# MEASURING THE COUNTER/ASSUMPTION MODEL'S EFFECT ON ARGUMENTATION QUALITY

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Evan Ovadia

December 2013

COMMITTEE MEMBERSHIP

| | |
|---|---|
| TITLE: | Measuring the Counter/Assumption Model's Effect on Argumentation Quality |
| AUTHOR: | Evan Ovadia |
| DATE SUBMITTED: | December 2013 |
| COMMITTEE CHAIR: | Franz Kurfess, Ph.D.<br>Professor of Computer Science |
| COMMITTEE MEMBER: | Aaron Keen, Ph.D.<br>Professor of Computer Science |
| COMMITTEE MEMBER: | John Patrick, Ed.D.<br>Professor of Communication Studies |

ABSTRACT

Measuring the Counter/Assumption Model's Effect on Argumentation Quality

Evan Ovadia

This thesis presents a new platform called See the Reason, built upon a tree-structured argumentation model called the Counter/Assumption model. In the Counter/Assumption model, a topic is posted first, then under that topic, reasons for and against, and for each reason, counterarguments, and for any counterargument, more counterarguments. The model enables us to systematically determine whether a claim is "tentatively true" or "tentatively false," in an effort to motivate people to make their side's claims tentatively true and the opposing side's claims tentatively false, thus encouraging conflict. Research suggests that debates with more conflict are better, so this thesis investigates whether Counter/Assumption model encourages better debates.

In this thesis, we have students debate on See the Reason and the closest existing platform, CreateDebate. We measure the number of uncaught bad arguments, the user satisfaction, and how far into the Interaction Analysis Model the debates progress. We find promising evidence that See the Reason progresses further into the IAM and encourages more logical debates, but sacrifices usability at the same time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

Debate is incredibly important in people's lives. In debate, people with opposing viewpoints clash, and create knowledge that is something more than they would have alone. Through argument, people can inform themselves and others and fill in the missing gaps of their knowledge, so they can make better decisions.

Debate and argumentation also serve an incredibly important role in our society. In business, we debate the best way to serve our clients. In philosophy, we debate the origins and meaning of life. In politics, we use debate to gauge candidates' knowledge and positions in various issues.

Two people on opposing sides of a debate will have some knowledge: some true, some false. But when they debate, they identify their opponent's false knowledge, fact-check their opponent's claims, and verify the opponent's logic by looking for flaws. Their false knowledge is identified and discarded, and in the end, both people come away more informed. In a way, their collective knowledge has evolved to be more complete and closer to the truth.

See the Reason tries to capture this interaction, this collaborative fact-checking, to make an ever-evolving fact-checked index of claims.

## 1.1 Indices of Claims

Before making a decision, it's useful to have all of the facts. For this reason, people construct entire maps of ideas related to a topic, so they can have a more complete picture of what they are talking about. For example, DebateGraph has a massive chart (Figure 1.1) called "Planet Under Pressure" which has every fact

imaginable related to Earth's environment.

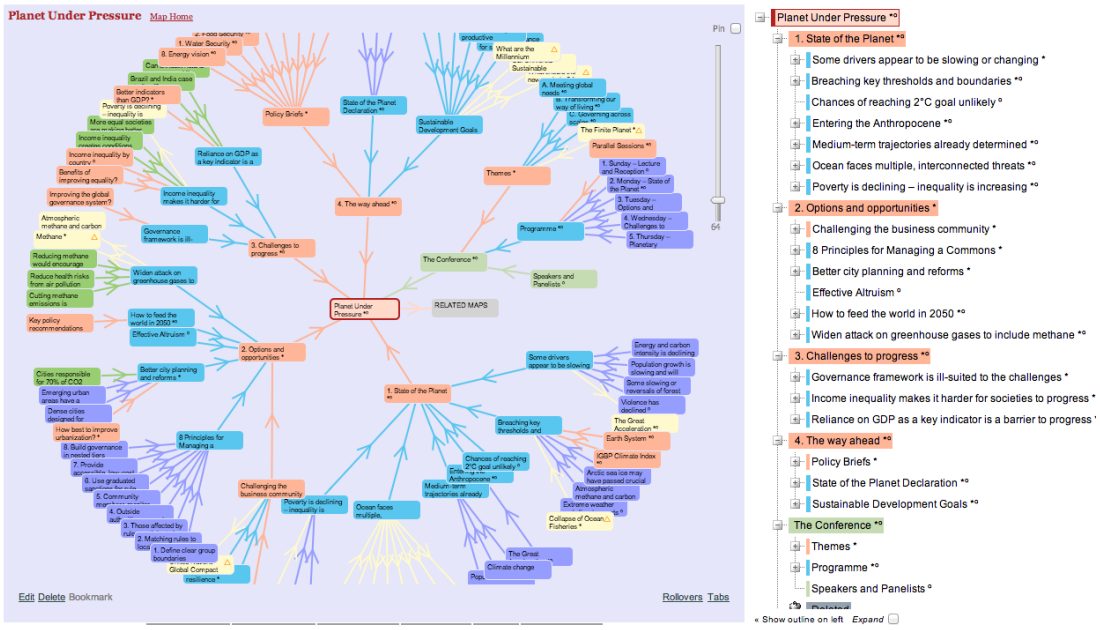


**Figure 1.1: DebateGraph showing the top 3 levels of the Planet Under Pressure map.**

The Talk Origins archive has a large index of claims (Figure 1.2) pertaining to the theory of evolution, and for every claim, it has a rebuttal, and even rebuttals of rebuttals.

These are incredibly useful resources for anyone who needs to make an informed decision or participate in a debate on the subject.

DebateGraph is especially useful tool because of its breadth; it's collaborative, so any user can contribute useful information to it. Talk Origins' list is useful because it doesn't only contain a list of facts, it contains a list of claims and rebuttals. See the Reason, in its simplest form, can be described as a combination of the two: a collaboratively constructed list of claims and rebuttals.

On top of those two basic properties, See the Reason is crafted to keep track of whether a claim is "tentatively true" or "tentatively false," in other words,

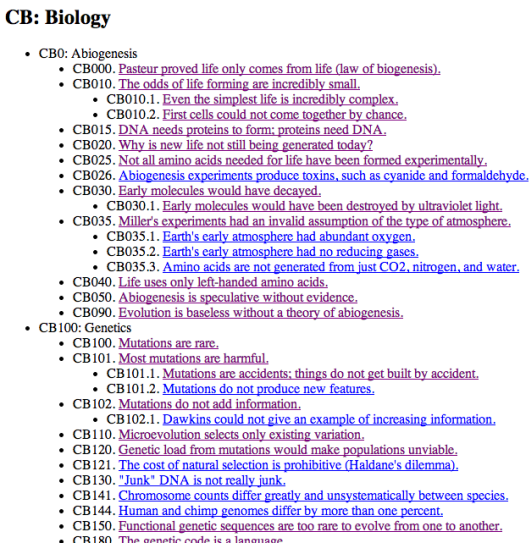


**Figure 1.2: A snapshot of Talk Origins.**

which side is winning for each given claim. Through the users' efforts of making tentatively true the claims they agree with, and making tentatively false the claims they disagree with, See the Reason also spurs them to factor out unstated assumptions, and uses those to customize the tentatively true/false judgment per-user based on their own assumptions. This is described in more detail in section 3.2.

Such a platform could be of great benefit to anyone who needs to make a decision. For example, a legislator, who soon has to make a decision on whether to enact a certain piece of legislation, can submit a topic to See the Reason, and ask his constituents to submit reasons for and against the topic. The users, in their efforts to submit reasons and keep them judged tentatively true, will unknowingly create an index of collaboratively fact-checked claims relevant to the decision. The legislator can then look at all of the claims that remain tentatively true[1], and use those to make his decision.

[1]The legislator in this example would also look at the certainty measure (see section D) to get an estimate of how true these "tentatively true" claims really are.

## 1.2 The Counter/Assumption Model

See the Reason is built upon our new argumentation representation, which we call the Counter/Assumption model. In this, there are two types of claims: objective and subjective.

In an "objective" claim, a user puts forward the facts, the assumptions, and the logic that lead to a given conclusion. Other users may then submit other objective claims in response to that objective claim, countering it. The system will look at users' inputs for assumptions, and related claims, and judge whether a claim is tentatively true or false.

A "subjective" claim has two categories of responses: reasons for, and reasons against. A "reason" is simply an objective claim, relevant to the subjective claim.

For more explanation and details on the Counter/Assumption model, see section 3.

## 1.3 Question

This thesis aims to answer this question: Does the Counter/Assumption encourage more productive arguments?

# CHAPTER 2

# PREVIOUS WORK

There are three main areas of the literature that relate to See the Reason: how to analyze debates, how to represent debates, and what moves a debate forward.

## 2.1 Analyzing Discussion

To analyze a discussion, it can be helpful to represent the arguments in a logical fashion.

One of the earliest, and still popular, schemas for representing arguments is the Toulmin model, published by Stephen Toulmin in his 1958 work "The Uses of Argument". He said that arguments could be broken down into six main types of components [56]:

- Grounds: The evidence, facts, or data from which the argument starts
- Warrant: The logical statements that imply the conclusion, given the grounds
- Backing: Evidence and reasoning that support the warrants
- Qualifier: indicates the strength or certainty with which the facts and warrants are claimed to lead to the conclusion
- Claim: The final conclusion of the argument.
- Rebuttal: Counter-arguments or statements that show the conclusion is false, or that the conclusion does not logically follow from applying the warrants to the grounds.

There are several models that build on top of the Toulmin model, such as that used by Parmenides [26]. There are also entirely different models for representing arguments, such as van Lehn's hypothesis-evidence-argument structures and EUCLID's model [20].

There are also content analysis methods for examining the quality of online discussions [42]. One such method is the Interaction Analysis Model [19], which says that discussions can go through five phases [60]:

- Phase 1: Sharing/comparing of information / statements of observation or opinion; statement of agreement between participants
- Phase 2: discovery and exploration of dissonance or inconsistency among ideas, concepts, or statements, identifying areas of disagreement; asking and answering questions to clarify disagreement
- Phase 3: negotiation of meaning / knowledge co-construction; negotiating meaning of terms and negotiation of the relative weight to be used for various arguments
- Phase 4: testing and modification of proposed synthesis or co-construction / testing the proposed new knowledge against existing cognitive schema, personal experience, or other sources
- Phase 5: phrasing of agreement and applications of newly constructed meaning / summarizing agreement and meta-cognitive statements that show new knowledge construction

Later in this thesis, we will use the IAM to approximate the quality of various debates.

Brook and Oliver (2003) explain, "The latter stages of the model require high levels of bidirectional influence between the individual and the group, an identifying characteristic of strong communities." [22]

There are other models as well, such as Henri's framework (1992) which analyzes content along five dimensions (participation, interaction, social, cognitive, metacognitive) [30], and Veerman and Veldhuis-Diermanse's model (2001) which analyzes content along eight dimensions (planning, technical, social, new facts, new experiences, new theory, explicitation, and evaluation) [58].

These models have been used throughout the literature to understand how various platforms perform in facilitating debate. Most work on this subject has been done in educational areas, to see if students could learn better and think more critically, given the right communication medium. There are three main types of communication media analyzed in the literature: threaded discussion, trees, and graphs.

## 2.2 Threaded Discussion

One of the more common types of discussion platform is the "threaded discussion." Linear conversations such as discussion boards, comment threads, and email threads fall into this category. Dollisso and Koundinya (2011) put it nicely:

> A threaded discussion is an asynchronous, web-based discussion that takes place in an on-line environment under a number of different topics that are called threads (Kirk and Orr, 2003). More simply, a threaded discussion involves posting of messages pertaining to a specific topic (Middlesex Community College, n.d.). It includes an initial message and subsequent posted responses that are sequentially linked to the initial message (Feng et al., 2006a). It is a form of conversation in which people express ideas, elaborate arguments, and answer questions of other group members (Feng et al., 2006b). [27]

Compared to discussing in real life, online threaded discussions have many advantages, including [27]:

- They improve higher-order thinking (Kirk and Orr, 2003; Meyer, 2003)
- They help students meet constructivist curricular objectives (Weasonforth and Meloni, 2002)
- They help students become participatory citizens (Larson and Keiper, 2002)
- They help build on-line learning communities (Edelstein and Edwards, 2002)
- They improve students' writing skills (Jordan, 2001)
- They help improve computer and on-line skills (Davidson-Shivers et al., 2001)
- They help facilitate student collaboration (Miller and Benz, 2008)
- They promote active and group learning (Kirk and Orr, 2003).

The IAM has been applied many times to online forums, with mixed results. Hou et al. (2008) apply the IAM to a problem solving activity, and the number of posts in the five phases of the IAM were respectively 67.45%, 8.43%, 22.75%, 1.37%, and 0%. [31] They said:

> This result shows that it is hard to achieve the C4 and C5[1] levels of knowledge construction, which is similar to other research conducted according to Gunawardena, Lowe, and Anderson's coding system (Gunawardena, Lowe, & Anderson, 1997; Jeong, 2003). Jeong (2003) also used the IAM to explore the state of knowledge construction from general online discussion activity; the teacher assigned the discussion topic, and, instead of problem-solving-based discussion activity, the students discussed the topic freely. The percentages were C1, 93.7%; C2, 2.4%; C3, 1.9%; C4, 1.0%; C5, 1.9%.

In Thanasingam and Soong's study (2007), they instructed students to post their thoughts about a presentation for 30 minutes, and then address other students' thoughts, and the analysis showed that there were 35.9%, 43.6%, 20.5%, 0%, and 0% posts in the phases respectively. [55]

In Brindley et al.'s study (2009), students met in an online forum to try to solve cases, and the were instructed to reply to at least one other post. The analysis showed that among the 5 different discussions, there were the following numbers of posts in the five phases respectively [21]:

- Case 1: 0%, 15.7%, 20%, 15.1%, 49.2%
- Case 2: 0%, 12.3%, 31.2%, 16%, 40.4%
- Case 3: 0.4%, 14.7%, 38.9%, 13%, 33%
- Case 4: 0%, 14%, 29.1%, 15.5%, 40.4%
- Case 5: 0.1%, 14.3%, 29.6%, 15, 41%

Buraphadeja (2012) had students meet in a forum to discuss case studies, and they found that their best discussion had 12.1%, 34.1%, 37.4%, 1.1%, and 15.6% posts in the five phases respectively [24].

In McLoughlin and Luca's study (2001), the students were assigned roles such as forum leader, questioner, and summariser, and there were the following numbers of posts in the five phases respectively [40]:

- Discussion 1: 73%, 25%, 6%, 3%, 0%
- Discussion 2: 65%, 23%, 10%, 3%, 0%
- Discussion 3: 66%, 21%, 9%, 4%, 2%

[1] C4 and C5 are the IAM phases 4 and 5, where participants come to a consensus or decision.

These studies show what online discussion can be like when a moderator is present, and the students are given instructions on what to do, and encouraged to interact with what the other users are saying.

### 2.2.1 Drawbacks

Even when online discussion is between students, and a moderator is present, discussion can be quite chaotic, with a poor signal-to-noise ratio. Gurkan (2010) notes,

> By far the most commonly used technologies, including wikis, blogs, and discussion forums, are what we can call sharing tools [21]. While such tools have been remarkably successful at enabling a global explosion of idea and knowledge sharing, they face serious shortcomings when applied for collective deliberation purposes. One is the signal-to-noise ratio. The content captured by forums is notorious for often being unsystematic and repetitive, making it hard for users to locate useful information. The content is also typically of highly variable quality, since online conversation tools do not inherently encourage or enforce any standard concerning what constitutes valid argumentation, so postings are often bias rather than evidence or logic-based. It becomes difficult, as a result, to separate the wheat from the chaff. [29]

Dollisso and Koundinya (2011) explain,

> Knowlton (2001) noted that online discussions could digress into chat that is not related to the intended purpose, thus hampering student learning. Consequently, not being able to maintain the focus of on-line discussions is a concern for many instructors (Gao and Wong, 2008). It has been the authors' personal experiences that some students lose focus and deviate from the discussion requirements, and can lead the discussion completely off track. [27]

Klein and Iandoli (2008) say,

> The content captured by such tools is notorious for having a poor signal-to-noise ratio, with many repetitive and low-quality posts, especially when addressing controversial issues. Coverage of a topic is

> generally unsystematic, since it is created bottom-up. Group interactions are all too easily hijacked by a narrow set of "hot" issues or loud voices, leading to such phenomena as forum "flame wars" and wiki "edit wars". OSPP[2] systems do not inherently encourage or enforce any standards concerning what constitutes valid argumentation, so postings are often bias- rather than evidence- or logic-based. Users of such systems also tend to self-assemble into groups that share the same opinions ("balkanization"), so they see only a subset of the issues, ideas, and arguments potentially relevant to a problem. [17]

Forums may be notorious for having shortcomings, but Hoven (2011) notes, "There is order and even traces of argumentative reasonableness immediately below the seemingly chaotic surface of this activity type." [32]

Many have attempted to add features to forums to amplify this argumentative reasonableness, with varying success. There are two main techniques to do this: staging and scaffolding.

### 2.3 Staged Threaded Discussion

Any threaded discussion platform can be used in a staged manner, where different kinds of posts are posted at different times.

For example, some users could be trying to decide whether to enact a piece of environmental legislation. A moderator could instruct the users:

- Days 1 and 2: Scour the entire internet and post links to all information relevant to this environmental regulation.
- Days 2, 3, and 4: Analyze the information, determine its validity, find patterns.
- Days 5, 6, and 7: Come to a consensus about whether we should enact this one environmental regulation.

Popular staged discussion platforms include:

- iPidato

[2] Open-Source/Peer-Production tools as email, instant messaging, news groups, chat rooms, blogs, wikis, podcasts, video and picture sharing sites, and the like. [17]

- debate.org
- economist.com/debate

In Dollisso and Koundinya's experiments (2011), they divided a discussion activity into two stages. In the first stage, users would talk about the case study, the relevant content, personal experiences, their perception, then would analyze, reflect, and formulate responses to it, and post their responses. In the second stage, they would look at all of their peers' responses, and the expert opinions on the subject, and would analyze, re-evaluate their position, make their decisions, post new responses, and read their peers' new responses. They note, "The findings indicated that by following this model, there was more student participation, more focused discussion, and less deviation from the intended purpose." [27]

### 2.4 Scaffolded Threaded Discussion

The second approach to improving threaded discussion is "scaffolding." Moore and Marra (2005) explain,

> Constrained or scaffolded discussion forums are pre-structured forms of conversation systems that require participants to label each of their postings from a pre-defined set of message types. [42]

Popular scaffolded threaded discussion platforms include:

- Argumentum
- ConvinceMe
- ProCon
- ForAndAgainst

Gao et al. (2013) used scaffolding, and had some success:

> Typically, it requires participants to start their notes with a predefined phase-a note starter, such as 'my argument is...' or label their notes using a predefined set of post types, such as evidence or elaboration (Oh & Jonassen, 2007). The rationale is that such structured environments can promote participants' metacognitive thinking and engage them in desired cognitive processes (Jonassen & Remidez, 2005;

> Scardamalia & Bereiter, 1994). ... In a constrained environment developed by Oh and Jonassen (2007), both post type labels (which are hypothesis cause, solution generation, verification, rebuttal, evidence and elaboration) and note starters (including "My experience is...," "I believe...," "Research shows...," "A scholar says...") were applied. By comparing the online argumentation occurred in this environment with that in a threaded forum, Oh and Jonassen (2007) concluded that participants in the constrained environment generated more evidence posts, more hypotheses and hypothesis testing posts. [28]

Brooks and Jeong (2006) also had some success and found, "pre-structured threads increased the frequency of challenges per argument by 64%, with a moderate and positive effect size of +0.47." [23]

Moore and Marra (2005) noted,

> Cho and Jonassen (2002) found that using a constraint-based argumentation scaffold positively affected the ability of groups to collaboratively construct arguments in an online environment. Students using the conversation scaffold provided more comments related to problem definition, orientation comments, criteria development, solution development, solution approval, and solution critique.

Not all the studies are unanimous in scaffolding's benefits though. Joung (2003) found,

> The results of the current study demonstrate that a high-structure cooperative learning strategy has statistically significant effects on improving the quantity of critical thinking, as well as dynamic and critical interaction patterns, but not on changes in individual decision-making.

In Moore and Mara's analysis (2005), even though scaffolding increased the percentage of posts in phase 1, it unfortunately reduced the number of posts in the other phases. [42]

### 2.4.1 Scaffolding Reduces Overall Discussion

There is also a downside to scaffolding. Jeong and Joung's (2004) study found that labeling messages reduced overall debate and reduced the proportion of

arguments that were supported with explanations. [42]

Another study by Jeong (2007) replicated the findings: "Students in the constraints-with-labels group were significantly less likely to (a) challenge other students, and (b) respond to challenges from other students." [35]

It seems that the complexity discourages posts. One student in that study said, "I might have submitted more if I were more secure in the proper method of submission." [42]

Macintosh et al. (2009) had a similar experience. They wrote,

> Research into getting argumentation systems to function efficiently has often been at the expense of refining the user interface; there is now an urgent need to address this imbalance by investigating what features are necessary for an interface if the system is to attract participants and encourage them to provide deliberated input. ... Developers need to strike a balance between imposing a formal structure upon contributions from the public, which some may find inhibiting, and providing a free text field, which imposes a considerable cost on consultation organizers in the task of extracting useful information. [38]

### 2.4.2 Remaining Weakness in Threaded Discussion

As much as staging and scaffolding may help or hurt, threaded discussions still have a fundamental problem: they are linear. Noroozi et al. (2012) explain, "Firstly, since an argument or the nature of argument is in fact complex and not linear (Toulmin, 1958), it is not a simple task to broaden and deepen the space of debate during sequential linear discussion (McCutchen, 1987)."

Dollisso and Koundinya (2011) found that because of this linear restriction, threaded discussions tend to lose focus:

> Research indicates that threaded discussions sometimes digress into chat that is not in line with the intended purpose, thus causing the discussions to lose their focus. ... Knowlton (2001) noted that online discussions could digress into chat that is not related to the intended purpose, thus hampering student learning. [27]

Gao et al. (2013) explain,

> Hewitt (2003) noticed that, because participants are more likely to respond to recent posts and less likely to revisit older posts, the excessive focus on new posts can unintentionally shift participants' attention away from discussing important issues. ... Herring (1999) believed the asynchronous threaded discussion system resulted in a high level of overlapping exchanges and topic decay. She cited Lambiase's (2010) work who found that during the first nine days of discussion, the percentage of posts on the group's global topic decreased steadily from 65% to 33%. Meaningful reflection, social interaction and knowledge construction can hardly occur when participants fail to maintain the focus of discussion on the central topics.

Gurkan et al. (2010) found that threaded discussions have a scalability problem: "Forum conversations do not scale well because it is virtually impossible for latecomers to make sense of and effectively join a conversation that has already been started by other participants." [29]

With these restrictions, many have shifted their focus away from threaded discussions and onto other, non-linear mediums, such as wikis, trees, and graphs.

### 2.5 Wikis

Wikis, such as the popular Wikipedia, can serve as a platform for argumentation. To have a debate on a wiki, the participants read the page, and if they have any new information or arguments to contribute, they edit it in. Popular platforms in this pattern include:

- DebatePedia.org
- iDebate.org's Debatabase
- DebateWise

Buraphadeja and Swapna found, "Results indicate that the wiki platform fosters collaborative knowledge construction and that is necessary to develop new frameworks to analyze content in new learning environments." [25]

Plain wikis are poor substitutes for threaded discussions in the realm of debate, because of "edit wars". While comparing forums and wikis, Gurkan (2010) said,

> Forum conversations do not scale well because it is virtually impossible for latecomers to make sense of and effectively join a conversation that has already been started by other participants. While wikis are able to cope with some of the limitations above, they tend to fare poorly when applied to controversial topics, often leading to such phenomena as "edit wars".

Edit wars are a common occurrence on even the most popular wikis, including Wikipedia. [9] An edit war occurs when one user edits into the article something he believes, someone else removes it, and the original user edits it back in, and so on.

However, this doesn't happen in wikis which have designated space for each person's arguments. DebatePedia (before it was shut down) and DebateWise have designated spaces for pro arguments and con arguments. Because of this, both sides views are represented, which avoids edit wars.

iDebate's Debatabase is similar to a wiki, in that anyone can send a request for an edit, but the edits need to be approved by a special class of users called "editors" before they are enacted. These editors also serve as a filter against bad edits. [3]

### 2.6 Trees

In tree-structured discussions, each post can have any number of replies. Another way to think of trees is that they are similar to threaded forums, but instead of only being able to put a post at the end of a thread, users can respond to any post in the middle, effectively "forking" the conversation. Popular systems implementing this pattern include:

- reddit.com
- livingknowledge.org

- ForAndAgainst

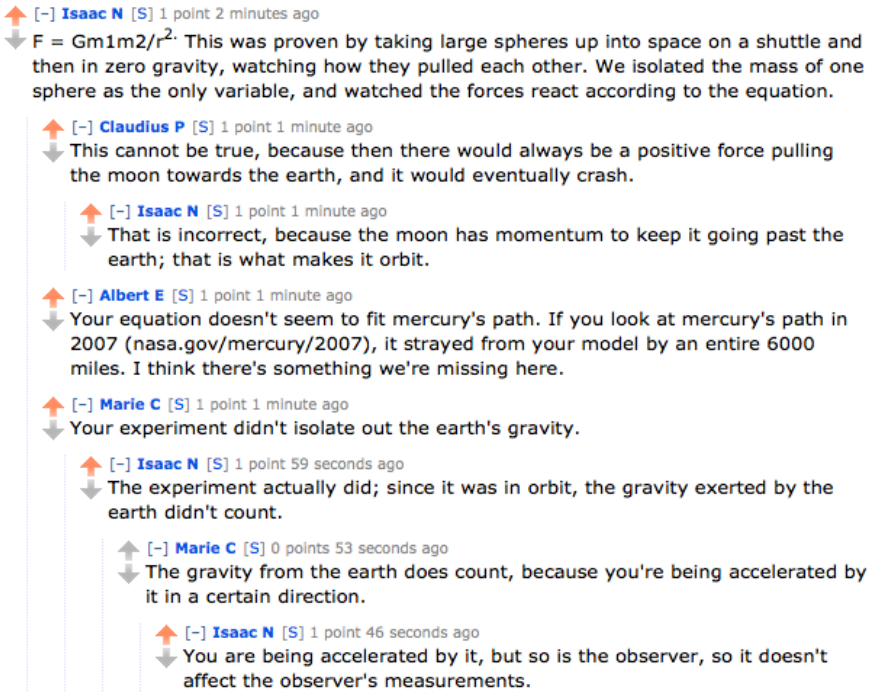


**Figure 2.1: An example debate on Reddit, a tree-structured discussion platform.**

Just like threaded discussions, tree-structured discussions can be scaffolded. Gao et al. (2013) compared them with and without scaffolding and remarked,

> In threaded forums[3], the hierarchical structure of discussion threads fails to represent the interrelationship of discussion posts. In a typical threaded forum, the hierarchical structure of the discussion only indicates the reply relationship between posts (by using indentation) and the time sequence of the replies (by showing the posts in chronological order). But Hewitt (2001) pointed out there is a significant distinction between the hierarchical structure imposed by the system and the linkages that are implicit in the text of the posts.

[3]Gao et al. are actually referring to tree-structured forums when they say "threaded forums"

## 2.7 Scaffolded Trees

Just as scaffolding can be applied to threaded discussions, it can also be applied to trees. Popular instances of this include:

- CreateDebate
- Knowledge Forum

Knowledge Forum is a very popular scaffolded tree-structured discussion board, where posts belong to a predefined set of categories. Some of those categories are:

- My theory
- I need to understand
- New information
- This theory cannot explain
- A better theory
- Putting our knowledge together

Marra et al. (2004) applied the IAM to Knowledge Forum, using the following methods:

> Discussion board participation was worth 5% of the final grade and consisted of weekly discussions on the assigned readings, instructional design topics, and case studies. For the case studies, participants were required to analyze and discuss instructional problems that emphasized one or more aspects of the instructional design process (Ertmer & Quinn, 1999). The case study activity, which was 4% of the final grade, required each student to participate on a team that analyzed one assigned case, post a set of controversial case issues to the discussion board, and facilitate the online discussion for one week. As part of the required discussion board participation, the remaining students debated the lead team's position on the controversial issues. [39]

and got the following numbers of posts in the five phases respectively [21]:

- 34%, 36%, 26%, 4%, 0%
- 21%, 36%, 38%, 4%, 0%
- 36%, 28%, 26%, 11%, 0%

Tan et al. (2008) also applied the IAM to various people discussing on Knowledge Forum. They had one group use it to discuss a topic, and another group use it to come up with a solution to a problem. Both groups had almost all posts in phase 1, and the latter group had a few posts in phases 2 and 3. One would have thought that the second group would have reached much further into the latter phases of the IAM, but the requirements, low number of posts, and the time constraints could have had a hand in suppressing that higher-order interaction. [54]

Albrecht (2003) did a study on DEMOS, a tree structured system from Hamburg, and he said,

> [T]he quality of debate was higher than one could expect from studies in computer-mediated communication, and even surpassing what could have been achieved if the experiment was conducted offline, without the help of Internet technology (for a similar consideration, see Beierle 2002: 49f.). [18]

In Sing and Khine's study (2006), they analyzed the interaction between teachers and students in a forum. They measured the number of posts in each phase respectively to be 63%, 21%, 13%, 5%, and 3%. They said:

> This is not an isolated phenomenon. Gunawardena et al.'s (1997) study obtained a result of 191; 5; 4; 2; 4 postings from Phase 1 to Phase 5 respectively. Her participants were practitioners of online education or graduate students. Schellens and Vackle (2005) used Gunawardena's model to analyze undergraduates' online postings and found 52%; 14%; 33%, 1.2% and 0.4% from Phase 1 to Phase 5 respectively. The results seem to indicate that higher phases of co-construction of knowledge are difficult to achieve. Reviews of studies on teacher networked-based learning had also yielded similar results (see Zhao & Rop, 2001). While the technological affordances of networked environment seems to provide an avenue for collaborative learning, there seems to be a higher possibility for the participants to share information and perhaps request for elementary clarification. These results also seem to corroborate with the quantitative results obtained by most studies (and this study) in terms of the average thread length. It seems reasonable to assume that high level of knowledge construction did not happen when the typical structure of a forum is one first level note followed by two to three responses. [51]

Besides trees, another solution to the nonlinear structure of discussion is to represent each message as a node in a graph.

## 2.8 Graphs

Graphs are the most flexible representation for discussions; in a tree model, one argument can only address a single other argument, while in graphs, one argument can be connected to multiple other arguments. All graph solutions are scaffolded, as opposed to threaded discussion and trees which have the option of being scaffolded or not. This is arguably the most active area of research in debate representations. Popular graph discussion platforms include:

- Belvedere
- DebateGraph
- Reason!Able / Rationale
- ArgMap
- ArguNet
- Araucaria
- bCisive
- Parmenides
- Deliberatorium

There is also a broader representation called a "mind map," which is used to represent any sort of graph, but can very easily be used to represent a debate. Popular mind map platforms include:

- Compendium
- QuestMap
- Theseus

There are two categories of graph systems out there [46]: those that facilitate arguments, such as Reason!Able, and those that analyze and assemble arguments from the outside, such as DebateGraph and Belvedere.

Belvedere is an argument mapping system aimed towards beginners:

These students can not be presumed to have either the skills of constructing scientific arguments or the specific knowledge of a domain. The design of Belvedere addresses the cognitive and motivational limitations and requirements of these unpracticed beginners, as presented in the psychological literature and as we encountered them in formative testing with 12-15 year olds in a lab study and in 10th grade classrooms in an inner-city public high school. A main goal of our system is to stimulate critical discussion that would not otherwise take place. [53]

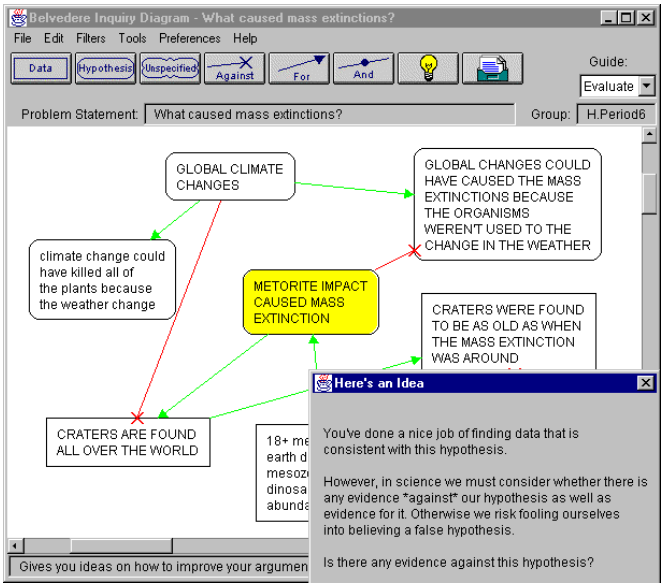


**Figure 2.2: An example argument using Belvedere, a scaffolded tree-structured discussion platform.**

Belvedere had great success in this regard. Gao et al. (2013) remarked,

> Researchers found that the use of Belvedere increased the generation of coherent arguments and problem-solving actions (Cho & Jonassen, 2002), and participants using Belvedere were more likely to state hypotheses early, elaborate on their hypotheses and integrate them with data than learners using a threaded forum (Suthers et al, 2008).

### 2.8.1 Complexity

The main drawback of graphing solutions is their inherent complexity. Reed and Rowe (2004) write, "The task of analysis and diagramming, however, is labor intensive and often idiosyncratic, which can make academic exchange difficult."

Tim van Gelder (2002), the author of the Reason!Able argument mapping system, writes,

> A great deal of philosophers' work involves articulating and communicating arguments, and identifying arguments as communicated by others, so you might have thought that a means of presenting arguments in which inferential structure is made completely explicit would be deemed very useful. Yet argument mapping has never really taken off among philosophers. One of the most important factors behind this neglect is that it just hasn't been easy to for your average philosopher to produce, modify and distribute diagrams of any kind, let alone diagrams of complex arguments. [57]

However, he remains hopeful that this is changing, with the advent of argument mapping programs.

Jodi et al. (2010) explored this problem:

> One fundamental question is what amount of complexity users are willing to adopt in order to reap the benefits of argumentation; previous research has emphasized incremental formalization [18] because users do not generally understand the larger structure of an argument from the outset (see e.g. [19], page 29), and even experienced users can have difficulty holding a complex argumentation model in their heads (page 27, ibid). This leads us to believe that only a simple argumentation model will gain use in social media, unless the complexity can be mitigated by good interfaces and familiar metaphors. [49]

Iandoli et al. (2007) found that because of graph-based systems' inherent complexity, heavy moderation is necessary.

