is a *leaf* node as it represents a complete outcome. Listing 6.2 describes how RAIRE uses leaf node evaluations to avoid having to explore the entire alternate outcomes tree.

Now that RAIRE has created and assigned a difficulty to *leaf* node, it has a better idea of how difficult the overall audit will be. RAIRE knows that it will have to rule out the alternate outcome with the elimination sequence Alice, Bob, Chuan, Diego somehow. It can only do this by attacking one of the nodes on the *branch* starting at the top of the tree and ending at node 14. This branch contains the nodes 3 (Diego), 9 (Chuan, Diego), 13 (Bob, Chuan, Diego), and 14 (Alice, Bob, Chuan, Diego). The cheapest, and in this case only, way to do this is to attack the leaf (node 14) with the assertion *NEN: Alice > Bob if only {Alice,Bob,Chuan,Diego} remain*. RAIRE adds this assertion to our audit, which now has a cost of 4.5.

Our Audit *NEN: Alice > Bob if only {Alice,Bob,Chuan,Diego} remain*
Cost: 4.5

Listing 6.2: Establishing a lower bound on the audit cost

RAIRE applies a key principle to avoid having to explore entire alternate outcomes trees. RAIRE's aim is to form a set of least-cost assertions while minimising how much of each alternate outcomes tree we need to explore. Visiting every node in these trees, for real-world elections, would be intractable. The method needs to return a set of assertions, or indicate that a full manual count is required, in a reasonable amount of time. The key mechanism RAIRE uses to do this is to maintain, and continually update, a *lower bound* on the difficulty of our audit.

RAIRE starts with a lower bound of zero, and revises this bound whenever we reach a *leaf* in our exploration of the trees. Recall that at a leaf, it becomes clear which node, N, is the *weakest point* in its branch. If the cost of the assertion used to attack the branch at N is C, then our audit must have a cost of at least C, as we have found the least cost way of ruling out all outcomes (branches of the tree) that pass through this weakest point, N. If C is *higher* than our current lower bound, we revise the bound to be equal to C. We then remove node N, and all of its descendants, from our alternative outcomes trees, and add the assertion that attacks N to our audit. We then look at all of the branches that remain in our trees, and whether their current weakest points have attacking assertions with costs that are less than or equal to C.

We repeat this process of adding these assertions to our audit, and removing these weakest point nodes, and their descendants, from our trees.

RAIRE then looks at any additional outcome that it knows can be ruled out with a cost of 4.5 or less. The outcomes represented by nodes 2 (Bob), 4 (Bob, Alice), 7 (Alice, Diego), 8 (Bob, Diego), and 10 (Bob, Diego, Alice) can all be ruled out with assertions that have a cost less than 4.5. RAIRE will now add the assertions used to attack these nodes to our audit.

Our Audit *NEN: Alice > Bob if only {Alice,Bob,Chuan,Diego} remain*
Chuan NEB Bob
NEN: Alice > Diego if only {Alice,Diego} remain

Cost: 4.5

Once these alternate outcomes have been ruled out, RAIRE can *prune* them from our set of alternate outcomes trees. *Pruning* essentially means that RAIRE removes those nodes from our trees. When all descendants of a node have been removed, we also remove that node from our set of tree. After this first pruning, Figure 6.9 shows the state of our set of alternate outcomes trees.

RAIRE will now take the slimmed down set of alternate outcomes trees, and continue its process of expanding the node with the highest difficulty, adding children to the trees, and evaluating them. Looking

at the trees in Figure 6.9, RAIRE will next expand node 5 (Chuan, Alice), creating nodes 15 (Bob, Chuan, Alice) and 16 (Diego, Chuan, Alice). Both nodes will be assigned a difficulty of infinity (Figure 6.10).

Alternate Outcomes Tree Set – First Pruning

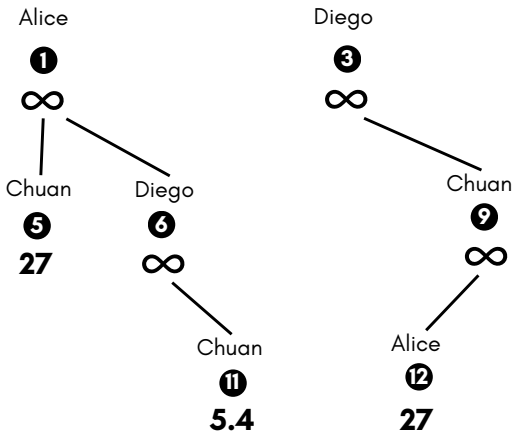

Figure 6.9: Alternate outcome tree set after their first pruning.

