

PROJECT REVIEW 3

AuxFlow: Static Analysis for Multi-Pass Document Compilation

1. Project Details

Project Title	AuxFlow: Static Analysis for Multi-Pass Document Compilation
Student Name(s)	Upayan Mazumder
Register Number(s)	24BDS0367
Programme / Branch	B.Tech CSE (Data Science)
Project Guide	Banupriya R
Review Date	

2. Project Overview

AuxFlow is a static analysis tool for LaTeX documents that need more than one compilation pass. It parses the source, builds the Auxiliary Reference Graph (ARG), and classifies the document as CERTIFIED, BENIGN_CYCLIC, AMPLIFYING, or UNKNOWN before the engine runs. For a certified document it predicts the engine run count. A runtime verifier then checks the prediction against the real engine.

Review 2 covered the front end and the IR (Phases 1 to 7). This document covers Phases 8 to 16: graph analysis, classification, build planning, runtime verification, and the evaluation.

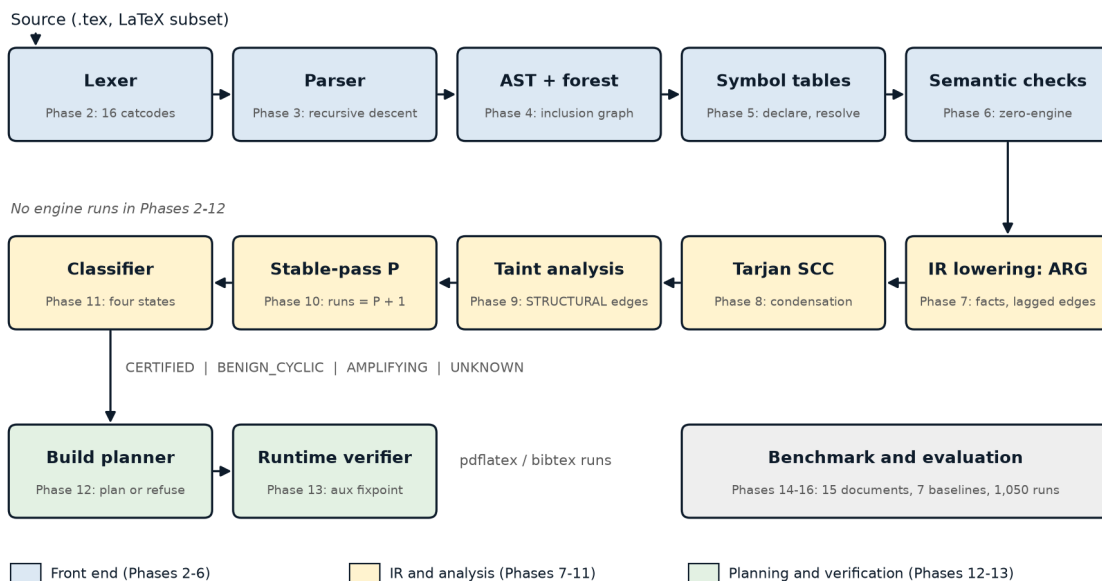


Figure 1. AuxFlow compiler pipeline. Phases 8 to 16 are the subject of this document.

3. Implementation Status

Progress: 16 of 16 implementation phases complete (100% of the implementation plan).

Phase	Module	Function
8	auxflow.analysis.scc	Iterative Tarjan SCC, condensation graph, cycle classes AMPLIFYING / UNKNOWN_CYCLE / BENIGN
9	auxflow.analysis.taint	Seeds from values read in loop bounds and conditions, worklist propagation, STRUCTURAL edges to CONTENT_VOLUME
10	auxflow.analysis.stable_pass	Stable-pass number P per fact on the condensation, predicted runs P + 1
11	auxflow.analysis.classify	Four-state classifier and source-located diagnostics, auxflow classify
12	auxflow.planner	Build plan: reachable files only, bibtex only with citations and a bibliography, derived run count, refusal of AMPLIFYING
13	auxflow.runtime	Normalised auxiliary records, fixpoint check, prediction-miss records, auxflow run
14 to 16	bench/	15-document corpus, 7 baselines, ground truth, 1,050 containerised runs, evaluation report

Build planner safety rule:

Only a CERTIFIED plan uses the derived run count. Every other executable state uses at least three engine runs, and an assertion in the planner enforces this. An AMPLIFYING document gets no plan unless the user passes --force, which runs it under a cap of six runs.

