

Argumentation Engine

Abstract Argumentation Semantics and Strategic Argumentation

Sreenitya Thatikunta

October 2026

Code, tests and data: <https://github.com/SreenityaThatikunta/argumentation-engine>

1 Introduction

In an argument, some points attack others. Which points should a fair judge accept? *Abstract argumentation* answers this by modelling a debate as a directed graph: each node is an argument, and an edge $a \rightarrow b$ means “ a attacks b ”. A *semantics* is a rule for picking the sets of arguments that can sensibly be accepted together.

This project has two parts, both following Chapter 5 of *Multiagent Systems* (Rahwan, “Argumentation among Agents”) [1]:

1. **An argumentation engine** that computes the standard semantics (grounded, complete, preferred, stable and semi-stable), the legal in/out/undec labellings, and the acceptance status of each argument.
2. **A strategic analysis.** Several agents each hold private arguments and choose which to reveal to a judge. The judge accepts the grounded extension of everything revealed. We ask *when can an agent gain by hiding arguments?* We test two theorems from the book, first on random instances and then exhaustively on every small graph.

The main findings are that the book’s sufficient condition for strategyproofness (Theorem 5.4) held in every one of over 200 million cases we checked, and that the condition is far from necessary. In most strategyproof cases where it fails, hiding an argument *can* rescue another, but always at an equal or greater cost.

2 Background

An **argumentation framework** is a pair $\langle A, \rightarrow \rangle$ of a finite set of arguments and an attack relation, as introduced by Dung [2]. For a set $S \subseteq A$, let S^+ be the arguments attacked by S , and α^- the attackers of α . The book’s definitions build on each other:

- S is **conflict-free** if no member attacks another ($S \cap S^+ = \emptyset$).
- S **defends** α if it attacks every attacker of α ($\alpha^- \subseteq S^+$).
- The **characteristic function** $F(S)$ is the set of arguments that S defends.
- S is **admissible** if it is conflict-free and $S \subseteq F(S)$, and **complete** if it is conflict-free and $S = F(S)$.

The semantics then pick out particular complete extensions. **Grounded** is the smallest, obtained by applying F to $\emptyset$ until nothing changes. **Preferred** extensions are the maximal ones. **Stable** extensions attack every argument outside them, and may not exist. **Semi-stable** extensions [3] maximise the *range* $S \cup S^+$; they always exist and coincide with the stable ones whenever those exist.

A **labelling** marks each argument *in*, *out* or *undec*. It is legal when an argument is *in* exactly when all its attackers are *out*, and *out* exactly when some attacker is *in*. Legal labellings correspond one-to-one with complete extensions. Finally, an argument is **skeptically accepted** if it is in every extension, **credulously accepted** if it is in at least one, and **rejected** otherwise.

3 The Argumentation Engine

The engine is a small Python package with no runtime dependencies.

Module	Contents
<code>framework.py</code>	framework class: conflict-free, defence, $F(S)$, admissible, complete
<code>semantics.py</code>	grounded, complete, preferred, stable, semi-stable
<code>labelling.py</code> , <code>acceptance.py</code>	legal labellings; skeptical/credulous/rejected status
<code>strategic.py</code>	mechanism, best responses, Theorem 5.3 and 5.4 checks
<code>visualize.py</code>	Graphviz rendering (used for every figure in this report)

Algorithms. The grounded extension is computed exactly as defined, by iterating F from $\emptyset$. To list complete extensions, we first compute the grounded extension G : every complete extension contains G and excludes G^+ . The search therefore only branches on the arguments G leaves undecided, and only along conflict-free sets. Preferred, stable and semi-stable extensions are then filtered from the complete ones, and each labelling is built directly from its extension.

Design decisions. (1) Frameworks are immutable, so they can be shared and cached safely. (2) If a semantics has *no* extensions (possible for stable), the book’s Definition 5.8 read literally would call every argument skeptically accepted but not credulously accepted. We return *rejected* in that case. (3) Asking about an argument that is not in the framework raises an error rather than silently returning a wrong answer.