Expanding node 15 (Bob, Chuan, Alice), with difficulty infinity, adds one new node, node 17 (Diego, Bob, Chuan, Alice), to our set of trees. The best assertion RAIRE can create to attack node 17 is *NEN: Diego > Bob if only {Diego, Bob, Chuan, Alice} remain*, with difficulty 5.4. Figure 6.10 shows the state of our alternate outcomes trees at this point. Node 17 is a leaf node representing an outcome with the elimination sequence Diego, Bob, Chuan, Alice. RAIRE now needs to rule out this outcome, somewhere on the branch involving nodes 1 (Alice), 5 (Chuan, Alice), 15 (Bob, Chuan, Alice), and 17 (Diego, Bob, Chuan, Alice). It could do this with the assertion *NEN: Diego > Bob if only {Diego, Bob, Chuan, Alice} remain*, attacking node 17, which would result in nodes 17 and 15 being removed from our trees at a cost of 5.4.

Alternatively, RAIRE could use *NEN: Chuan > Alice if only {Chuan,Alice} remain* to attack the branch leading to node 17 at node 5 (Chuan, Alice), at a cost of 27. This would result in node 5, and all of its descendants (nodes 15-17), being removed. The attack at node 17 is cheaper, however, and RAIRE will choose the cheaper option. The assertion *NEN: Diego > Bob if only {Diego, Bob, Chuan, Alice} remain* is added to our audit, increasing its cost to 5.4, and nodes 17 and 15 are removed from out tree set.

Our Audit *NEN: Alice > Bob if only {Alice,Bob,Chuan,Diego} remain*
Chuan NEB Bob
NEN: Alice > Diego if only {Alice,Diego} remain
NEN: Diego > Bob if only {Diego, Bob, Chuan, Alice} remain

Cost: 5.4

Alternate Outcomes Tree Set – Expansion of Node 15

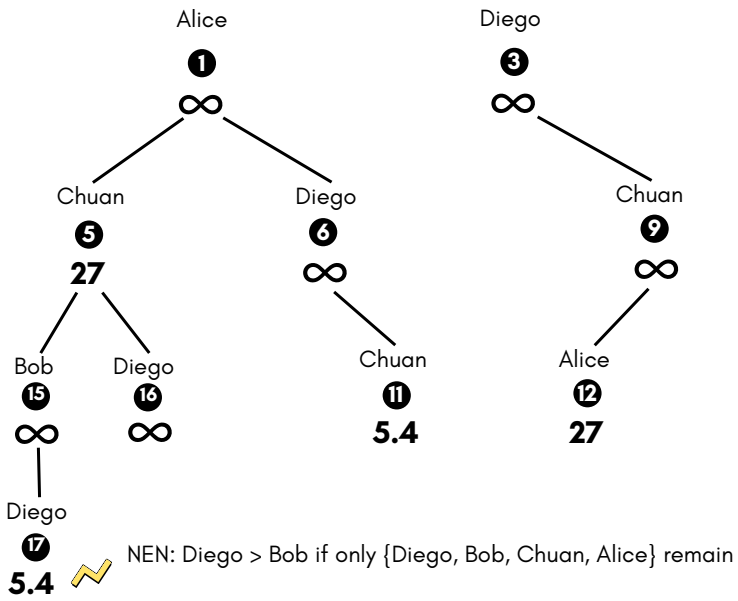

Figure 6.10: Alternate outcome tree set after expansion of node 15.

RAIRE increases its lower bound on expected audit cost to 5.4, and looks at what alternate outcomes it can rule out with a cost of 5.4 or less. We can rule out node 11, the outcome (Chuan, Diego, Alice), with an assertion of cost 5.4. This assertion, *NEN: Chuan > Diego if only {Chuan,Diego,Alice} remain*, is added to our audit and nodes 11 and 6 are removed from our trees.

RAIRE then finds another node to expand. It selects node 16 (Diego, Chuan, Alice) as it has the highest difficulty (of infinity). Node 18 (Bob, Diego, Chuan, Alice) is added to the tree, as shown in Figure 6.11. It will be assigned a difficulty of infinity as we cannot show that Bob has more votes than any of the other candidates at the point where no one has been eliminated. We know that we need to rule this outcome out, and the only way to do so is to attack node 5 (Chuan, Alice) with the assertion *NEN: Chuan > Alice if only {Chuan,Alice} remain*, at cost 27. This assertion is added to our audit.

