

COMPILER DESIGN LAB · REVIEW 2

AuxFlow: Front-End, Semantics, & IR

Lexer Catcodes, Recursive Descent, Multi-File Symbol Tables, and the Auxiliary Reference Graph

Student: Upayan Mazumder (24BDS0367)

Course: Compiler Design Lab

Milestone: Review 2 — Phases 1 to 7 Implementation Complete

From Proposed Architecture to Working Front-End & IR

PHASES 1–4: FRONT-END

- **TeX Catcode Lexer**
Single-pass scanner supporting all 16 TeX category codes with full source fidelity.
- **Recursive Descent Parser**
Hand-written, 1-token lookahead over declared macro signature table.
- **Fault-Tolerant AST**
Missing braces or unterminated groups record diagnostics without crashing.
- **Multi-File Forest**
Resolves \input/\include chains and detects circular inclusions via DFS.

PHASES 5–6: SEMANTICS

- **6 Symbol Tables**
LabelTable, RefTable, CiteTable, PackageTable, FileTable, CounterTable.
- **Two-Pass Resolution**
Declare pass defines all symbols; Resolve pass binds cross-file references.
- **Zero-Engine Check**
auxflow check surfaces undefined refs, duplicate labels, and missing assets in milliseconds.
- **Opacity Boundary**
Unmodelled packages flagged immediately under strict default-deny policy.

PHASE 7: INTERMEDIATE REP.

- **Instance-Level ARG**
Auxiliary Reference Graph nodes are fact instances, not generic types.
- **Typed Lagged Edges**
Edges carry pass-lag (0 or 1) and dependency type (AUX_ROUNDTRIP, TOOL, LAYOUT).
- **Exists / Value Split**
List entries split into structure (exists) vs page (value) to keep standard docs acyclic.
- **Graphviz DOT Export**
auxflow graph serializes the complete IR with visual component coloring.

Lexical Analysis: TeX Category Codes & Scanners

THE TEX CATCODE SCANNER

- 16 Category Codes: Characters are classified by dynamic catcode table (escape '\', group '{ }', comment '%', math '\$').
- Single-Pass Linear Scan: $O(N)$ table-driven state machine without backtracking.
- Comment Semantics: '%' consumes through the newline and all leading whitespace on the following line.
- Delimiter-Driven Verbatim: \verb|...| dynamically reads its delimiter character and bypasses tokenization.
- Precise Span Attribution: Every token retains exact (line, col, offset, length) in the input file.

CLI OUTPUT: auxflow tokens

```
$ auxflow tokens snippet.tex
1:1  CONTROL_SEQUENCE '\section'
1:9  BEGIN_GROUP      '{'
1:10 LETTER          'I'
1:11 LETTER          'n'
1:12 LETTER          't'
1:13 LETTER          'r'
1:14 LETTER          'o'
1:15 END_GROUP       '}'
2:1  CONTROL_SEQUENCE '\label'
2:7  BEGIN_GROUP      '{'
2:8  LETTER          's'
2:9  LETTER          'e'
2:10 LETTER         'c'
2:11 OTHER           ':'
2:12 LETTER          '1'
2:13 END_GROUP       '}'
```

```
Property: round_trip(tokens) == source bytes
```

Recursive Descent Parsing & Multi-File AST Forest

SYNTACTIC RECONSTRUCTION

- Declared Subset: Focuses on structure-bearing macros (`\section`, `\label`, `\pageref`, `\cite`, `\begin{...}`).
- Signature-Driven: Command arguments (mandatory {}, optional []) parsed by signature table.
- Error Recovery: Unterminated groups or missing arguments log a Diagnostic and close gracefully at EOF.
- Inclusion Graph: `\input` and `\include` nodes connect file ASTs; 3-color DFS flags circular inclusion immediately.
- AST Nodes (16): Frozen immutable dataclasses, each with mandatory `Span(file_id, start, end)`.

CLI OUTPUT: `auxflow ast`

```
$ auxflow ast main.tex
=== main.tex ===
Document [main.tex:0-142]
  Preamble [main.tex:0-48]
    MacroCall(\documentclass) [main.tex:0-23]
      Text('article') [main.tex:15-22]
    MacroCall(\usepackage) [main.tex:24-43]
      Text('varioref') [main.tex:36-42]
  Sectioning(section) [main.tex:49-68]
    Text('Introduction') [main.tex:58-67]
  MacroCall(\label) [main.tex:69-85]
    Text('sec:intro') [main.tex:76-84]
  Include(include) [main.tex:86-112]
    Text('chapters/methods')
=== chapters/methods.tex ===
Document [chapters/methods.tex:0-320]
  Sectioning(chapter) ...
```

Symbol Tables & Zero-Engine Semantic Analysis

TWO-PASS CROSS-MODULE RESOLUTION

- Pass 1: Declare: Walks all ASTs in the forest. Populates LabelTable definitions, PackageTable declarations, FileTable reachability, and CounterTable ops.
- Pass 2: Resolve: Binds RefTable use sites to defining labels across file boundaries; matches citations to bibliography entries.
- Multi-File Forward References: Reference in intro.tex to label in appendix.tex resolves cleanly without ordering hazards.
- Diagnostics Engine: auxflow check surfaces errors before running any engine pass: undefined refs, duplicate labels, missing graphic files, inclusion loops.