Validation. The test suite (255 tests, 97% line coverage of the library, run in CI on Python 3.10–3.13) is built from the book. On the book’s Figure 5.1 it checks that $F(\emptyset) = \{\alpha_3, \alpha_5\}$, $F^2(\emptyset) = \{\alpha_1, \alpha_3, \alpha_5\}$, and that $\{\alpha_3, \alpha_5\}$ is admissible but not complete (Examples 5.5–5.6). On Figure 5.2 it checks the three complete extensions $\{\alpha_3\}$, $\{\alpha_1, \alpha_3\}$, $\{\alpha_2, \alpha_3\}$ and the two preferred ones (Examples 5.7–5.8). Beyond the book, complete extensions are compared with a brute-force check of all 2^n subsets on 50 random graphs, and labellings with all 3^n candidate labellings on 30 more. Edge cases include odd cycles, self-attacks, and a graph where semi-stable picks fewer extensions than preferred. Figure 5.3 is the one book example not tested, because its edges appear only in the image.

4 Strategic Argumentation

Setting. This follows the argumentation mechanism design model of Rahwan, Larson and Tohmé [4], as presented in the book. Each agent i privately holds a set of arguments A_i (its *type*). Agents report a subset of A_i : they may hide arguments but not invent them. The **grounded direct mechanism** builds the graph induced by all reported arguments and accepts its grounded extension. An agent’s utility is the number of its own arguments accepted, $u_i = |o \cap A_i|$. The mechanism is **strategyproof** if revealing everything is always a best response, whatever the others report.

Checking Theorem 5.3 (exact test). Theorem 5.3 says the mechanism is strategyproof exactly when, for every agent and every combination of subsets the other agents might reveal, truth-telling maximises utility. We check this by brute force: for each agent, each combination of the others’



(a) All agents truthful: outcome $\{\alpha_1, \alpha_3\}$, $u_x = 1$.

(b) x hides α_1 : outcome $\{\alpha_2, \alpha_4, \alpha_5\}$, $u_x = 2$.

Figure 1: Example 5.10. Green arguments are accepted (in), red are rejected (out); the dashed argument is withheld. Each agent has its own node shape.

reports, and each subset the agent could reveal. Many profiles reveal the same union of arguments, so grounded extensions are cached by that union. An instance with n arguments then needs at most 2^n grounded computations (256 for $n = 8$), which makes thousands of instances cheap.

Checking Theorem 5.4 (graph condition). Write $\alpha \hookrightarrow \beta$ if there is an *odd-length* path from α to β . Along a chain of attacks, odd steps act as attacks and even steps as defences, so $\alpha \hookrightarrow \beta$ means α can work against β . Theorem 5.4 says: if no agent holds α, β with $\alpha \hookrightarrow \beta$, the mechanism is strategyproof. Cycles make “odd-length path” ambiguous, so we implement two readings:

- **Walks** (arguments may repeat). We search over states (argument, parity of path length so far). Each state is visited once, so cycles cannot cause infinite loops. Going round an odd cycle flips the parity, so one pair can be joined both ways.
- **Simple paths** (no repeats). We enumerate paths directly. This is exponential, but fast for graphs of this size.

Every simple path is a walk, so the walk reading is stricter. In the graph $g \rightarrow u$, $u \rightarrow v \rightarrow w \rightarrow u$, $u \rightarrow t$, the only simple path from g to t has length 2, but the walk $g u v w u t$ has length 5, so the readings disagree. Following the theorem’s formal statement, we also count $\alpha = \beta$; a self-attack is the length-1 case, so this enforces the theorem’s requirement that each type be conflict-free.

Examples 5.10 and 5.11. In Example 5.10, agent x holds $\{\alpha_1, \alpha_4, \alpha_5\}$, y holds $\{\alpha_2\}$ and z holds $\{\alpha_3\}$, with attacks $\alpha_1 \rightarrow \alpha_2 \rightarrow \alpha_3$, $\alpha_3 \rightarrow \alpha_4$ and $\alpha_3 \rightarrow \alpha_5$. Revealing everything gives $\{\alpha_1, \alpha_3\}$, so x gets 1. If x hides α_1 , then α_2 defeats α_3 , freeing α_4 and α_5 , and x gets 2 (Figure 1). Our checker finds exactly this deviation, and Theorem 5.4 flags $\alpha_1 \hookrightarrow \alpha_4$ and $\alpha_1 \hookrightarrow \alpha_5$. Example 5.11 adds $\alpha_6 \in A_x$ with $\alpha_6 \rightarrow \alpha_3$. Now x gets all four arguments by telling the truth: the mechanism is strategyproof even though the condition still fails.