> Moderators played a crucial role: in an important sense they led the community. They supported users with comments and suggestions and, by ensuring a logically-organized argument map, helped users rapidly locate the contexts where their piece of knowledge can best be linked. For these reasons it is crucial to have enough moderators working to ensure fast certification and timely reorganization of the argument map. With the existing data we can roughly estimate the requisite number of moderators per users. A cadre of from 2 to 5 moderators (the number varied from day to day according to their

> other commitments) was able to more or less keep up with 180 active authors, but only by dint of an unsustainably heavy investment of their time. We estimate that a more realistic time commitment would require that between 5 and 10% of the active users be moderators. [33]

## 2.9 Conflict

The literature seems to suggest that the more phase 2 interaction there is, the more learning occurs. In other words, conflict spurs learning.

Brooks and Jeong (2006) said,

> The underlying assumptions used to ground the research questions and methods used in this study were based on assumptions of the dialogic theory of language (Bakhtin, 1981). The main assumption in dialogic theory is that social meaning is renegotiated and reconstructed as a result of conflict brought about through social interactions. Accordingly, conflict is the primary catalyst that drives critical inquiry and discourse. ... Some current research in CMC supports these assumptions that "conflict and consideration of both sides of an issue" (Baker, 1999; Jeong, 2003, 2005a; Johnson & Johnson, 1992; Wiley & Voss, 1999) produce critical inquiry and deeper understanding. [23]

Jin and Jeong (2013) found that "Chi square tests indicated that higher levels of learning were most likely to be exhibited in critique and argument postings." [36]

Rourke and Kanuka (2007) have some possible insight into this phenomenon. In their study, they found that users "perceived critiques as personal attacks" [47] One user "seemed to interpret differing opinions as win-lose competitions, not as the opportunities for higher-order learning that many commentators imagine" [47]

Such behavior could make sense; people enjoy winning arguments in the real world, so it's not unreasonable to say that they enjoy winning arguments online.

See the Reason's approach to online argumentation is to emphasize the conflict

aspect of online debate. In other words, we will try to maximize phase 2 of the IAM. However, such emphasis on phase 2 has its risks. Trena (2006) says,

> Findings show that rather than a challenge model of argumentation discourse, participants engaged in a relationship-oriented discourse of connection. Educators should be aware of both models of discourse, challenge and connect, because emphasizing only argumentation before trust has been developed among members of the group could result in unproductive conflict. [44]

Most of the literature regarding online debate has overwhelmingly been in educational contexts, to analyze how students learn in an online environment, and much less applied to online debates. This thesis will apply the IAM to a general online discussion.

# CHAPTER 3

# THE COUNTER/ASSUMPTION MODEL

See the Reason is built upon the Counter/Assumption model. It makes a few modifications to make it easier to use, which will be mentioned in section 4.1.

The Counter/Assumption model can represent any discussion, but to explain it, let's explore the simplest use case first: mathematical proofs.

## 3.1 Objective Claims, Counters, Claim Status

An "objective claim"[1] is one that is presented as "objectively true beyond a reasonable doubt."[2]

For example, let's say someone came up with a mathematical proof, shown in Figure 3.1:

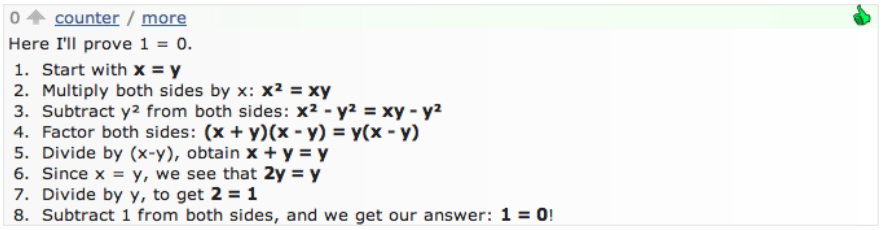


**Figure 3.1: A simple proof in See the Reason.**

This entire proof is presented as an "objective claim" because the author sees it to be true beyond a reasonable doubt.

Every objective claim has a status, either "tentatively true" or "tentatively false"[3] (there is one more status introduced in section 3.2). Unless there is something specifically making it tentatively false, every claim is by default tentatively true. There are various ways in which a claim can become tentatively false,

[1] as opposed to a subjective claim, explained in section 3.3

[2] For more information on how we determine what "true beyond a reasonable doubt" means, see section E.1.

[3] The term "tentatively" is here because one cannot use these statuses to determine whether a claim is really true or not. For that, we have a different feature, "certainty." See section D for details.

which will be covered below. Right now, none of those are happening; this claim is tentatively true.

Another user can see this claim, and notice that something is wrong with it. He can say what is wrong with it, by adding another objective claim to the discussion, as a "counter" to the original objective claim, shown in Figure 3.2

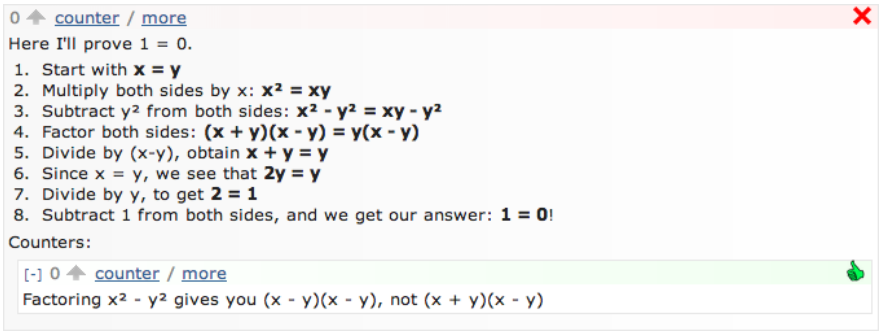


**Figure 3.2: A counter.**

An objective claim can have any number of countering objective claims.[4]

At this point, the top claim becomes tentatively false, because one way a claim is made tentatively false is if it has any tentatively true counters.

If the original author comes back and sees that the critique itself is actually invalid, he can submit another objective claim as a counter to the counter, shown in Figure 3.3.

Now we determine the status for each claim. We start at the bottom. The bottom-most claim is tentatively true, because there are no reasons for it to be tentatively false. The middle claim is tentatively false, because it has a tentatively true counter. The top claim may have a counter, but the counter itself is tentatively false, so the top claim is left as tentatively true. Remember, only a tentatively true counter can make its parent claim false.

Someone else could come along and think they see something wrong with the proof, and submit it, shown in Figure 3.4

[4]Note, objective claims cannot have "supporting" claims. This is a major difference between the Counter/Assumption model and every other model. See section I.16 for why we can't have supporting claims, and how we compensate for it.

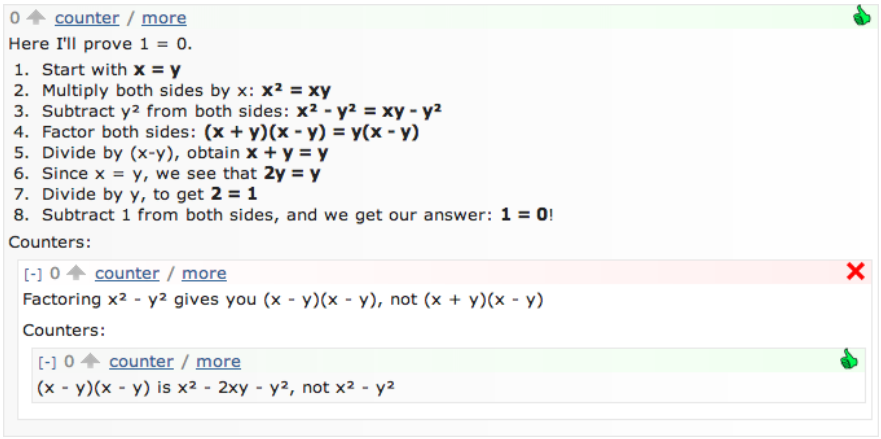


**Figure 3.3: A counter of a counter.**

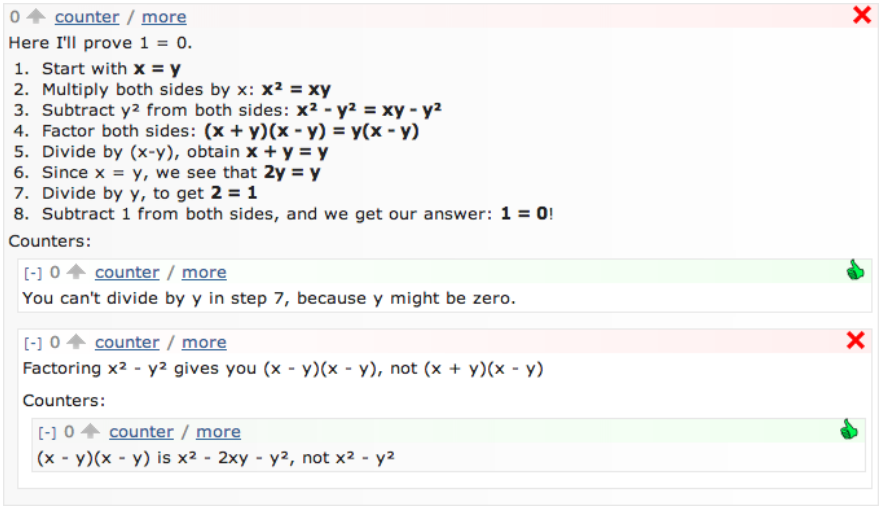


**Figure 3.4: A second counter.**

When this last claim is submitted, the top claim becomes tentatively false, because it has a tentatively true counter.

At this point, the author can't figure out a way to counter it, because it makes a good point. However, he can submit an entirely new discussion, fixed so that it cannot be countered in the same way: he adds, "x and y are positive integers," shown in Figure 3.5.

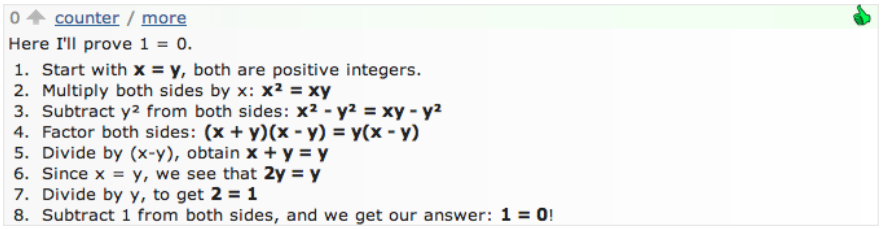


**Figure 3.5: A new and improved version of the previous claim.**

In this new discussion, this claim is tentatively true again because there are no counters.

Let's say a user comes along and adds a new counter, shown in Figure 3.6.

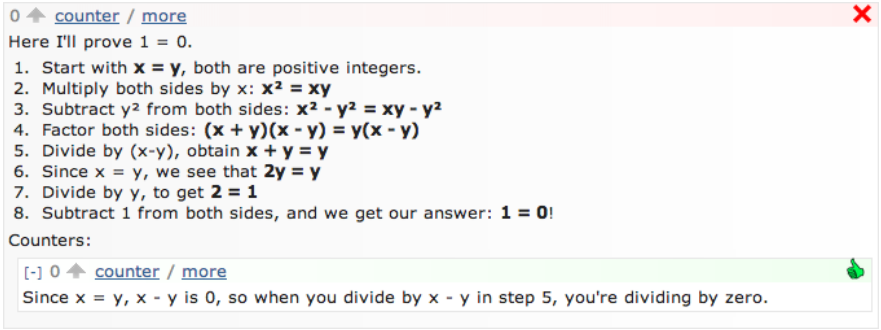


**Figure 3.6: A user countering the new version of the claim.**

The original author sees this, and realizes his mistake. There's no way to counter the counter, and there's no way to edit[5] the proof to make it correct, so he gives up, leaving this claim tentatively false.

[5]For usability reasons, in See the Reason, instead of submitting an entire new discussion, users are also allowed to edit their original claims, as long as they don't change the core meaning of the claim. In See the Reason, the original author would have edited his claim instead of submitting a new discussion.

As explained so far, the counter/assumption model can be a very useful tool for tracking critiques of mathematical proofs, or any purely objective claims. The next feature, assumptions, will allow the users to factor subjective premises into their claims.

## 3.2 Assumptions

See the Reason, as explained so far, would only be able to handle objective arguments, such as mathematical proofs. However, interesting situations arise in discussions like the one shown in Figure 3.7.

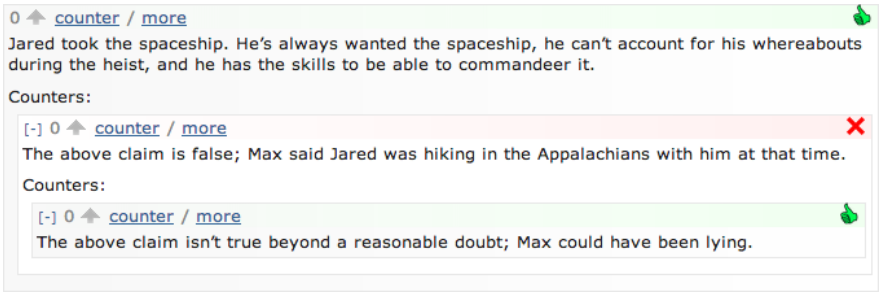


**Figure 3.7: A discussion where users have different underlying assumptions.**

If one person believes that Max is lying, and one person believes that Max is telling the truth, then the debate can go no further. Until they figure that out, this debate is paralyzed. However, it would be very helpful to be able to argue in a hypothetical sense, as if Max was telling the truth. For this, we can submit a new counter, this time with an "assumption," shown in Figure 3.8.

For claims with assumptions, See the Reason will suspend its normal rule of "Every claim is true until proven false" and wait for input from the user. If the user hits the "accept" button, it will look like Figure 3.9.

If the user hits the "reject" button, the tree will look like Figure 3.10.

An assumption means that, for the claim's subtree, users have to argue in a hypothetical sense, as if the assumption is true[6].

[6]If someone mistakenly tries the same counter again, then the counter can be countered as "contradicting an above assumption"

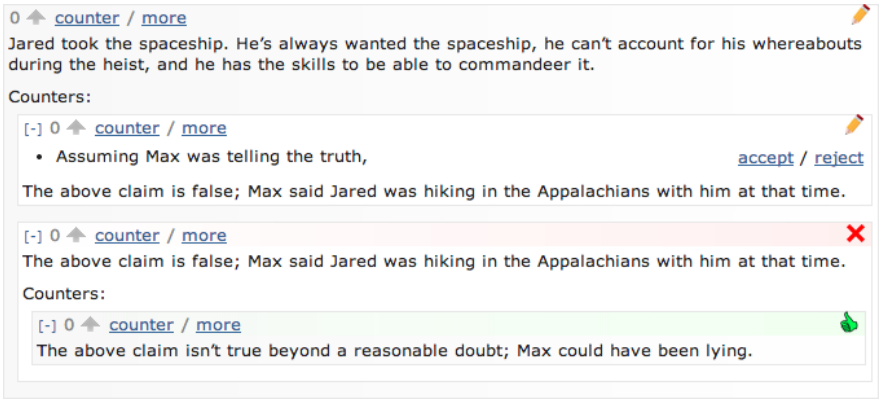


**Figure 3.8: A discussion with a factored-out assumption.**

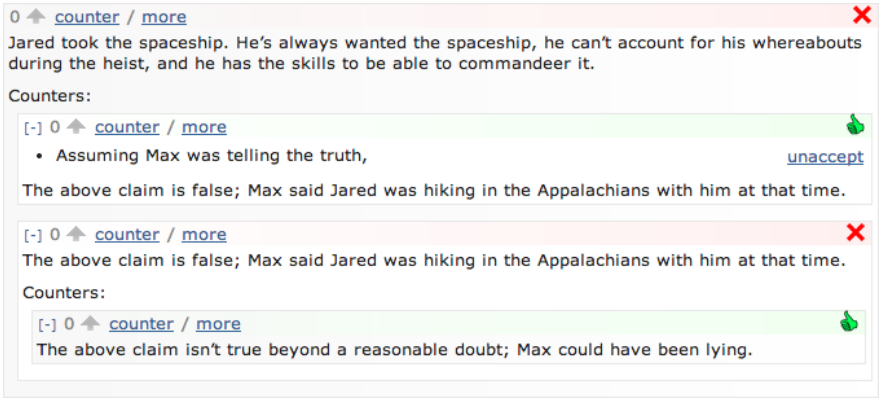


**Figure 3.9: An accepted assumption.**

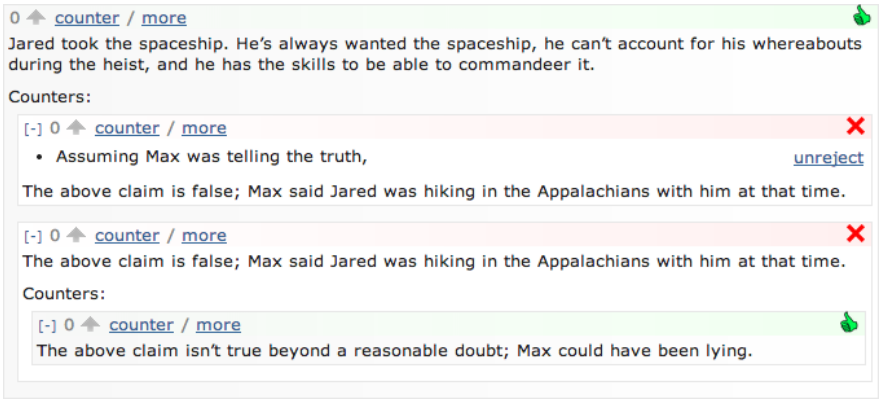


**Figure 3.10: A rejected assumption.**

Now, just because the users are arguing hypothetically as if Max was telling the truth, it doesn't mean that the claim cannot be countered for a different reason, such as in Figure 3.11.

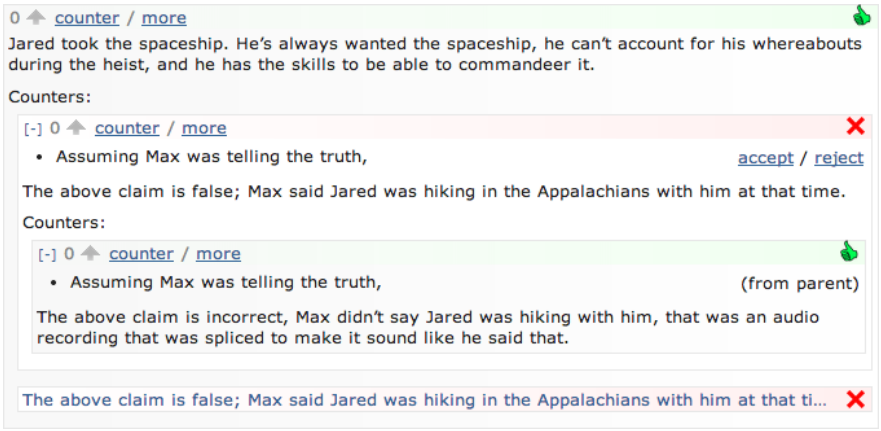


**Figure 3.11: Countering a claim that has an assumption.**

In this case, the assumption is made moot; it doesn't matter whether one believes Max was telling the truth, because the claim is wrong for a different reason. It doesn't matter if they hit accept, or reject, or leave it unjudged, the claim will still remain tentatively false.

## 3.3 Subjective Claims

We can represent an individual claim and factor some subjectivity out of it. But most of the great debates of our time cannot be represented yet. For example, this debate happened on createdebate.com:

- (topic) Optimus Prime is better than Bender
  - (dispute) Bender can bend things to very precise angles, and Optimus can't.
    - (support) Indeed, evidence can be found here: http://www.imdb.com/title/tt0584449/quotes?item=qt0336866
  - (support) Optimus Prime has saved the world and Bender hasn't.
    - (support) Optimus prime has indeed saved the world, see http://www.imdb.com/character/ch0003342/

    - (dispute) Bender actually has saved the world, in the global warming episode.
- (support) Optimus Prime is more moral than bender.
    - (support) Optimus actively tries to protect humankind.
    - (support) Bender sets a bad role model for the kids in the show, because he drinks and smokes.
    - (support) Bender has been shown to steal or earn enormous amounts of money, jewels, and other valuable objects.

To handle these kinds of complex claims with many factors that must be weighed against each other, the counter/assumption model has the "subjective claim." For example, in the counter/assumption model, the above discussion would look like Figure 3.12.

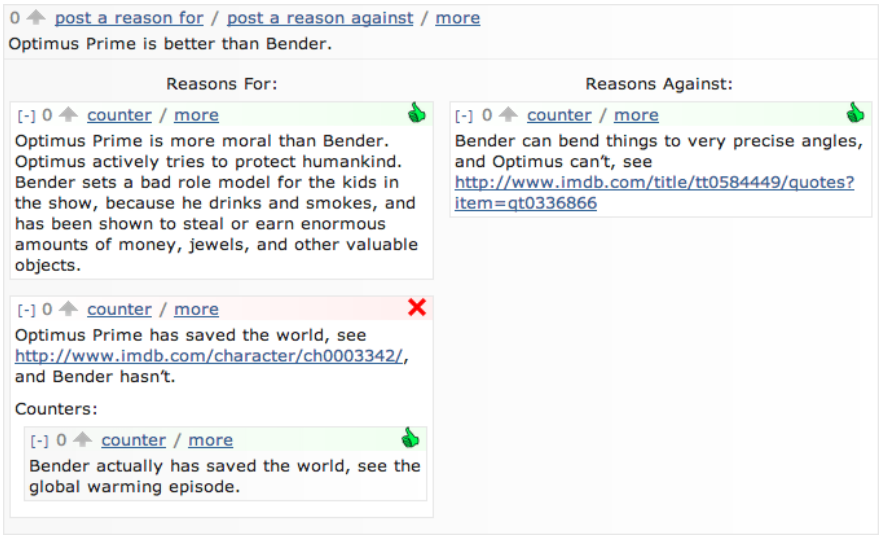


**Figure 3.12: See the Reason's representation of a subjective claim.**

A subjective claim, instead of having counters, can have "reasons," for and against. A "reason for" is defined to be "something that, if true, makes the parent subjective claim more believable," and a "reason against" is defined to be "something that, if true, makes the parent subjective claim less believable."

A reason can be either an objective claim or a subjective claim. Although, in See the Reason, because of usability concerns (see section I.16), we don't allow any reasons to be subjective claims.

Remember, there's no such thing as "supports" for objective claims in the Counter/Assumption model. What would have been a supporting claim is instead factored into the parent claim.

One way to think about a subjective claim is that it's a "grouping of relevant objective claims."

Subjective claims have no status; they can't be tentatively true or tentatively false.

Sometimes, reasons need assumptions, shown in Figure 3.13.

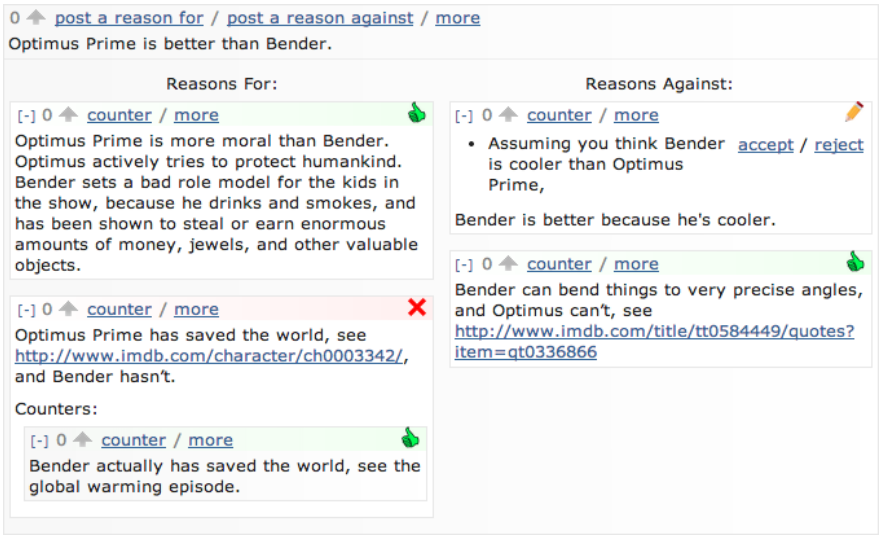


**Figure 3.13: A subjective claim whose reasons have assumptions.**

## 3.4 In Toulmin's Terms

In terms of Toulmin's model, in which arguments are comprised of containing grounds, warrant, backing, qualifier, claim, and rebuttal, an objective claim's body must contain the grounds, warrants, backing, and claim. The assumptions serve the same purpose as Toulmin's qualifiers. The rebuttals would just be entirely new objective claims.

### 3.5 Potential Benefits and Drawbacks

The main drawback of the Counter/Assumption model is that it is unintuitive. Almost all existing debate models include a notion of a "supporting" claim, but such a thing is impossible in the Counter/Assumption model; the same purpose is accomplished by submitting a new version of the claim, with the necessary changes.

The Counter/Assumption model's main benefit is that it can judge whether a claim is tentatively true or false, given a user's assumption judgments. This can have many potential secondary benefits.

One such secondary benefit is that the model generates an archive of false claims. Some platforms will try to prune the false claims away. Osborne (2010, p. 463) aptly explains, "knowing what is wrong matters as much as knowing what is right,"[43] If an argumentation model did not include a history of what people have tried to say, then people will ultimately try to say it again, leading to an unproductive loop.

Another potential secondary benefit is that if people see that an opposing claim is marked "tentatively true," they will be motivated to counter it. If they see that their own claim is marked "tentatively false," they will be motivated to address its counter.

The Counter/Assumption model emphasizes counters as the way to interact with the system. This approach is promising, because conflict is what moves a discussion forward[7]. As Brooks and Jeong (2006) said,

> The main assumption in dialogic theory is that social meaning is renegotiated and reconstructed as a result of conflict brought about through social interactions. Accordingly, conflict is the primary catalyst that drives critical inquiry and discourse. ... Some current research in CMC supports these assumptions that "conflict and con-

[7] One could say say conflict could keep people from reaching a consensus, but See the Reason's goal is not to achieve consensus on the overall topic, but consensus on which facts are true or not. Once the participants agree on the facts, See the Reason's goal is achieved, and the users can then use a different platform (such as LiquidFeedback or Loomio) to come to a consensus about the overall topic, given the facts.

sideration of both sides of an issue" (Baker, 1999; Jeong, 2003, 2005a; Johnson & Johnson, 1992; Wiley & Voss, 1999) produce critical inquiry and deeper understanding.

# CHAPTER 4

# SEE THE REASON FEATURES

## 4.1 Changes to the Counter/Assumption Model

See the Reason is based on the Counter/Assumption model, but makes a few modifications to it and adds a few features to make it more usable.

The first big change is that See the Reason doesn't allow subjective claims' reasons to be subjective claims, all of a subjective claim's reasons must be objective claims. This is because users could not understand the difference between the two. It was an acceptable limitation because in the rare case that a subjective reason is needed, it can be represented by an objective reason with an assumption which is linked to another root subjective claim. See I.16 for more details.

The second big change is that, when a user wants to add supporting information to a claim, See the Reason allows the user to edit it in instead of creating an entire new claim. There is a restriction on editing: users may only edit if they are not changing the core meaning of the claim. See I.24 for more details on how this could be enforced.

## 4.2 Guide Mode

To anyone who hasn't seen it before, See the Reason is a very complex system. There are dozens of actions available to any user. Users are easily paralyzed by the multitude of choices[52], and it happened in See the Reason. To illustrate this, figure 4.1 highlights all of the actions available in one view.

A user would not know when he could do something, and even then, the user would not know what to do. On top of that, it's hard for the user to figure it out because there are so many confusing words on the page. To address this, we introduced Guide Mode, enabled by a checkbox in the top right corner. There are three manifestations of guide mode: the sidebar, "helpables," and "wizards".

Figure 4.2 shows what See the Reason looks like when Guide Mode.

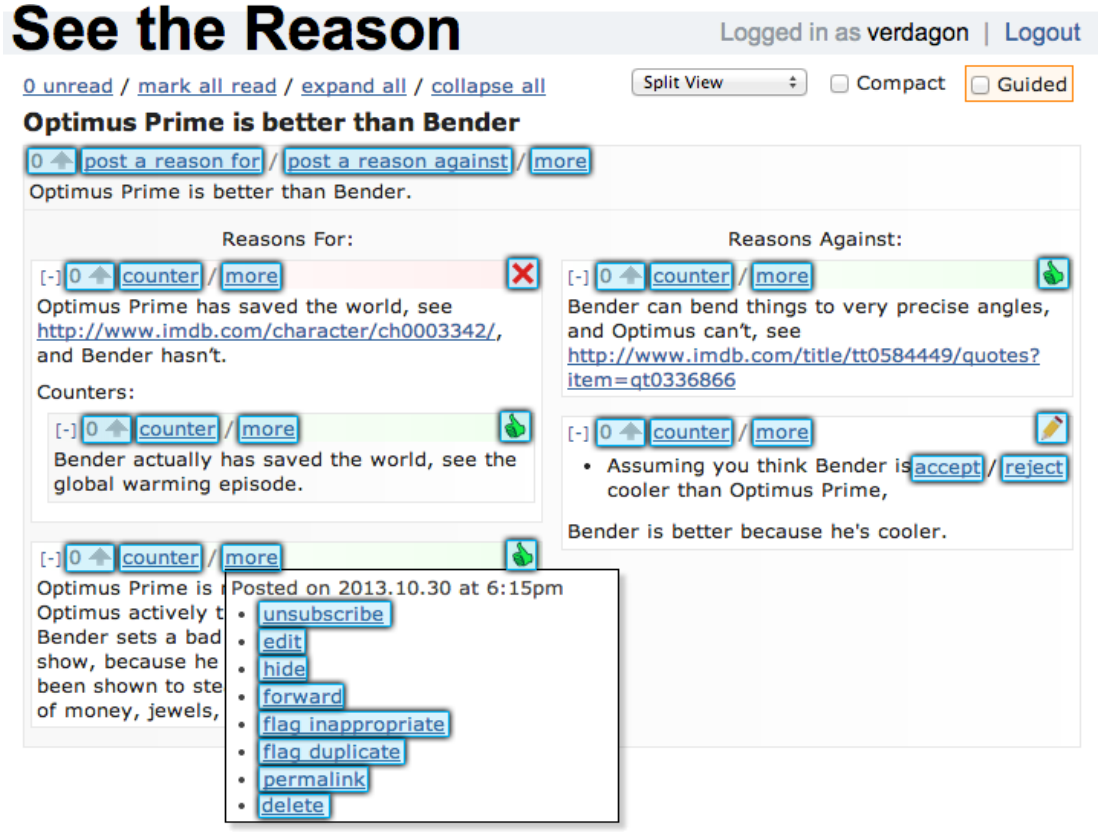


**Figure 4.1: A complex view on See the Reason.**


**Figure 4.2: A view of See the Reason with Guide mode on.**

### 4.2.1 Sidebar

When enabled, a sidebar is visible (see Figure 4.2) which gives the user a broad overview of See the Reason. [1].

The goal of the sidebar is to give the user a good overview of the system, if that's how they choose to learn. If they choose to instead dive into the system and learn by seeing it in action, they will benefit from the "Helpables" feature.

### 4.2.2 Helpables

There are a lot of confusing terms, phrases, and icons on See the Reason. When a user hovers the mouse over one, the sidebar will change to show an explanation of it. For example, if the user doesn't know what the "counter" link means in Figure 4.3, he can hover the mouse over it, which causes the sidebar to show an


**Figure 4.3: See the Reason in guide mode, before the user hovers over "counter."**

[1]The sidebar's full text is in appendix C

explanation, as shown in Figure 4.4.

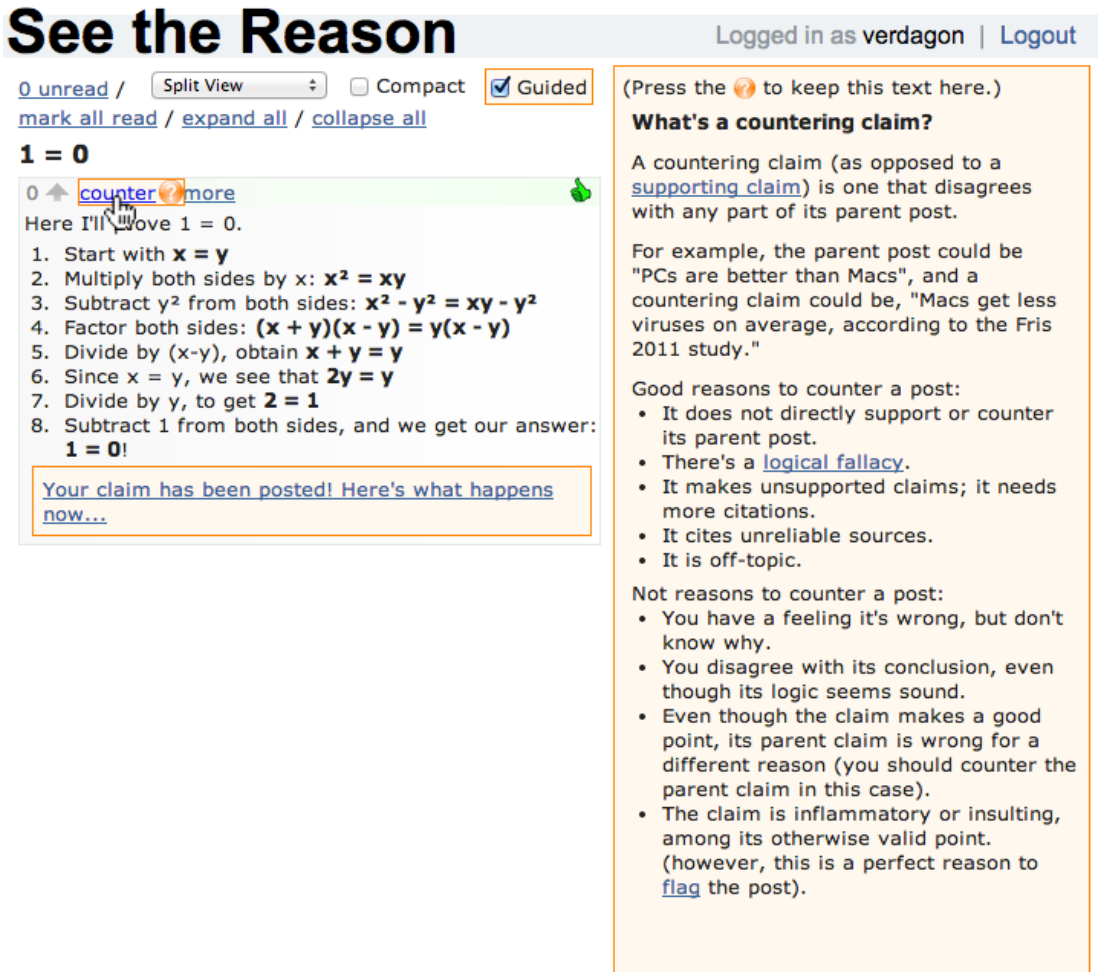


**Figure 4.4: See the Reason in guide mode, while the user's hovering over "counter."**

There are dozens of "helpables" throughout See the Reason. For a full list, see appendix B.