CLI OUTPUT: auxflow symbols & check

```
$ auxflow symbols main.tex
labels:
  sec:intro -> main.tex:2:1 (counter: section)
  ch:methods -> chapters/methods.tex:1:1
refs:
  sec:intro -> bound to main.tex:2:1 (ref)
  ch:methods -> bound to methods.tex:1:1 (pageref)
cites:
  knuth1984 -> present in refs.bib (1 use)
packages:
  varioref -> modelled (narrow syntax)
  tikz -> unmodelled (OPAQUE)

$ auxflow check broken.tex
error: undefined reference 'fig:missing' (broken.tex:14:22)
error: duplicate label 'sec:intro' (broken.tex:45:1)
[Exit status: 1]
```

The Auxiliary Reference Graph (ARG) Intermediate Representation

IR FORMAL SPECIFICATION

- Instance Facts: ARG vertices are specific fact instances in code, not coarse categories. Nodes carry exact source spans.
- Component Tagging: Every fact is tagged either exists (structural presence) or value (typeset content / number / page).
- Edge Semantics: Edges $E = (u, v, \text{lag}, \text{type}, \text{reason})$.
 - AUX_ROUNDTRIP (lag 1): read from prior .aux pass.
 - TOOL (lag 1): consumed by external tool (BibTeX).
 - LAYOUT (lag 0 or 1): pagination and list extents.
- Acyclic Normal Case: Ordinary TOC documents remain strictly acyclic in the ARG, matching the fixed-point convergence of standard LaTeX.

THE CRITICAL EXISTS / VALUE SPLIT

THE PROBLEM IN NAIVE MODELING:

A Section produces a TOC entry on page P.

The TOC entry shifts document pagination.

Pagination alters page P.

-> Forms a cyclic graph for EVERY document!

AUXFLOW'S ARCHITECTURAL SOLUTION:

Split TOC_ENTRY into two nodes:

1. TOC_ENTRY [exists] (depends only on structure)
 - > feeds LIST_EXTENT
 - > feeds CONTENT_VOLUME
2. TOC_ENTRY [value] (the rendered page number)
 - <-- fed by CONTENT_VOLUME

Result: Edges flow in ONE direction.

Ordinary documents are provably acyclic.

Tracing a Forward Reference: Code to IR Graph

1. SOURCE SNIPPET

```
\section{Intro}  
\label{s:intro}  
...  
See page  
\pageref{s:intro}  
\newpage
```

2. AST & SYMBOLS

- AST Node: MacroCall(\label)
- LabelTable: 's:intro'
 counter: 'section'
- RefTable: 's:intro'
 kind: 'pageref'
- Bound across files: True

3. LOWERED IR FACTS

```
f0: CONTENT_VOLUME  
  [value]  
f1: LABEL_NUMBER(s:intro)  
  [value]  
f2: LABEL_PAGE(s:intro)  
  [value]  
f3: PAGEREf_USE(s:intro)  
  [value]
```

4. ARG EDGES

- f0 -> f2: LAYOUT (lag 0)
 'pagination sets page'
- f2 -> f3: AUX_ROUNDTRIP
 (lag 1)
 'pageref reads label'

Result: 2-run chain
Acyclic DAG (Pass 1 -> 2)

Review 2 Deliverables Complete; Ready for Review 3

REVIEW 2 DELIVERABLES ACHIEVED

- ✓ Phase 1–7 Implementation Complete: Lexer, Parser, AST Forest, Symbol Tables, Semantics, and ARG IR fully functional.
- ✓ Zero-Engine Operation: All front-end analysis runs in Python without requiring TeX Live installed.
- ✓ Robust Test Suite: 100+ unit tests covering token round-trips, error recovery, forward references, and graph lowering.
- ✓ Type Safety: strict mypy enabled across 40+ source modules with zero type errors.

REVIEW 3 ROADMAP (PHASES 8–16)

→ Phase 8–9: SCC & Taint Analysis

Iterative Tarjan algorithm detects cycles; taint analysis identifies content amplification risks.

→ Phase 10–11: Fixpoint & Classification

Compute stable pass P; classify into CERTIFIED, BENIGN_CYCLIC, AMPLIFYING, UNKNOWN.

→ Phase 12–13: Planner & Verifier

Emit safe execution plans with dead-code elimination; verify canonical auxiliary state.

→ Phase 14–16: 15-Member Benchmark Corpus

Execute 1050 containerized runs across 7 baselines; prove Gates 1 and 2.