Our Audit
NEN: Alice > Bob if only {Alice,Bob,Chuan,Diego} remain
Chuan NEB Bob
NEN: Alice > Diego if only {Alice,Diego} remain
NEN: Diego > Bob if only {Diego, Bob, Chuan, Alice} remain
NEN: Chuan > Diego if only {Chuan,Diego,Alice} remain
NEN: Chuan > Alice if only {Chuan,Alice} remain

Cost: 27

Alternate Outcomes Tree Set – Expansion of Node 16

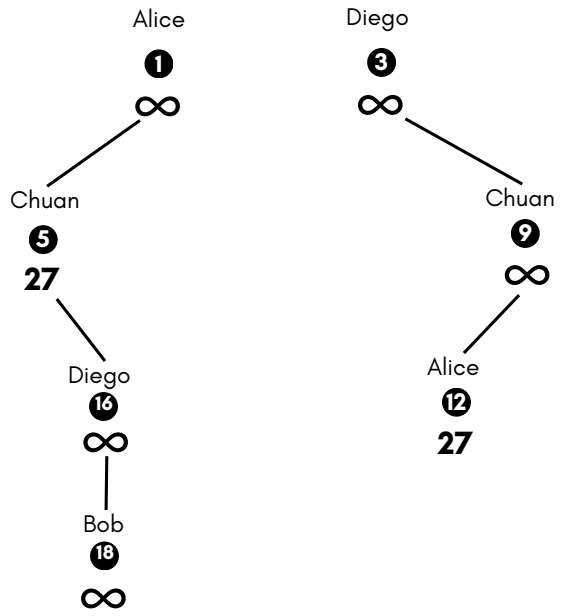

Figure 6.11: Alternate outcome tree set after expansion of node 16.

RAIRE increases its lower bound on expected audit cost to 27, and looks at what alternate outcomes it can rule out with a cost of 27 or less. It can rule out the single remaining outcome–the branch starting at node 3 (Diego) an ending at node 12 (Alice, Chuan, Diego)–by including the assertion that attacks node 12. This assertion is *NEN: Alice > Diego if only {Alice,Chuan,Diego} remain*. Adding this assertion to our audit gives us a set of assertions that rule out all outcomes in which the reported winner did not win.

Our Audit
NEN: Alice > Bob if only {Alice,Bob,Chuan,Diego} remain
Chuan NEB Bob
NEN: Alice > Diego if only {Alice,Diego} remain
NEN: Diego > Bob if only {Diego, Bob, Chuan, Alice} remain
NEN: Chuan > Diego if only {Chuan,Diego,Alice} remain
NEN: Chuan > Alice if only {Chuan,Alice} remain
NEN: Alice > Diego if only {Alice,Chuan,Diego} remain

Cost: 27

In this example, we have a redundant assertion: *NEN: Diego > Bob if only {Diego, Bob, Chuan, Alice} remain*, of cost 5.4. At the time this assertion was added to our audit, we were trying to see if we could avoid the more expensive assertion, *NEN: Chuan > Alice if only {Chuan,Alice} remain*, costing 27. The latter assertion rules out all outcomes that end with Chuan as the runner-up and Alice as the winner.

The former assertion rules out a specific outcome, one where Diego is eliminated first, Bob next, with Chuan as the runner-up, and Alice as the winner. Once *NEN: Chuan > Alice if only {Chuan,Alice} remain* has been added to our audit, there is no need for *NEN: Diego > Bob if only {Diego, Bob, Chuan, Alice} remain* to remain as the outcome it was targeting is also ruled out by the more costly assertion.

Refining our Assertion Set RAIRE currently uses two rules for detecting certain kinds of redundancies in an assertion set. These rules are described below.

1. Each NEN assertion is designed to rule out outcomes where a specific candidate w is eliminated at an elimination step when candidates $\mathcal{S}$ remain standing. Consider an NEB assertion with winner w and loser l. If w is eliminated before l in all outcomes that the NEN assertion rules out, then the NEB makes the NEN assertion redundant. For example, let's consider the following two assertions:

   Chuan NEB Bob
   NEN: Chuan > Bob if only {Chuan,Bob,Alice} remain

   The NEB rules out all outcomes in which Chuan is eliminated while Bob remains. The NEN rules out a smaller set of outcomes where Chuan is eliminated at the point where Chuan, Bob, and Alice remain. We can see that this smaller set of outcomes is a subset of those that the NEB rules out.

   Consider a different NEN assertion: *NEN: Chuan > Alice if only {Chuan,Bob,Alice} remain*. This is also made redundant by the assertion *Chuan NEB Bob*.

   We say that the assertion *NEN: $w > l$ if only $\{\mathcal{S}\}$ remain* is made redundant by any *w' NEB l'* for which: w' does not appear in $\mathcal{S}$ but l' does; or where $w' = w$ and l' appears in $\mathcal{S}$.