### 4.2.3 Participate Hints

The "Participate Hints" are little orange boxes that advise the user how to do things, like the six in Figure 4.5.

For subjective claims, it always says, "If you know anything (even if the topic seems to broad for it) that might factor into someone's decision, click here!"

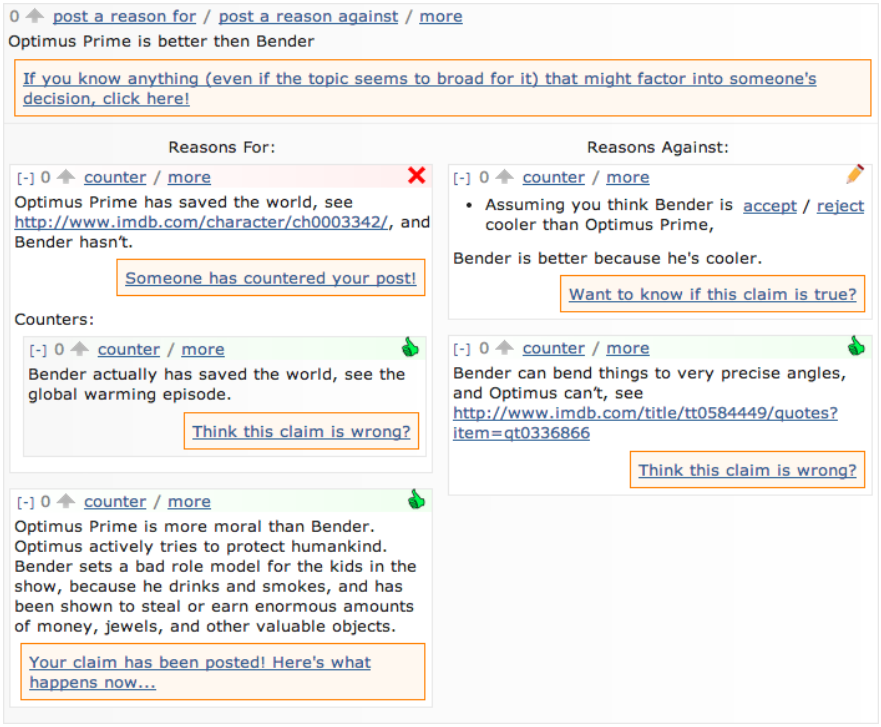


**Figure 4.5: See the Reason's "participate hints."**

If it's a objective claim with a "no judgment" status, it says, "Want to know if this claim is true?" and when clicked, explains that the user needs to accept or reject some assumptions.

If it's a tentatively true objective claim, it says, "Think this claim is wrong?" and when clicked, explains how to counter the claim. If it's the same user, it will say, "Your claim has been posted! Here's what happens now..." and explains to the user that he should wait for someone to counter it.

If it's a tentatively false objective claim, it says, "Think this claim actually correct?" (or, if its the same user, "Someone has countered your post!") and when clicked, it opens the Address Counter Wizard, explained in section 4.2.4.

For a full listing of all of the things the Participate Hints say, see 4.2.3

### 4.2.4 Address Counter Wizard

Before it lets the user fill in their new claim's text, the new claim form will ask them the general reason why they are posting. For example, given the discussion in Figure 4.6, another user might notice that the claim is wrong, and want

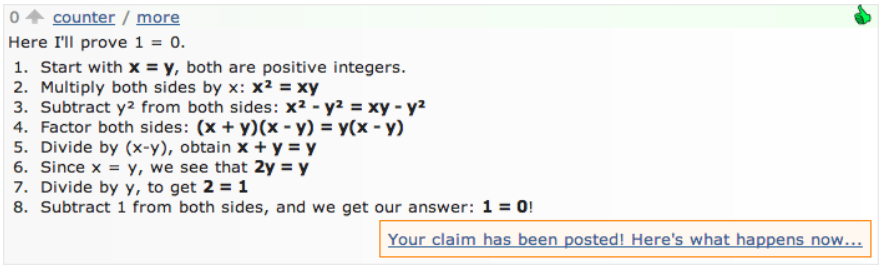


**Figure 4.6: An example discussion.**

to counter it. He would hit the "counter" link, and would see something like Figure 4.7.

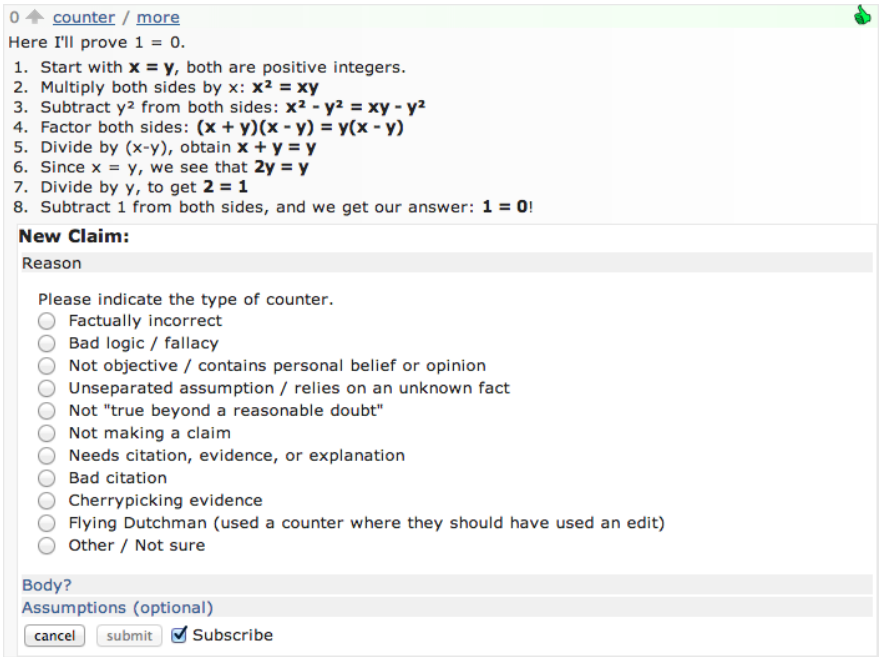


**Figure 4.7: The "New Claim Form," asking for the type of counter.**

The user would then select "Factually incorrect," and then it would ask the user to type in the claim's body, as seen in Figure 4.8.

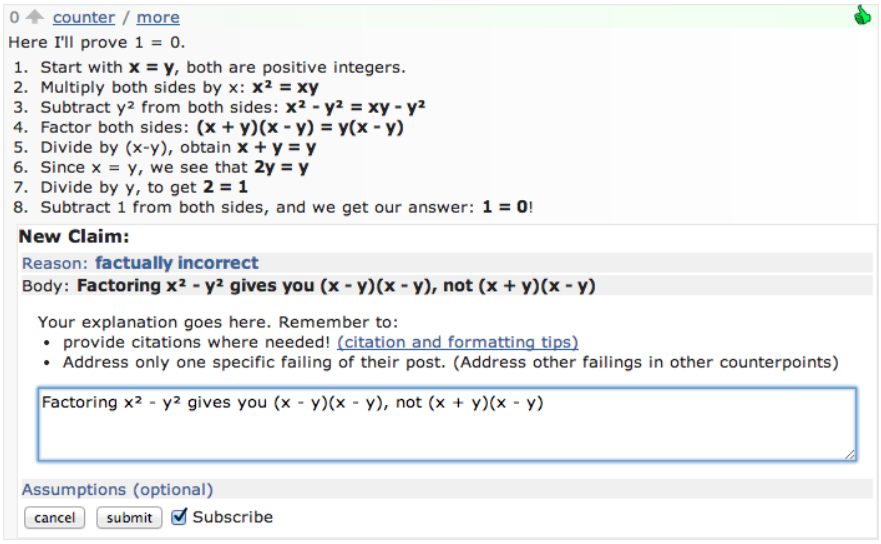


**Figure 4.8: The "New Claim Form," asking for the counter's body text.**

He hits submit, and then the claim's body and counter reason is sent to the server. It appears on the other user's page, with a "Someone has countered your post!" link, as seen in Figure 4.9.

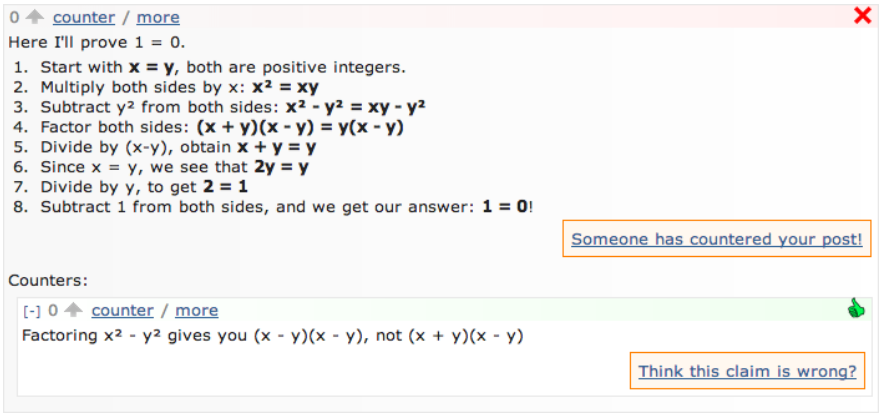


**Figure 4.9: The system hinting the user that someone has countered their claim.**

When "Someone has countered your post!" is clicked, the page will display different information depending on the counter reason of the first (in this case, only) counter. In this case, since the counter is a "Factually incorrect" counter, it will display the text seen in Figure 4.10.

The text it shows them depends on the type of counter that the other person

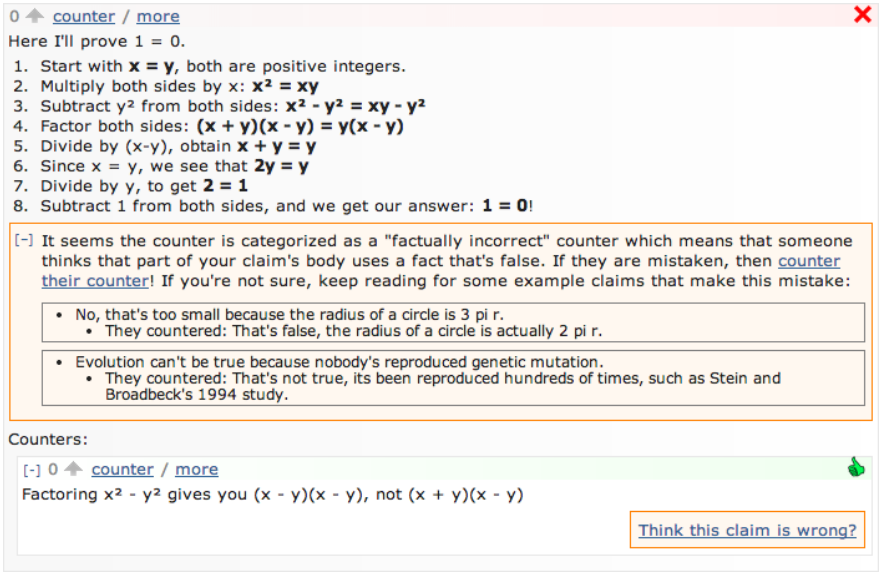


**Figure 4.10: The Address Counter Wizard offering advice.**

posted.[2]

In short, if a claim is tentatively false, the Address Counter Wizard will advise the user on possible courses of action depending on the first tentatively true counter's reason.

The Address Counter Wizard, the participate hints, the helpables, and the sidebar are all meant to be training wheels for new users. Once a user is experienced enough, we expect him to disable guide mode.

## 4.3 Ordering

Whenever claims are displayed, we order them according to our algorithm, which aims to have these properties:

- More tentatively true claims are displayed near the top than tentatively false ones
- Claims with higher certainty will be displayed near the top
- More upvoted[3] claims are displayed near the top then those less upvoted

[2]For a full list of what the Address Counter Wizard could say for each type of couter, see section 4.2.4.

[3]If a user thinks a claim is a positive contribution to the debate, then they can hit the arrow

- New claims will not be crowded out by old claims
- Every claim will always have a chance to be near the top

See section I.10 for how the ordering algorithm works.

## 4.4 Citations and Formatting

Users can format their claims according to the WikiCreole syntax, with our own additions for citations.

For example, this text:

> Bender is **way** better than Optimus; he can bend at very price angles, see [[http://futurama.com/bender.php—this page]]!

becomes the claim seen in Figure 4.11.

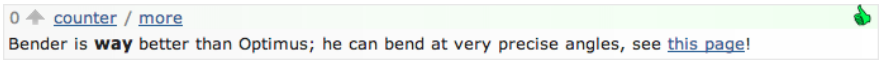


**Figure 4.11: A formatted claim.**

There are many other formatting options in WikiCreole (see Figure 4.12).

There's also a special syntax for citations. This text:

> Abraham Lincoln was actually a vampire hunter in his early twenties, not early thirties. [cite http://abethehunter.com/beginnings.html] You're probably thinking of Vampus Slayus, from the Third World series.

will become the claim in Figure 4.13.

## 4.5 Subscriptions

Users can subscribe to any given claim. If the claim is objective, they will be emailed whenever its status changes. If the claim is subjective, they will be

icon at the top left of the claim. This will affect where the claim appears in relation to other claims.

| | | |
|---|---|---|
| //italics// | → | *italics* |
| **bold** | → | **bold** |
| * Bullet list<br>* Second item<br>** Sub item | → | • Bullet list<br>• Second item<br>..• Sub item |
| # Numbered list<br># Second item<br>## Sub item | → | 1. Numbered list<br>2. Second item<br>2.1 Sub item |
| Link to [[wikipage]] | → | Link to wikipage |
| [[URL\|linkname]] | → | linkname |
| == Large heading<br>=== Medium heading<br>==== Small heading | → | **Large heading**<br>**Medium heading**<br>**Small heading** |
| No<br>linebreak!<br><br>Use empty row | → | No linebreak!<br><br>Use empty row |
| Force\\linebreak | → | Force<br>linebreak |
| Horizontal line:<br>---- | → | Horizontal line:<br>--- |
| {{Image.jpg\|title}} | → | Image with title |
| \|=\|=table\|=header\|<br>\|a\|table\|row\|<br>\|b\|table\|row\| | → | Table |
| {{{<br>== [[Nowiki]]:<br>//**don't** format//<br>}}} | → | == [[Nowiki]]:<br>//**don't** format// |

www.wikicreole.org

**Figure 4.12: WikiCreole's syntax.**

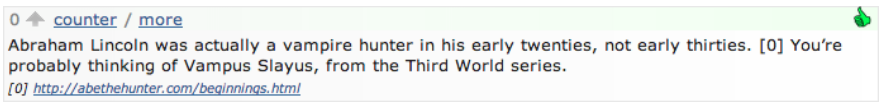


**Figure 4.13: A claim with a citation.**

emailed whenever any of the reasons for or reasons against are added or have their status change.

## 4.6 Next Unread Link

The system tracks which claims a user has viewed by noting the position of their screen every five seconds. If a give claim has been in the user's screen for ten seconds, then it's marked as "read".

When the user clicks the "next unread" link, it will move the screen to the next unread claim, and highlight it in blue for about a second.

## 4.7 View Modes

There are two modes with which a user can view the tree page: "Split View Mode" and "Combined View Mode".

Split View mode is the default view mode; users will always see the page in Split View mode unless they manually change it. Split View mode separates the reasons for from the reasons against: the reasons for appear in the left column, and the reasons against appear in the right column. An example is shown in Figure 4.14.

We also have Combined View mode. In Combined View mode, the reasons for and against both appear in one big column. First is a reason for, then a reason against, then a reason for, and so on until one of them runs out, at which point the rest of the other side will be displayed. An example is shown in Figure 4.15.

## 4.8 Compact Mode

The regular mode has a lot of spacing to make the page feel less daunting [4]. For those who are experienced and aren't intimidated by very compact text, they can enable compact mode. It removes a lot of the padding, to have a maximum of

[4] see the experiments in section I.11.6

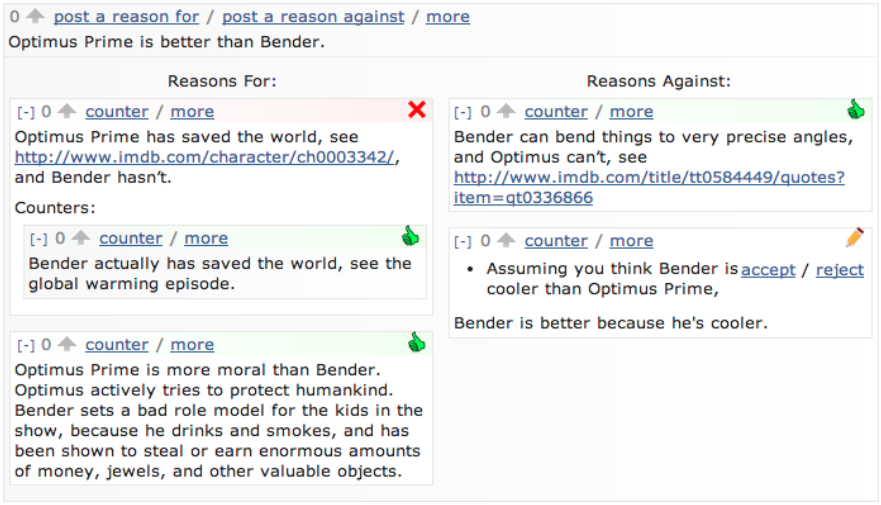


**Figure 4.14: See the Reason in Split View mode.**

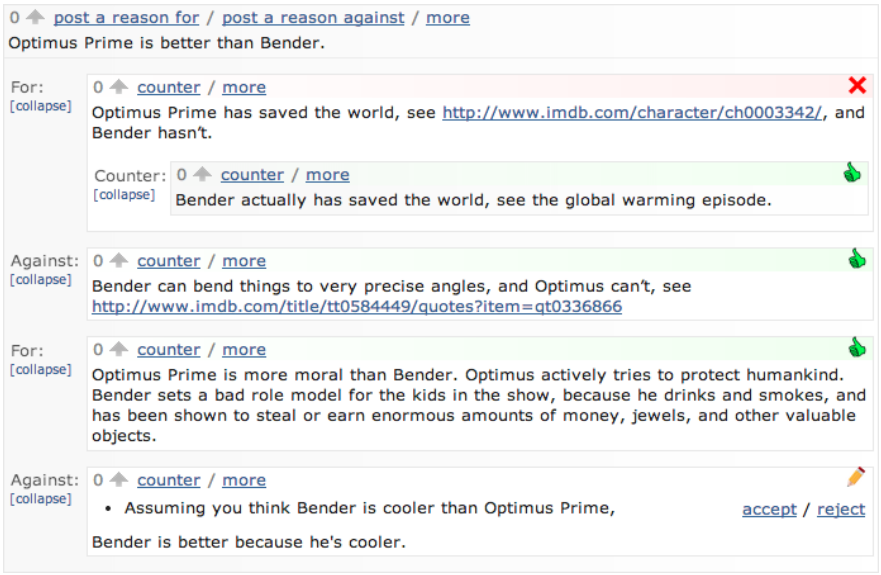


**Figure 4.15: See the Reason in Combined View mode.**

text and minimal unused space on the screen. It's also very helpful for smaller screens, such as mobile devices. An example is shown in Figure 4.16.

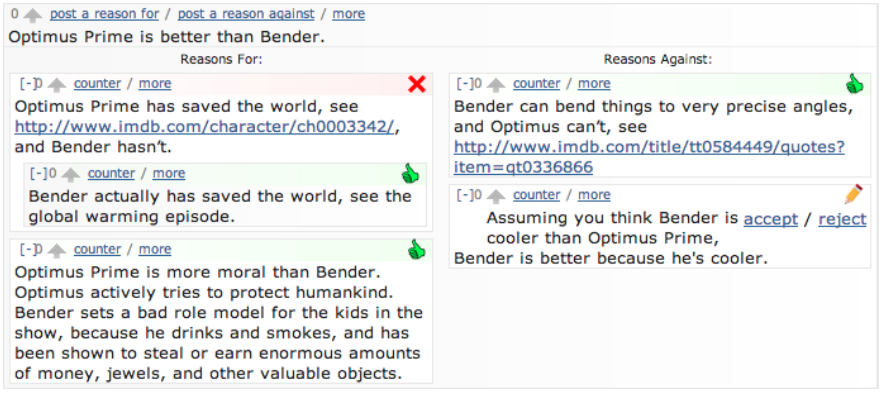


**Figure 4.16: See the Reason in Split View mode and Compact mode at the same time.**

The page looks best on mobile devices using combined view, with helpful mode disabled, and compact mode enabled, as shown in Figure 4.17.

## 4.9 Collapsing Subtrees

The user can expand and collapse a claim and all of its children, in case he's done reading it and doesn't want to see it anymore. If a user started with the page in Figure 4.18, he could hit the collapse icons, to get Figure 4.19. When all the claims are collapsed, it looks like Figure 4.20.

## 4.10 Real-time Updating

So the users don't have to reopen the page constantly to figure out when someone else has posted, the page will automatically check for them. When someone has posted to the site, it will show a small blue notification in the corner. When the user clicks this notification, the tree will update itself with the new claims.

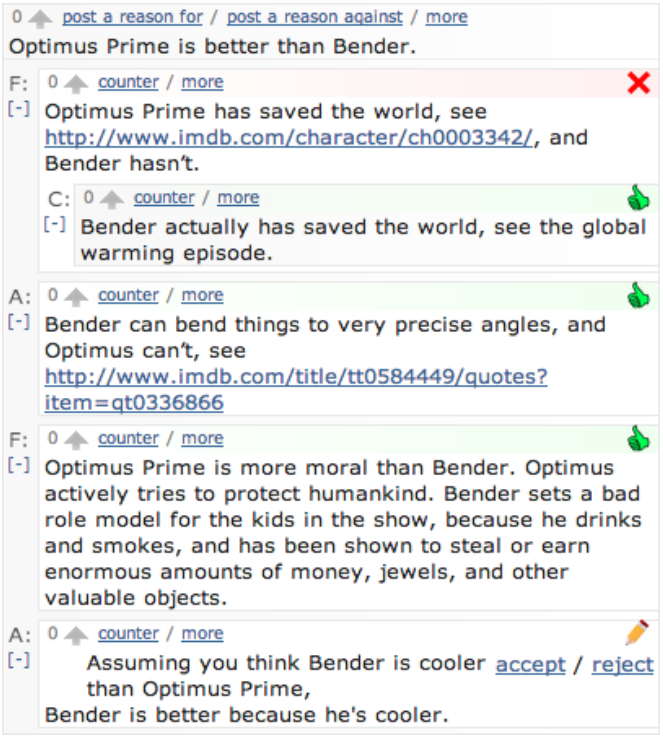


**Figure 4.17: See the Reason in Combined View mode and Compact mode at the same time.**

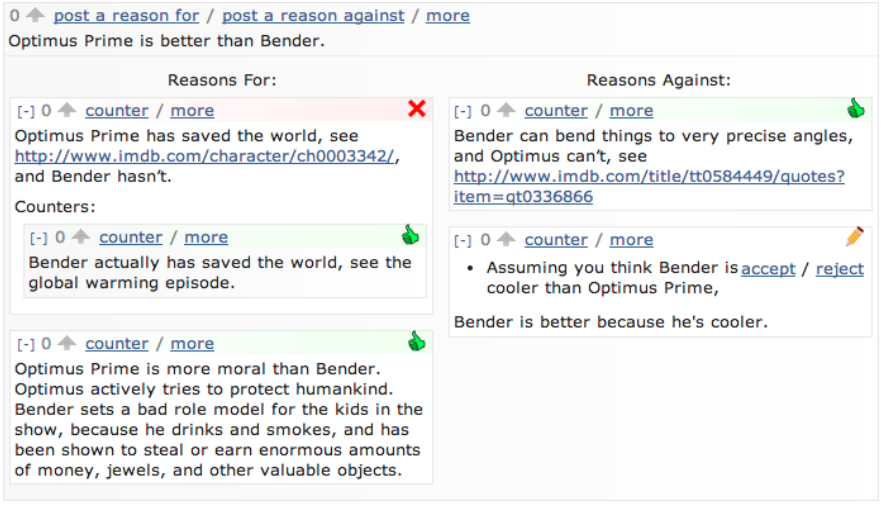


**Figure 4.18: A regular discussion, with all claims expanded.**


**Figure 4.19: A discussion, with some claims collapsed.**


**Figure 4.20: A discussion, with all claims collapsed.**

# CHAPTER 5

# EXPERIMENT

We performed an experiment to try to gauge how well the Counter/Assumption model affects argumentation quality, and also to gauge how well See the Reason performs in general.

We decided to compare it to CreateDebate, an online scaffolded tree-structured platform.

## 5.1 Choice to compare to CreateDebate

There were a multitude of platforms out there, but because of participation constraints we had to choose only one to compare with See the Reason.

We decided not to compare with any of the graph systems because the difference was too extreme; it would be impossible to determine if See the Reason performed better because it's based on the Counter/Assumption model, or because it uses a tree-structured model instead of a graph.

The closest existing solutions were in the "scaffolded trees" category, which include Argumentum, ConvinceMe, ProCon, ForAndAgainst, and CreateDebate. CreateDebate's feature set is a superset of all of the others; none of the others have all of the features CreateDebate does. For this reason, we chose CreateDebate.

## 5.2 Experiment Questions and Hypotheses

With this experiment, we hope to answer the following questions:

1. Does See the Reason motivate users to participate more than CreateDebate?
2. Are there fewer uncaught "bad arguments" in See the Reason?[1]
3. Does See the Reason progress further into the IAM than CreateDebate?

[1] Originally, this question was two questions: "Are there less bad arguments presented in See the Reason?" and "Are more of the presented bad arguments addressed by the users?" but because of a misunderstanding with some of the judges, some of the data was suspect and had to be thrown out. However, we still had valid measurements of which fallacies were present and not mentioned by participants.

4. Is See the Reason as easy to use as CreateDebate?

### 5.2.1 Does See the Reason motivate users to participate more than CreateDebate?

Our hypothesis is that users will be more motivated to participate in See the Reason, because of the instant feedback provided by the Claim Status feature[2].

### 5.2.2 Are there fewer uncaught "bad arguments" in See the Reason?

See the Reason tries to encourage people to counter others' claims, so we hope that there will be more people checking the claims' facts and logic. On top of that, the new claim form gives hints to the user about some fallacies to look for (see Figure 4.7). For this reason, we think that there will be fewer uncaught arguments in See the Reason.

### 5.2.3 Does See the Reason Progress Further into the IAM than CreateDebate?

Our hypothesis is that See the Reason's debate will go further into the IAM, because of it's emphasis on counters. We expect that it will have less posts in phase 1, where people present their reasons, and most posts in phase 2, where people counter each others' claims.

Phase 3 is about agreeing on the facts relevant to the debate. Whenever someone edits out parts of a post that are unsound, that is a manifestation of phase 3. Another example of this in See the Reason could be the negotiation of what assumptions for which a given claim is true[3]. We hope that See the Reason's claims will progress into phase 3, but we hypothesize that it won't, because in our similar previous experiments we've never seen the sustained interest and

[2]See section 2.9 for why we could expect this.

[3]See section 3.2, where the users negotiate the meaning of Figure 3.7, by factoring out assumptions to find the core logic that both can agree on, even though their premises might differ.

time commitment from users that is required to actually negotiate meaning with others.

Phase 4 is about the users negotiating an agreement about the topic as a whole. We don't expect any posts in See the Reason or CreateDebate to be in phases 4 or 5. See the Reason is built to judge the correctness of relevant claims, not to come to a conclusion about the topic as a whole. Similarly, CreateDebate, through its voting procedure, can come to its own conclusion, but it does not represent the users' conclusion. Both platforms aren't so much meant to come to a conclusion, they're meant to inform people so they can better reach their own conclusions via their own means.

Phase 5 posts are those which express agreement of the negotiated conclusion. Since there is no negotiated conclusion, there will be no posts in phase 5 for either platform.

### 5.2.4 Is See the Reason as Easy to Use as CreateDebate?

The Counter/Assumption model is somewhat unintuitive compared to CreateDebate's model. However, we did a lot of usability tests during See the Reason's construction (see appendix I) and refined the user interface as much as we could, to try to compensate for the unintuitive underlying model. Our hypothesis is that they should be roughly equal in terms of usability.

## 5.3 Method

In this experiment, we had 79 students debate the topic, "Who is better, Bender or Optimus Prime?"[4] The topic was chosen because it is fairly broad (there's a lot of relevant information), and because it was relevant to the students. We chose a topic relevant to the students to simulate the aspect of online discussions where someone only participates because they have interest in and knowledge

[4]Bender is the robotic character from Futurama, Optimus Prime is the robotic character from Transformers.

about the topic. The topic was also chosen because it's a very subjective one and it didn't have a clear answer.

We created pages for the topic on createdebate.com and seethereason.net. We divided the users into two groups according to their student IDs: students with even IDs were instructed to participate in the createdebate.com debate first, and those with odd IDs were instructed to participate in the seethereason.net debate first. They were instructed that to receive full credit, they should to submit at least 3 posts to the platform the first week, and then to submit at least 3 posts to the other platform the second week, making six in total. At the end of the two weeks, they filled out a survey.

They were told,

> The goal of See the Reason is to make as many of your side's claims tentatively true (green) as possible, and as many of the other side's claims tentatively false (red) as possible. First, try to counter the other side's claims, if they can be countered. If you can't think of any counters, submit some reasons of your own!

For this experiment, there were no moderators and the moderation features were disabled, to cut down on external interference.

To measure the first question, "Does See the Reason motivate users to participate more than CreateDebate?" we measured the number of posts per person that are not part of the students' requirements to receive credit. In other words, we counted the number of posts after the required three posts. For example, if a student posted 10 posts to one of the debates, and he was only required to post three, then we recorded that he posted seven posts past the requirements.

To measure the second question, "Are there fewer uncaught "bad arguments" in See the Reason?" we had four people[5] look at both of the debates, and identify as many flaws with the arguments, that have not been mentioned or addressed by the participants themselves.

To measure the third question, "Does See the Reason Progress Further into

[5]These four people have all had lessons in identifying logical fallacies as part of their courses.

the IAM than CreateDebate?" we counted the number of posts in three categories: reasons, supports, and counters. "Reasons" are the posts that are directly for or against the root claim. Any posts below the reasons that are on the same side as their parents are "supports." Any posts below the reasons that are on the opposite side as their parents are "counters." For example, there are three reasons, five supports, and one counter in the following discussion:

- Optimus Prime is better than Bender
  - (Bender) Bender can bend things to very precise angles, and Optimus can't.
    - (Bender) Indeed, evidence can be found here: http://www.imdb.com/title/tt0584449/quotes?item=qt0336866
  - (Optimus) Optimus Prime has saved the world and Bender hasn't.
    - (Optimus) Optimus prime has indeed saved the world, see http://www.imdb.com/character/ch0003342/
    - (Bender) Bender actually has saved the world, in the global warming episode.
  - (Optimus) Optimus Prime is more moral than bender.
    - (Optimus) Optimus actively tries to protect humankind.
    - (Optimus) Bender sets a bad role model for the kids in the show, because he drinks and smokes.
    - (Optimus) Bender has been shown to steal or earn enormous amounts of money, jewels, and other valuable objects.

To measure the fourth question, "Is See the Reason as easy to use as CreateDebate?" we designed the below survey. It contained two of each of these questions, one for See the Reason, one for CreateDebate, making twenty questions in total.

1. How difficult was it to post to (platform)? (scale, 1-5, 5 is better)
2. How much did you enjoy using (platform)? (scale, 1-5, 5 is better)
3. How was using (platform) compared to arguing on an email thread? (scale, 1-5, 5 is better)
4. How was using (platform) compared to arguing in person? (scale, 1-5, 5 is better)
5. If any, about how many times did (platform) help you come up with things to post? (number, more is better)

6. How quickly did you run out of things to say on (platform) before any research was needed? (multiple choice)
    - Option 1: "Immediately, all the things I would have said before researching were already said"
    - Option 2: "I only posted one or two things before I needed to research to find more things to post"
    - Option 3: "Never, there were plenty of other things I could have said"
7. How easy/difficult was it to participate in the discussion on (platform)? (scale, 1-5, 5 is better)
8. Did (platform) present the information in a way that was easy to understand? (scale, 1-5, 5 is better)
9. Was there ever a time when you wanted to submit something on (platform), but ended up not submitting it? (yes/no + text)
10. Was there anything you didnt like about (platform)? (text)

## 5.4 Weaknesses

### 5.4.1 Scale of Experiment

The largest weakness of this experiment was its small size. See the Reason has a lot of hypothesized long-term emergent behavior, such as how a tree could evolve over time to become more complete and correct[6], and how users could come to a consensus on a given claim; in other words, it would proceed further into IAM phase 3. For these things to happen, there needs to be a lot more participation than what we would see in an experiment of this size; the participants needed to be given enough time to exhaust their phase 2 posts. Since each student was only required to submit at least three claims to each debate, it was unlikely that we'd reach far into phase 3.

### 5.4.2 Judges

We used four people to try to find things that were obviously wrong with the arguments, in order to measure the question, "Are there fewer uncaught "bad

[6]See section F for how a tree could evolve in this way.

arguments" in See the Reason?" but like any human, these four people were fallible; they might not have identified every bad argument.

### 5.4.3 Difference from Online Behaviors

In this experiment, users were required, as part of a class, to submit at least three claims to each debate in order to get full credit. A large number of participants obviously didn't spend very much time formulating their arguments, and didn't spend any time reading any existing arguments. A lot of students just submitted three top-level reasons, one sentence long, and left. This also means that there were a large number of duplicates, which might have skewed question 6: "How quickly did you run out of things to say on (platform) before any research was needed?" If a user didn't read the other posts, then he would less likely choose "Immediately, all the things I would have said before researching were already said."