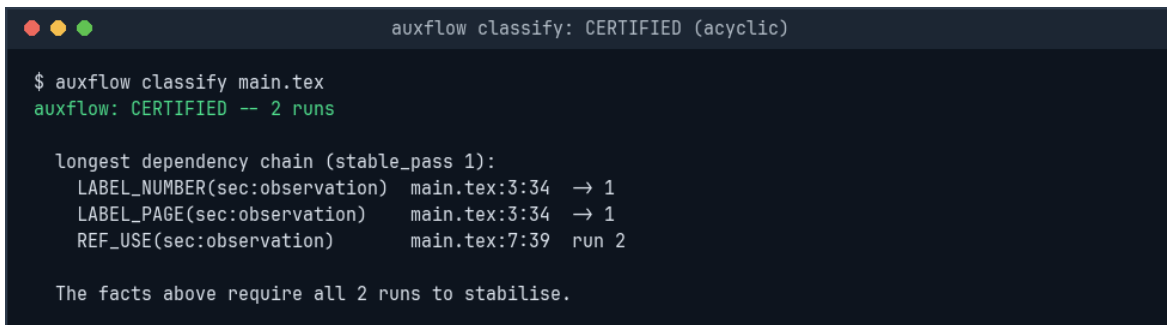
Runtime verification:

The verifier normalises each auxiliary file into an order-independent set of records and compares the state after run k with the state after run k-1. Rerun warnings miss changes in generated lists, so the verifier does not use them. When a certified document needs more runs than predicted, the verifier records a prediction miss.

Code and quality figures:

- About 3,930 lines of implementation code in 41 modules, and about 4,080 lines of test code in 46 files. No third-party runtime dependencies.

- 245 tests pass and 1 is skipped. mypy --strict reports no issues in 41 source files. ruff reports no lint issues in src, tests, and bench.



```

auxflow classify: CERTIFIED (acyclic)

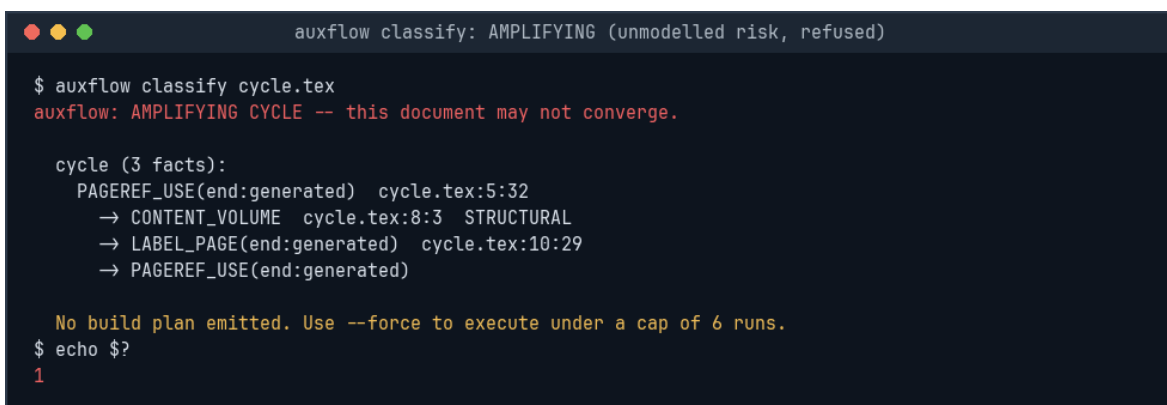
$ auxflow classify main.tex
auxflow: CERTIFIED -- 2 runs

longest dependency chain (stable_pass 1):
  LABEL_NUMBER(sec:observation)  main.tex:3:34  → 1
  LABEL_PAGE(sec:observation)    main.tex:3:34  → 1
  REF_USE(sec:observation)       main.tex:7:39  run 2

The facts above require all 2 runs to stabilise.