2. The assertion *NEN: $w > l$ if only $\{\mathcal{S}\}$ remain* makes the assertion *NEN: $w' > l'$ if only $\{\mathcal{S}'\}$ remain* redundant, and vice versa, if $w = w'$ and the set $\mathcal{S}$ is equal to the set $\mathcal{S}'$. Note that when viewed as *sets*, there is no ordering among the candidates in $\mathcal{S}$ and $\mathcal{S}'$. They will be viewed as equal if they contain the same candidates. By this rule, the assertions *NEN: Chuan > Bob if only {Chuan,Bob,Alice} remain* and *NEN: Chuan > Alice if only {Bob,Chuan,Alice} remain* are essentially equivalent – they both rule out the same set of outcomes. We don't need both of them in an audit.

Neither of these rules will detect the redundancy of *NEN: Diego > Bob if only {Diego, Bob, Chuan, Alice} remain* given the presence of the assertion *NEN: Chuan > Alice if only {Chuan,Alice} remain*. The reason is that while in this instance the first NEN assertion was generated to attack a specific outcome, (Diego, Bob, Chuan, Alice), and is not being used to rule out anything else, the assertion itself can be used to rule out outcomes that *NEN: Chuan > Alice if only {Chuan,Alice} remain* does not. For example, *NEN: Diego > Bob if only {Diego, Bob, Chuan, Alice} remain* will also rule out the outcome (Diego, Bob, Alice, Chuan). So, the outcome set that *NEN: Diego > Bob if only {Diego, Bob, Chuan, Alice} remain* can rule out is not a subset of that which *NEN: Chuan > Alice if only {Chuan,Alice} remain* rules out.

Optimizing RAIRE

We have incorporated a key optimization within RAIRE – called *diving* – to reduce the number of nodes it has to evaluate in order to form a least-cost assertion set. *Diving* comes into play when RAIRE *expands* a node. Let's cast our minds back to Figure 6.2, where RAIRE had added the first level of nodes to our set of alternate outcome trees, and evaluated them. In this first level, we had node 1 (Alice) with a difficulty of infinity, node 2 (Bob) with a difficulty of 3.375, and node 3 (Diego) with a difficulty of infinity. Node 1 was first selected for expansion. Before we expand it fully, we perform a *dive*.

Figure 6.12 shows the result of this dive. We take our to-be-expanded node and consider only *one* of its children. We choose a runner up for Alice that will allow us to most closely replicate the reported outcome. Chuan lasts longer than Bob and Diego in the reported outcome, and so we add the node (Chuan, Alice) to our tree as node 4. We then evaluate the added node. The best assertion we can find to attack node 4 is *NEN: Chuan > Alice if only {Chuan,Alice} remain* with a cost of 27. We then take the node we just added (node 4) and consider *one* of its children. Diego lasts longer than Bob, and so we add the node (Diego, Chuan, Alice) to our tree as node 5. When evaluating node 5, RAIRE finds it cannot form any assertions to attack it, so it is assigned a difficulty of infinity. We then take the node we just added (node 5), and consider *one* of its children. There is only one candidate left, and we add the node (Bob, Diego, Chuan, Alice) to our tree set as node 6. When RAIRE evaluates this node, it finds it cannot find an assertion to attack it. Diving continues like this until we reach a leaf, or a node that can be attacked with an assertion whose cost is less than or equal to the current lower bound on audit cost.

Alternate Outcomes Tree Set – First Dive

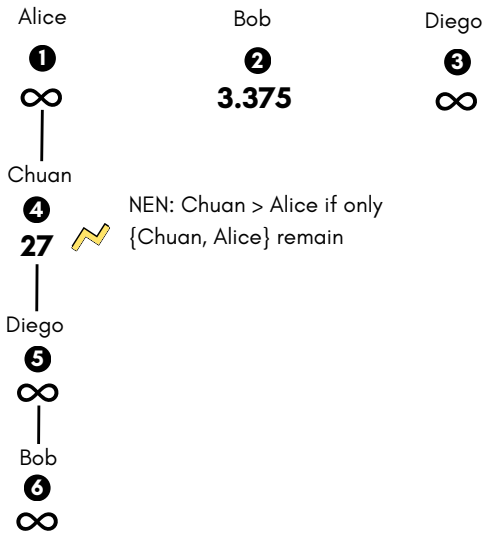

Figure 6.12: Alternate outcome tree set after first dive when expanding node 1.

Alternate Outcomes Tree Set – Expansion of Node 1

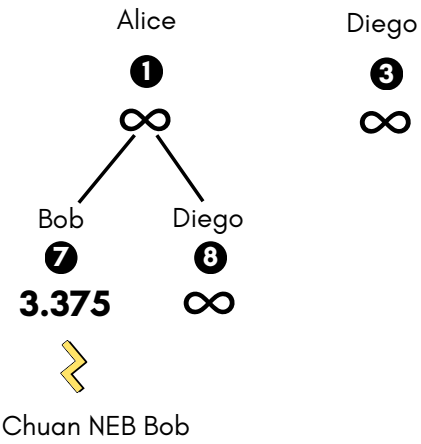

Figure 6.13: Alternate outcome tree set after expansion of node 1.