It turns out, these problems are not uncommon. Noroozi et al. (2012) encountered similar disinterest in their experiments:

> Learners rarely respond to one another's points and tend to repeat points already constructed by others (Koschmann, 2003); they may thus refuse to challenge arguments made by peers (Nussbaum, 2002), resulting in narrow discussions with low quality (Pena-Shaff, Martin, & Gay, 2001) and low consistency (Brooks & Jeong, 2006). [43]

Brooks and Jeong also encountered this problem:

> Despite the affordances of using asynchronous threaded discussions, studies still find that students rarely respond to one another's points, often repeat points already made by other students (Koschmann, 2003; Veerman, 2003), and often produce discussions that lack coherence and depth (Herring, 1999). [23]

Another difference we encountered was that in regular online debate, people would re-visit the debate multiple times, but in this experiment, students never came back to defend their original claims. Phase 3 posts could only happen if

people engaged each other directly to negotiate the meaning of a given claim, but since people posted and never came back to the site, phase 3 messages could not happen.

### 5.4.4 Difference in Platforms Besides Underlying Models

This experiment was designed in part to gauge the effectiveness of the underlying argumentation models, but it's impossible to measure that without receiving interference from the two platforms' other difference. For example, CreateDebate had many features, and had years of use and refinement. If See the Reason and CreateDebate performed differently, it would be difficult to know if the differences were because CreateDebate was more refined, or because the underlying argumentation models are different. Because this could be an issue, we spent a lot of time refining See the Reason, making the interface as simple and clean as possible, and fine-tuning Guide Mode, so that the two platforms could be on somewhat even footing, so that any differences could be more reliably attributed to the different underlying argumentation models.

### 5.4.5 Bias towards See the Reason

We gave the participants no indication that we designed See the Reason, or that we hoped See the Reason would succeed, but the participants could have figured it out anyway, just from the appearances of each site. CreateDebate had the appearance of a mature site that's visited very often, while See the Reason had the appearance of a prototype. If the users figured this out, then it could have interfered with the results in some way.

## 5.5 Results

The results of the experiment follow. In this section's tables, the "N" column contains the counts of the posts in question, the "N/T" column contains the percentage of the posts in question, and the "N/U" column contains the average

number of those posts per user.

| | CreateDebate | See the Reason |
|---|---|---|
| **Users** | 79 | 73 |
| **Posts** | 223 | 255 |

**Table 5.1: Number of users and posts for each platform.**

The number of users and posts are shown in table 5.1.

| | CreateDebate | | | See the Reason | | |
|---|---|---|---|---|---|---|
| | **N** | **N/T** | **N/U** | **N** | **N/T** | **N/U** |
| **Reasons** | 134 | 60% | 1.70 | 135 | 53% | 1.85 |
| **Supports** | 23 | 10% | 0.29 | (n/a) | | |
| **Counters** | 64 | 29% | 0.81 | 120 | 47% | 1.64 |

**Table 5.2: Number of reasons, supports, and counters in each debate.**

The number of reasons, supports, and counters are shown in table 5.2.

| | CreateDebate | | | See the Reason | | |
|---|---|---|---|---|---|---|
| | **N** | **N/T** | **N/U** | **N** | **N/T** | **N/U** |
| **Phase 1** | 157 | 70% | 1.98 | 135 | 53% | 1.85 |
| **Phase 2** | 64 | 29% | .81 | 120 | 47% | 1.64 |
| **Phase 3** | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 5.3: Number of posts in the IAM phases.**

The number of posts in each phase is in table 5.3[7]. There were no posts in phase 3. See the Reason made it further into phase 2 than CreateDebate, but only with a p-value of 0.069, which makes it a promising difference, but not a statistically significant one.

The number of posts past requirements is shown in table 5.4. See the Reason did much better in this metric.

The number of bad arguments, as determined by the judges, is shown in table 5.5. There were less in See the Reason, with a p-value of 0.014, making it a statistically significant difference.

[7]Note, phase 1 is simply the number of reasons plus the number of supports, and phase 2 is simply the number of counters.

| | CreateDebate | | | See the Reason | | |
|---|---|---|---|---|---|---|
| | N | N/T | N/U | N | N/T | N/U |
| **Num Posts Past Requirements** | 7 | 3% | .09 | 53 | 21% | .70 |

**Table 5.4: Number of posts past the requirements.**

| | CreateDebate | | | See the Reason | | |
|---|---|---|---|---|---|---|
| | N | N/T | N/U | N | N/T | N/U |
| **Bad Arguments** | 60 | 27% | 0.76 | 51 | 20% | 0.70 |

**Table 5.5: Bad arguments in each platform.**

The results of the survey are shown in table 5.6. In See the Reason, users reached the limits of their knowledge faster and helped users more to come up with things to post (Q5), but did worse in every other question.

### 5.5.1 Summary of Results

The promising, but not quite statistically significant, results were:

1. See the Reason's debates progressed further into the IAM than CreateDebate's (p-value: 0.069)
2. Users believed that See the Reason did not compare against email threads as well as CreateDebate (p-value: 0.073)
3. Users believed that it was more difficult to participate in the discussion on See the Reason (p-value: 0.062)

The statistically significant results are:

1. There were fewer uncaught bad arguments in See the Reason (p-value: 0.014)
2. Users believed it was more difficult to post to See the Reason (p-value: 0.010)
3. Users believed that See the Reason presented the information in a way that was more difficult to understand (0.034)

## 5.6 Discussion

The first thing to note is that a lot of the differences were overwhelmingly because of two users: one who posted 23 claims in See the Reason compared to their 4

| Question | CD | StR | P-value |
|---|---|---|---|
| **Q1**: How difficult was it to post to (platform)? (scale, 1-5, 5 is better) | 4.2 | 3.76 | 0.010 |
| **Q2**: How much did you enjoy using (platform)? (scale, 1-5, 5 is better) | 3.34 | 3.16 | 0.278 |
| **Q3**: How was using (platform) compared to arguing on an email thread? (scale, 1-5, 5 is better) | 4.48 | 3.79 | 0.073 |
| **Q4**: How was using (platform) compared to arguing in person? (scale, 1-5, 5 is better) | 2.76 | 2.59 | 0.392 |
| **Q5**: If any, about how many times did (platform) help you come up with things to post? (number, more is better) | 1.15 | 1.47 | 0.397 |
| **Q6**: How quickly did you run out of things to say on (platform) before any research was needed? (multiple choice) | | | 0.464 |
| Q6 option 1: "Immediately, all the things I would have said before researching were already said" | 7 | 9 | |
| Q6 option 2: "I only posted one or two things before I needed to research to find more things to post" | 33 | 35 | |
| Q6 option 3: "Never, there were plenty of other things I could have said" | 36 | 32 | |
| **Q7**: How easy/difficult was it to participate in the discussion on (platform)? (scale, 1-5, 5 is better) | 4.24 | 3.92 | 0.062 |
| **Q8**: Did (platform) present the information in a way that was easy to understand? (scale, 1-5, 5 is better) | 3.97 | 3.57 | 0.034 |

**Table 5.6: Survey results.**

in CreateDebate, and one who submitted 22 claims in See the Reason compared to their 3 in CreateDebate. The distribution of posts is shown in Figure 5.1. We considered discounting them as outliers, but it does seem to follow the basic power law distribution described in Gurkan et al.'s study[27] (2010). Similarly, Klein and Iandoli (2008) noted,

> If the Collaboratorium works like most OSPP systems, user contributions will follow a power law, so we can expect that a relatively small corps of "power users" will take on the bulk of the argument mapping tasks.

### 5.6.1 Does See the Reason motivate users to participate more than CreateDebate?

As seen in table 5.4, See the Reason's users contributed more past their requirements. This suggests that the claims' statuses of tentatively true/false might help motivate users[8].

### 5.6.2 Are there fewer uncaught "bad arguments" in See the Reason?

According to our judges, there were significantly fewer uncaught "bad arguments" in See the Reason. This suggests that the counter/assumption model is either helping users present less bad arguments, or helping users catch the ones that are presented, or both.

### 5.6.3 Does See the Reason Progress Further into the IAM than CreateDebate?

As shown in table 5.3, See the Reason had 47% of its posts in Phase 2, and CreateDebate only had 29%. This suggests that See the Reason progresses further into the IAM than CreateDebate, however, this only has a p-value of 0.069, so we cannot say that it is statistically significant.

[8]though as stated in section 5.4.4, this could also be because of other factors besides the counter/assumption model.

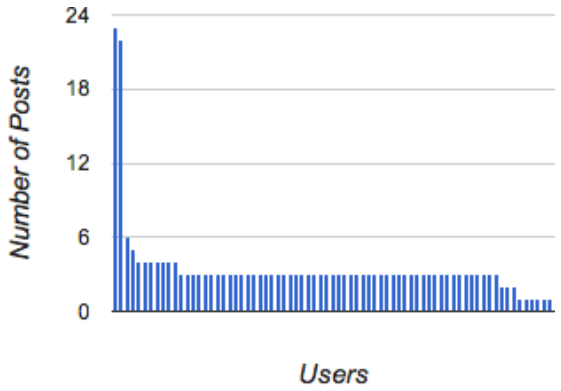


**Figure 5.1: Number of posts by each user.**

Twice in See the Reason, users successfully factored out their assumptions, but neither of those were cases of IAM phase 3 interaction, because they were both preemptive, neither was brought about by interaction.

### 5.6.4 Is See the Reason as Easy to Use as CreateDebate?

The answer to this question is a resounding "no," as shown by the survey results. CreateDebate outperformed See the Reason in all significant and near-significant survey results.

Oddly enough, we didn't get any complaints about the Counter/Assumption model; nobody complained that they couldn't submit any supporting claims. Most of the complaints were about how all of the claims were collapsed by default, and users had to click on them to see them.[9]

[9]Of course, in our experiments, we found in previous experiments that users were even more unhappy when they were left expanded; the view is too dauntingly complex if they're not collapsed.

# CHAPTER 6

# CONCLUSION

We found in this experiment that See the Reason makes a significant tradeoff between usability and argument quality. CreateDebate is more usable and users seem to like it more, but See the Reason produces arguments of higher quality.

Taking those results, plus the weaknesses into account, we can conclude that See the Reason does in fact make online argumentation better.

## 6.1 Future Work

### 6.1.1 Future Experiments

In the future, it would be good to run more experiments to see if the effects that our almost-statistically-significant results suggest are correct.

There are a few different kinds of experiment that should be run.

One promising experiment would be like this one, but with less people, and where each person takes more time. If this could happen, then we could see the hypothesized behavior in See the Reason, where people refine their claims to be tentatively true. Unlike this experiment, we might see the assumption used correctly, something incredibly important to See the Reason.

Another promising experiment would be to deploy See the Reason online and open it up to the public, to see what happens. See the Reason was designed for that kind of environment, so it might perform better in that situation.

One more promising experiment would be to have a "driven" debate. In this, a skilled person would be the only user, and he or she would take input from different people and put them into the correct representation (with assumptions, etc.) If we have this person acting as a middleman between the user and the system, it might drastically improve the quality. Higher quality claims might amplify the benefits of See the Reason.

Any future experiment should be conducted over a longer period of time than

this experiment. Having users only submit three posts to each debate did not create much interaction, and more interaction could really show the differences between See the Reason and CreateDebate.

### 6.1.2 Comparison with Calculemus and TruthMapping

See the Reason's objective claims are quite different from any tool out there, but we did come across two interesting platforms that might be somewhat relevant. TruthMapping.com has a tool which allows users to critique deductive claims, which supposedly promotes the constant evolution of a claim to become more and more correct, or crumble under its weaknesses, very similar to how our objective claims work. [1] Another interesting platform is Calculemus [37] which aims to factor out assumptions, similar to how See the Reason does it. It could be enlightening to compare See the Reason's objective claims to these two tools, to see which method best promotes the constant improvement of a claim.

# BIBLIOGRAPHY


[1] About truthmapping. http://truthmapping.com/about.php.

[2] bCisive. http://bcisive.austhink.com/solutions.

[3] Be a debatabase editor. http://idebate.org/content/be-debatabase-editor.

[4] Belvedere. http://lilt.ics.hawaii.edu/belvedere/.

[5] ClaiMaker. http://globalsensemaking.net/video/2052744:Video:3222.

[6] For impatient web users, an eye blink is just too long to wait. http://www.nytimes.com/2012/03/01/technology/impatient-web-users-flee-slow-loading-sites.html?pagewanted=all.

[7] idebate debatabase. http://idebate.org/debatabase.

[8] iLogos. http://www.phil.cmu.edu/projects/argument_mapping/.

[9] Lamest edit wars. http://en.wikipedia.org/wiki/Wikipedia:Lamest_edit_wars.

[10] Oracle the clear leader in 24 billion rdbms market. http://itknowledgeexchange.techtarget.com/eye-on-oracle/oracle-the-clear-leader-in-24-billion-rdbms-market/.

[11] Philoctopus. http://www.archelogos.com/philoctopus/toolDetails.htm.

[12] Php: A fractal of bad design. http://me.veekun.com/blog/2012/04/09/php-a-fractal-of-bad-design/.

[13] Rationale. http://rationale.austhink.com/.

[14] Theseus. http://www.skymark.com/Theseus/overview.asp.

[15] Truth-trees. http://kleene.ss.uci.edu/lpswiki/index.php/Truth-Trees.

[16] Why nosql? http://www.couchbase.com/why-nosql/nosql-database.

[17] *MIT Sloan Research Paper No. 4691-08*, 2008.

[18] S. Albrecht. Whose voice is heard in the virtual public sphere? a study of participation and representation in online deliberation, 2003.

[19] T. Anderson, C. N. Gunawardena, and C. A. Lowe. Analysis of a global online debate and the development of an interaction analysis model for examining social construction of knowledge in computer conferencing. *Journal of Educational Computing Research*, 17(4):397–431, 1997.

[20] B. Bernstein. *Euclid: Supporting Collaborative Argumemtation with Hypertext*. CU-CS. Department of Computer Science, University of Colorado, 1992.

[21] J. Brindley, L. Blaschke, and C. Walti. Creating effective collaborative learning groups in an online environment. *The International Review of Research in Open and Distance Learning*, 10(3), 2009.

[22] C. Brook and R. Oliver. Online learning communities: Investigating a design framework, 2003.

[23] C. D. Brooks and A. Jeong. Effects of prestructuring discussion threads on group interaction and group performance in computersupported collaborative argumentation. *Distance Education*, 27(3):371–390, 2006.

[24] V. Buraphadeja. An assessment of knowledge construction in an online discussion forum: The relationship between content analysis and social network analysis. Master's thesis, 2010.

[25] V. Buraphadeja and K. Swapna. Content analysis of wiki discussions for knowledge construction: opportunities and challenges, 2012.

[26] D. Cartwright and K. Atkinson. Using computational argumentation to support e-participation. *Intelligent Systems, IEEE*, 24(5):42–52, 2009.

[27] A. Dollisso and W. Koundinya. A model for using threaded discussions in on-line agricultural education courses. *NACTA Journal*, 55(3):70, 2011.

[28] F. Gao, T. Zhang, and T. Franklin. Designing asynchronous online discussion environments: A review of current research. *British Journal of Education Technology*, 44(3), 2013.

[29] A. Gurkan, L. Iandoli, M. Klein, and G. Zollo. Mediating debate through on-line large-scale argumentation: Evidence from the field. *Information Sciences*, 180(19):3686 – 3702, 2010.

[30] F. Henri. Computer conferencing and content analysis. In A. Kaye, editor, *Collaborative Learning Through Computer Conferencing*, volume 90 of *NATO ASI Series*, pages 117–136. Springer Berlin Heidelberg, 1992.

[31] H.-T. Hou, K.-E. Chang, and Y.-T. Sung. Analysis of problem-solving-based online asynchronous discussion pattern. *Educational Technology & Society*, 11(1):17–28, 2008.

[32] P. Hoven. Marcin lewinski: Internet political discussion forums as an argumentative activity type. a pragma-dialectical analysis of online forms of strategic manoeuvring in reacting critically. *Argumentation*, 25(2):255–259, 2011.

[33] L. Iandoli, M. Klein, and G. Zollo. Can we exploit collective intelligence for collaborative deliberation? the case of the climate change collaboratorium. *MIT Sloan Research Paper No. 4675-08*, 2007.

[34] ioseb. Rest vs json-rpc? http://stackoverflow.com/a/15116562/1424454.

[35] A. Jeong and S. Joung. Scaffolding collaborative argumentation in asynchronous discussions with message constraints and message labels. *Computers & Education*, 48(3):427 – 445, 2007.

[36] L. Jin and A. Jeong. Learning achieved in structured online debates: levels of learning and types of postings. *Instructional Science*, 41(6):1141–1152, 2013.

[37] Kyle. Calculemus. http://serptopia.org/?p=140.

[38] A. Macintosh, T. F. Gordon, and A. Renton. Providing argument support for e-participation. *Journal of Information Technology & Politics*, 2009.

[39] R. M. Marra, J. L. Moore, and A. K. Klimczak. Content analysis of online discussion forums: A comparative analysis of protocols. *Educational Technology Research and Development*, 52(2):23–40, 2004.

[40] C. McLoughlin and J. Luca. Investigating processes of social knowledge construction in online environments. 2001.

[41] R. Mills. Researching social news - is reddit.com a mouthpiece for the 'hive mind', or a collective intelligence approach to information overload? *Ethicomp*, 2011.

[42] J. L. Moore and R. M. Marra. A comparative analysis of online discussion participation protocols. *Journal of Research on Technology in Education*, 38(2):191–212, 2005.

[43] O. Noroozi, A. Weinberger, H. J. A. Biemans, M. Mulder, and M. Chizari. Argumentation-based computer supported collaborative learning (abcscl): A synthesis of 15 years of research. *Educational Research Review*, 7:79–106, 2012.

[44] T. Paulus. Challenge or connect? dialogue in online learning environments. *Journal of Computing in Higher Education*, 18(1):3–29, 2006.

[45] C. Reed and G. Rowe. Araucaria. http://araucaria.computing.dundee.ac.uk/doku.php.

[46] C. Reed and G. Rowe. Araucaria: Software for argument analysis, diagramming and representation. *International Journal of AI Tools*, 14:961–980, 2004.

[47] L. Rourke and H. Kanuka. Barriers to online critical discourse. *International Journal of Computer-Supported Collaborative Learning*, 2:105–126, 2007.

[48] W. A. Sandoval and B. J. Reiser. Explanation-driven inquiry: Integrating conceptual and epistemic scaffolds for scientific inquiry. *Science Education*, 88(3):345–372, 2004.

[49] J. Schneider, A. Passant, T. Groza, and J. G. Breslin. Argumentation 3.0: How semantic web technologies can improve argumentation modeling in web 2.0 environments. In *Proceedings of the 2010 Conference on Computational Models of Argument: Proceedings of COMMA 2010*, pages 439–446, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[50] S. B. Shum. Compendium. http://compendium.open.ac.uk/.

[51] C. C. Sing and M. S. Khine. An analysis of interaction and participation patterns in online community. *Educational Technology & Society*, 9(1):250–261, 2006.

[52] D. Sivers. http://sivers.org/jam, 2009.

[53] D. Suthers, A. Weiner, J. Connelly, and M. Paolucci. Belvedere: Engaging students in critical discussion of science and public policy issues. 1995.

[54] J. Tan, C. S. Chai, and H.-Y. Hong. The analysis of small group knowledge building effort among teachers using an interaction analysis model. 2008.

[55] S. Thanasingam and S. K. A. Soong. Interaction patterns and knowledge construction using synchronous discussion forums and video to develop oral skills. *ascilite Singapore*, 2007.

[56] S. Toulmin. *The Uses of Argument*. Cambridge University Press, 2003.

[57] T. van Gelder. Argument mapping with reason!able. *APA Newsletter*, 2(1), 2002.

[58] A. Veerman and E. Veldhuis-Diermanse. Collaborative learning through computer-mediated communication in academic education. 2001.

[59] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, and D. Wilkins. A comparison of a graph database and a relational database: A data provenance perspective. In *Proceedings of the 48th Annual Southeast Regional Conference*, ACM SE '10, pages 42:1–42:6, New York, NY, USA, 2010. ACM.

[60] M. Zheng and H. Spires. Teachers' interactions in an online graduate course on moodle: A social network analysis perspective. http://www.ncsu.edu/meridian/winter2011/zheng/02.htm.

# APPENDIX A

## OTHER SYSTEMS

This appendix is a list of all the other systems we came across that are related to debate. This is not a list of all platforms in existence, but should serve as a good sample.

Regular threaded discussion systems:

- onlinedebate.net
- Facebook's status threads

Scaffolded threaded discussion systems:

- procon.org
- convinceme.net
- forandagainst.com
- lettucedebate.com

Regular tree structured debate systems:

- reddit.com
- slashdot.com's comment systems
- livingknowledge.org

Scaffolded tree structured systems:

- arg.umentum.com
- createdebate.com

Online argument mapping systems:

- bcisiveonline.com
- rationaleonline.com
- debategraph.org

Offline argument mapping systems:

- Philoctopus [11]
- iLogos [8]
- Reason!Able / Rationale [13]
- ClaiMaker [5]
- ArgMap
- ArguNet
- Araucaria [45]
- bCisive [2]
- Calculemus [37]
- Belvedere [4]

These systems are more wiki-like:

- debatewise.org
- iDebate's Debatabase [7]

These systems do facilitate online debate, but only between two people, and each debate lasts for a set time or number of rounds.

- ipidato.com
- debate.org
- economist.com/debate

These systems enable video debate, instead of text-based debate:

- meevsu.com
- vbate.idebate.org

These are general mind map tools which could be used for debate (all offline):

- Theseus [14]
- Compendium [50]
- QuestMap

These systems are used for organizing a claim and its evidence, but do not support conflict:

- truthmapper.com

- Explanation Constructor [48]

These system are geared towards reaching a consensus, which involves debate to a certain degree but doesn't have the features or emphasis on it that the other systems have:

- LiquidFeedback
- Loomio

These systems may have once facilitated online debate, but they are no longer active, and information on them is scarce:

- whereistand.org
- RiledUp
- Argutect
- Athena
- Argumo

# APPENDIX B

# HELPABLES' CONTENTS

When a user has guide mode enabled, and hovers the mouse over certain elements in the view, the sidebar will change its contents to an explanation about the hovered element. For an example of a helpable in action, see section 4.2.2. This appendix lists all of the explanations for those elements.

## B.1 Certainty

What does certainty mean?

Just because a claim is standing, doesn't mean it's necessarily true, and just because a claim is successfully countered, it doesn't mean it's necessarily false.

If someone just posted a claim, and nobody has had the chance to counter it yet, then we can't assume it's true.

However, if someone posted a claim, and a thousand people have seen it and not been able to successfully counter it, then we can feel a little more comfortable about its truth.

Certainty is an indirect measurement of how many people have seen this claim since it's status last changed. It goes from 0 (not certain at all; nobody has seen it) to 1 (very certain, a lot of people have seen it)

Can I rely on the certainty measurement?

The certainty measurement has a few weaknesses:

- It only measures whether the site's viewers consider it certain or not. It's very possible that the people who view this site are the type of people who wouldn't know much about the given claim.
- It assumes that in any 1000 people, there are enough people who know enough about the claim to be able to identify if it is false, and counter it accordingly.

In the end, it's up to you whether to rely on it. Click here to learn about how we measure certainty, and judge for yourself.

### B.2 "post a reason for" link

What's a "reason for"?

A reason for, as opposed to a reason against, is a claim that explains a reason why its parent claim is true. It's a claim that agrees with its parent post.

For example, the parent post could be "Reddit is better than Digg," and a reason for could be, "Reddit has more content posted per day."

In other words...

Another way to think about it is that a "reason for" claim explains one aspect in which the parent claim is true.

For example, the parent post could be "Lower taxes on everyone would be good." and a reason for could be "Everyone would have more money, and having more money is good." because from the money aspect, lowering taxes on everyone is indeed good.

In that same example, there could be another post, this time a reason against, saying "Lowering taxes would mean less money for services, and services are good, so lowering taxes would be bad." because from that aspect, the parent claim is not good.

### B.3 "post a reason against" link

What's a "reason against"?

A reason against, as opposed to a reason for, is a claim that explains a reason why its parent claim is false. It's a claim that disagrees with its parent post.

For example, the parent post could be "PCs are better than Macs," and a reason against could be, "Macs get less viruses on average, according to the Fris 2011 study."

In other words...

Another way to think about it is that a "reason against" claim explains one

aspect in which the parent claim is false.

For example, the parent post could be "Lower taxes on everyone would be good." and a reason against could be, "Lowering taxes would mean less money for services, and services are good, so lowering taxes would be bad." because from that aspect, the parent claim is not good.

In that same example, there could be another post, this time a reason for, saying "Everyone would have more money, and having more money is good." because from the money aspect, lowering taxes on everyone is indeed good.

What's a countering claim?

A countering claim (as opposed to a supporting claim) is one that disagrees with any part of its parent post.

For example, the parent post could be "PCs are better than Macs," and a countering claim could be, "Macs get less viruses on average, according to the Fris 2011 study."

Good reasons to counter a post:

- It does not directly support or counter its parent post.
- There's a logical fallacy.
- It makes unsupported claims; it needs more citations.
- It cites unreliable sources.
- It is off-topic.

Not reasons to counter a post:

- You have a feeling it's wrong, but don't know why.
- You disagree with its conclusion, even though its logic seems sound.
- Even though the claim makes a good point, its parent claim is wrong for a different
- reason (you should counter the parent claim in this case).
- The claim is inflammatory or insulting, among its otherwise valid point. (however, this is a perfect reason to flag the post).

### B.4 Upvote

What is this arrow?

This is the "good point!" button. If you think this claim makes a good point, click on the arrow!

What is this number?

This is the number of "good point!" marks this claim has. Each user can only mark each claim once.

What do these marks do?

The more marks a claim has, the higher up in the list it will be displayed.

Should I mark this claim?

If a claim is on-topic, or makes a good point, or helps you in some way, mark it!

Try not to mark it for being funny, or for agreeing with it. Try to mark claims that contribute to the discussion in a positive way!

### B.5 Objective Claim

What's a "normal" claim?

A normal claim (as opposed to a complex claim) is one that's put forward as logic, as fact, and explains why it is fact, with enough evidence to show it to be true beyond a reasonable doubt.

See some examples!

Normal claims are more visible in See the Reason, but they're held to a much higher standard of proof: the logic within must be true beyond a reasonable doubt.

A normal claim can have no supporting claims; there should be enough evidence within the claim to show that it is true beyond a reasonable doubt. It can, however, have countering claims.

If a normal claim has any countering claims, it is considered "tentatively false" in the eyes of See the Reason. It can be considered tentatively true again

by countering those counterclaims. more...

### B.6 Subjective Claim

What's a "complex" claim?

A complex claim (as opposed to an normal claim) has evidence both supporting and countering it. It's up to the viewer to judge whether or not he believes it.

It is often thought of as a "subjective claim."

While a normal claim is expected to include all of the relevant proof, a complex claim is more for presenting just the conclusion. The proof should be supplied in supporting child claims.

Complex claims are not shown as often as normal claims, but complex claims are more open-ended and flexible in what can be said with them.

For example, one can say "Macs are better than PCs" in a complex claim, but not in a normal claim.

more...

### B.7 Assumption

This is an assumption, and it is one that was introduced by one of this claim's ancestors.

An assumption is a way to have a hypothetical discussion as if something were true.

When a post makes an assumption, all of its children must discuss as if the assumption was true or not.

The assumption is shown in all of the child posts as well as a reminder of that.

## B.8 Claim Judgments

### B.8.1 "tentatively true" icon

What does this icon mean?

It means that See the Reason thinks this claim is tentatively true.

A normal claim is "tentatively true" if See the Reason believes it is true, assuming all that can be said about the topic has been said already.

If you know that this claim is false, hit the "counter" link and type in why. See the Reason will then change its status to "tentatively false."

### B.8.2 "tentatively false" icon

What does "tentatively false" mean?

See the Reason will consider a claim tentatively false if it has any tentatively true counterpoints.

To make a claim "tentatively true" again, one needs to counter all of the tentatively true counterclaims.

### B.8.3 "no judgment" icon

What does this icon mean?

It means the system needs a bit more information before it can tell you whether the claim is tentatively true or false.

To do this, it needs you to accept or reject any relevant assumptions. Click on the "(need input)" and it will help you.

## B.9 New Claim Form

### B.9.1 assumption

What's an assumption?

An assumption is a way to have a hypothetical discussion as if something were true.

Then, each user can tell the system whether they believe the assumption or not, and see how it affects the rest of the discussion.

For example, a person could have a claim "The experiments show that oranges have more citric acid than apples." and an assumption "Assuming the experiment's detectors were calibrated."

This means that they don't exactly know whether or not the detectors were calibrated, but would like to have a discussion as if it was.

This could be useful if perhaps we're still trying to figure out whether the detectors were calibrated, but we want to move the discussion forward anyway.

If that same argument was presented without the assumption, such as "The experiments show that oranges have more citric acid than apples," then someone could easily counter it by saying "That is not the only logical conclusion, it's also possible the detectors weren't calibrated, which happens a lot in our lab."

Note: If someone can prove beyond a reasonable doubt that an assumption is false, they could still counter the claim.

### B.9.2 "personal opinion"

Indeed, many claims mistakenly rely on personal opinions, such as "We should offer the course because programming is the most important field," or "Yes, because red matches her orange dress better."

That's fine in a subjective claim, but not in an objective claim. Objective claims need to be objective.

Here are some examples:

- Macs are better than PCs because they are more beautiful than PCs
  - (counter) Above claim is not objective.
- Macs are way better for programming.
  - (counter) Above claim is not objective.
- Yes because my mom likes Macs better.
  - (counter) Above claim is not objective.

### B.9.3 "based on unknown fact"

If someone makes an argument which is based on an unknown fact, then you can counter it. Here's some good examples:

- No, because upon returning to the scene of the crime, he would have called the police.
  - (counter) But we don't know that the witness returned to the scene of the crime.
- Burst mode is better because it will be better for client-side prediction.
  - (counter) No, we don't know if we're going to do client-side prediction

In this case, the poster should have explicitly separated the assumption, like in this post:

- Assuming the witness returned to the scene of the crime, accept / reject
- No, because upon returning to the scene of the crime, he would have called the police.

Notice the "accept" and "reject" links. If those are missing, such as in this post:

- No, because upon returning to the scene of the crime, he would have called the police, assuming the witness returned to the scene of the crime.

then the user posted it wrong, and it counts as an "unseparated assumption." This should be countered; an assumption needs to be separated when posting, and thus have its accept and reject links, to be valid.

### B.9.4 "not true beyond a reasonable doubt"

A claim must convince you that its conclusion is "true beyond a reasonable doubt." If there's any part of the claim that's not true beyond a reasonable doubt, then it should be countered. For example, if someone makes the claim:

- Eevee had orange juice on his tail and there was orange juice knocked over on the ground. On top of that, Eevee loves climbing on the counter. We can safely conclude that Eevee knocked the orange juice over.

a good counter would be:

- Eevee had orange juice on his tail and there was orange juice knocked over on the ground. On top of that, Eevee loves climbing on the counter. We can safely conclude that Eevee knocked the orange juice over.
  - The above is not true beyond a reasonable doubt: the other cats do walk on the counter, so they could have knocked it over. Eevee's tail could have brushed against the orange juice once it was already spilled.

If you're ever not sure whether something is true beyond a reasonable doubt, ask a moderator, or just factor it out into an assumption.

### B.9.5 "not a claim"

A claim must actually be making a statement. In other words, it must be reaching a conclusion. For example,

- Eevee had orange juice on his tail and there was orange juice knocked over on the ground. On top of that, Eevee loves climbing on the counter. We can safely conclude that Eevee knocked the orange juice over.

Here are some examples of posts that are not making any claim:

- Why would you say that?
- You sound pretty smart, can you email me that research later?
- I agree!
- Turquoise bicycle shoe fins actualize radishes greenly!
- I would like to invest in your country. Please send me your bank details so I can transfer $2,000,000 USD into your account. Sincerely, Prince Fayed W. Bolkiah.

### B.9.6 "factually incorrect"

If any parent of a claim is factually incorrect, then it must be countered. For example,

- We need 10 cars because we have 60 people and each honda civic fits 6 people.

should be countered like this:

- We need 10 cars because we have 60 people and each car only fits 5 people.
  - That's not right, each honda civic only fits 5 people.

Even if someone says something unrelated to their main point, and it's incorrect, you should counter it. For example,

- No, because the car is unreliable; it's broken down dozens of times. That's what you get when you take it to a bad mechanic 6 times.
  - That's not right, we only took it to the mechanic twice.

it may seem like a nitpick, but if we're willing to counter even these things, then it will encourage other users to keep their claims small, so they're not as counterable, which is a good thing for the readers.

### B.9.7 "bad logic / fallacy"

If any part of a claim is making a mistake in logic, or committing a fallacy, then it should be countered.

- Well, Isaac Newton believed in Alchemy, do you think you know more than Isaac Newton?
  - That's not right, even a smart man can believe in something false.
- People with Macs tend to have higher IQs. If you want to be smart, get a Mac!
  - That's not correct, it could be that something else causes someone to both be smart and have a mac; one doesn't necessarily cause the other.
- Everybody hates those Anonymous hackers, they must be evil!
  - That's an appeal to popularity; there are plenty of things that are good that everybody hates because they don't understand it, such as math and skydiving.

### B.9.8 "not a reason for the root"

If someone makes a claim that is not a reason for the root, then it should be countered. For example, if the root is "Macs are better than PCs," and someone makes this claim:

- Yes, because Steve Jobs is a lot cooler than Bill Gates.

it should be countered like this:

- Yes, because Steve Jobs is a lot cooler than Bill Gates.
  - Not a reason for the root; their relative coolness has nothing to do with Macs being better than PCs.

### B.9.9 "not a reason against the root"

If someone makes a claim that is not a reason against the root, then it should be countered. For example, if the root is "PCs are better than Macs," and someone makes this claim:

- No, because Steve Jobs is a lot cooler than Bill Gates.

it should be countered like this:

- No, because Steve Jobs is a lot cooler than Bill Gates.
  - Not a reason against the root; their relative coolness has nothing to do with PCs not being better than Macs.

often times, this is known as a "straw man" fallacy.

### B.9.10 "doesn't counter parent"

Indeed, just because someone says something after you, doesn't mean they've countered what you've said.

And often times, someone posts something that looks like it counters you, but it doesn't (this is often called a "straw man" fallacy).

They may even derive their conclusion with beautiful, perfect logic, but if their conclusion doesn't actually show that its parent claim is false, that's bad.

### B.9.11 "contradicts assumption"

We use assumptions to carry on a debate in a hypothetical sense. For example, "Assuming Kennedy was shot from the building, the shooter would not have been able to run away."