Textbook errata. We confirmed both known misprints. (1) In Example 5.10 the utilities of y and z are written with each other’s argument sets; correctly, $u_y = 0$ (holding α_2) and $u_z = 1$ (holding α_3). (2) After Example 5.11, the argument that cancels the indirect self-defeat is α_6 , which attacks α_3 , not α_5 .

5 Experiments

Setup. For each $p \in \{0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5\}$ we generated 1000 random instances (7000 in total): 3 agents, 6–8 arguments, each possible attack $a \rightarrow b$ ($a \neq b$) present independently with probability p , and each argument given to a random agent. For every instance we recorded whether the mechanism is strategyproof (Theorem 5.3) and whether Theorem 5.4’s condition holds under

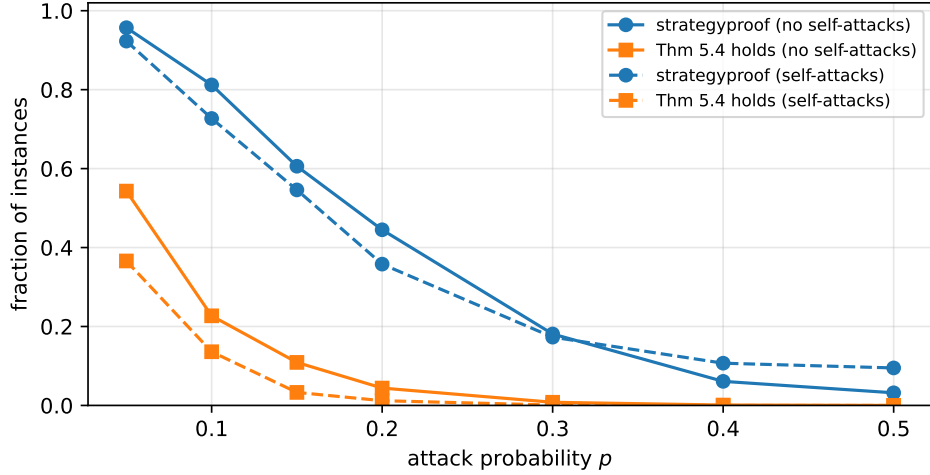


Figure 2: Fraction of instances that are strategyproof (blue) and that satisfy Theorem 5.4’s condition (orange), without (solid) and with (dashed) self-attacks. The gap between the curves is Q3.

both readings. A second run of 7000 instances also allowed self-attacks. Everything is seeded and reproducible.

Q1: How often is the mechanism strategyproof?

p	0.05	0.10	0.15	0.20	0.30	0.40	0.50
Strategyproof	95.7%	81.2%	60.6%	44.5%	18.1%	6.1%	3.2%
Theorem 5.4 condition holds	54.3%	22.7%	10.9%	4.4%	0.8%	0.1%	0.0%
Q2: condition holds, not strategyproof	0	0	0	0	0	0	0
Q3: strategyproof, condition fails	414	585	497	401	173	60	32

Table 1: Results over 1000 instances per p (no self-attacks).

Strategyproofness drops quickly as attacks get denser (Table 1, Figure 2). With more attacks there are more chains linking an agent’s own arguments, and so more chances for one of them to work against another. The condition falls even faster, reaching zero at $p = 0.5$. The walk and simple-path readings gave the same verdict on every random instance. With self-attacks, strategyproofness is lower at small p (72.7% vs 81.2% at $p = 0.1$) but higher at large p (9.5% vs 3.2% at $p = 0.5$). Self-attacking arguments are never accepted, so in dense graphs almost nothing is accepted at all: on average 0.08 arguments per instance at $p = 0.5$, against 0.22 without self-attacks. With so little ever accepted, there is little to gain by hiding.

Q2: Does the condition ever hold without strategyproofness?

No. None of the 14 000 random instances, with or without self-attacks, satisfies the condition yet fails strategyproofness, under either reading. Random sampling can miss rare cases, so we also checked *exhaustively*. Because agent types do not overlap, the other agents together can reveal *any* subset of the arguments the agent does not own. Strategyproofness for one agent therefore depends only on the graph and the agent’s own set, so we checked every directed graph on up to 5 arguments against every possible own set. A fast bitmask version of the engine, cross-checked against the main library in the test suite, makes this take seconds (or about four minutes with self-attacks, which allow 2^{25} graphs at $n = 5$).