```

auxflow classify on a CERTIFIED document



```

auxflow classify: AMPLIFYING (unmodelled risk, refused)

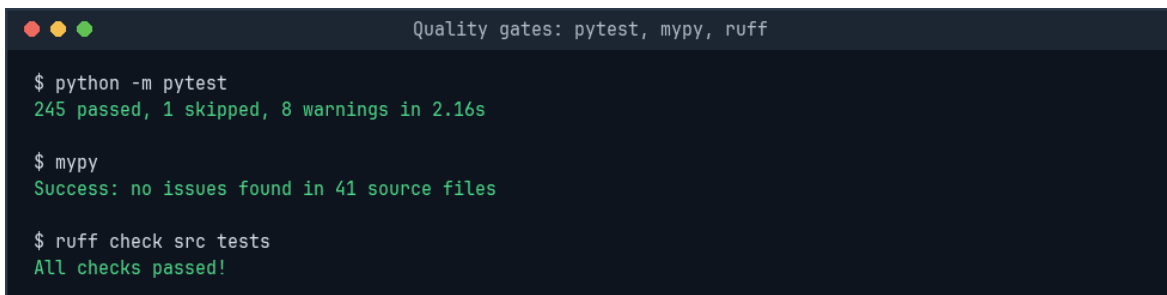
$ auxflow classify cycle.tex
auxflow: AMPLIFYING CYCLE -- this document may not converge.

cycle (3 facts):
  PAGeref_USE(end:generated)  cycle.tex:5:32
    → CONTENT_VOLUME  cycle.tex:8:3  STRUCTURAL
    → LABEL_PAGE(end:generated)  cycle.tex:10:29
    → PAGeref_USE(end:generated)

No build plan emitted. Use --force to execute under a cap of 6 runs.
$ echo $?
1

```

auxflow classify on an AMPLIFYING document: no plan



```

Quality gates: pytest, mypy, ruff

$ python -m pytest
245 passed, 1 skipped, 8 warnings in 2.16s

$ mypy
Success: no issues found in 41 source files

$ ruff check src tests
All checks passed!

```

Test suite, strict type check, and lint

4. Evaluation and Results

Setup:

15 benchmark documents (b01_plain to b15_ibid) cover the non-convergence mechanisms M1 to M9, including a realistic multi-chapter thesis. Seven baselines were measured: A (fixed three runs), B (rerun warnings), C (checksum), C' (inputs), C'' (read/write set), D (AuxFlow static), and E (AuxFlow static plus runtime verification). Each document and baseline ran ten

times in a pinned Docker image with one CPU and no network: 1,050 runs in total. Ground truth N* comes from eight unconditional engine passes, independent of every baseline.

Gate 1, classification separation: PASS

Document	Expected	Observed
b08_cycle	AMPLIFYING	AMPLIFYING
b05_toc	not AMPLIFYING	CERTIFIED
b11_width, b12_float, b13_varioref	CERTIFIED (declared false negative)	CERTIFIED
b07_footnote, b15_ibid	UNKNOWN	UNKNOWN

All 15 documents classified exactly as expected.json predicted before implementation.