Every child post to that post, and their children, and their children, etc. all need to speak as if the assumption is true. If they accidentally contradict the assumption, that's bad, and means they should be countered.

### B.9.12 "defends assumption"

We use assumptions to factor an unknown out of the debate. If someone then defends their assumption, they're not using assumptions properly, and should be countered.

### B.9.13 "citation needed"

This is probably the most common mistake: forgetting to explain or cite something not obvious that a claim relies on.

For example, "We should have programming courses, because programming raises your IQ."

The claim may be true, but we need evidence before we can accept it.

### B.9.14 "bad citation"

If a claim cites an article, and the claim says that the article says something that it doesn't, then the claim should be countered.

### B.9.15 "cherrypicking"

Sometimes, there's a topic that has a lot of study behind it. For example, there are a few studies that show global warming is not happening, but there are a vast

amount of other studies that show that it is happening. Someone would have to look pretty hard to find the former, and wouldn't be able to find them without at least seeing the latter.

When someone looks for the few bits of evidence that agree with their claim, and obviously ignore all the rest of the evidence, that's called cherrypicking.

### B.9.16 "flying dutchman"

A "Flying Dutchman" is when someone uses a counter when they should have used an edit instead.

For example, if we start with this claim and counter:

- No, Charmander's actually a Virgo.
  - (counter) I somewhat doubt that; citation needed.

and then someone posts a counter to the counter:

- No, Charmander's actually a Virgo.
  - (counter) I somewhat doubt that; citation needed.
    - (counter) Oh, I forgot. Here's the citation: chr.com/virgo.htm

it may look proper, but instead of countering the counter, he should have edited the original post:

- No, Charmander's actually a Virgo, see chr.com/virgo.htm
  - (counter) I somewhat doubt that; citation needed.
    - (counter) Edited, should be correct now, thanks.

Editing like this is better because the citation is now inside the claim, which means the readers don't have to hunt for counters-to-counters which "add" to the claim.

### B.9.17 "other"

If your counter doesn't fit any of the other reasons, then select "Other." Also, if you think it should be included in the list, message a moderator! We may have overlooked it.

## B.10 Claim Menu

### B.10.1 "hide" link

If you don't want to see this claim again (because you've already read it), you can click on the "hide" link to hide it.

You can still see all claims including hidden ones if you click on the "show hidden claims" button at the top of the page.

To unhide a link, click on the "unhide" button in its menu.

### B.10.2 "subscribe" link

If you want to receive an email when this claim is successfully countered or becomes standing again, click on the subscribe link!

We'll send only one email until you visit again.

Once subscribed, you can always unsubscribe by hitting the link again.

### B.10.3 "unsubscribe" link

You are currently subscribed to this claim, which means you'll get an email whenever it is successfully countered or becomes standing again.

Click on the unsubscribe link to stop receiving these emails.

### B.10.4 permalink

What's a "permalink"?

A permalink is just a link to the current claim, that can copy into an email or a document, or send to someone else. When they open the link, their screen will scroll to and highlight the claim.

To use a permalink, just right click and "Copy Link Address."

### B.10.5 "forward" link

If you've posted a better version of this claim, then you can make this claim forward to the new one.[1]

What does forward mean?

"Forward" is just a fancy word for putting in a link to the new claim. All users will see the link to the new claim, in the old claim.

It's a way to let the user know that there is a better version of this claim somewhere.

### B.10.6 "flag inappropriate" link

If you think this claim is inappropriate, then please hit the "flag inappropriate" link.[2]

What counts as inappropriate?

The claim is inappropriate if:

- It's spam or advertisement.
- It's blatantly off-topic.
- It's discriminatory or offensive.

In these cases, please flag it.

The claim is not inappropriate if:

- It has a rather extreme opinion.
- It is only wrong (in this case, you should just counter the claim)
- It has grammatical mistakes

In these cases, please do not flag it.

What will flagging a claim as inappropriate do?

[1] This is not yet implemented.
[2] This is not yet implemented.

Flagging a claim as inappropriate will bring it to the moderators' attention. The moderator can then do a number of things such as deleting the post, banning the user, or investigate further.

### B.10.7 "flag duplicate" link

If you think this claim is a duplicate of another claim, then please hit the "flag duplicate" link.[3]

What's a duplicate claim?

If two claims' texts are identical, then they are definitely duplicate. Also, if two claims are not identical, but they present exactly the same point, then they are duplicates.

Example duplicate claims:

- Claim A: "The moon landing was faked because there are three astronauts in the picture, but only two went up."
- Claim B: "Only two astronauts went up, but three were in the picture, so the moon landing couldn't be real!"

What will flagging a claim as a duplicate do?

Flagging a claim as a duplicate will bring it to the moderators' attention. Often times, duplicates are posted on accident, and in this case, the moderator will delete the post. Sometimes, the moderator will forward one claim to the other, or merge them.

[3]This is not yet implemented.

# APPENDIX C

## SIDEBAR

When a user has guide mode enabled, the right third of the page contains an overview of See the Reason. This appendix contains the text visible there.

*(This bar will help you navigate and understand See the Reason. Uncheck "Guided" to hide it.)*

See the Reason is a tool for collaboratively figuring out if something is true or not.

To participate, try to make as many of your side's posts tentatively-true (green) as possible, and as many of the other side's posts tentatively-false (red) as possible. Try to post some reasons, and counter the other side's posts!

So how do I find out if a claim is true or false?

> If there is no little image in the top right of the claim, such as for "complex" claims (see below for what that means), then you should look at the reasons for and against, weigh them against each other, and make your own decision.
>
> If there is a little image there, such as for "normal" claims, then it's easy to tell:
>
> A green thumbs-up icon means it's tentatively true, A red X icon means it's tentatively false, A pencil means the system needs you to answer some things.

Want to say something?

> First, see if anyone has posted about it before. If nobody has posted before, then feel free to post it yourself!
>
> Get people to participate, and post claims under yours. The more claims there are, the more information we have for figuring out which claims are true or false.

There are four basic ideas at work here:

counters:

A "normal" claim (as opposed to a "complex" claim, explained later) is a regular claim that presents evidence. When someone disagrees with any part of it, they "counter" it by posting another regular claim under it. Then, if the original poster disagrees, he can counter the counter, and so on.

tentatively true / false:

Every "normal" claim (as opposed to a "complex" claim, explained later) has a status. By default, it is "tentatively true". "Tentatively true" means that nobody has countered it. "Tentatively false" means that it has a counter which is tentatively true.

Indeed, one cannot mistake "tentatively true" to mean "true" because See the Reason only knows as much as is posted to it. In the future we plan to introduce statistical analysis, but until then, the claim status is a useful approximation.

assumptions:

If you want to speak in a hypothetical sense as if something was true, you can add an assumption. For example, "Assuming Jack was indeed the culprit," / "Jill couldn't have caused the seg-fault."

If a claim has an assumption, See the Reason will not determine whether the claim is tentatively true or false, but will instead wait for the user to hit the "accept" button or the "reject" button next to the assumption. Unless the user hits "accept" for all the assumptions, the claim will not appear as tentatively true to that user.

All counters must speak as if the assumptions are true, and must try and counter a different aspect of the claim.

Remember, if a claim has a tentatively true counter, it is tentatively false. Even if the user has accepted all the assumptions, a tentatively true counter will make a claim tentatively false.

We can also make assumptions about the reader. For example, "Assuming you care about the temperature of the rooms" / "Room A is better because it has air conditioning."

This way, users who care about the temperature can accept the assumption, and those who don't can reject it.

normal/complex claims

There are two types of claims: “normal” and “complex”. A “normal” claim is one that makes a statement, presents evidence, and lists any assumptions that must be true for it to be true. Normal claims can be “countered” by other normal claims. A “complex” claim is something like “Macs are better than PCs” or “We should pass gun control laws”. A complex claim can have normal claims be reasons for, and normal claims be reasons against.

“Complex” claims will have a list of normal claims that sway the reader to agree with it (“reasons for”), and a list of normal claims that sway the reader to disagree with it (“reasons against”).

See the Reason will judge the reasons for and against, and the reader can look at those to come to his own conclusion about the complex claim above.

# APPENDIX D

## CERTAINTY MEASURE

Claims can be tentatively true or tentatively false, depending on the existence of tentatively true counters, or any rejected assumptions. However, just because a claim is "tentatively true" doesn't mean it's true, and just because a claim is "tentatively false" doesnt mean it's false.

One case in which a claim's status does not make sense is when it has just been submitted. In the few seconds since a claim has been submitted, it has been seen by very few people. In this case, we cannot trust the claim's status to reflect its actual truth; nobody has seen it yet, so nobody has been able to counter it yet.

However, if it's been an hour, and a thousand people have seen the claim, and nobody has been able to counter it, then it's much more probable the claim is true. We can't be absolutely sure, but we can be more certain than before.

A much better approximation would be how many people have seen the claim ever since it's become tentatively true. If nobody has seen it, then we're not certain it's true. If a thousand people have seen it and nobody's countered it yet, then we can be much more certain it's true.

The certainty measure can be thought of as a rough measure of the number of views since the claim was submitted. If not many people have had a chance to react to the claim's new status, then the certainty is low, near 0. If a lot of people have had a chance to react to the claim's new status, and nobody has, then the certainty is high, near 1. In our implementation, a claim's certainty will be low, near 0.1, until 100 people have viewed it, and then once perhaps 2000 people have seen it, it will be high, near .95. Then, we taper off the certainty so that no matter how many people have viewed it, it never gets to 1. A logistics curve fits these requirements nicely. The full equation is this:

$$\frac{1}{1 + e^{(-.5 \cdot steepness \cdot (t/numViewsTo50Percent - 1))}} \tag{D.1}$$

In Java, this would look like:

```
private static final double numViewsTo50Percent = 1000.0;
private static final double steepness = 10.0;

static double certainty(int numViews) {
        return 1/(1 + Math.pow(Math.E, -.5 * steepness * (numViews /
            numViewsTo50Percent - 1)));
}
```

With "steepness" set to 10 and "numViewsTo50Percent" set to 1000, the graph looks like Figure D.1.

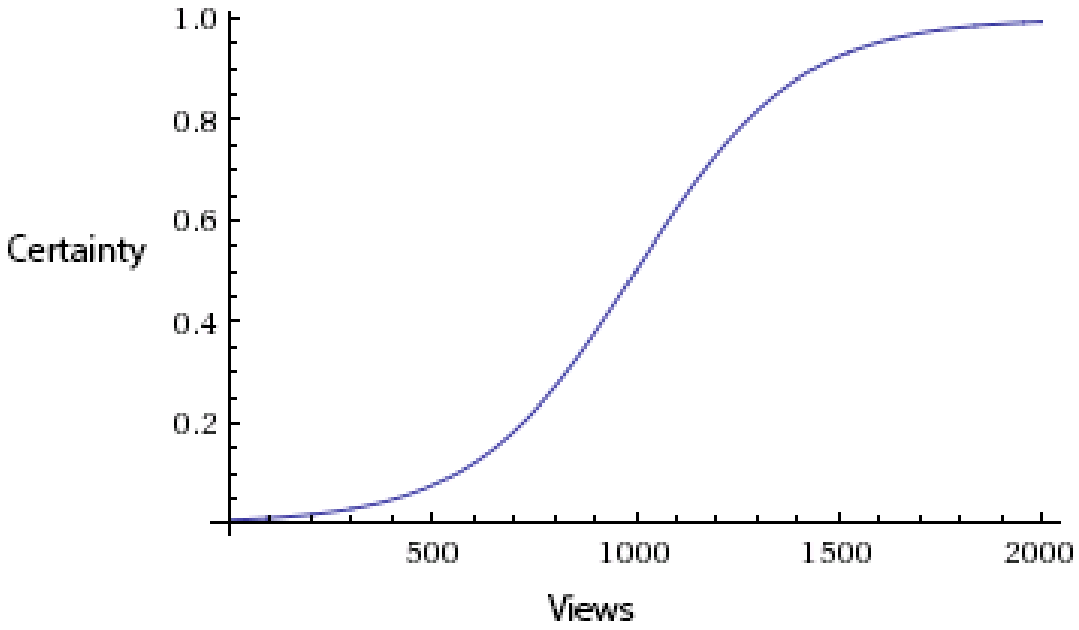


**Figure D.1: A graph of certainty given number of views**

Table D.1 contains some examples of claims with different numbers of views, and their resulting certainty values.

This method is a very basic estimate, and has many theoretical weaknesses. It can suffer from Reddit Hivemind Syndrome, and it's oversensitive to very short-term changes in status. Below are ideas on how to solve these problems.

| Views | Certainty |
|---|---|
| 0 | .0067 |
| 1 | .0067 |
| 2 | .0068 |
| 10 | .0070 |
| 100 | .0110 |
| 250 | .0230 |
| 500 | .0759 |
| 750 | .2227 |
| 1000 | .5000 |
| 1250 | .7773 |
| 1500 | .9241 |
| 1750 | .9770 |
| 1950 | .9914 |
| 2000 | .9993 |
| 3000 | .9999 |

**Table D.1: Example certainty values given number of views.**

### D.0.8 Defining Certainty Recursively

It would make sense to have a counter's certainty negatively affect a parent claim's certainty. So, much like how claim status is determined recursively, we can determine certainty recursively. In java, this would be:

```
class Claim {
        int numViews;
        List<Claim> counters;
}

private static final double numViewsTo50Percent = 1000.0;
private static final double steepness = 10.0;

static double baseCertainty(int numViews) {
        return 1/(1 + Math.pow(Math.E, -.5 * steepness * (numViews /
            numViewsTo50Percent - 1)));
}
```

```
static double certainty(Claim claim) {
        double c = baseCertainty(claim.numViews);
        for (Claim counter : claim.counters)
                c = c * (1 - certainty(counter));
        return c;
}
```

For example, if a claim is posted, and after 4000 views, a counter is posted, it would look like Figure D.2.

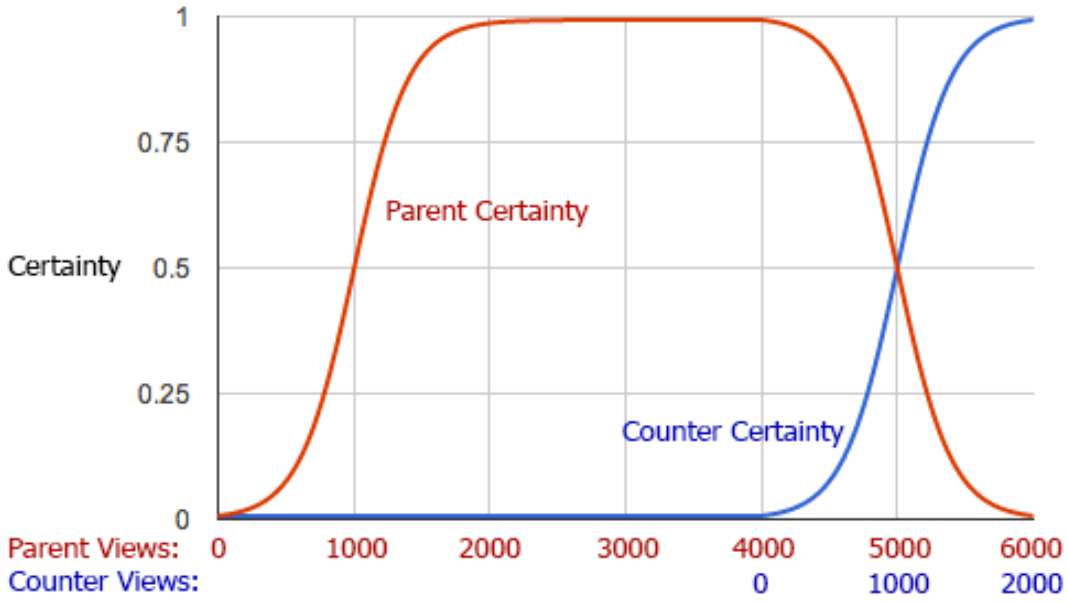


**Figure D.2: A graph of a claim and its late counter's certainties, when certainty is recursively defined**

If a claim is posted, and after just 1000 views, a counter is posted, it would look like Figure D.3.

For example, if a claim is posted, and a counter is posted immediately, it would look like Figure D.4.

### D.0.9 Avoiding Reddit Hivemind Syndrome

(Details on Reddit Hivemind Syndrome can be found in appendix H.)

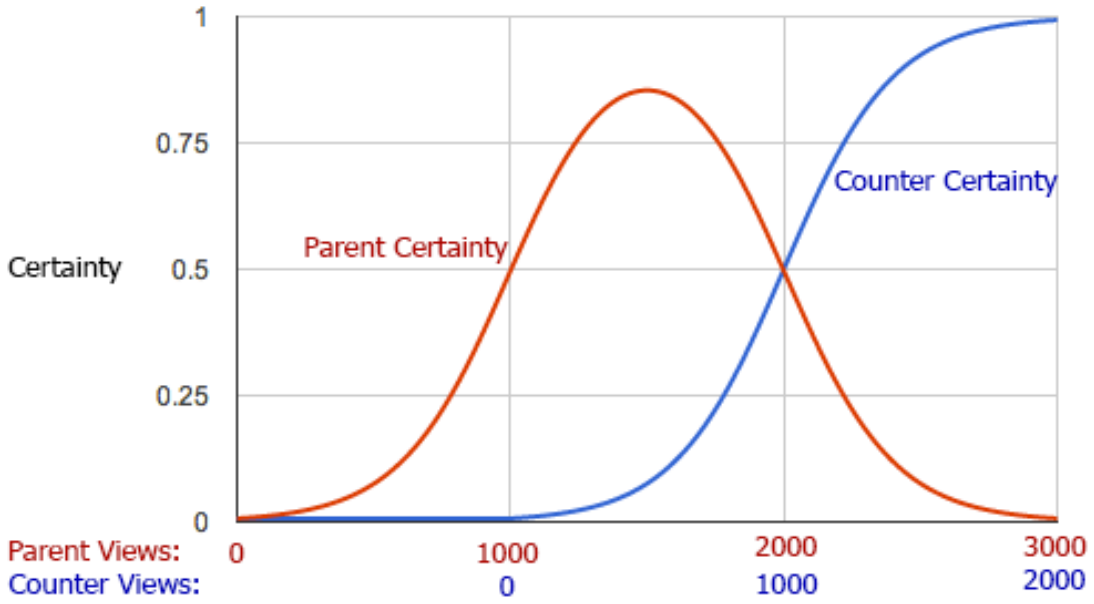


**Figure D.3: A graph of a claim and its counter's certainties, when certainty is recursively defined**

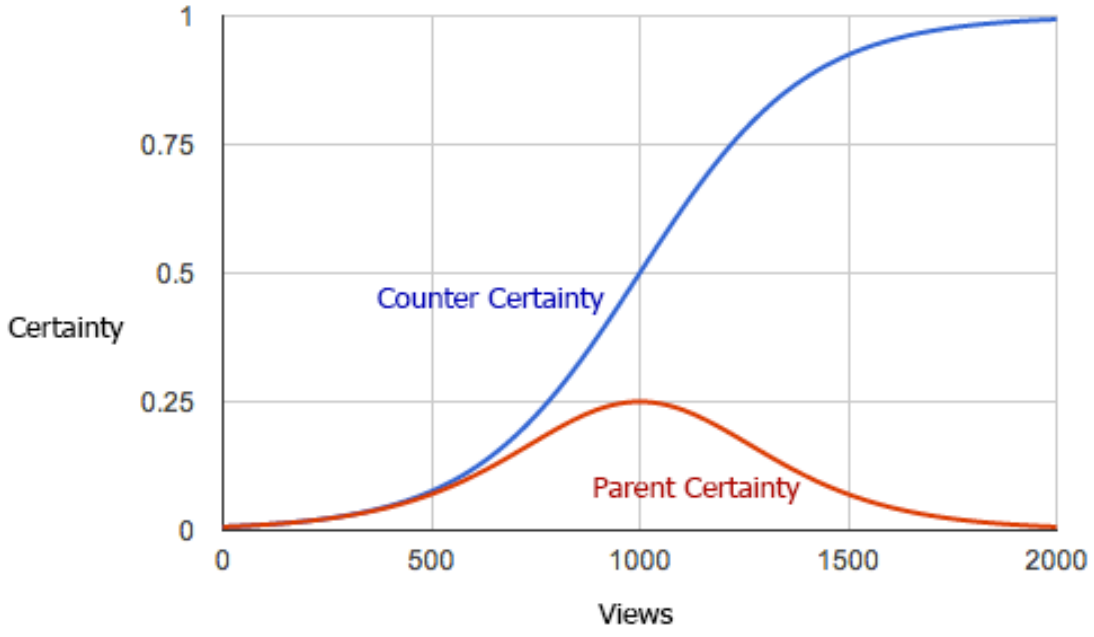


**Figure D.4: A graph of a claim and its immediate counter's certainties, when certainty is recursively defined**

Imagine we have a debate where there are 4000 people for the root claim, and 40 people against the root claim. If one of the 4000 submits a claim, there's a good chance that 1000 of the "for" people will see it before any of the "against" people have a chance to react. This means that the certainty will be very high, when it shouldn't be.

For this, instead of counting the raw number of views since the last status change, we will weight each user's view differently. In this case, we will give each of the 40 people 100x the weight of any of the 4000 people.

Of course, we have another basic problem: how do we identify who is on which side? For that, we can use the bias measure, explained in appendix G.

These two measures should eliminate the Reddit Hivemind Syndrome in See the Reason.

# APPENDIX E

# COMMON QUESTIONS

## E.1 How do you know if something is true beyond a reasonable doubt?

There are three things that we use to address this problem:

First, we leave it up to the users to decide. If any user thinks that it's not true beyond a reasonable doubt, then they can counter it as such.

Second, if the users disagree on what is true beyond a reasonable doubt, then we bring in a moderator. If a moderator thinks it is true beyond a reasonable doubt, then he can step in and put in the right counters. If people think the moderator is wrong, then the decision can be appealed to random other moderators (see the moderation section).

Third, if a moderator rules that your claim is not true beyond a reasonable doubt, then you can use an assumption. For example, "Assuming that Eevee being at the scene of the crime and having orange juice on his tail establishes it as true beyond a reasonable doubt that he knocked over the orange juice," and then viewers can decide for themselves whether it's beyond a reasonable doubt.

## E.2 How will you enforce that edits do not change the core meaning of claims?

In the current version, we use the honor system.

In the future, when a user edits a claim, the edit will be recorded in the claim's "history." There will be another link on the claim widget (perhaps in the "more" menu) that will show any edits made to the claim, similar to how StackOverflow shows it:

Under the claim, there will be a link that the user can click if any of the edits changed the core meaning of the claim.

An additional possibility is that we won't even wait for the user to hit the "more" menu; if the claim has been recently edited, we'll show the edits anyway.

Then, after a certain number of views, the system will conclude that the edit is probably not changing the core meaning of the claim (otherwise, someone would have clicked on the link by then) and will no longer show the edits right away.

# APPENDIX F

# EVOLUTION OF A TREE

A future experiment would be to determine if in See the Reason makes a topic more complete and useful as more people participate and as time goes on.

This can happen on argument mapping platforms. Klein and Iandoli (2008) said,

> As the system scales, we can assume that the argument-mapping burden per user will decrease. This is because the number of mapping-savvy users should scale linearly with the overall user population, but the number of novel ideas that need to be mapped should scale (as we mentioned above) less than linearly. If the Collaboratorium works like most OSPP systems, user contributions will follow a power law, so we can expect that a relatively small corps of "power users" will take on the bulk of the argument mapping tasks. Most people will just have to read or comment on, as opposed to create, argument maps.

However, on threaded discussions like forum threads, the argument-mapping burden per user will not decrease. Even though a debate on a forum thread can become more complete as time goes on, just because more and more posts are added to it, the new information in each new post drops off over time; the longer a thread is, the more likely that what you are saying has already been said. On top of that, even if users are careful, it becomes harder and harder to avoid repeat posts because the thread gets larger over time. As more posts with less new information are added, the signal to noise ratio decreases, and the argument becomes less useful as a whole.

The burden per user will also increase on systems like Reddit. If we assume that the average user only looks at the top 100 posts in a given topic, then any information that is not in those top 100 posts will be posted over and over again. Because of this, the signal-to-noise ratio will unfortunately decrease.

In the same way, See the Reason might decrease its signal-to-noise ratio over time. Even though the judgment on each particular claim might become more certain over time (see section D), people will not take the time to read the entire

argument before submitting. In this way, the chance of duplicates increases over time, and the signal-to-noise ratio will decrease.

This has been solved on Stack Overflow, which can detect potential duplicate posts before they are submitted. When someone is typing a question into Stack Overflow, before the user submits the question, it suggests similar questions that were already submitted. Often, these posts will contain the information that the user needed. Because of its duplicate detection, less and less new posts will be posted, because the chance of something answering your question increases over time. In this way, a given category of stack overflow questions becomes more and more complete and useful over time.

The planned duplicate-detection feature of See the Reason aims to accomplish the same thing as Stack Overflow did. If See the Reason can detect duplicate claims before they are submitted, then it will hold true that potential duplicate claims will be less likely submitted, while the claims that contribute new information will not be hampered. In this way, the signal-to-noise ratio will not decrease infinitely over time, and the new useful claims will not be drowned out by the constant flow of duplicate claims.

We will now try to prove that the more people view a point, a given tree will get closer and closer to the ideal tree.

For any claim, it can be either tentatively true or false, and it can be true or false in reality. That means that there are four possible situations.

In the first situation, when a claim is marked tentatively true, and it is actually false, then as more users view it, the chance of someone recognizing it as false increases. Once someone recognizes it as false, they can counter it explaining why. At that point, it will become tentatively false and false, which brings the tree closer to an ideal tree. This case also happens for the second situation, when tentatively false claims that are actually true (such as when something is true, but mistakenly countered).

In the third situation, when a claim is tentatively true, and it is indeed true, then as more people see the claim, the chance of someone mistaking it as false

when it is actually true increases. The user will then counter it, bringing us further from the ideal tree. This case also happens for the fourth situation, when tentatively false claims are false (someone might mistakenly counter a false claim's counter, thus making the original claim tentatively true)

It may seem like these two forces on a given point are equal and opposite, but in fact, they aren't: when someone mistakenly counters a claim and then their counter is countered, anybody who tries to counter the original claim for the same reason will be advised by the duplicate-detection feature that it has already been tried; that what they are about to claim is a duplicate. This will inform the user and discourage him from trying to counter the original claim in that way.

There are plenty of ways to mistake a true claim as false, but as more and more people see the claim, more of those will be tried, and less will be available to try (because repeats are discouraged by the duplicate detection). Because of this, the second case (someone countering a true claim) will probably become less common as time goes on, while the first case remains just as probable as time goes on.

# APPENDIX G

## THE BIAS MEASURE

It would be very useful to be able to programmatically determine a user's bias. In other words, whether a user wants the root claim to be true or false. It would be useful for many reasons:

- for factoring into the certainty measure by weighing more heavily the views from people biased against the claim,
- for factoring into upvotes by weighing more heavily upvotes from people biased against the claim
- for choosing moderators from people with low bias scores
- for identifying trends in assumption judgments for those with similar biases

For this, we have the bias measure. We gauge how biased someone is by how unevenly they post to a given claim. For example, we have these three users:

- User 1 is consistently posting in favor of the root claim
- User 2 is consistently posting against the root claim
- User 3 is posting evenly for and against the root claim

Users 1 and 2 will have a high bias measure, and user 3 will have a low bias measure.

# APPENDIX H

# REDDIT HIVEMIND SYNDROME

Reddit functions according these basic principles:

- Each user can give one upvote or one downvote to each post.
- A post's "score" is the number of upvotes minus the number of downvotes.
- Posts are ordered according to score: posts with higher scores appear before those with lower scores.
- Posts with low enough scores are hidden.

These principles have served Reddit quite well, but some interesting behavior has emerged, something called "the hivemind."

A hivemind is where everyone in a Reddit community shares mostly the same opinion, and downvotes any opposing opinions. A hivemind is an echo chamber, where only one type of view is expressed. For example, the prevalent posts on the r/politics subreddit have much more liberal views than the general population.

It's unknown exactly what causes a hivemind to form, but our theory is that even a slight imbalance in the population's viewpoints will become more and more extreme, in a positive feedback loop:

- Imagine there are more people with viewpoint A than viewpoint B.
- Aside from upvoting posts people thing are well-crafted and informative, people also tend to upvote posts they agree with. This means that posts expressing viewpoint A will tend to get more upvotes than viewpoint B, and viewpoint B will get more downvotes than viewpoint A.
- Because of this, posts expressing viewpoint A will have positive scores, and posts expressing viewpoint B will have negative scores.
- This creates two effects:

  - The author of the post expressing viewpoint B will see that he has a negative score, and feel rejected by the community, and be less likely to stay compared to an author of a post expressing viewpoint A.
  - Because posts with viewpoint A have higher scores, they are more visible. Users will see that there are more posts they disagree with, and be more likely to move to a different website than users that see many posts they agree with.

- In the end, people with viewpoint A are more likely to stay, and people with viewpoint B are more likely to leave. It's an endless cycle that leads to extremely polarized communities.

Also encouraging the positive feedback loop is the tendency of users to mimic successful posts. Mills (2011) writes,

> Reddit's voting system contains a number of positive feedback loops. Firstly, positive votes for a post are associated with increased visibility and in turn further votes leading to the emergence of a small number of superstar' posts. These superstar' posts are displayed on reddit's front page for the whole community to see, and serve as shining examples of the kind of content which is successful on reddit. If a user wished to submit content which the reddit community would appreciate, a potentially fruitful strategy would be to mimic the types of post which they see on the front page. It would seem that many users adopt such a strategy; it is quite common for a sub- reddit to be flooded with new submissions which emulate a post that is currently performing well in that sub-reddit.
>
> Reddit's penchant for up-voting certain types of post is often discussed by users. The tendency for users to submit posts of a type which have previously been received well is known in the community as circlejerking' and a sub-reddit exists for the purpose of highlighting and satirising these trends (the circlejerk' sub-reddit). This, and the perception that posts which go against certain commonly held beliefs or mores are always down-voted, are two of the main characteristics of what reddit users often refer to as the "reddit hive mind". [41]

See the Reason was crafted to avoid this behavior. Specifically,

- See the Reason discourages duplicates, which should cut down on users mimicking successful posts.
- See the Reason has upvotes, but not downvotes.
- See the Reason does not compare posts on one side of the discussion with posts on the other side.

For example, there could be 3 reasons for the root claim, with 9, 8, and 6 upvotes, and 3 reasons against the root claim, with 100, 91, and 50 upvotes. In reddit, the three reasons against would be displayed before the three reasons for. In See the Reason's Split View mode, the reasons for are displayed in a separate column than the reasons against, so they are equally visible. In See the Reason's Combined View mode, the view alternates between displaying reasons for and reasons against, ensuring that both sides are equally visible.

# APPENDIX I

# MAKING SEE THE REASON

The original idea for See the Reason was nothing like it is today. It had the basic cores features of objective/subjective claims and judging objective claims to be tentatively true/false (back then known as "uncountered" and "countered,") but not much else.

## I.1 Assumptions

After a couple years of having this idea, we would look at various arguments and see if they would fit into the model. It seemed to fit nicely, but the discussions seemed to reach standstills when people hit different underlying assumptions, or any bit of uncertainty. At that point, the model broke down. Once we added the assumption though, it seemed to handle all use cases. The assumption feature makes the entire model very flexible.

## I.2 Standard of Truth

Some people brought up a very important issue: what should "uncountered" mean to the user? We had a notion of what it meant, but it was very nebulous and hard to explain. We concluded that it means that "this claim is true beyond a reasonable doubt as far as our community knows."

## I.3 Supports for Objective Claims

The most important disagreement within the team was about whether objective claims should be able to have supporting objective claims.

The biggest argument for having subjective claims was that it was awkward and unintuitive to shoehorn an entire debate, with supports and counters, into a mode of debate with only counters[1]. In other online discussion platforms, if

[1] for an example of this, see Figure 3.12

someone makes a claim, other people can jump in and add their own experiences and evidence to make the claim stronger and stronger.

In a system with just counters, if someone sees a claim that they wish to support, then they should just add a new claim with the same text as the old, except with the new evidence added to it. The major drawback of this is that more and more near-duplicate claims will flood the system. This could be alleviated by the fact that we sort the claims, so the older claims can just fall to the bottom, out of sight. But still, arguing in this system is awkward.[2]

However, if we had supports for objective claims, the system would no longer be able to calculate if a claim was tentatively true or tentatively false. For example, if we started with this:

- Mankind caused global warming, it's way hotter nowadays than before.

and someone countered with this:

- Mankind caused global warming, it's way hotter nowadays than before.
    - Counter: That can't be true, the warming could have been caused by the massive chain of volcano eruptions we've seen in the past fifty years.

and someone else supported with this:

- Mankind caused global warming, it's way hotter nowadays than before.
    - Counter: That can't be true, the warming could have been caused by the massive chain of volcano eruptions we've seen in the past fifty years.
    - Support: We've seen a vast rise in N2O, which is only emitted from cars, and the temperature has gone up in tandem with its rise in concentration.

We would run into an interesting question: how would we judge the original claim to be tentatively true or tentatively false? It indeed had a valid counter,

[2]The system became a bit less awkward when we introduced editing, see section I.24

but in the meantime, someone has made the claim stronger with a support. Is the argument now strong enough to overcome the counter? The system can't know, with only the information given.

There were a few ideas on how to modify the tentatively true/false system to handle this.

Perhaps we could have said, "a claim is tentatively true if it has more supports than counters," but it would have been a meaningless comparison because a hundred weak supports shouldn't necessarily beat one strong counter.

Perhaps we could have said, "a claim is tentatively true unless it has a tentatively false support or a tentatively true counter," but it had a weakness: a crafty opponent would only have to make a bogus support such as "Eevee's a good cat" and then immediately counter it himself with "The above claim is not objective," and the parent claim would be forever tentatively false because it is supported by a tentatively false claim.

Perhaps we could have said, "a counter claim has to identify a weakness that exists in a claim and all of its supports," in other words, consider a claim and all of its supports to be one large argument, and the counter would have to address the argument as a whole. But if this was the case, then these "arguments" would grow larger and larger as the number of supports grew. Anyone who wanted to counter a claim would have to read through all of the supports, to make sure that their counter makes sense when considering all of those supports together. In fact, they wouldn't just look at all of the supports, they need to look at all of the supports' supports and the supports of those supports and so on. The situation could get out of hand quickly.

In the end, we couldn't figure out a system that would be able to judge whether a claim is tentatively true or tentatively false. It seemed that objectives' supports and the claim judgments were mutually exclusive features. After much debate, we decided to continue not allowing supports for objective claims.