Now that we have reached a leaf node (node 6), we need to rule out the outcome represented by the branch starting at node 1 (Alice) and ending at node 6 (Bob, Diego, Chuan, Alice). The only way we can do this is with the assertion currently attacking node 4 (Chuan, Alice). This assertion is *NEN: Chuan > Alice if only {Chuan,Alice} remain* with cost 27. We add this assertion to our audit, remove nodes 4, 5 and 6 from our trees, and increase our lower bound on expected audit cost to 27. Now that RAIRE has a non-zero lower bound on expected audit cost, it looks at what additional outcomes it already knows how to rule out with a cost of 27 or less. There is one such set of outcomes – those that end with Bob as the winner. We add *Chuan NEB Bob* to our audit and remove node 2 (Bob) from our set of trees.

Our Audit *NEN: Chuan > Alice if only {Chuan,Alice} remain*
Chuan NEB Bob

Cost: 27

RAIRE will then complete the expansion of node 1, adding and evaluating its two additional children. Figure 6.13 shows the result. As we an attack node 7 with a cost that is less than our current lower bound on the cost of the audit (27), with the assertion *Chuan NEB Bob*, we remove node 7 from our trees. (As this NEB is already in our audit, we do not need to replicate it). RAIRE will then select node 3 (Diego) for expansion, first completing a dive until it reaches a node that is either a leaf or that can be attacked with an assertion that costs less than or equal to 27. Figure 6.14 shows the result.

Figure 6.15 shows the result of the completed expansion of node 3 (Diego). It's remaining two children, not formed during the dive, have been added to the tree and evaluated. Both can be attacked with assertions that have a cost less than our current audit cost. So, the assertions attacking these nodes are added to our audit, and the nodes removed from our tree.

Alternate Outcomes Tree Set – Diving from Node 3

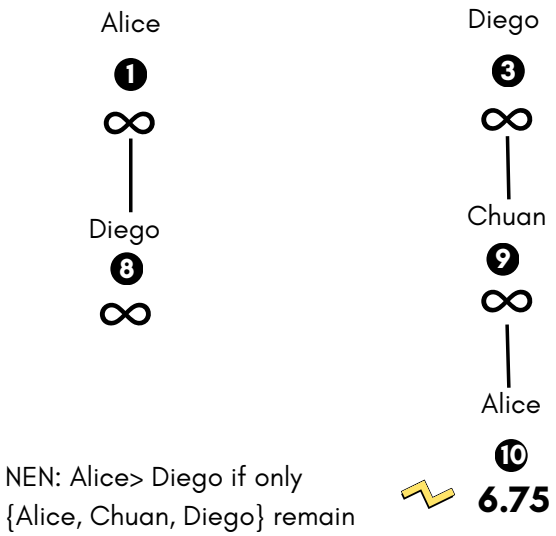

Figure 6.14: Alternate outcome tree set after diving from node 3.

Alternate Outcomes Tree Set – Expansion of Node 3

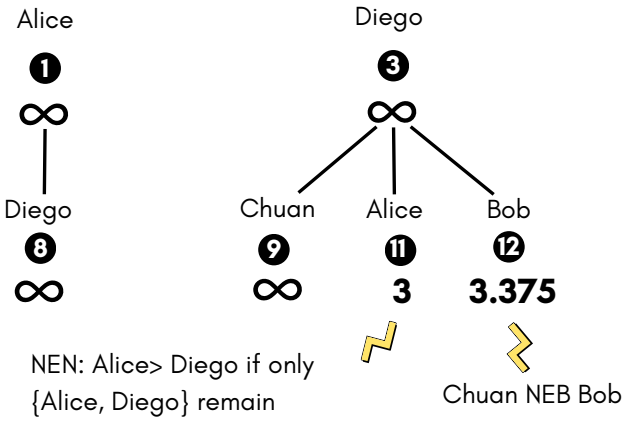

Figure 6.15: Alternate outcome tree set after expansion of node 3.