Gate 2, run-count bound (EC1): PASS

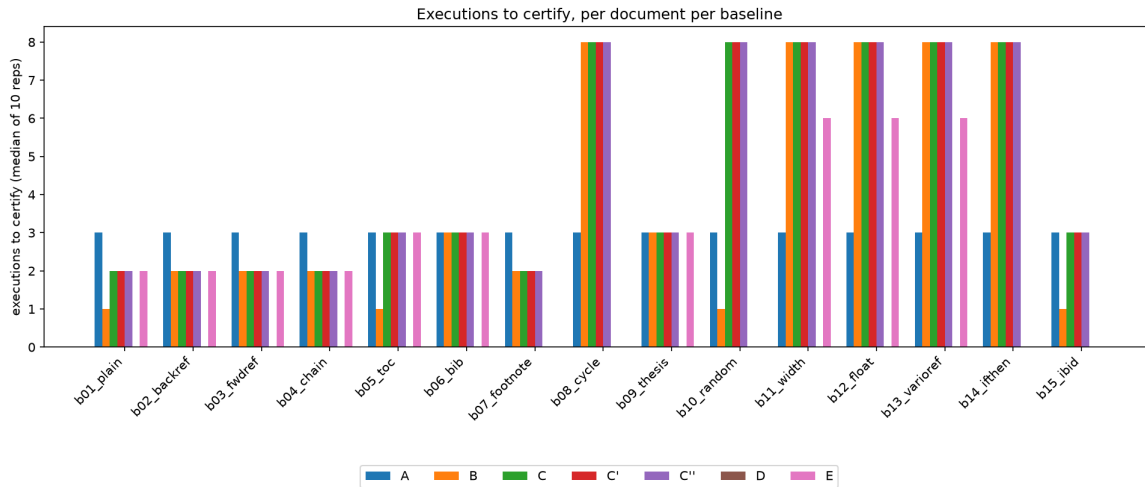
Document	Predicted N	Ground truth N*	N >= N*
b01_plain	1	1	yes
b02_backref	2	2	yes
b03_fwdref	2	2	yes
b04_chain	3	2	yes
b05_toc	3	3	yes
b06_bib	4	3	yes
b09_thesis	4	3	yes
b11_width	2	divergent(8)	no (declared)
b12_float	2	divergent(8)	no (declared)
b13_varioref	2	divergent(8)	no (declared)

The only three violations are the three documents declared as false negatives before implementation. There are no undeclared violations.

Findings:

- Baseline A reports success on all six non-convergent documents (b08, b10, b11 to b14) and is wrong on each of them.
- Baseline B reports CONVERGED after one run on b10_random in all ten repetitions. Ground truth for b10_random is divergent(8).
- On b11_width, b12_float, and b13_varioref, static analysis alone (D) certifies the document wrongly, as declared. Baseline E keeps running past the prediction and reports cap_exceeded at six runs in all 30 repetitions.
- latexmk and latexrun also fail to converge on b11_width, b12_float, and b13_varioref. Tectonic stops b08_cycle at its 6-pass limit, and Typst rejects a port of b08_cycle with 'loop seems to be infinite'.
- Static analysis (D) finishes in under 10 ms on every document. The largest case, b09_thesis with 91 facts and 73 edges, has a median of 9.65 ms.

- b01_plain needs two runs under E although N^* is 1. pdfTeX writes \@abspage@last on every run, so the verifier needs a second run to see two equal states.



Engine executions to certify, per document and baseline

Limitations:

- The corpus has 15 documents, and most are synthetic. All runs used one machine.
- The static model does not see line-break width, float placement, or varioref text feedback. Runtime verification catches these cases.
- Tectonic ran fully offline on only b01_plain and b08_cycle. The latexmk and latexrun cross-check covers all 15 documents.

5. Novelty

- Static classification of multi-pass convergence from source, before any engine run. latexmk, latexrun, and Tectonic decide only after each run.
- The exists/value split in the IR, which keeps ordinary documents acyclic.
- Four states that separate value-only cycles from cycles that change how much content is generated.
- A static prediction paired with a runtime verifier that records prediction misses.

6. Remaining Work and Future Scope

- Write the research paper. The paper/ directory has the section structure.
- Model more packages, so fewer documents classify UNKNOWN.
- Prove a run-count bound for benign cycles.

7. Reproduction

`bash bench/demo.sh` runs the full pipeline on `b08_cycle` and `b11_width` and compares each verdict with the published measurements. Every number in this document comes from `bench/results/published/` and `docs/EVALUATION-REPORT.md`.