## I.4 Editing

The team's next big decision was whether we should allow or disallow the editing of claims. It would be very convenient to be able to edit a claim; that way, if you didn't offer enough evidence, you could edit it in. However, there was a basic problem here. For example, given this discussion,

- Mankind caused global warming, it's way hotter nowadays than before.
    - Counter: There's no evidence that it's hotter nowadays.

if the original poster edited their post to be this:

- Mankind caused global warming, it's way hotter nowadays than before. [http://www.nrdc.org/globalwarming/f101.asp]
    - Counter: There's no evidence that it's hotter nowadays.

then the counter wouldn't make sense anymore; someone who looked at the argument in that state might be slightly confused. Perhaps we could make a rule that "if you edit a claim in response to someone's counter, you have to counter that counter to say that you edited it." For example,

- Mankind caused global warming, it's way hotter nowadays than before. [http://www.nrdc.org/globalwarming/f101.asp]
    - Counter: There's no evidence that it's hotter nowadays.
        - Counter: Edited to include evidence, thanks.

If we have that rule, then it could possibly work. However, there was another problem. If we started with this debate again,

- No, Eevee did not knock the orange juice over because nobody saw him go in the house.
    - Counter: That's false, Mike saw him go into the house.

and the original poster didn't know what he was doing, he might have edited his post in one of the following ways:

- No, Eevee did not knock the orange juice over because he can't jump onto the table.
    - That's false, Mike saw him go into the house.

In the above tree, the original poster edited his post to describe a different reason why Eevee did not knock the orange juice over. Instead, he should have left the original post as tentatively false, and made a separate post with this new information. This was bad because:

- It meant the tree no longer made sense to the readers.
- It meant we've lost some valuable information. People who also thought that nobody saw him go into the house will no longer be able to find out that they are mistaken.

The original poster might also instead have edited the post to be one of these, which are bad for the same reasons:

- Alright, nevermind.
    - That's false, Mike saw him go into the house.

- The guy below me is wrong, Mike's not reliable.
    - That's false, Mike saw him go into the house.

Because we couldn't figure out a good solution to these problems, we decided we wouldn't have editing.[3]

### I.5 Experiment A: the 508 experiment

In our first experiment, we had the basic building blocks in place, but as is true for any first prototype, the experiment uncovered some user interface issues.

Our prototype would often change the claims' ordering when new posts were added; every time someone posted something, it would request the entire tree from the server, and every response's claims were randomized. Users did not

[3]Though later on, we did figure out a good approach though (see section I.24)

appreciate this. We decided that our next version would have to order the posts in a consistent way.

Another issue with the random ordering was that the well-said posts had as high of a chance of being at the top as the lesser posts. From this, we learned that we should use an upvoting system like Reddit's. Of course, we wanted to avoid the Reddit Hivemind Syndrome, so we decided to try something very interesting: instead of ordering by "upvotes minus downvotes," we would order by "upvotes plus downvotes." in an effort to bring more controversial points to the top of the tree. We later abandoned this approach because universally hated points, such as those containing gibberish or spam, would be shown just as much as beautifully crafted, logical posts. We decided that just ordering by upvotes, and ignoring downvotes, would be sufficient.

Some participants mentioned that it was slightly difficult to visually distinguish between subtrees of the argument. While this was certainly exacerbated by the random movement of subtrees described above, there was still an important point there. We learned that we needed to do everything we could to make it easy for the user to track how to get from the root of the tree to the argument they are currently reading, so they could more easily know the context of a given claim. We planned to make the ancestors of posts more obvious. We also considered a few ways to keep the overall topic and the direct parents for the post that is being read at the top of the screen.[4]

There was one area in which the system excelled though: hiding irrelevant posts. There were several posts that were completely irrelevant to the discussion (jokes, etc.), and the users countered them easily, making them tentatively false. Our system would automatically collapse any tentatively false posts, so that to see it, a user would have to click on it first. This greatly increased the signal-to-noise ratio, which was an unexpected triumph.

At this point, the year was over, and we resolved to start over and use different technologies to implement See the Reason, because we had been using Google

[4]Single Branch Tree View (see Figure I.2) did this quite well.

Web Toolkit (GWT), which had proven to be a terrible mistake.

GWT was an interesting way to program in Java for both the back-end and the front-end; it would automatically compile Java code to Javascript for use in the browser. As an added benefit, it also managed all cross-browser coding, so the programmer wouldn't have to worry about compatibility with less popular browsers like Opera and Safari, and universally reviled browsers such as Internet Explorer.

We originally chose GWT because it used Java, which is arguably a much better language than Javacript, mostly because of its static typing, and lack of the "undefined" value.

However, GWT had a myriad of problems:

- At the core of GWT was the Element class, which represented an html element. However, it lacked some very basic abilities, such as the ability to find descendant elements via a CSS selector. For example, in jQuery (a library commonly used with Javascript), one could say:

  ```
  $("> div li.helpItem", myElement)
  ```

  to look inside all of the child div elements and find any li element anywhere inside them that have the "helpItem" class. With GWT, one would need to write a 10-line function with a 15-line recursive helper function to accomplish the same thing.

- The webpages it generated were incredibly slow, even when compiled in release mode. We were never quite sure why this was the case.

- Its built-in library of widgets was extremely limited and not at all extensible. For example, we would not have been able to create the "Helpable" widget as shown in section 4.2.2. To fit it into a GWT-centric project, we would have had to create an external library in Javascript, and wrap it in Java bindings. Needing to write a library in Javascript defeats the purpose of using GWT; we chose GWT to avoid Javascript in the first place.

While the static typing from Java was a godsend, it was not worth GWT's other troubles, especially how the most complex widgets in our application had to be written in Javascript anyway. For all of these reasons, we decided to abandon GWT.

## I.6 Choosing Languages and Systems

With the new year and a fresh start, we reconsidered our entire stack of technologies behind See the Reason. The first thing we reconsidered was which database technology to rely on.

### I.6.1 Database

Our requirements for a database system were as follows:

- Must be fast.
- Must be able to store a directed acyclic graph.
- When asked for all descendants of a given node, must respond immediately, but when asked to put a new child to a given node, changes do not need to be immediate.
- Must be run on commodity hardware, distributed.
- Schema may change every month or two, as features are added.
- Must be data-mining friendly; must be easy to build complex queries.

There were three main contenders:

- Relational Database using transitive closure tables (nested interval trees and nested sets would not work because they only handle trees, not directed acyclic graphs)
- NoSQL Databases such as Accumulo or Cassandra
- Graph Databases such as Neo4j or HyperGraphDB

#### Graph Databases

Since we were storing a directed acyclic graph, the graph databases option was an obvious contender. There were a few drawbacks:

- We had no experience with graph databases
- Graph databases were rather new and untried.
- It's difficult to distribute graph databases over multiple machines.

and a few benefits:

- Storing a directed acyclic graph is extremely simple.
- In the case of Neo4j, it came bundled with some advanced text searching functionality [59]

### NoSQL Databases

NoSQL databases, such as Cassandra and Accumulo, were basically gigantic hash maps, with a string key and a string value. There were some drawbacks:

- Requesting a given DAG containing n claims from the database would require $O(n)$ requests, which is extremely slow.
- It's impossible to search by value. For example, to find all users that have the password "moo123" is an $O(n)$ operation.

and a few benefits:

- Easy to distribute across machines compared to the other two methods.
- Since there was no structure to the "columns," adding new features would be very easy. [16]
- In the case of Accumulo, it came with a very good security model, where every cell has an access control list.

### Relational Databases with Transitive Closure Tables

One of the most common models in use at the time was the relational database [10]. Paired with a technique called transitive closure tables[5], they can be very fast representations of directed acyclic graphs.

Specifically, we were considering MySQL. There were a few cons:

- It would be very complicated to store directed acyclic graphs in a relational database, requiring at least two tables: one for the claims themselves, and one for the relations between them.

[5] transitiveclosuretables

- Using a transitive closure table would have meant memory usage of up to $O(n^2)$ depending on how interconnected the different discussions are. This could be prohibitively large in the long run.
- It would be somewhat difficult to distribute simple data over multiple machines, and distributing a transitive closure table might have been downright impossible.

and of course, there were a few benefits:

- Requesting a given graph would be lightning fast: a simple query on the transitive closure table could get all descendants for a given node much faster than a graph database could.
- Everyone on our team was experienced with MySQL.
- MySQL was a very stable and secure technology.

We ended up choosing MySQL because most of the drawbacks were long-term and all of the benefits were short-term. See the Reason will likely go through many rewrites and redesigns anyway before becoming popular enough that the drawbacks would manifest themselves.

We didn't immediately use transitive closure tables, because we didn't want to prematurely optimize. For the time being, we simply used recursive calls to the database to get all the children for a given node. By repeating that call multiple times, we could get all of the descendants for a given node.

Given that we chose MySQL, another question was whether we would put a database abstraction layer on top of it, such as Hibernate. These layers could be very helpful for loading the serialized MySQL result data into Java or PHP objects. We were reluctant to choose Hibernate though, because it would be difficult to optimize since Hibernate generates the underlying SQL schema itself. After looking through many libraries that could serve our purposes without hindering us in that way, we settled on a small library called MyBatis.

### I.6.2 PHP vs Java

The choice to use Java was obvious. PHP was a terribly designed language, for too many reasons to write here, but here are the two main ones [12]:

- No static typing. Most of our errors in dynamic-typing languages like Javascript or PHP were because somewhere, we've passed to a function values of the wrong type, or added extra parameters to a function without updating all of the calls to that function.
- The built-in functions have inconsistent parameter orders. Even after many years of PHP, this had been a constant source of bugs. For example:
    - `array_key_exists($needle, $haystack)`
    - `property_exists($haystack, $needle)`
    - `in_array($needle, $haystack)`
    - `strstr($haystack, $needle)`

One might wonder why we would choose the dynamically-typed Javascript over GWT's Java in the front-end, but choose the statically-typed Java over the dynamically-typed PHP in the back-end. While we prefered static-typing in the front end as well, the other drawbacks of GWT overwhelmed and outweighed the benefits there. GWT's devastating inflexibility overcame the massive benefits of its static typing.

### I.6.3 Model

#### Abstraction over the Database

We knew that someday, eventually we would have to move away from MySQL; its space requirements (specifically the transitive closure table) are so heavy that it's just not scalable for storing directed acyclic graphs. Knowing this, we decided to make an abstraction between the server and the database, so that we could more easily switch databases in the future, when those disadvantages start to manifest.

This presented an interesting challenge. Abstracting over different relational databases is easy, all sorts of systems do it. However, abstracting over relational, NoSQL, and graph databases all at the same time is quite unusual.

Luckily, they seemed to all share one representation in common: the ER diagram. We figured out a way to represent our data as an ER diagram, and then we came up with an algorithm to turn an ER diagram into a set of Java interfaces:

- Create an empty IDatabase interface
- For each entity type [E]:
  - Make an interface [IE] with a getID() method, and a setter and getter for each of the rest of the attributes.
  - Make a createTransient[E] method in the IDatabase which returns [IE]
  - Make an update[E] method in IDatabase
  - Make a find[E]ByID method in IDatabase
  - Make a delete[E] method in IDatabase
  - Make an insert[E] method in the IDatabase, which:
    - For every one-to-many relationship [ER] involving [E]
      - let [ERO] be [ER]'s "one" end,
      - let [ERM] be [ER]'s "many" end,
      - if [E] is [ERM]: Add an argument to insert[E] that takes in [ERO]
- For every one-to-many relationship [R]
  - let [ERO] be [ER]'s "one" end,
  - let [ERM] be [ER]'s "many" end,
  - Add a method to IDatabase which retrieves the [ERO] given a [ERM]
  - Add a method to IDatabase which retrieves all [ERM] for a given [ERO]
- For every many-to-many relationship [R]
  - let [ER1] and [ER2] be the two ends of the relationship
  - Add a method to IDatabase which retrieves all [ER1] for any [ER2]
  - Add a method to IDatabase which retrieves all [ER2] for any [ER1]

So for example, given the ER diagram in figure I.1,

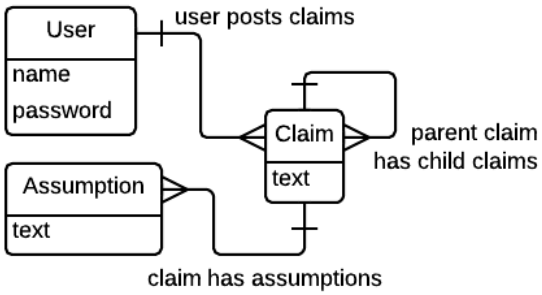


**Figure I.1: An ER Diagram**

the corresponding interfaces would be:

```
interface IClaim {
   UUID getID();
   void setText(String text);
   String getText();
}

interface IAssumption {
   UUID getID();
   void setText(String text);
   String getText();
}

interface IUser {
   UUID getID();
   void setName(String name);
   String getName();
   void setPassword(String password);
   String getPassword();
}

interface IDatabase {
   // Relations
   IUser findPostingUserForClaim(IClaim claim);
   List<? extends IClaim> findClaimsPostedByUser(IUser user);
   IClaim findParentClaimForClaim(IClaim childClaim);
   List<? extends IClaim> findChildClaimsForClaim(IClaim
      parentClaim);
   IClaim findClaimForAssumption(IAssumption assumption);
   List<? extends IAssumption> findAssumptionsForClaim(IClaim
      claim);

   // Claim methods
   IClaim createTransientClaim();
```

```
    IClaim findClaimByID(UUID id);
    void updateClaim(IClaim claim);
    void deleteClaim(IClaim claim);
    void insertClaim(UUID id, IClaim newClaim, IUser postingUser,
        IClaim parentClaim);

    // User methods
    IUser createTransientUser();
    IUser findUserByID(UUID id);
    void updateUser(IUser claim);
    void deleteUser(IUser user);
    void insertUser(UUID id, IUser newUser);

    // Assumption methods
    IAssumption createTransientAssumption();
    IUser findAssumptionByID(UUID id);
    void updateAssumption(IAssumption assumption);
    void deleteAssumption(IAssumption assumption);
    void insertAssumption(UUID id, IAssumption newAssumption, IClaim
        claim);
}
```

The nice thing about this algorithm is that since an ER diagram is implementation-agnostic, and it is the input to this algorithm, the results of the algorithm are implementation-agnostic. In other words, the resulting set of interfaces is compatible with NoSQL databases, graph databases, java serialization databases, or relational databases. Because the resulting interfaces are compatible with any kind of database, they are perfect for decoupling the database and the server, and perfect for reserving the option of switching to a different database down the road. Also, since ER diagrams are incredibly flexible, the resulting interfaces are flexible as well.

### IModel

Once we had a flexible, decoupled database abstraction, we could design the rest of the model, which specifies the computation using that data.

There were six main operations that the model needed to handle[6]:

- Judge claim status
- Assemble tree page data
- Assemble front page data
- Email subscriptions (called by server every ten minutes or so)
- Heartbeat: called by browser every 5 seconds to see if a tree has changed, involves hashing an entire tree into a checksum to be compared with on the client side
- Methods for adding/updating/deleting/finding underlying entities for the REST service

### Ensuring Quality

Since we were starting with a clean slate, we decided to put into place a few measures to ensure quality: using assertions everywhere, unit testing with 100% code coverage, and double-implementing the IModel.

The main subclass of IModel used MySQL of course, but we decided to also develop a simpler subclass which didn't need to be as fast, which used java serialization as storage. On top of that, we made a proxy subclass of IModel which, when called, called the corresponding method on both of the two subclasses. Before returning, the method will compare the result of both of the methods, and make sure they were the same. The proxy class would look something like this:

```
public class DoubleModel implements InvocationHandler {
        IModel mysqlModel;
        IModel simpleModel;
        ...
        public Object invoke(Object p, Method method, Object[] args)
            throws Throwable {
```

[6]The model also later handled caching of the claims' statuses and heartbeat results, though those could theoretically be moved down into the IDatabase in the future.

```
                Object resultFromMySQLModel = method.invoke(mysqlModel,
                    args);
                Object resultFromSimpleModel =
                    method.invoke(simpleModel, args);
                assert
                    resultFromMySQLModel.equals(resultFromSimpleModel);
                return resultFromMySQLModel;
        }
}
```

The theory motivating this unusual practice was that if we had two models that used wildly different approaches (MySQL versus Java serialization) to solving the same problem, then they would be implemented very differently. Since they would be implemented differently, if there were any bugs in one, the chance would be quite low that the other one would have the same exact bug, which means that the assertion will be tripped. The whole idea was that the two approaches act as "oracles" to each other. This way, we could drastically reduce the chances of any bug slipping past us. There was of course a large cost associated with this approach: we doubled the implementation time. We decided that the cost seemed worth the benefit.[7]

### I.6.4 Server

The next arena of decisions to be made was in vast expanse between the model and the browser: the server.

The server did various things such as session management, security, wrapping database calls in transactions, scheduling the subscriptions checker, and other things that are not part of the model.

As for the servlet container, we chose Tomcat over other solutions like Jetty, JBoss, etc. because that was the container we had the most experience with.

[7] We later found that the added robustness was not worth the extra implementation time; we had so many other measures enforcing quality, that very few bugs even made it to the point where this double-implementation would catch them, see section I.29

### Division between Server and Servlets

Tomcat required that the application be divided into sections called "servlets." The servlets received input as HttpRequests and returned output in the form of HttpResponses. We had one for each type of entity, in a very RESTful fashion. For example, some of the servlets were:

- /claims
- /users
- /assumptions

We kept the logic inside the servlets to a minimum. Their main job was to take HttpRequests, and call code inside our real server, the IServer. Then, once it had the results from the IServer, it would wrap the results into HttpResponse objects.

The IServer housed the real server logic. The reason we had the logic inside IServer instead of the servlets was because servlets are not easily called by test code; all tests would have to wrap their calls in HttpRequest objects, and read the resutls out of HttpResponse objects. By making an IServer interface, the tests could just call methods and read results directly. In other words, the IServer was introduced so that server code could be called from the test code, as a shortcut around the servlets; the servlets would simply act as translators between the HTTP world and the Java world, and the IServer subclass would house the actual server code.

The IServer subclass turned out to be a very useful abstraction for another reason: proxying. We were able to use Java's proxy subclass feature to generate an IServer subclass that would automatically transform Java method calls into HTTP requests, block until an HTTP response was received, and then transform those responses back into Java results. This was useful because we could make Java programs that could interact with the live server and the local server from the outside, such as XMLTreePoster and XMLTreeReader, which could save and restore entire discussions.

### I.6.5 Communication

Back when we used GWT, sending data back and forth between the browser and the server was handled automaticall, which was very nice. Since we decided to abandon GWT, we had to consider our options.

#### Embedding the Data in the Page Response

Embedding the data into the page response was a very common approach with many other systems that we had seen. In this approach, when the server responds with the HTML for a page, it also includes the data with it. It looks like this:

```
<html>
<head>...</head>
<body>
   <div id=''pagecontents''>...</div>
   <script>
      var data = <?php echo json_encode($pageData); ?>;
      populatePage(data);
   </script>
</body>
</html>
```

which is evaluated by the server and sent to the browser like this:

```
<html>
<head>...</head>
<body>
   <div id=''pagecontents''>...</div>
   <script>
      var data = {
         claimsTextsByID: {
             100: "Bender is better than Optimus Prime",
             101: "Bender is fueled by liquor, which is way classier.",
             102: "Bender cost less money to build",
```

```
            103: "The above claim is false, we don't know how much
                Optimus costs."
        },
        usersByID: {
            4: "mattschirle",
            5: "evanovadia"
        }
    };
    populatePage(data);
  </script>
</body>
</html>
```

This is a good pattern because it's simple, and only requires one request to the server to retrieve everything the client needs.

However, there is one massive drawback to this approach: if the user wanted to see a more up-to-date version of the data, he would need to refresh the entire page to get it, since the data only comes with each page load.

### Making the page static, and have it AJAX for the data

Another tactic we considered was serving a static page, which didn't come with data embedded in, and have that page programmed to make a secondary request to get the data. The benefit of this approach is that that static page would be cached, which meant future page loads could just use the cached version of the static page, and the browser would only need to get the new data, instead of fetching the HTML again.

A nice benefit from this option was that the page could re-request only the data if it wanted to, and refresh the view's contents on the fly, without reloading the rest; the server is already capable of sending only the data. This was very important for us, because in the near future, we wanted to have the page automatically update itself every few seconds.

The drawback was that it involved two requests the first time someone loads the page. Even though the two requests' total data added up to the same amount as the other solution's one request, we had to wait for two requests, which meant waiting for two round-trip times.

It may seem insignificant that the load time would be longer only on the first page request, but that first page request is incredibly important.

An article by Steve Lohr (2012) explains,

> These days, even 400 milliseconds—literally the blink of an eye—is too long, as Google engineers have discovered. That barely perceptible delay causes people to search less. [6]

We didn't want to ruin our first impression to improve subsequent page load times, especially if the ruined first impression prevented those subsequent page loads.

### Embedding the Data with the Page, AJAXing for real-time updates

Interestingly enough, a third option was to combine the two approaches. If we embedded the data into the page itself for the first request, and also let the browser make data-only requests, then we would have the same initial page load times as solution 1, and the same flexibility as solution 2. The drawback was that it would be more complex; the server would have to the server support two different ways to retrieve the data: one way embedded in the HTML, one way standalone.

In the end, we decided to go with the second solution, with plans to switch to the third solution further on. The transition should be very simple; a few lines on the front-end and perhaps ten lines on the back-end. When the need for speed becomes great enough, we will make the change.

### Web Service: RPC vs REST

Since we decided in the previous section to have a web service instead of embedding the data into the page, the question arose: how should we design the web service?

There are two main styles of web service: Remote Procedure Calling (RPC) and REST (Representational State Transfer). In RPC, the server can be thought of as a massive interface with methods which "perform actions" and may or may not take arguments or return results. In REST, the server can be thought of as "a collection of objects" and all actions are boiled down to "add," "update," "delete," and "retrieve."

We decided on REST, because it's the more decoupled approach[8]; since there is only one way to do anything, it's not coupled to a given use case. One StackOverflow post [34] sums up our considerations nicely:

> The fundamental problem with RPC is coupling. RPC clients become tightly coupled to service implementation in several ways and it becomes very hard to change service implementation without breaking clients:
>
> - Clients are required to know procedure names;
> - Procedure parameters order, types and count matters. It's not that easy to change procedure signatures(number of arguments, order of arguments, argument types etc...) on server side without breaking client implementations;
> - RPC style doesn't expose anything but procedure endpoints + procedure arguments. It's impossible for client to determine what can be done next.
> - etc...
>
> On the other hand in REST style it's very easy to guide clients by including control information in representations(HTTP headers + representation). For example:
>
> - It's possible(and actually mandatory) to embed links annotated with link relation types which convey meanings of these URIs;

[8] It turns out that we were overlooking a very critical drawback of decoupling though, see section I.14.1.

- Client implementations do not need to depend on particular procedure names and arguments, instead clients depend on message formats. This creates possibility to use already implemented libraries for particular media formats(e.g. Atom, HTML, Collection+JSON, HAL etc...)
- It's possible to easily change URIs without breaking clients as far as they only depend on registered(or domain specific) link relations;
- It's possible to embed form like structures in representations giving clients possibility to expose these descriptions as UI capabilities if end user is human;
- Support for caching is additional advantage;
- Standardised status codes;
- etc...

There are much more differences and advantages on the REST side.

### I.6.6 Front-end

#### Library

Since we had dropped GWT, we knew we were going to be using Javascript, so the next question was, "what library should we use on top of Javascript?" Some of us knew YUI and preferred it for its design philosophies and architecture, but jQuery was the only library that everyone on the team knew, so we chose that.

#### Data and Modules

After choosing the library, the remaining question was how to organize the front-end. We decided to have each page be separated into different modules. For example, the front page was divided into these modules:

- header module
- sidebar module, which contained these modules:
  - New claim module
  - fallacy-of-the-day module

    - fallacy-quiz module
- the main section module, which contained these topic title modules

The tree page used these modules:

- header module, same as the one on the front page
- a different sidebar module
- the TreeView module, which contained many nested claimwidget modules.

The decision to use modules was mainly to help with decoupling, encapsulation, and reuse; some decoupling is needed in any large project just to be able to isolate bugs and grasp the data flow, and the encapsulation was to make sure that we didn't use data where we expected not to use it. The reuse was the main reason though. In the future, if we wanted to use claimwidgets in other places, such as the FAQ, making it into its own reusable, decoupled module would be the best option. For the same reason, we made TreeView its own module, so that if we ever wanted a sandbox TreeView for training or FAQ purposes, we could easily put it into a new page, separate from the sidebar module and other dependencies.

The encapsulation was implemented via the strategy pattern; if any module needed access to any information from above, it would have to request it via a callback which was supplied to it by whoever constructed the module. Similarly, if any event happened to a module, and the behavior affects anything outside the module, then the module would call another callback supplied to it.

For example, the claimwidget module has these callbacks supplied to it during creation:

- counterLinkClicked
- reasonForLinkClicked
- reasonAgainstLinkClicked
- editLinkClicked
- addRelevantMark
- deleteRelevantMark
- judgeAssumption
- unjudgeAssumption

- focusOnAssumption
- focusOnClaimWithID
- bodyClickedWhileCollapsed
- subscribe
- unsubscribe
- addressCounterClicked"

Another example is that the treeview module has these callbacks:

- onTreeLoaded
- getData
- makePermalinkForClaimWithID
- addClaimBundle
- editClaimBundle
- addAssumptionJudgment
- addRelevantMark
- deleteRelevantMark
- deleteAssumptionJudgment
- heartbeat
- addSubscription
- deleteSubscription
- addUserLatestClaimView
- suppressClaim

Another rule we imposed was that we would not have any globals. For debugging, we needed to be able to track all the inputs and outputs to every module and function, and every single global represents an implicit extra input and output to every function in the program.

### Dynamic Type-Checking

The last big architecture feature in our front-end was the dynamic type-checking. For example, look at this piece of raw javascript:

```
init: function(viewState, menuManager, helper, claim, delegate) {
        ...
},
```

in this, the arguments are dynamically typed, and we have to hope that the caller has passed in values of the right types. With our new dynamic type-checking system, it would look like this:

```
init: checked(null,
   IViewState,
   bp.or(null, IMenuManager),
   interface(``showHelp'', ``hideHelp'', ``stickifyHelp''),
   bp.instanceOf(GraphClaim),
   bp.or(``split'', ``combined''),
   IDelegate,
   function(viewState, menuManager, helper, claim, mode, delegate) {
      ...
   }),
```

The checked() function is the core of the concept: it takes in a return type, then parameter types, and lastly, the function itself. checked() will generate a new function that effectively works like this:

```
init: function(viewState, menuManager, helper, claim, delegate) {
   assert(IViewState.conforms(viewState));
   assert(menuManager === null || IMenuManager.conforms(menuManager));
   assert(interface(``showHelp'', ``hideHelp'',
       ``stickifyHelp'').conforms(helper));
   assert(claim instanceof GraphClaim);
   assert(mode === ``split'' || mode === ``combined'');
   assert(IDelegate.conforms(delegate);
   ...
},
```

That's right, checked() is a function that generates a function. The source code for checked() is this:

```
function checked() {
       var resultBlueprint = null;
```

```
        assert(arguments[0] !== undefined);
        if (arguments[0] !== null)
                resultBlueprint = bp(arguments[0]);

        var argumentsBlueprints = [];
        for (var i = 1; i < arguments.length - 1; i++) {
                assert(arguments[i] !== undefined);
                argumentsBlueprints.push(bp(arguments[i]));
        }
        var argumentsBlueprint = new BArguments(argumentsBlueprints);

        var func = arguments[arguments.length - 1];

        return function() {
                if (argumentsBlueprint !== null)
                        argumentsBlueprint.assert(arguments);
                var result = func.apply(this, arguments);
                if (resultBlueprint !== null)
                        resultBlueprint.assert(result);
                return result;
        };
}
```

This relies heavily on the function bp() which is an alias for blueprint(), a function that transforms something like this:

```
var myblueprint = bp.or(``herp'', ``derp'');
```

to something similar to this:

```
        var myblueprint = {
                assert: function(value) {
                        assert(this.match(value));
                },
                match: function(value) {
```

```
                return value === ``herp'' || value === ``derp'';
            }
        };
```

In the above code, bp.or() is a blueprint generator. At the time, all of the built in blueprints and blueprint generators were:

- bp.undef: only the value "undefined"
- bp.args: any arguments object (similar to array)
- bp.bool: any boolean
- bp.obj: any object
- bp.number: any integer or float
- bp.int: any integer
- bp.str: any string
- bp.jelly: a jquery handle on an html element
- bp.element: an html element
- bp.array: any array containing any number of any type of elements
- bp.arrayOf(valBP): any array containing any number of the given valBP blueprint
- bp.uuid: any uuid string
- bp.instanceOf(constructor): any object that's an instance of the given constructor
- bp.value(val): any data that === val.
- bp.and(bp1, bp2, bp3, ...): any data that meets all of the given blueprints
- bp.or(bp1, bp2, bp3, ...): any data that meets any one of the given blueprints
- bp.mapTo(valBP): any object for which all values conform to the valBP blueprint
- bp.func: any function
- bp.custom(func): any value for which func(value) returns true

While it did make the initial code more verbose, the system saved us an immeasurable amount of time. Every time one of those assertions was tripped, it represented a chunk of time that we saved. The assertions were tripped a lot; we saved a lot of time. Because of all the saved time, we were able to add more features.

With that, we had made most of the large design decisions, and proceeded to fine-tune our system according to the first experiment's feedback.

### I.7 Users were Lost and Confused

The most important thing we learned from the first experiment was that See the Reason was a very complex platform. Even though the system looked simple to us (since we created it), it looked complex from the outside. The entire system could be broken down into two main features (assumptions and claim status) but when looking at it for the first time, those two features and all the minor features around it seemed bundled up in an enigmatic jumble of mystery and confusion.

On top of that, people were countering posts for the wrong reasons, and not countering posts that should have been countered. We had many posts lacking citations, many posts containing straw-man fallacies, etc. People just didn't know the appropriate ways to interact with the system. Looking at the system again, we found that it really wasn't as intuitive as we'd thought, and nowhere were explanations as to how the system should work.

To address this, we introduced the Guide Mode and the contribution wizards. Guide Mode was the same back then as it is in the latest version; users could hover over things they don't understand and the system would show them an explanation. The contribution wizards were radically different than they are in the latest version though[9].

There were three contribution wizards:

- Counter Objective Wizard (COW)
- Improve Objective Wizard (IOW)
- Improve Subjective Wizard (ISW)

Before the COW, when a user clicked on "counter," a form would appear where they could fill out their counter, which simply asked for the counter's body and any assumptions. Once the COW was introduced, clicking on "counter" would bring up a list of links asking the user what to do, such as:

> Why do you want to counter this claim?

[9]In the latest version, the contribution wizards are known as the Participate Hints and the Address Counter Wizard.

- It's lacking citations.
- It's not directly countering its parent.
- It's not objective; it contains personal beliefs/opinions.
- I just don't agree with it.
- It contains inappropriate language.
- (other)

Some of the links, such as "I just don't agree with it" would bring them to dead-end paragraphs explaining why they shouldn't counter it, such as "Disagreeing with a post isn't enough of a reason to counter it. If there is a specific reason why the post is wrong, please counter it for that reason!" We hoped that this would curb this kind of invalid counter.

Most of the other links would then bring up the form where they could fill out their counter, but it would be pre-filled for them, such as with "Citation needed" or "The above claim does not counter its parent." or "The above claim is committing a fallacy" and it would give them a tip that they should modify the pre-filled text to better suit them.

### I.8 Single Branch vs Multi Branch

Our team came up with two ways to display the debate to the user, and couldn't decide which was best.

The first option was called Multi Branch Tree View (MBTV), which structured the view very much like Reddit; children were displayed indented under their parents. Multibranch tree view is still in the current version, as "Combined View mode."

The second option was called Single Branch Tree View (SBTV), where the page would be a two-column layout, like figure I.2.

There were many differences between the two views:

- In MBTV, you could have the entire tree open at once, which is theoretically pretty good for just browsing and reading. In SBTV, you wouldn't be able

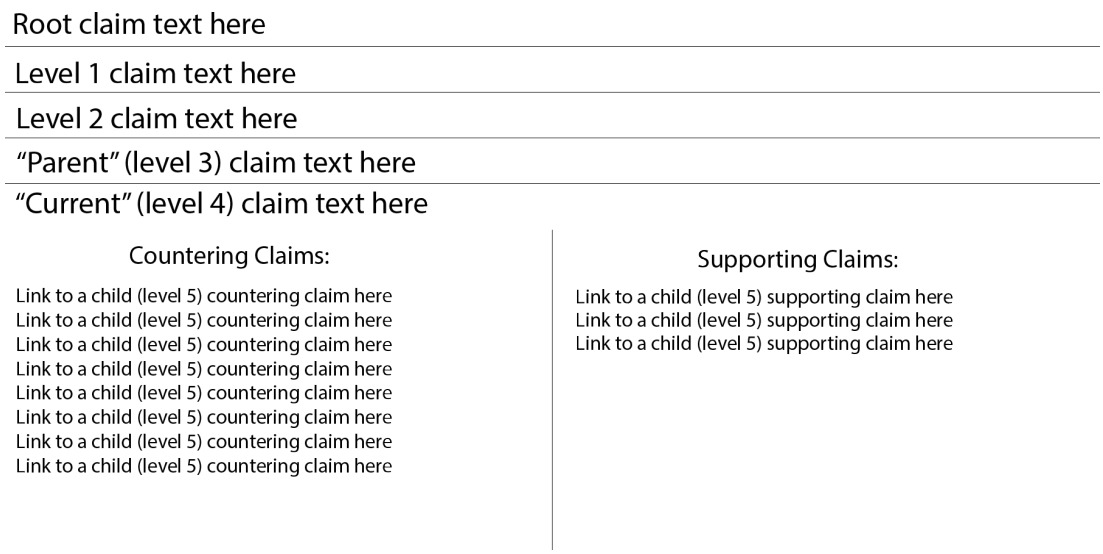


**Figure I.2: Single Branch Tree View layout**

to do that, you'd have to do a lot of clicking to read through the entire thing.