Our Audit *NEN: Chuan > Alice if only {Chuan,Alice} remain*
Chuan NEB Bob
NEN: Alice > Diego if only {Alice,Chuan,Diego} remain
NEN: Alice > Diego if only {Alice,Diego} remain

Cost: 27

Alternate Outcomes Tree Set – Diving from Node 8

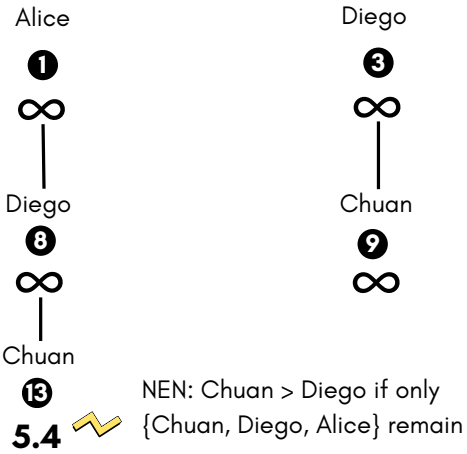

Figure 6.16: Alternate outcome tree set after diving from node 8.

Alternate Outcomes Tree Set – Expansion of Node 8

Figure 6.17: Alternate outcome tree set after the expansion of node 8.

RAIRE then selects node 8 (Diego, Alice) to expand. First, it will perform a dive, adding a child (node 13) representing the outcome (Chuan, Diego, Alice). When evaluated, this new node can be attacked by an assertion with a cost of 5.4 (*NEN: Chuan > Diego if only {Chuan,Diego,Alice} remain*). This cost is less than our current audit cost, and so diving stops. Figure 6.16 shows the state of our alternate outcome tree set at this point. We can rule out node 13, adding the attacking assertion to our audit. When RAIRE completes the expansion of node 8, adding the one remaining child (Bob, Diego, Alice), it finds this child can be ruled out with the assertion *Chuan NEB Bob* at a cost of 3.375. This is within our current audit cost, and incidentally the assertion is already in our audit, and so node 13 (along with node 8, and node 1) is removed from our tree. All outcomes in which Alice wins have now been ruled out.

Figure 6.17 shows our much simplified alternative outcome tree set.

Alternate Outcomes Tree Set – Diego ruled out as a Winner

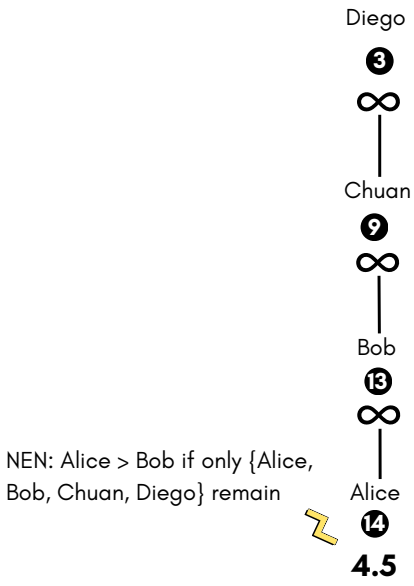

Figure 6.18: Alternate outcome tree set after expansion of nodes 9 and 13.

Our Audit *NEN: Chuan > Alice if only {Chuan,Alice} remain*
Chuan NEB Bob
NEN: Alice > Diego if only {Alice,Chuan,Diego} remain
NEN: Alice > Diego if only {Alice,Diego} remain
NEN: Chuan > Diego if only {Chuan,Diego,Alice} remain

Cost: 27

RAIRE will now select node 9 (Chuan, Diego) for expansion. This node has already been involved in a dive – that is how it was added to the tree – and so we don't perform the dive again. We simply consider the children of node 9 that were not added to our tree in the earlier dive from node 3. There is only one such child. We add a node for the outcome (Bob, Chuan, Diego) to our tree as node 13. It will be assigned a difficulty of infinity, however, as we cannot show that Bob has more votes than either Chuan or Diego at the point where only they remain. This newly added node will be the next selected for expansion. As node 13 has only one child, there is no difference between dive-then-expand and just expanding the node without diving. Node 14 (Alice, Bob, Chuan, Diego) will be added to the tree and evaluated. We can attack it with the assertion *NEN: Alice > Bob if only {Alice, Bob, Chuan, Diego} remain* of cost 4.5 (Figure 6.18). This cost is lower than our current lower bound on total audit cost, and so we add the assertion to our audit. In doing so, we rule out the last outcome in which Diego was a winner, and remove nodes 3 (Diego), 9 (Chuan, Diego), 13 (Bob, Chuan, Diego), and 14 (Alice, Bob, Chuan, Diego) from our tree set.

Our Audit *NEN: Chuan > Alice if only {Chuan,Alice} remain*
Chuan NEB Bob
NEN: Alice > Diego if only {Alice,Chuan,Diego} remain
NEN: Alice > Diego if only {Alice,Diego} remain
NEN: Chuan > Diego if only {Chuan,Diego,Alice} remain
NEN: Alice > Bob if only {Alice, Bob, Chuan, Diego} remain

Cost: 27

We have now ruled out all outcomes in which someone other than Chuan won. We have arrived at a slightly smaller set of assertions as we did without diving, and we created and evaluated four fewer nodes in the process! This doesn't sound like a substantial difference in effort, and it isn't for this small example. Diving is a crucial feature of RAIRE's implementation, however, for larger elections.