This is what the theorem predicts. Intuitively, if none of an agent’s arguments has an odd path to another of its arguments, then each of them can only help (defend) the agent’s other arguments,

Graphs with $n \leq 5$	(graph, own set) pairs	Counterexamples (walks / simple)
No self-attacks	6,312,206	0 / 0
With self-attacks	201,656,372	0 / 0

Table 2: Exhaustive verification of Theorem 5.4.

never harm them. Revealing more can therefore never reduce the agent’s utility. A notable side result: the *simple-path* reading never failed either, even though it accepts more cases than the walk reading (33,549 extra pairs at $n = 5$, all strategyproof). For graphs of this size, the looser condition is still a valid test.

Q3: Strategyproof although the condition fails

Yes, very often: 2162 of the 3094 strategyproof instances (70%) fail the condition. This is expected, since the condition is only sufficient. To see *why* each one is still strategyproof, the experiment script labels every such instance with one of three reasons:

Reason	Count	With self-att.	Meaning
Rescue costs as much	2128 (98%)	1619	Hiding can get a rejected own argument accepted, but every such move loses at least as many own arguments.
Never rescues	31	447	Whatever the others reveal, hiding never turns a rejected own argument into an accepted one.
Self-cycles only	3	315	The only violations are $\alpha \hookrightarrow \alpha$: an odd cycle back to the same argument.

One worked instance of each, taken from the saved results:

- **Rescue costs as much.** Agent x holds $\{\alpha_1, \alpha_2, \alpha_5\}$ and the only attack is $\alpha_5 \rightarrow \alpha_1$, a direct conflict inside x ’s own set. Telling the truth, x gets $\{\alpha_2, \alpha_5\}$. Hiding α_5 rescues α_1 but loses α_5 itself, giving $\{\alpha_1, \alpha_2\}$: still 2. The condition only looks at the graph, but utility *counts* arguments, so a one-for-one swap is never a strict gain. This is the main reason the condition is so conservative.
- **Never rescues.** Agent y holds $\{\alpha_1, \alpha_5\}$, with attacks $\alpha_5 \rightarrow \alpha_6$, $\alpha_6 \rightarrow \alpha_1$, $\alpha_6 \rightarrow \alpha_4$ and $\alpha_4 \rightarrow \alpha_1$. The path $\alpha_5 \alpha_6 \alpha_4 \alpha_1$ has length 3, so $\alpha_5 \hookrightarrow \alpha_1$. But α_5 also *defends* α_1 through the length-2 path $\alpha_5 \alpha_6 \alpha_1$. Whenever the harmful route is active, the helpful one is too, so hiding α_5 never gets α_1 accepted. This is the same effect that α_6 has in Example 5.11.
- **Self-cycles only.** With $\alpha_1 \rightarrow \alpha_2 \rightarrow \alpha_6 \rightarrow \alpha_1$ (plus $\alpha_2 \rightarrow \alpha_1$), each of $\alpha_1, \alpha_2, \alpha_6$ lies on a 3-cycle and so has an odd path to itself. All three belong to different agents, and hiding your own argument from a cycle cannot make that same argument accepted. The violation is harmless. Self-attacks make this case much more common (315 instances).

6 Limitations and Conclusion

The exhaustive check stops at 5 arguments; $n = 6$ has 2^{30} graphs and would need symmetry reduction to be practical. The single-agent reduction relies on agent types not overlapping, as in our experiments; the book allows overlap. Brute-force checking is exponential, which is fine for 6–8 arguments but not for large debates. Figure 5.3 remains untested.

In summary, we built and tested a complete argumentation engine, implemented the grounded direct mechanism with an exact strategyproofness check, and studied Theorem 5.4 both statistically and exhaustively. The theorem held in every case, including under the looser simple-path reading. Its condition is very conservative, however: in most strategyproof cases where it fails, hiding an argument could rescue another but never at a profit, because the condition ignores that the agent is *counting* its accepted arguments.

References

- [1] I. Rahwan. Argumentation among agents. In G. Weiss (ed.), *Multiagent Systems*, 2nd ed., Chapter 5. MIT Press, 2013.
- [2] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.
- [3] M. Caminada. Semi-stable semantics. In *Proc. COMMA 2006*, pages 121–130. IOS Press, 2006.
- [4] I. Rahwan, K. Larson and F. Tohmé. A characterisation of strategy-proofness for grounded argumentation semantics. In *Proc. IJCAI 2009*, pages 251–256, 2009.