- Since MBTV could display the entire page at once, it could be very useful for printing.
- In MBTV, a child is often right below its parent, but sometimes, a child is very far from its parent, and when someone comes across the child, its hard to know what it's replying to
    - such as when we have a parent claim up here
        - and its first child is here, all good
            - but then the first child has a ton of descendants,
        - and leaving this other child very far from its parent.
- In SBTV, one can get a very good overview of the entire argument. For example, in the evolution debate, MBTV didn't show that only 3 reasons for and 30 reasons against. SBTV made that abundantly clear. SBTV is a much better way to get a feel for who's "winning," therefore enhancing the gamification.
- SBTV can handle arbitrary depths. MBTV, because of its indentation scheme, has its depth limited by the screen's width.
- In SBTV, its easy to tell if a claim is for or against the parent. In MBTV, you have to look to the left at the relation icon.
- MBTV is more intuitive to reddit users, but SBTV is more intuitive to people who haven't experienced reddit, and even some who have.

Of course, this was all made moot when we removed the option to post sub-

jective children (see section I.16), at which point we hybridized the views and made "Split View mode."

## I.9 Choosing a New Name

The name was not always See the Reason. Before it changed, it was actually "The Truth Tree." The name fit nicely because it's a platform used to find the truth, and the claims are in a tree structure. However, truthtree.com and truthtree.net were already taken, and more importantly, there was already a technique in the field of philosophy called the "truth tree" [15] which was used to solve a system of boolean equations, as shown in figure I.3.

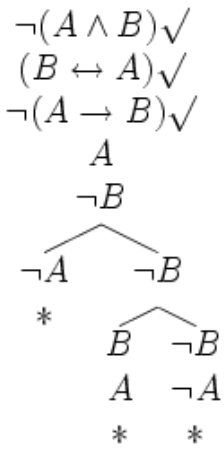


**Figure I.3: A "truth tree"**

We kept the name Truth Tree for a year or so, keeping our eyes out for a better name, and asking friends and family for naming ideas, but none quite felt right. Eventually, we realized that we'd never find a name that felt right, and decided to use a different method for finding a name.

We did one last round of asking people for name ideas, and had a poll. We also put in an extremely bad option, logiconomicon.com. It was so bad that we weren't going to include it in the poll, but we did anyway, to serve as a red flag to find people who mess up or are being silly. If anyone chose that one as the best name, then we would look into it and make sure their results were usable.

The poll asked the users to score each name from 1 to 5, 5 being the best. The results of the poll, in order, were:

- seethereason.com 3.32
- reasonexplorer.com 2.91
- truthsorter.com 2.86
- letstalkreason.com 2.82
- icanlogic.com 2.68
- reasonsorter.com 2.68
- largereasoncollider.com 2.64
- canyoulogic.com 2.59
- icantruth.com 2.59
- truthr.com 2.5
- logicreasontruth.com 2.36
- havetruthwilllogic.com 2.36
- letslogos.com 2.22
- claimcarnage.com 2.14
- offactsandlogic.com 2.14
- canhaslogic.com 2.09
- oureason.com 2.09
- tsorter.com 2.05
- logicapocalypse.com 2
- logiconomicon.com 1.64

There was indeed one person who mixed it up and thought that 1 was the best and 5 was the worst; we found him because he rated logiconomicon.com way higher than the rest, something no sane person would do. We reached out to him, figured this out, and reversed his responses before tallying the final results.

In the end, seethereason.com won, so See the Reason officially became the new name.

### I.10 Ordering Algorithm

The previous experiment showed us that the ordering of the claims was important, so we looked into how other systems ordered their content. The most helpful system in this regard was Reddit.

The details of Reddit's ordering algorithm were secret, but the basic idea is

that posts are ordered by "number of upvotes minus number of downvotes." For example, a post with 1000 upvotes and 200 downvotes would have a score of 800, which would show up before a post with 400 upvotes and 300 downvotes. However, scattered throughout these ranked posts are a few new posts. This is done on purpose so that new posts' scores have a chance to grow. We're not interested in their "upvotes minus downvotes" model, because of the emergent Reddit Hivemind Syndrome (see section H). Instead, we'll measure quality by number of upvotes divided by number of views. That ratio measures "what percentage of people liked it enough to upvote it" which seems to us like a much better approach.

We were particularly interested in how they interleave the good posts with the new posts. To accomplish that, we reserve some of the top space for posts with less than 100 views.

There was something in their algorithm that made posts disappear from the page after a while, no matter how highly ranked they were. This was to ensure that the page changed day to day and never got old. Of course, we couldn't retire old posts the way Reddit did. In fact, we needed to keep all good posts around forever.

One worry we had was that there would be so many good posts and new posts, that other posts, not as good but still valuable, would be buried forever. For this, we reserved an eighth of the top spots that would randomly cycle through claims. This way, every post always had a chance of being seen.

Lastly, we wanted to show more often the posts that were standing with high certainty[10]. We reserved part of the top spots for those, too. In the end, the algorithm could be described thusly:

1. Choose 9 claims like this:
   (a) Choose 2 "tentatively true" claims, weighted to ones with higher certainty
   (b) Of the remaining, choose 3 claims with the best upvotes-to-views ratio.

[10] see section D for more information on certainty.

(c) Of the remaining, choose 1 random claim of all those with under 100 views.

(d) Of the remaining, choose 1 random "no judgment" claim

(e) Of the remaining, choose 1 random "fallen" claim.

(f) Of the remaining, choose 1 random claim.

2. Put those 9 claims in a list and shuffle it.
3. Add them to the page in that shuffled order.
4. Repeat.

In other words, claims were added in groups of nine, and inside the group were 3 claims with a high upvotes-to-views ratio, 2 tentatively true claims, 1 no-judgment claim, 1 tentatively false claim, 1 random claim, and 1 new claim.

The algorithm has served us quite well, but in the future we plan to investigate other algorithms to see if they do better, see section J.5.

### I.11 Experiment B

At this point, we had another experiment. We had 80 students use the site to argue the claim "Macs are better than PCs" and had them fill out a questionnaire. The main takeaways were:

- The site was confusing
- The single-branch tree view was hard to navigate
- Users liked the separate for and against columns
- Users didn't like having to expand to see the claims
- Users didnt understand assumptions
- Users didn't understand objective vs subjective
- Users wanted to support objective claims
- Users were nervous about posting duplicates

#### I.11.1 The Site was Confusing

The most common complaint by far was that the whole site was very confusing:

- I did not understand what I was doing until after I read everything.
- It was easy once you read through everything and figured out how to do it.

- Took me a few minutes to figure it out but reading the text made it very simple.
- There was a lot of reading required to understand what was happening. In other words, the interface was not intuitive.
- It was a bit confusing to counter someone and to confirm your post, but overall I got the hang of it quickly.
- It took me a few minutes to understand how to use the web page.
- It was a little confusing, but easy enough to figure out.
- The application took a bit of thinking and second guessing myself before i was able to figure out how to use it.
- There were times I was a little confused because it was so dependent on carefully reading the material.
- Half of the time, I was trying to figure out and read through all of the instructions.
- I did not know how you wanted us to counter examples already stated.
- I did not really understand what to do at first, but after a few minutes I got the hang of it and it became easy.
- It was confusing at first, but then I figured it out.
- Very confusing at first, but with time, it got easier. Not good structure for quick and easy understanding. Should have the directions and goal more clear.
- I was confused by the symbols at first but I liked that it was explained by rolling your mouse over it.
- I got lost in trying to make anything work!
- I was a bit confuse on how to put up my post at first.
- It took me a little while to figure out how to create a counter claim or claim
- This was a pretty easy userface, but there was a lot of text... almost too much text and so it made things a little confusing.
- There are a lot of different things you can click on. It could be more simplified.
- It was not easy to respond to others' claims
- Maybe a basic overview directions?
- Guide Mode did seem to help, but we definitely have a lot of work to do in simplifying the site. After this feedback, we spent a lot of time working on the site in response to this to try to reduce clutter and make it look more simple.

### I.11.2 Single-Branch Tree View was Hard to Navigate

A very common complaint from users of the SBTV was that it was difficult to navigate. They didn't like the basic idea of clicking on a child post to make it a new layer in the view (see Figure I.2).

They said:

- It was a little difficult to navigate back and forth between a single claim and the main menu of the list of claims that people had made.
- It was not clear how to get back to see the discussion as a whole. I just ended up refreshing the page after each of my posts.
- It was not as easy to navigate back to the topic after posting, as the back button took me back to the home page.
- It was not clear at first if you were responding to a claim or making a new one.
- Wish that [back] would bring me back one step in the process (to primary claims), as opposed to going back to the topics page.
- Once I went to expand an argument, the back option was disabled.
+ and [-] should be changed to different terms that are more intuitive. [+] doesn't just expand the post, it opens the post so [open] might be better. [-] doesn't close the post, it goes back to the previous page. [back] might be a better option.
- I also didn't really like the [+] and [-]...it took me a while to figure it out. Maybe if it said [open] and [back] or something more intuitive like that.
- It would be great if the browser's "back" button or the "return to index" button returned you to the current topic, not to the topic list.
- I would design it so that the expand button either opens a new tab with the argument or opens an expansion within the window which can easily be clicked out of - similar to how clicking on a Facebook picture works.
- I didn't quite understand the method where you had to open up one branch and then respond to that comment. I wish I could see all of the comments vertically, with responses stacked below them (similar to the Reddit UI).
- It would be helpful to have comments nested in single-branch view, as it was a little confusing once a claim had two responses.
- It was kinda difficult to maneuver. Not that it wasn't intuitive, it just took more steps to go back and forth from claim to claim.

Most of these weren't complaints about SBTV's central idea, but about its execution. The plus and minus buttons which collapsed and expanded a claim could have been made much clearer, and if we had animation to transition from

one state to the next, they would more immediately see that clicking on the plus icon was expanding into the new claim. As it stood, the page's response to the click seemed unrelated to the click itself.

Also, adding a collapse-all link would help greatly. Technically, there was one, in the form of the minus link on the top claim, but having another link on top do the same thing would have been very helpful.

Some of these complaints were about SBTV's central idea, but it's possible that once we polish the user interface, the users wouldn't complain anymore. We decided to keep SBTV in, and have it working much better before the next experiment.

### I.11.3 Users Liked the For and Against Columns

There were many complaints from users of MBTV that the reasons for were interleaved with the reasons against:

- I would have rather been able to see the claims in the table lay out that showed the for's and against's
- I didn't really like it - having both pros and cons mixed made it more work for the viewer to see the differences.
- I think the information should maybe be separated between those for and those against.
- I think if you had the fors and againsts seperated into two different columns it would be easier to read and logically understand the arguments.
- When I first got on the page I liked the single-branch view better than the multi-branch.
- The multi-branch view made it a little complicated to grasp at first.
- However, I do understand and appreciate the filtering of comment to For and Against under the main claim.

These users would have liked the SBTV's separate columns.

We tried to make it easier to distinguish reasons for from reasons against by adding icons, and by alternating for-against-for-against, but it didn't seem to be enough.

It seems the users liked the separate columns of SBTV, but also liked the indented child claims of MBTV. We couldn't see a way to combine them at this point though, so we resolved to try to keep both options, and polish them both.[11]

### I.11.4 Users Liked the Helpables

Purely on accident, we didn't include some of the helpables in the SBTV. Because of that, we got three complaints from SBTV users who didn't know what the claim status icons were:

- It would be nice if there was a key or something in the guide bar that says what the thumbs up means. Now that more people are posting I understand that it's a claim that still holds up but as one of the first people that posted, it was a little strange.
- I didn't really understand the thumbs up and red x things. I think I would use something else.
- I wasn't sure what the green thumbs up and the cross were.

One user of SBTV even suggested:

- Or you could do roll over or pop up boxes, so when you roll over a feature then text pops up and tells you what to do.

We received no such complaints from MBTV users, and users even mentioned that it was helpful:

- While the guide for each link was helpful, I felt confused that clicking on something didn't lead me to the next thing, it led me to steps.
- I liked it! I was confused by the symbols at first but I liked that it was explained by rolling your mouse over it.
- The little labels such as [on] the red x made things clear.

### I.11.5 Users Didn't Like Having to Expand to See Claims

Many users said that they didn't like how we cut off the claim texts to fit more on the page, and how they have to click on the claims to see them.

Users of SBTV said:

[11] Though we later figured out a way to combine them, see section I.16.

- Also, "quick view" feature that would expand the claim so you could see the what it says would be nice, otherwise it's very inconvenient to look at a claim and get pack to the overview of claims that have been made about the topic.
- Counters in the claims are a bit difficult to sort through. An expand all option might be useful, as well as the aforementioned nesting.
- When looking at the sides of an argument, you could only see a few words of the argument, leaving a large amount of empty space available. I would make more of the text visible and reduce the empty space.
- It would have been easier to choose claims to read if there was a subject or something simple. Instead, it was just the first 40-50 characters of their post, which didn't necessarily give a feel for their argument.
- There should be more room to display the posts so you can read more before clicking on it.
- Posts were a little difficult to read. It would be nice that if you placed your mouse over the text it would elaborate and show what was cut off.
- The actual view of the single branched view was simple and clean but was lacking information and you could not easily get the idea of the topic without actually clicking on it and reading the entire post.
- I wish the arguments weren't all cut off to fit a single line. I had to click each one to read it.
- Too much clicking to open submenus
- I didn't like that in order to read the full post you had to click on it and go to a seperate browser.
- A (+) button that expands topics on the same screen without opening a new page.
- Don't cut off the arguments to fit one line. Allow users to see all the arguments without having to click anything.
- I thought [the MBTV] was much easier to use than the single branch view because it allowed you to see more discussion threads.

This was also brought up a few times in MBTV:

- It was kind of annoying having to expand everything to read the whole post, but it was fine. I am sure it is necessary to an extent.
- Maybe show the whole response instead of hiding them.
- Having to expand everything to read the posts involves a lot of clicking

Both the MBTV and SBTV cut off collapsed claims to one line, but we got more of these complaints for SBTV because the one line was only half the screen's width instead of the full width. This suggests that the problem isn't just that

we cut off the lines, it's that we cut them off to be too short. We decided to eventually add an option to cut it off after two or three lines, instead of just one.

Some users were also frustrated that they had to click on a claim to see its whole contents. One user suggested that we add "Different tabs for things like expand all,"' which would also help with some of SBTV's other problems, so we decided to add that feature as well.

### I.11.6 The Page was too Compact

In our quest to cut white-space and fit more and more information onto the page, we failed to realize the legitimate need for spacing. Several users said:

- Everything was compacted into the top of the page, it was a little difficult to follow. Maybe if it was spaced out more, it would work better.
- Give the claim a slightly bigger font size and a little more padding, so it isn't squashed in by the logo/site name.
- More spacing throughout would make it more aesthetically pleasing but overall good
- Again, fairly straightforward. I would minimize clutter - make as "clean" looking as possible. But still, an easy application (overall) to use and to post stuff

Also, the site wasn't just complicated–it looked complicated too. Because everything was so compact, there was a lot of information on the screen at once, which was daunting to the users. We think that adding more padding around various elements and making the site look more comfortable will not just improve aesthetics, but will also improve usability and lessen confusion.

### I.11.7 Users Wanted to Support Objective Claims

Many users didn't understand that you can only counter objective claims, and thought they should be able to support:

- to make supporting claims to claims instead of just counters to counters
- the inability to making supporting claims

- I would like to add a spot where you could either counter or agree with others statements.
- It didn't seem that there was an easy way to respond to claims made and to have an ongoing discussion.
- The one thing that confused me was I thought you could add a side note to a statement that you agreed with, but that option was either unavailable or I couldn't find it.

There's nothing we can really do about these responses; supporting objective claims is something we just cannot do (see section I.3). We tried to ease their frustration by putting in the Improve Objective Wizard and the Improve Subjective Wizard to redirect their desire to the proper actions (see section I.7), but it seems that it wasn't enough.

### I.11.8 Users Wanted to Edit and Delete Claims

As expected, users wanted to edit claims, and complained when it wasn't allowed:

- Editing a post was difficult because there was no edit button but instead had to edit as duplicate.
- I didn't like that I wasn't able to edit a post after I posted it.
- Unable to edit posts.
- No delete button
- I posted a comment on the against side when I was for it and couldn't figure out how to delete it. I would enable a delete button for you comments because I could only figure out how to edit it.
- A delete button (for typos)

### I.11.9 Users were Nervous of Posting Duplicates

We didn't expect this at all, but users were quite nervous about posting duplicates:

- It was time consuming to read through every discussion to see if your point had already been made. If it were categorized [by] similar subject heading it might be easier.
- Many of the things I know about the subject were already said
- It was hard to think of creative and new things to address.

- At first it seemed like everything I would want to say was already said and it would just seem redundant to repeat it.
- Coming into the conversation later, it was hard to make points that weren't already made.
- It seemed very accessible, but I had to sift through claims and counter-claims a lot to make sure I wasn't being redundant.

We don't like duplicates, so this is a good thing, but there is something worrying in here. People are even nervous about posting something thats not a duplicate, because it might be a duplicate. The users think they have to read the entire discussion before posting, which is true, but not something we like, because as the discussion gets larger and larger, the users will be more and more demotivated to post.

We had always planned to add automatic duplicate detection, so users can just start typing their post, and the system will tell them if it already exists, but this feedback moved it up on our priority list.

### I.11.10 Topics were too Broad

This was another surprising bit of feedback. "Macs are better than PCs" was a very broad topic on purpose, to accept a broader range of claims that the users could make. We hadn't considered that it could be too broad.

- A lot of the claims were general and it was difficult to add anything to them
- I was very confused on what the discussion wanted. I could post claims but I did not think it was relevant to collaborating with others.

If a user thought of posting "Macs are based on Unix and Unix is better for programming," he might have thought that the claim was too specific, because nobody else in the discussion was a programmer, he would be reluctant to post. However, even specific claims like this should be posted.

To address this, we added a hint to the top that said, "If you know anything (even if the topic seems to broad for it) that might factor into someone's decision, click here!" which when clicked would slide open a little explanation for what can be posted.

### I.11.11 Explanations were Verbose

A lot of users complained that the explanations in the sidebar were too lengthy:

- The directions when trying to post on the discussion were very wordy. The length was overwheleming and made me less interested in posting.
- Lots of wordy descriptions. Not super user friendly.
- The links describing the comments were too long and not summarative enough
- The instructions were easy to follow, but they were also a little bit lengthy. I don't know that everyone participating would read every line of direction given.

The solution to this one was simple: we went through the explanations and got rid of any extra words, phrases, and repetitions to make the explanations shorter.

### I.11.12 Users Didn't Understand "Countered"

Back then, "tentatively false" was called "successfully countered." This confused some users:

- The symbols on the side didn't seem to be helpful at all. A thumbs up was 'still standing'? Also, the x for 'successfully countered' seems like it's overstepping its bounds. How does the program know that that claim was perfectly shut down?
- once one counter argument was produced it said countered... that makes no sense. I can write jellyfish as a counter and then the statement is no longer valid? I don't get that.

To address this, we changed "successfully countered" to "tentatively false" and "still standing" to "tentatively true."

### I.11.13 Users Didn't Understand Assumptions

Only one user said, "the whole claim and assumption thing made it sort of confusing," but more worrying was the fact that not a single claim had an assumption.

Assumptions are the core feature that make the entire platform work for most discussions. If assumptions aren't usable, then the entire system isn't usable.

We weren't sure how to address this; for the time being, we put in a little more explanation into the sidebars about why assumptions are needed.

### I.11.14 The COW was Confusing

The Counter Objective Wizard was a bit confusing to users:

- When I chose to counter a claim, there were quite a few options to choose from, some of which could have been grouped together.
- To post a counter you had to pick one of the questions that you found faulty and as an inexperienced debater I wasn't sure which applied all the time. Plus you couldn't see who was posting what.
- When writing a counter argument, the second interface before you submit the topic was confusing, it made it sound like "are you sure this is correct?"

These were easily addressed by changing the phrasing of various links and prompts.

### I.11.15 Users Didn't Understand Objective vs Subjective

One thing we saw consistently is that users were posting subjective posts as objective, and vice versa. Even though we added a lot of explanation, the users didn't quite grasp the difference. Note that, back then, objective claims were known as "claims" and subjective claims were known as "topics."

- I was also a little confused about the difference between claims and topics. I tried to go with statements I was more sure about.
- Claims and topics were unclear
- When you first begin "posting reasons for" it is confusing about selecting claims or topics and then until you refresh the page it does not become clear that you have posted a reason. I didn't think it was working until I refreshed the page.
- It was easy to figure out for the most part. But I was a bit confused on posting a claim/topic.

This was the most concerning feedback for the platform. The categorization of subjective and objective claims is critical to the system functioning properly. If a user posts a subjective claim as objective, that's alright, because it can be countered as "not objective" and the poster can re-post it as a subjective claim. But if a user posts an objective claim as a subjective claim, there's nothing similar that the system can do to correct it.

Since miscategorizing objective posts as subjective is so much more harmful than miscategorizing subjective posts as objective, we decided to put another hint next to the choice, saying, "if you're not sure, choose objective claim, and it can be corrected later if needed."

### I.11.16 Other Good Suggestions

Some users put in their feedback some good suggestions.

- I would add a way to keep track of my topics and claims.

A list of one's claims would be very useful, because then you could know what you should respond to next.

- Also, auto-generated links would be nice for URLs. At the very least, add support for html

  `<a>`

  tags to allow linking.
- But I really like to organize my text using indents and line spacing. Please include that later on. It makes the text much more readable.

These two users were asking for basic formatting. We addressed this, and added in WikiCreole formatting syntax (see Figure 4.12).

- It would have been nice if it was sorted in some way because I was looking for a post but I was having trouble finding it.
- It was a bit hard to sort through the differences between comments, replys, etc.

These users wanted to be able to find certain posts much more easily. We had already planned to add a search feature eventually, but hearing this from the users bumped it up on the priority list.

Coming out of this experiment, we had a massive to-do list ahead of us, so we dived in. Our next tasks were about performance and testing.

### I.12 Transcripts

At this point, we did an experimental approach to testing. The idea was that when we knew the server was working, we would record all incoming requests and all outgoing responses for a given operation into a log file we called a "transcript." Then later, after we had implemented a bug fix or a new feature, we would run that same operation again, and compare the requests and responses to the ones in the transcript. If they matched, then the server was still working. If they didn't match, then something might be broken.

The approach was good in theory, but the test cases were incredibly tedious to maintain. Every time we added another field to a response, we'd have to go back and remake the transcript for each test case.

This reminded me of why Selenium IDE is not a good tool for making web tests: the resulting test cases are much too brittle; any change to the system would mean a rewrite was needed.

A better approach would be to look for specific fields in the inputs and the outputs and compare those. That way, only when the specific features that affect those inputs or outputs are changed will we need to change the test case. Because of this, we abandoned the transcript approach and went back to only having unit tests.

### I.13 Cherrypicking

One interesting case arose in a climate change debate. One person said "The climate is changing because of this article" and another person said "No, the climate is not changing, see this other article." It was a difficult case, because both seemed right, and it seemed a logical counter; the latter study contradicted the original post's study.

However, this could lead to an infinite chain of valid replies. The original person could counter the counter with a third study, and then the other person could counter that counter with a fourth study, and so on.

We decided that the correct behavior in this case was to counter the first one with a "cherrypicking" counter, basically saying, "The above claim isn't representing the current state of knowledge, and isn't including studies A, B, and C." In other words, they're cherrypicking which evidence to use to suit their own conclusion.

We also added a hint about cherrypicking to the COW so that people could know how to deal with these kinds of claims.

### I.14 Optimizing

The users didn't complain about this, but the site was rather slow once the discussion got to about 100 posts. It took twenty seconds for each page load. Some optimization was sorely needed.

At this point, we did some profiling. We loaded up that same discussion onto a local server, and did a page request, while a profiler was watching the server. We were somewhat surprised to learn that a vast amount of time was spent waiting for requests to the MySQL instance. This is how it had worked:

First, a request comes in for an entire tree of claims, starting at a given root claim's ID. We would request the claim with that ID first. Then, we would request all claims whose parentClaimID matched that ID. Then, for each of those

child claims, we would request all claims whose parentClaimID matched those child claims IDs, and so on. The number of requests was equal to the number of claims, because for each claim, we'd request all children for that claim.

Then, for each claim, we'd request all assumptions belonging to that claim.

Then, for each assumption, we'd request any judgment made by the user who's making the entire page request.

Then, for each claim, we'd read its postingUserID and request the corresponding user. This was potentially a number of requests that was equal to the number of claims, in the case that each claim was posted by a different user.

In the end, we had an average of perhaps 2N MySQL requests per page request, where N is the number of claims. In each MySQL request, only a tiny fraction of the time is actually spent searching the database. The vast majority of the time was spent on latency between the server and the MySQL instance.

Once we had learned all of this, we decided that the obvious solution was to reduce the number of requests as much as possible. Ideally, we'd have three requests: one that returned all of the claims, one that returned all of the assumptions for those claims (with the requesting user's judgments tacked on each), and one that returned all of the users for those claims.

However, this was fundamentally incompatible with the IDatabase pattern introduced in section I.6.3. This caused a lot of soul-searching, as the realization dawned on us that speed was not always compatible with elegance. The IDatabase was very elegant, but the way it simplified requests into just following one relation at a time meant that we would be making a lot of requests to the MySQL instance for any complex operation.

### I.14.1 The Shortcut Method

After much thought, we decided that we would make a special method of IModel and IDatabase called treePage() which would return all data relevant to the tree page. In other words, we made a shortcut around all of the beautiful architecture.

It felt dirty, but it meant that we could speed up the entire operation.

We always knew that speed and elegance were not always 100% compatible, but this predicament really brought to light the gravity of their competition. We spent a lot of time looking for an architecture that was both efficient and flexible, but eventually accepted the compromise.

### I.14.2 The Transitive Closure Table

Up until this point, we had been using recursive requests to retrieve all of the descendants for a given node. This meant we could have as many requests as there were descendants.

The solution was to use a transitive closure table. This meant a possible space usage of $O(n^2)$ but that could possibly be simplified in the future to $O(n)$ if we restricted ourselves to only requesting descendants of root claims. For now, we decided to use $O(n^2)$, because we weren't struggling for space, we were struggling for speed.

The basic idea of a transitive closure table is that for each node, we would maintain a list of all of its descendants. So for something like table I.1, its

| **claimID** | **parentID** |
|---|---|
| A | (null) |
| B | A |
| C | A |
| D | C |
| E | D |

**Table I.1: A table of claim-to-parent relations**

transitive closure table would be table I.2. Keep in mind, every claim is an ancestor of itself and a descendant of itself.

For A, it has a row for each descendant it has: B, C, D, and E. C has two descendants, D and E. D has one descendant, E.

Having a table like this around means we can request all the descendants in one fell swoop, with a query like this:

| claimID | parentID |
|---|---|
| A | A |
| A | B |
| A | C |
| A | D |
| A | E |
| B | B |
| C | C |
| C | D |
| C | E |
| D | D |
| D | E |
| E | E |

**Table I.2: A transitive closure table.**

```
select descendantID
from AncestorsIDsToDescendantsIDs
where ancestorID='A'
```

Then, if we join that with the query for the claims themselves, we can find all of the claims in a given subtree in one query instead of N queries.

Using some clever joining, we could also request all of the assumptions in one fell swoop with a query that looked roughly like this:

```
select *
from AncestorsIDsToDescendantsIDs where ancestorID='A'
join Claim c on c.id = descendantID
join Assumption a on a.claimID = c.id
```

and we could requests all of the users in a similar fashion:

```
select *
from AncestorsIDsToDescendantsIDs where ancestorID='A'
join Claim c on c.id = descendantID
join User u on c.postingUserID = u.id
```

In the end, we sped up a request that took twenty seconds to one that just took 198 milliseconds, an over 100x speedup.

## I.15 Experiment C

At this point, we did an informal experiment among a few friends, to see if the system functioned properly, and to see if the measures we put in place after the last experiment actually had their intended effect. The big claim was "Evolution is true, creationism is false." we decided to debate for the creationism side, and drew claims from a website called "Index of Creationist Claims" which had a massive list of all the things a creationist could say when arguing about evolution and creationism.

We didn't consider this a real experiment with statistically valid results, because we were directly participating. This was a rough experiment to see what would happen when experienced people debated with inexperienced people, and to see if the other features, like subscriptions and refreshing, were received well.

We let loose a storm of claims, and our opponents would skillfully counter them over and over. In the end, almost all the claims for evolution were tentatively true, and all the claims for creationism were tentatively false, except for those with assumptions that "the bible is the true word of god."

### I.15.1 Objective vs Subjective is Still Not Intuitive

Specifically, changing the wording on the objective/subjective choice for a new claim to "select objective if you're not sure" did have a large effect: almost all of the miscategorized claims were objective when they should have been subjective, which is a lot better than the subjective claims being miscategorized as objective. However, the few cases of objective claims being mistakenly posted as subjective were still rather devastating.

### I.15.2 Equal-but-Opposite Subtrees

An interesting phenomenon happened in the experiment.

- (subjective root) Macs are better than PCs.
  - (subjective support) Macs are better for programming.
    - (objective support) Macs are based on Unix which is better for programming.
    - (objective support) Macs come with GCC.
  - (subjective counter) PCs are better for programming.
    - (objective counter) Macs are Unix which is better for programmers.
    - (objective counter) Macs come shipped with GCC.
    - (objective counter) Macs have XCode which is free.

Here we have an almost exact duplication of content, but reversed. This is bad because if a claim is posted in one, it should be posted in the other, which means that that claim is using twice the screen space as other claims. Also, the two subtrees might become out of sync like this one (the last claim there is in the second subtree but not in the first).

It turns out, there is a solution to both this problem and the objective vs subjective problem: removing subjective children.

## I.16 Removing Subjective Children

We eventually figured out that we could completely remove the idea of subjective child claims. For example, if we had the tree:

- (root subjective) Macs are better than PCs
  - (support subjective) Macs are cuter than PCs

Most of the time it could be represented like this:

- (root subjective) Macs are better than PCs

- (support objective) Assuming you think Macs are cuter than PCs / Macs are better than PCs because they're cuter.

of course, this means that you can't post reasons for and against the claim that macs are cuter than PCs directly in this tree, but if you wanted to do that, you could make a separate root claim:

- (root subjective) A: Macs are better than PCs
  - (support objective) B: Assuming you think Macs are cuter than PCs accept reject discuss / Macs are better than PCs because they're cuter.

- (root subjective) C: Macs are cuter than PCs

so if the user clicks on the "discuss" link, they'll be brought to the "Macs are Cuter than PCs" page. The benefit of this approach is that only root claims can be subjective, which removes the choice from the users who are posting child claims.

This also presents an interesting opportunity to solve the problem of equal-but-opposite subtrees. Recall this discussion:

- (subjective root) Macs are better than PCs.
  - (subjective support) Macs are better for programming.
    - (objective support) Macs are based on Unix which is better for programming.
    - (objective support) Macs come with GCC.
  - (subjective counter) PCs are better for programming.
    - (objective counter) Macs are Unix which is better for programmers.
    - (objective counter) Macs come shipped with GCC.
    - (objective counter) Macs have XCode which is free.

This can be represented as this:

- (subjective root) A: Macs are better than PCs.

    - (objective support) B: Assuming macs are better for programming (linked to D) / Macs are better than PCs because they're better for programming.
    - (objective counter) C: Assuming macs are better for programming (linked inversely to D) / PCs are better than Macs because they're better for programming.

- (subjective root) D: Macs are better for programming.
    - (objective support) E: Macs are Unix which is better for programmers.
    - (objective support) E: Macs come shipped with GCC.
    - (objective support) E: Macs have XCode which is free.

In this, we've made two assumptions linked to the same discussion, and one of them is inverted.

This approach has two main benefits. First, there are no more equal-but-opposite or duplicate subtrees. Second, users no longer have to figure out if their reason for / reason against / counter is a subjective or an objective post, because all children are objective. If it would have been a subjective post, it can be represented as an objective post with an assumption.

### I.17 Making SBTV into "Split View Mode"

The only reason SBTV could only show one branch of a tree and not the entire tree is because if it tried to, something like figure I.4 would happen. To display the entire tree, each claim would need to reserve half of its width for its supporting child claims, and the other half for its countering child claims. We would eventually get claims that are prohibitively thin.

Interestingly enough, removing subjective children meant that this could no longer happen. Reasons for and against would only have one type of child (the counter) and therefore only need one column. The page can be represented like how Split View mode is now.

This approach combined MBTV's ability benefit of seeing all the claims at once with SBTV's benefit of having separate columns for reasons for and reasons

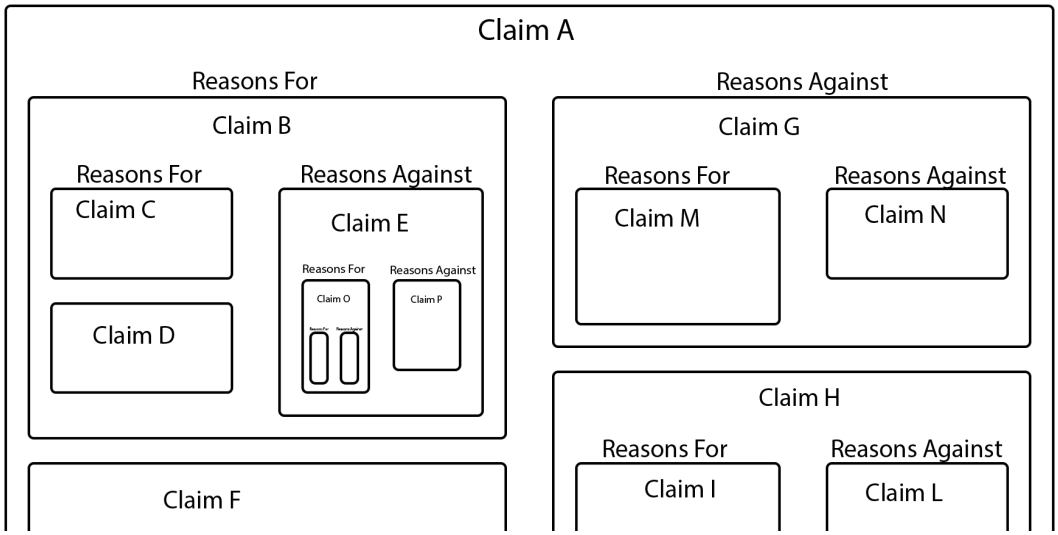


**Figure I.4: A bad layout**

against.

### I.18 Users Have to Keep Refreshing the Page

When testing out the new changes, we often had to refresh the page every few minutes to see if the other person had posted anything. This was extremely inconvenient; it meant that the users had to be motivated enough to refresh the page themselves.

For this, we made the browser page automatically re-request the entire page every five seconds or so. It wouldn't change the actual tree view, but if it noticed that its local copy was different than what it was getting from the server, it would show a small notification on the bottom left saying "Someone has posted to the discussion! Click here to refresh your view."

Of course, asking for the entire tree every five seconds was extremely expensive. To address this, we had to do another bunch of optimizations.