Interestingly, we did not end up with our redundant assertion, *NEN: Diego > Bob if only {Diego, Bob, Chuan, Alice} remain*, this time. When we didn't use diving, this assertion was added by RAIRE when it was uncertain about how much the eventual audit was going to cost. It could have removed more outcomes with the assertion *NEN: Chuan > Alice if only {Chuan,Alice} remain* at the time, but went for the cheaper option in case it could get away with a cheaper audit in the long run. Using the optimized version of RAIRE, we quickly realise our audit is going to cost at least 27 with our first dive, when expanding our first node. With this knowledge, RAIRE accepts the more expensive option to rule out the outcome (Chuan, Alice), without exploring that branch of our tree set any further.

Bibliography

Michelle Blom, Peter J Stuckey, and Vanessa Teague. RAIRE: Risk-Limiting Audits for IRV elections. *arXiv:1903.08804*, 2019.

Michelle Blom, Andrew Conway, Dan King, Laurent Sandrolini, Philip B. Stark, Peter J. Stuckey, and Vanessa Teague. You can do RLAs for IRV. In *E-Vote-ID*, TalTech Press Proceedings, pages 296–310, 2020.

Philip B Stark. Super-simple simultaneous single-ballot risk-limiting audits. In *EVT/WOTE*, 2010. https://www.usenix.org/legacy/events/evtwote10/tech/full_papers/Stark.pdf.

Philip B. Stark. Sets of half-average nulls generate risk-limiting audits: SHANGRLA. In *Financial Cryptography and Data Security. FC 2020*, volume 12063 of *Lecture Notes in Computer Science*, pages 319–336, 2020. Preprint: arXiv:1911.10035.

Appendix A

Specification

Note: this appendix is very technical and will eventually be separated into a distinct document. It is included now to allow CDOS developers to start considering how to integrate RAIRE audits into the existing colorado-rla codebase. A more detailed document with more specific references to existing colorado-rla java code and endpoints will be specified over the next few months.

In this section we develop a specification for the logic of the reduction from IRV to plurality audits. The reduction consists of two separate steps:

1. a reduction from each assertion to a two-candidate contest, specifying what constitutes an over- or under-vote for each situation; and
2. a way of verifying that a set of assertions implies a particular IRV winner.

Both of these are explained in the RAIRE paper Blom *et al.* [2019], but we specify the logical details here as an aid to implementation. The reduction follows the auditing nomenclature of Stark's Super Simple Simultaneous RLAs Stark [2010].

Complete assertion scoring rules

This section describes how to implement the *Assertion Discrepancy Calculator* in Figure 1.1.

First some definitions. A *Cast Vote Record (CVR)* is the electronic record of a ballot produced by the voting system, for example from a scanner. A *Manual Vote Record (MVR)* is a manually-entered record of a paper ballot, generally one sampled for audit.

An IRV vote may be invalid for various reasons, such as repeated, incomplete, or ambiguous preferences. Jurisdictions generally define some *interpretation rules* for such ballots, which specify how to transform an invalid vote into a valid one (generally, for IRV, an ordered list of non-repeated candidates, starting with the first preference). Raw CVRs and MVRs might be invalid votes that need to have the interpretation rules applied before counting. Most of the rest of this specification assumes that ballots are valid, *i.e.* have already had the interpretation rules applied.

Definition 1 (Interpretation rules). *The Interpretation rule function I takes a raw CVR or MVR $\tilde{b}$ and outputs a valid ballot. Notation:*

$$b = I(\tilde{b}).$$

The exact definition of I varies between jurisdictions according to relevant regulations.[1]

The following definitions are taken from Blom *et al.* [2019].

Definition 2 (IRV election). *An IRV election is defined as a tuple $\xi = (c, \mathcal{B})$ where c is the set of available candidates, and $\mathcal{B}$ a multiset[2] of ballots. Each ballot $b \in \mathcal{B}$ is a sequence of candidates in c, with no duplicates, listed in order of preference (most preferred to least preferred).*

We refer to sequences of candidates π in list notation (e.g., $\pi = (c_1, c_2, c_3)$), and use such sequences to represent both ballots and the order in which candidates are eliminated. We use the notation $\mathit{first}(\pi) = \pi(1)$ to denote the first candidate in a sequence π. For example, $\mathit{first}((c_1, c_2, c_3)) = c_1$.

We use the concept of *projection* to formally define a candidate's tally at any stage in the IRV counting process.

Definition 3. ***Projection*** $\mathbf{p}_{\mathcal{S}}(\pi)$ *We define the projection of a sequence π onto a set $\mathcal{S}$ as the largest subsequence of π that contains only elements of $\mathcal{S}$. (The elements keep their relative order in π). For example:*

$p_{\{c_2,c_3\}}([c_1, c_2, c_4, c_3]) = [c_2, c_3]$ *and* $p_{\{c_2,c_3,c_4,c_5\}}([c_6, c_4, c_7, c_2, c_1]) = [c_4, c_2]$.

This allows for a more formal version of the scoring table—see Table A.1.

The *Discrepancy* $\Delta_A(mvr, cvr)$ is the extent to which cast vote record *cvr* overstates assertion A compared with actual (paper) ballot *mvr*. This generalizes the notion of discrepancy for plurality audits—rather than comparing the straightforward tallies for the winning and losing candidates according to the *mvr* and *cvr*, we compare the assertion scores for the *mvr* and *cvr* using the score function from Table A.1.

[1] For example, Colorado's rules are here: https://www.sos.state.co.us/pubs/rule_making/CurrentRules/8CCR1505-1/Rule26.pdf

[2] A multiset allows for the inclusion of duplicate items.

Scoring vote $\mathbf{b} = [\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_n}]$		
Assertion A	SCORE$_A(b)$	**Notes**
c_1 *NEB* c_k where $k > 1$	1	Supports 1st prefs for c_1
c_j *NEB* c_k where $k > j > 1$	0	c_j precedes c_k but is not first
c_j *NEB* c_k where $k < j$	-1	a mention of c_k preceding c_j
NEN: $c_i > c_k$ *if only* $\{\mathcal{S}\}$ *remain*		
where $c_i = \mathit{first}(p_S(b))$	1	counts for c_i (expected)
where $\mathit{first}(p_S(b)) \notin \{c_i, c_k\}$	0	counts for neither c_j nor c_k
where $c_k = \mathit{first}(p_S(b))$	-1	counts for c_k (unexpected)

Table A.1: Complete scoring for vote $(c_1, c_2, \ldots, c_n)$ for all assertions. Over-statements are counted by subtracting the ballot paper score from the CVR score (and vice versa for under-statements). See Definition 4.

Definition 4. *For assertion A, cast vote record cvr and actual (paper) ballot mvr, the* discrepancy *is*

$$\Delta_A(mvr, cvr) = \textsc{score}_A(I(cvr)) - \textsc{score}_A(I(mvr)).$$

Assertion margins

This section describes how to implement the *Assertion Margins* component in Figure 1.1.

In order to estimate sample sizes, we need to understand the concept of "margin" as it applies to an assertion, or rather to the two-candidate contest described by that assertion.

The Assertion Margin for an assertion A is simply the sum of the scores of all ballots for A. This is the minimum number of ballots that would have to be added or removed to switch a true assertion to false.[3]

Definition 5. *The* Assertion Margin m_A *for assertion A and a set of ballots $\mathcal{B}$ is*

$$m_A(\mathcal{B}) = \sum_{b \in \mathcal{B}} \textsc{score}_A(b).$$

This margin can be used directly to calculate the *diluted margin* μ for each assertion, in order to estimate the required sample size. The diluted margin for an IRV contest is the smallest diluted margin for any of its assertions.

[3]This use of the term "margin," follows Stark's definition, i.e. the difference between the winner and the runner-up. Note that we do *not* divide the difference by 2.

Verifying that a set of assertions implies an IRV winner

This section describes the underlying logic of the *Assertion Visualizer and validator* in Figure 1.1. UI Design for presenting the assertions to users will be specified in consultation with UI design experts.

A set $\mathcal{A}$ of assertions implies that w won if no other winner is possible when all the assertions in $\mathcal{A}$ are true.

Proposition 1. *A set of assertions $\mathcal{A}$ implies that w won if for all other candidates $c_n \neq w$, for all possible elimination sequences $(c_1, c_2, \ldots, c_{n-1}, c_n)$ in which c_n wins, there exist $i < k \in [1, n]$ s.t. either*

$$\text{“NEN: } c_i > c_k \text{ if only } \{c_i, c_{i+1}, \ldots, c_n\} \text{ remain”} \in \mathcal{A}$$

or

$$\text{“}c_i \text{ NEB } c_k\text{”} \in \mathcal{A}.$$

Proof. By contradiction. Assume that there is some other possible winner l. Then l can win according to a specific elimination sequence $(c_1, c_2, \ldots, c_{n-1}, l)$. But then this sequence must satisfy one of the conditions above, making it impossible. □