## I.19 Optimizations: Round 2

### I.19.1 Caching Claim Judgments

We always knew it would come to this. We always knew we would have to cache the results of the claim judgment algorithm. In a perfect world, it would be a simple mapping of:

```
claim ID -> judgment
```

The tricky part here is that the tree page is different for different people, depending on their assumption judgments. This meant that instead of a simple mapping, it had to be a mapping of:

```
claim ID + entire subtree's assumption judgments -> judgment
```

To make a hash map out of that, we had to create a class called ClaimIDAndSubtreesAssumptionJudgments with a recursive hashCode() and equals() method, and the resulting hash map was

```
HashMap<ClaimIDAndSubtreesAssumptionJudgments, ClaimJudgment>
```

.

Even more tricky was the fact that when a claim was added to a tree, its parent's cache entry had to be invalidated, and it's parent's, and so on, all the way up to the root. For this, we needed to know all of a given claim's ancestors, which required a call to the database. To save time on this call, we did another bit of caching.

### I.19.2 Caching the Transitive Closure Table

To figure out all of a given claim's ancestors, we needed to query the database, and select from its transitive closure table. Database queries are rather slow (because of the latency involved) so we decided to cache the transitive closure table.

In java, the transitive closure table is represented by two

```
HashMap<UUID, HashSet<UUID>>
```

instances: one mapping from the claim to all of its ancestor IDs, and one mapping from the claim to all of its descendant IDs.

### I.19.3 Tree Hashing and Caching

Another optimization we made was that instead of requesting the entire tree every five seconds[12], we could just request from the server a small hash of the entire tree's claim's IDs. Then, when we get that hash back, we can calculate the same hash on the client. If there was an extra claim on the server that the client didn't know about, then the hashes would be different, which would cause the browser to request a new version of the entire tree.

Calculating this hash of IDs was still expensive, so we decided to cache those as well. Luckily, this was a simple mapping of

```
claim ID -> hash code
```

and it was simple to store.

## I.20 Sanity Checking

With all these extra mappings (the tree hash cache, the transitive closure table cache, and the judgment cache), we had a lot of data everywhere that had to be perfectly in sync. It was a tricky problem; many things had to work perfectly in sync, performing an intricate dance to a very strict rhythm. We've learned in our past projects that when this daunting situation arises, we need sanity checking to ensure that we don't make mistakes.

A sanity check is something we've used often in complex applications, to make sure the model is consistent. For example, in a game, if a puzzle piece has

[12] See section I.18

a reference to the game board, and the game board has a list of references to all of the puzzle pieces, then we would have a simple for-loop of asserts, making sure that, for every puzzle piece, the game board it referenced also had the puzzle piece in its list of references. If a puzzle piece thought it was on the board, we made sure that the boad also thought that it had the puzzle piece on it. These asserts would make sure that the two never fell out of sync, which could easily happen when any modification is made to the model's code. A sanity check is a function that checks the entire model, which can contain dozens if not hundreds of lines of assertions.

These asserts are extremely expensive, of course. However, asserts have a handy property that they can be run in debug builds, and disabled in release builds. When we're developing, we can have these asserts running to make sure we don't make mistakes, and then once we're confident that there are no mistakes, we can turn them off and deploy to the live site. When we start developing again on the local server, we turn them back on. This way, we get all the benefits of sanity checking, with no performance impact for our users.

We introduced a sanity check function to the model, which is run before and after every single request. It would make sure that:

- the transitive closure table in the database was exactly the same as the one in java
- the transitive closure table contained every claim
- if the claim judgment cache didn't contain a judgment for a cache, then it didn't contain a judgment for any of its ancestors

and other miscellaneous assertions. We added the sanity checking before we added the actual feature itself, in a form of test-driven development. Whenever we found any emergent properties of the new features, we put assertions in the sanity check to make sure that those stayed true.

Sanity checking has always served us well in the past, and didn't let us down this time either. This caught quite a few bugs during development, and by our estimates, saved entire days of programming.

### I.21 Subscriptions

Whenever we asked someone to help test the platform, they would help out, but of course would never know to come back unless we told them that we had responded and that they should check it. This is quite unrealistic in the real world, we can't manually tell thousands of people that people have responded and that they should check back. So we did what plenty of other sites did, and introduced subscriptions.

Whenever a claim's status is changed, a subscription email is sent out to all people who are subscribed to that email.

### I.22 Experiment D

This was another informal experiment, where we would argue with a few friends. We had a debate about the free market, a debate about what architecture a certain 3D game should use, and a few other minor debates.

### I.23 "Flying Dutchman" Posts

One thing we noticed is that people didn't correctly react to "citation needed" posts. For example, take this discussion:

- (root) Evolution is true, creationism is false.
    - (reason for) The basic premises of evolution have been scientifically verified, so we know that evolution happens.
    - (counter) Citation needed, I'm skeptical that it's been verified.

an incorrect reaction would be to counter the counter with the missing citation:

- (root) Evolution is true, creationism is false.
    - (reason for) The basic premises of evolution have been scientifically verified, so we know that evolution happens.
        - (counter) Citation needed, I'm skeptical that it's been verified.

- (counter) Here's the citation: http://www.icr.org/article/2605/

We deem this "incorrect" because it leads to an undesirable situation: to see if you can counter a claim, you have to see if it's been "corrected" like this in any of the descendant claims. Only if it hasn't been corrected like this, can you post your counter. Otherwise you'd be posting a duplicate.

The correct response should be to post a new claim:

- (root) Evolution is true, creationism is false.
  - (reason for) The basic premises of evolution have been scientifically verified, see http://www.icr.org/article/2605/.
  - (reason for) The basic premises of evolution have been scientifically verified.
    - (counter) Citation needed, I'm skeptical that it's been verified.

### I.24 Nitpick Counters and Refining is Frustrating

In the debate about the 3D game's architecture, one of us was explaining to the opponent how he would deal with some of the counters we submitted. For example, if we said "citation needed," he should post a new version of his claim, but with citations this time.

We made this happen, and it turned out that there were so many versions of so many claims, that it became quite cumbersome. We always knew that this might happen, but not that it would make things so difficult and unwieldy.

With this, we revisited the decision not to allow editing. The reason we disallowed editing is that we were afraid that a user would completely change the core meaning of his post, thus getting rid of a valuable tentatively false claim, and making all the descendants nonsensical.

We decided that if we introduced a "history" feature, so that users could see what a claim used to say, then we could mitigate that problem and allow editing. It's a bit awkward, but it's worth it if we can keep the system from being flooded with near-duplicate claims. Editing a post to include something it didn't mean

before is against the rules; a user can only edit a claim to strengthen, supply more evidence, and clarify, but not to add additional meanings to the post.

Since we re-introduced editing, it also changes how users should avoid the flying dutchman problem. Remember this discussion:

- (root) Evolution is true, creationism is false.
    - (reason for) The basic premises of evolution have been scientifically verified, so we know that evolution happens.
        - (counter) Citation needed, I'm skeptical that it's been verified.

The correct next step is no longer to introduce a new reason, but to edit the evidence into the original reason, and counter the counter with an "Edited in the citation, thanks":

- (root) Evolution is true, creationism is false.
    - (reason for) The basic premises of evolution have been scientifically verified, see http://www.icr.org/article/2605/.
        - (counter) Citation needed, I'm skeptical that it's been verified.
            - (counter) Edited in the citation, thanks.

### I.25 The Address Counter Wizard

In the beginning, the users had trouble knowing what they could counter a post with, but this was no longer a problem when we added the Counter Objective Wizard. However, now we see that people don't know how to address those counters.

For example, in this discussion,

- (root) Evolution is true, creationism is false.
    - (reason for) The basic premises of evolution have been scientifically verified, so we know that evolution happens.
        - (counter) Citation needed, I'm skeptical that it's been verified.

The person who posted the reason would not know how to address this counter. Some people might intuit that they should counter the counter with citations there. Some people might think that they should flag as inappropriate the "citation needed" post. Some people might correctly guess that they should edit their reason, and counter the counter with an "edited, thanks." There are a lot of ways to react to this, but nothing to show them the way.

However, we couldn't show them what to do, because our system didn't know what type of counter it is.

There are many types of counters:

- Bad logic / fallacy
- Factually incorrect
- Not "true beyond a reasonable doubt"
- Not "true beyond a reasonable doubt" given its assumptions
- Not objective / contains personal belief or opinion
- Unseparated assumption / relies on an unknown fact
- Not making a claim
- Not a reason for the root claim
- Not a reason against the root claim
- Doesn't try to directly counter the parent
- Contradicts its own or an ancestor's assumption
- Defends its own assumption
- Needs citation, evidence, or explanation
- Bad citation
- Cherrypicking evidence
- Other / Not sure

To know which type a given counter is, the system would either have to analyze the text of their post, or have the user specify the type of post when they post it. We chose the latter approach.

In fact, most of the time, we're having them think about the type of post anyway, in the Counter Objective Wizard. The difference here is that instead of just using it to help the user in the browser, we also send that categorization to the server. Each claim is now stored with its categorization, and included in the tree page response. This means that the system can now give users hints on how to address a given counter.

### I.26 Encouraging Assumptions

Another problem we noticed in the evolution vs creationism debate was that we were the only one who ever used assumptions. It seems that using assumptions is still not intuitive enough. However, since assumptions mostly come into play as a reaction to counters, we think that the Address Counter Wizard will indirectly cause users to use assumptions.

### I.27 Complex Subjective Claims

In our debate of "Evolution is true and creationism is false," one person said that that almost none of the reasons for or against were valid. He said that all of the reasons for needed to have two parts to them: one part that gave a reason why evolution is true, and one part that gave a reason why evolution is false. Or, some way to say that evolution being true means creationsim is false, and vice versa. Neither of the reasons for had both of these.

In other words, if you consider "Evolution is true" to be A, and "creationism is false" to be B, and the entire claim as "A & B" he was saying that to show that the claim was true, one had to have both something showing that A is true, and something that shows B is true.

We told him that we can think of a "reason for" as "something that makes the root more likely" and a "reason against" as "something that makes the root less likely." If you think about it in this way, then all of the reasons for and against were valid.

In other words, if you can show that A is more likely, then A & B as a whole is more likely (think in terms of fuzzy logic if it helps).

### I.28 IObjectiveClaim and ISubjectiveClaim

In the beginning, when we designed the IDatabase, we decided to make separate IObjectiveClaim and ISubjectiveClaim interfaces under an IClaim interface,

instead of just having an IClaim with a getType() method.

The idea was that they should be different classes because:

- They are conceptually "different types" of claims
- Objective claims have only a List¡IClaim¿ counterClaims, but subjective claims also have a List¡IClaim¿ supportClaims.

However, maintaining these as three separate classes was extremely cumbersome, and led to a bunch of unnecessary code, and led me to reevaluate the decision.

The second reason was conceptually correct, but nowhere in the actual code did we treat the two differently. There was only one area, at the very top of the server, which checked that we weren't posting any supporting children to objective claims, and that wasn't worth the massive headache of maintaining two entirely separate claim classes.

The first reason, we eventually concluded, was a bit of object-oriented dogma. The dogma could be useful, because usually, different types do things differently, which means that they have different definitions for the same function (basic object-oriented polymorphism). However, we never had any functions inside IClaim to begin with, besides getters and setters. We still had polymorphism all over the place, but outside of the IClaim, in a way more similar to functional programming. For example, the sanity check will act one way when checking an objective claim, and another way when checking a subjective claim. But instead of making it into an abstract method in IClaim and putting those different behaviors into the subclasses, we simply used an if statement to check the type there.

With these reasons no longer valid, we removed IObjectiveClaim and ISubjectiveClaim, and just used IClaim with a getType() method. It made the codebase cleaner, and has caused no harmful side effects since then.

### I.29 SimpleModel

In section I.6.3, we decided to implement two versions of the model: SimpleModel and MySQLModel. The goal was to improve quality, and find bugs earlier. We were finding a few bugs with this approach, but it just wasn't worth the massive amount of time spent maintaining it. We think there were a few things that took away from its usefulness:

- The sanity check caught a ton of bugs that would have been caught from this approach
- The model just wasn't that complicated, so not many bugs could have occurred.

If we didn't have the sanity check, or if the model was a little more complicated, then the double-implementation strategy could have yielded more benefits, but it didn't. With this realization, we let the SimpleModel become out of date and eventually deleted it.

### I.30 Parallel Discussions

A big problem in one of the discussions was that we had to repeat some things in different claims. For example:

- (root) We should make our game turn-based on the inside.
  - (reason for) Turn-based games have faster development because there are less bugs because the internal state easier to keep in sync because XYZ.
  - (reason for) Turn-based games lead to higher quality code because they make the internal state easier to keep in sync (see my other post for why its easier to keep in sync)

This becomes a very difficult discussion, because if someone counters the first reason:

- (root) We should make our game turn-based on the inside.

    - (reason for) Turn-based games have faster development because there are less bugs because the internal state easier to keep in sync because XYZ.
        - XYZ was proven false by the Kearns 96 study.
    - (reason for) Turn-based games lead to higher quality code because they make the internal state easier to keep in sync (see my other post for why its easier to keep in sync)

then they have to add a counter to the second reason:

- (root) We should make our game turn-based on the inside.
    - (reason for) Turn-based games have faster development because there are less bugs because the internal state easier to keep in sync because XYZ.
        - XYZ was proven false by the Kearns 96 study.
    - (reason for) Turn-based games lead to higher quality code because they make the internal state easier to keep in sync (see my other post for why its easier to keep in sync)
        - The other post has been countered.

and if someone counters the counter,

- (root) We should make our game turn-based on the inside.
    - (reason for) Turn-based games have faster development because there are less bugs because the internal state easier to keep in sync because XYZ.
        - XYZ was proven false by the Kearns 96 study.
            - That's false, the Kearns 96 study actually disproves ABC, not XYZ.
    - (reason for) Turn-based games lead to higher quality code because they make the internal state easier to keep in sync (see my other post for why its easier to keep in sync)
        - The other post has been countered.

then the second reason's counter is out of date, and someone has to counter that:

- (root) We should make our game turn-based on the inside.

- (reason for) Turn-based games have faster development because there are less bugs because the internal state easier to keep in sync because XYZ.
    - XYZ was proven false by the Kearns 96 study.
        - That's false, the Kearns 96 study actually disproves ABC, not XYZ.
- (reason for) Turn-based games lead to higher quality code because they make the internal state easier to keep in sync (see my other post for why its easier to keep in sync)
    - The other post has been countered.
        - I countered the other post's counter.

and it goes on forever. The solution until now was to not allow claims to reference other claims, but instead they had to copy the text into it, but that led to trees like this:

- (root) We should make our game turn-based on the inside.
    - (reason for) Turn-based games have faster development because there are less bugs because the internal state easier to keep in sync because XYZ. XYZ was proven false by the Kearns 96 study.
        - That's false, the Kearns 96 study actually disproves ABC, not XYZ.
    - (reason for) Turn-based games lead to higher quality code because they make the internal state easier to keep in sync because XYZ.
        - XYZ was proven false by the Kearns 96 study.
            - That's false, the Kearns 96 study actually disproves ABC, not XYZ.

We plan to soon implement a feature that lets claims cite other claims. It would look like this:

- (root) A: We should make our game turn-based on the inside.
    - (reason for) B: Turn-based games have faster development because there are less bugs because the internal state easier to keep in sync because XYZ.
        - D: XYZ was proven false by the Kearns 96 study.
            - E: That's false, the Kearns 96 study actually disproves ABC, not XYZ.

- (reason for) C: Turn-based games lead to higher quality code because they make the internal state easier to keep in sync (cites B)

The (cites B) denotes that claim C has an internal reference to claim B, so that if claim B is countered, it makes claim C countered.

There is one problem here though: if claim D was countering the part of claim B that was not the part that claim C was citing, then claim B's tentatively false status would mistakenly carry over to claim C:

- (root) A: We should make our game turn-based on the inside.
  - (reason for) B: Turn-based games have faster development because there are less bugs because the internal state is easier to keep in sync because XYZ.
    - D: The internal state being easier to keep in sync may be true, but it doesn't lead to less bugs.
  - (reason for) C: Turn-based games lead to higher quality code because they make the internal state easier to keep in sync (cites B)

In this case, users will have to factor out the "internal state is easier to keep in sync because XYZ." into a separate claim. We will have to make this clear to the users.

This citation feature will lead to some interesting challenges. In the claim judgment algorithm, a claim's status before only relied on its descendants. Now, it will have to rely on its descendants, its citations, and its descendants' citations, and its citations' descendants, and so on. This is tricky, but it will be possible.

# APPENDIX J

# FUTURE FEATURES

This section contains various ideas we hope to implement into See the Reason.[1]

## J.1 Karma

Sites like Reddit and StackOverflow reward their users' participation with points called "karma" or "reputation." Karma is displayed next to users' names, so it's easy to tell who has contributed to the site a lot, and who hasn't. Karma does not have to be used for anything else, it's just a number that users can grow and be proud of. It even sometimes motivates users to contribute more, and compete with other users to see who can contribute the most.

See the Reason could benefit from something like this. After all, as more and more posts get added to a tree, it becomes closer and closer to the truth, in theory[2].

In StackOverflow, users can even offer points to users who do things for them, such as helping with a question. In this way, it's similar to a currency, which motivates users to contribute in that way.

Karma could be used in a similar way to See the Reason to motivate people to make tentatively false a certain claim. If a user says "whoever can successfully counter this claim gets 50 of my karma," and nobody is able to counter it, it could increase a claim's certainty score.

## J.2 Moderation

In a perfect world, users would respect other users and present only what they believed to be valid arguments. Unfortunately, in this world, there are people who get frustrated and start misusing the web site. For example, this has happened:

[1] Not to be confused with section 6.1, which describes various ideas to make the experiments better, not the platform in general.

[2] see section F

- Person A: Eevee knocked over the orange juice.
- Person B: No, there's no proof that he did.
- Person A: Are you calling me a liar?
- Person B: The above is not a claim.
- Person A: WTF are you talking about
- Person B: The above is not a counter
- Person A: Stop saying that.

In this case, Person B has acted correctly, but Person A can just keep on posting things, not knowing how the system works, and Person B can do nothing to stop them. In this case, some outside help is needed, such as from moderators.

Moderators are people who know the system very well, and who are chosen because they've proven that they can be fair. They have the power to keep certain people from posting to certain topics.

### J.2.1 How do the moderators find spam and inappropriate posts?

A common problem with moderation is that the moderators must actively search for spam, and searching for spam is a large time investment, and some can get through. A better idea is to use the users themselves to bring certain posts to the moderators' attention. Reddit does this in the form of "reporting." If a user sees a post that he thinks is inappropriate, he hits the "report" button, and it bring it to a moderator's attention.

To prevent the moderators being flooded with requests, we'll only bring a given claim to a moderators attention once there are a certain number of reports on a given claim. The number can be a percentage of viewers, or some other metric.

### J.2.2 How do you keep users from abusing the reporting feature?

Someone might think they are gaming the system by consistently reporting every claim that they disagree with. If the system notices that a user keeps reporting things, and the moderators that investigate the reportings end up not taking

action, then it will conclude that the user is not a reliable reporter, and start devaluing the reports, perhaps by making their report worth only half of another user's reports, or lower.

### J.2.3 How do you keep the moderators from abusing their power?

There are two methods to address this potential problem: choosing fair moderators that won't abuse their power, and making them accountable for when they do.

We try to choose moderators that are fair by looking at their bias measure (see section G).

In theory, users with low bias scores would be much better moderators. If someone has low bias scores for many of the discussions they participate in, then we will ask them to be a moderator.

This method may have some false negatives; it may reject some users with bad bias measures even if they might have made good moderators, but it should have very few false positives, and that's the important thing. Using this method, we will much better be able to choose fair moderators.

Second, if a bad candidate gets through, or a good moderator turns bad, we'd have an appeal process. Whenever someone wants to appeal a moderator's decision, they can hit an "appeal" link in the claim's menu. They make their case and the moderator and the user can discuss it. Once the user is satisfied that he has made his case, he can trigger a vote between four other random moderators. If at least two of the four random moderators agree with him, then the moderation is undone.

Any moderators that are consistently ruled against in these votes will be put on notice and possibly have their moderation abilities revoked.

### J.2.4 How do you motivate people to be moderators?

There are many ways we can do it: karma, "flair"[3], money, fame, access to an exclusive chat room, access to interesting statistics about the site, the exclusivity of being a moderator, etc.

## J.3 Marking duplicate

A lot of duplicate posts should be caught before they are posted, because of the duplicate detection in the replier. Of course, a lot of duplicate posts will make it through. For these, we let the users mark which claims they think are duplicates.

A user would click on a "mark duplicate" link in the claim's menu. This causes two things to happen:

The first thing that happens is that it will increment the claim's "possibleDuplicate" counter. When the "possibleDuplicate" counter reaches a certain threshold (perhaps a certain percentage of the number times the claim has been viewed), then whenever any user views the claim, the system will try to guess which claims are the most similar, choose one of them at random, and present the viewer with a question that asks "Is this claim similar to..." and then that claim's text, and a "yes" link. If enough users click on the "yes" link, then the claim will be brought to the attention of a moderator, who can then either put a forward in[4] or merge the claims.

## J.4 Searching

One of the main points of See the Reason is to generate a collection of relevant claims for a given debate. For example, someone who is going into a debate about evolution can open up a laptop, go to the page that has the evolution debate, and hit ctrl+F to find given terms on the page. That works great when the user

[3] special decoration that is displayed next to their username

[4] See section J.10

can find the right page, but when a user doesn't know where the page is, then they will need a way to search the entire site at once.

For this, we can introduce a search page, where a user can type in keywords for what they are looking for, and the site will return to them links to the relevant debates. This feature will be useful and increasingly necessary as the number of debates grows.

### J.5 A Better Ordering Algorithm

The current ordering algorithm aims to have various properties, explained in section 4.3. The algorithm we came up with in section I.10 is somewhat unusual and has a few weaknesses, so we hope to have a modified bandit algorithm with coefficients that self-adjust over time to maintain the desired composition of various types of posts.

### J.6 Sorting options

Right now, the claims are sorted according to our own algorithm (see Section I.10 and Section J.5), but a user might find it more helpful to sort it in another way, depending on their situation.

For example, let's say a user is done arguing, and wants to make his decision. He would prefer to show all of the tentatively true claims before the tentatively false ones.

Another example: let's say a user believes the root claim is true, and he's very knowledgeable and wants to argue for it. His goal is to make tentatively true the reasons for the root claim, and make tentatively false the reasons against the root claim. For him, it would be helpful to have the tentatively false reasons for show up before the tentatively true reasons for, and the tentatively true reasons against show up before the tentatively false reasons against.

## J.7 New Claim Form's Duplicate Detection

While the user is typing in the text for their new claim, the system could analyze it in real-time, and comparing it to all existing claims, and showing the user the five most similar ones. This way, if the post already exists, the user can avoid posting a duplicate.

This technique is used on StackOverflow, a popular website where users can ask other users programming question. When a user is typing their question, the system searches for similar ones, and if there are some, it displays a list of them (see figure J.2).

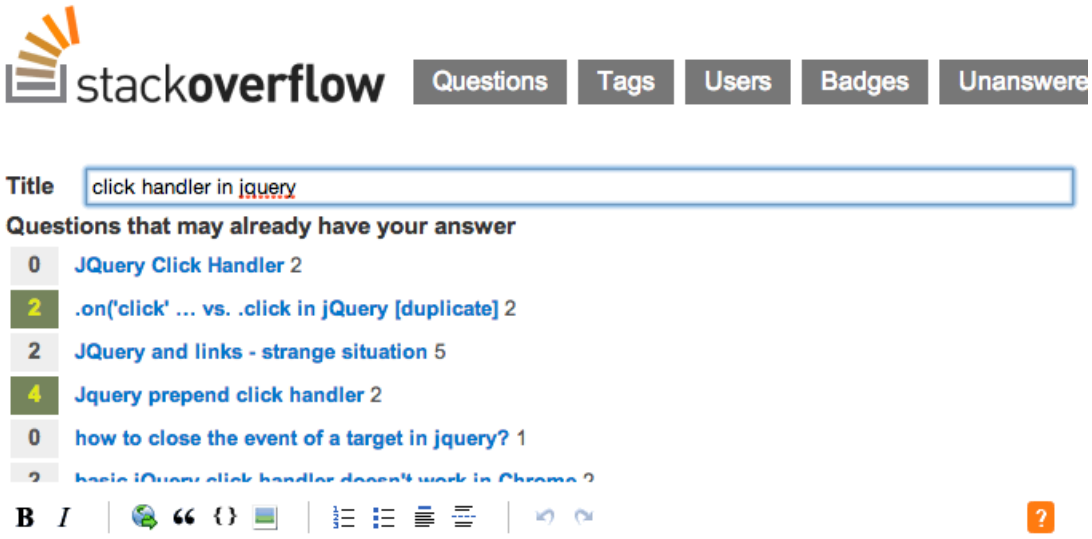


**Figure J.1: Stack Overflow's Duplicate Detection.**

This duplicate detection is critical to the long-term evolution of a given claim, see Section F.

## J.8 A Better Subjective Claim

It's a problem when a user posts as a subjective claim something that is objectively true beyond a reasonable doubt, or objectively false beyond a reasonable doubt. It would be better if it was posted as an objective claim, so that the

platform could do its judgment and inform users of the claim's strengths or shortcomings.

Right now, we can only hope that another user will come along and submit a new version of the claim, this time as an objective claim.

However, there could a way to bring the objective claim's features into subjective claims. Right now, subjective claims have two types of children: reasons for, and reasons again. We could also add two more: proof for and proof against.

If there's a tentatively true "proof for" child, then it makes the parent subjective claim tentatively true. Or, if there's a tentatively true "proof against" child, then it makes the parent subjective claim tentatively-false. If both a proof for and a proof against are tentatively true, then the subjective claim will have a special "indeterminate" status, and ask the user to counter one of the two child claims.

This way, if a user posts as a subjective claim something like "1 = 0," the users could counter it and the system will be able to show it to be tentatively false. Also, if a subjective claim like "Evolution is how it happened" is posted, and a few years later someone finds some amazing new evidence that proves it beyond a reasonable doubt, they can post it. This way, See the Reason can handle some controversial topic becoming provably false or provable true.

### J.9 Incompatible Assumptions

There may be a way to automatically detect incompatible assumptions. For example, the assumption "Assuming god exists," and the assumption "Assuming god doesn't exist," are incompatible. Having the system know this automatically would be very helpful, so it can help users find inconsistencies in their logic.

It might be possible to do this with some statistics. For example, if we find that 99% of users who have accepted the first assumption have also rejected the second assumption, then we might be able to conclude that they are incompatible, or at least suspect. We would at least be able to warn the user that they should

doublecheck their assumptions.

We could also ask the users to flag specific assumptions they see to be incompatible. Perhaps they can even start entire new objective claims that try to prove the assumptions are incompatible.

### J.10 Forwarding

Forwarding is a way to put a hyperlink from one claim to another. It's kind of like a "see also" link. Moderators can put these in if two claims are very similar, but not similar enough to merge. Users can put these in if they've updated one claim to another; they can put a forward from the old one to the new one, with a little caption saying what has changed to make it better.

### J.11 Post Prefixing

Post Prefixing is when we mandate that certain claims start with a specific phrase. For example, if a claim is countering its above parent for not counting its parent claim, the system can require that the claim's body text contains "The above claim does not counter its parent: ..." In this way, we can make sure the claim stays on-topic.

Gao et al. (2013) used post prefixing, he called them "note starters," and it seemed to have some success. [28]

### J.12 Structured Objective Claims

TruthMapping [1] represent claims in very interesting ways. Instead of freeform text, TruthMapping breaks an objective claim into steps.

In TruthMapping, a claim is structured like a mathematical proof; every step in the proof is a section of the post, and all of the premises are explicitly listed. This is nice because users can counter specific parts of a claim, instead of the entire claim.

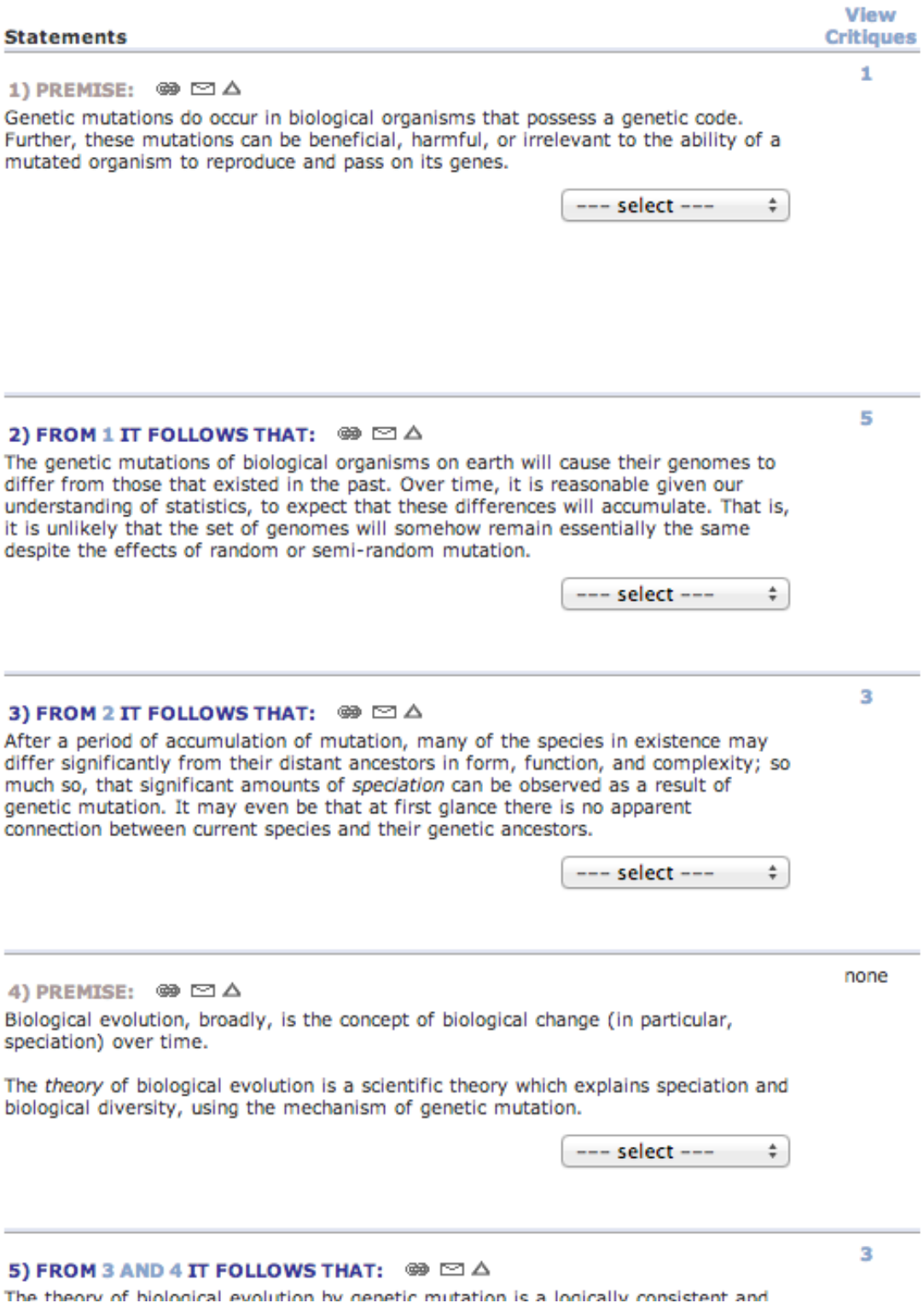


**Figure J.2: TruthMapping's representation of claims.**

Parmenides has the users structure their claims according to predetermined templates. [26] For example, one template could be "(A) is in a position to know about (B). (A) claims (B). Therefore, (B)." and the users could fill in just A and B. Then, it would give opponents options and suggestions on various ways to counter that template. For this template, one can say, "(A) is not actually in a position to know (B) because..." or "(A) does not actually claim (B)."

Cartwright and Atkinson (2009) explain their system:

> Since the last major publication of our progress on the Parmenides system,3 we have implemented significant new functionalities, some of which we have already described. Further new functionalities allow statements within the system to be accompanied by a piece of supporting evidence, provided in the form of a different argumentation scheme. The debate administrator is free to select the most appropriate argumentation scheme from the set currently built into the system. At the time of writing, we have implemented and tested three additional argumentation schemes, with more to follow soon. The currently available schemes for providing evidence, again taken from Walton,1 are as follows:
>
> - *Argument from expert opinion.* This scheme can be used to provide evidence from the written or spoken word of an expert source.
> - *Argument from position to know.* This scheme can be used to provide evidence from the written or spoken word of a source who is in a position to know some fact.
> - *Argument from correlation to cause.* This scheme can be used to state that because of a positive correlation between two events A and B, then A causes B.
>
> For example, in the Smalltown supermarket scenario, the circumstance statement, "Smalltown has high unemployment," might itself be supported with another statement from the town mayor. The town mayor would probably be considered as a source who is in a position to know this fact, rather than an expert in employment rates. Thus, the administrator could use the argument from position to know argumentation scheme, stated as follows:
>
> > *Person A is in a position to know whether Fact F. Person A assert(s) that Fact F. Therefore, Fact F.*

The debate administrator must provide elements Person A and Fact F to instantiate this scheme, and the Debate Creator tool provides prompts for the user to enter the relevant details. Once instantiated, the argument for our example could be as follows:

> *The mayor of Smalltown is in a position to know whether unemployment in Smalltown is high. The mayor of Smalltown assert(s) that unemployment in Smalltown is high. Therefore, unemployment in Smalltown is high.*

This argumentation scheme has its own associated set of characteristic critical questions. One example is, "Is Person A an honest (trustworthy, reliable) source?" By providing a response to such a critical question, we can again determine which parts of the evidence a user agrees or disagrees with, thus helping to identify exact points of disagreement within the debate.

Alternatively, the user could employ the argument from correlation to cause argumentation scheme, stated as follows:

> *There is a positive correlation between A and B. Therefore, A causes B.*

This argumentation scheme could be used to give evidence related to the change in unemployment rate after a supermarket was built elsewhere, for example, as follows:

> *There is a positive correlation between opening new businesses and employment rate rise. Therefore, opening new businesses causes employment rate rise.*

Again, the system poses the critical questions associated with this argumentation scheme to determine which part (if any) of this underlying evidence the user agrees or disagrees with. For example, there might not be enough cases of new businesses opening and causing employment rate rise for this to be seen as strong supporting evidence.

This is a very powerful suggestion system, similar to how the Address Counter Wizard works. This feature could drastically improve debate, so we should try implementing it.