A FORMALISATION OF CAUSAL MAPPING

Abstract

Draft for an IJSRM submission. This paper proposes a lightweight grammar and logic for encoding, aggregating, and querying causal claims found in qualitative text data.

The specification is grounded in a "Minimalist" (or "Barefoot") approach to causal coding: it prioritises capturing the explicit causal claims made by sources, without imposing complex theoretical frameworks that may not align with how people naturally speak.

1. Data Structures

The foundation of the specification is the **Project Data** package, which strictly separates the causal claims from the source material.

Definition Rule DS-PROJECTDATA: Project Data

- **Definition:** `ProjectData = LinksTable + optional SourcesTable`
- **Note:** `SourcesTable` may be omitted (or empty). If it is present, the evidence constraint in DS-LINKS-SOURCES applies.

Syntax Rule DS-LINKS: The Basic Links Table

A list of causal connections where each row represents one atomic claim.

- **Structure:** A table containing at least the following columns:
- `Cause`: Text label for the driver (influence factor).
- `Effect`: Text label for the outcome (consequence factor).
- `Link_ID`: Unique identifier for bookkeeping.
- `Context`: Optional identifier for distinguishing contexts.
- `Source_ID`: Identifier for the origin of the claim (e.g., document ID, participant ID).
- **Extensions:** The table may contain additional columns (e.g., `Sentiment`, `Time_Period`) or tags.

Syntax Rule DS-SOURCES: The Sources Table (optional)

A registry of documents, interviews, or respondents.

- **Structure:** A table containing:
- `Source_ID`: Unique key.
- `Full_Text`: The complete text of the source document.
- `Metadata`: Optional custom columns representing attributes of the source (e.g., Gender, Region, Date).

Syntax Rule DS-LINKS-SOURCES: Evidence Constraint (if Sources table is present)

- **Additional column:** The Links table also contains:
- `Quote`: The specific text segment evidencing the claim.
- **Constraint:** If a Sources table is provided, then for every link `L` in the Links table:
- `L.Source_ID` MUST match a `Sources.Source_ID`, and
- `L.Quote` MUST be a substring of `Sources.Full_Text` for the matching `Source_ID`.

Coding is strictly evidence-based.

Definition Rule DS-FACTORS: The Factors Derivation

There is no independent table for Factors stored in the Project. Factors are derived entities.

- **Definition:** `Factors = unique values in LinksTable.Cause + LinksTable.Effect`
- **Note:** A "Factors Table" is a transient structure created only during analysis.

2. Coding and Semantics

This section defines how text is translated into the data structures defined above.

Definition Rule COD-DEF: The Definition of Coding

Coding is the process of extracting links from source text into the Links table.

Semantics Rule COD-ATOM: The Atomic Causal Claim

A single row in the Links table, `Link | Cause="A" | Effect="B" | Source_ID="S1"`, is interpreted semantically as:

- *"Source S1 claims that A causally influenced B in virtue of its causal power to do so."*

Semantics Rule COD-BARE: Bare Causation

The relationship `A -> B` implies **influence**, not determination.

- **Negative Definition:** It does NOT imply `A` is the only cause of `B`.
- **Negative Definition:** It does NOT imply `A` is sufficient for `B`.

Example:

- **Source Text:** "Because of the drought, we had to sell our livestock." (from Source 12)
- **Coded Link:**
- Cause: `Drought`
- Effect: `Selling livestock`
- Source_ID: `12`

- Quote: "Because of the drought, we had to sell our livestock"

Semantics Rule COD-MIN: Minimalist Coding Principles

The coding schema prioritizes explicit claims over complex theoretical frameworks.

- **The 90% Rule:** Code the simple form **"X causally influenced Y"**. Explicit coding of complex logic (enablers, sufficiency, non-linearity) is not provided for.
- **Propositions, Not Variables:** Factors are simple propositions (e.g., "Jo started shouting"), not variables with values. Distinct concepts should not be merged prematurely.
- **Example (why this matters):** Code `Poverty` and `Wealth` as distinct factors rather than values of one "Economic Status" variable. A source may claim `Poverty -> Stress` and also `Wealth -> Investment`; collapsing these too early can obscure distinct narratives.
- **No Absences:** Do not code absences. If a source does not mention a factor, it is unknown, not absent.
- **Realism & Partiality:** "X influenced Y" means X had the causal power to affect Y in this context. It implies a contribution, not total determination.

Surface cognition

- **Goal:** Model the speaker's expressed causal thinking (cognition), not necessarily underlying objective reality.
- **Method:** Code only what is explicitly said; avoid inferring hidden variables or unstated counterfactuals.

Semantics Rule COD-MULTI: The meaning of multiple links

A table containing multiple links simply asserts the logical conjunction of the links:

- Source S1 claims that A causally influenced B in virtue of its causal power to do so.
- Source S1 claims that C causally influenced D in virtue of its causal power to do so.
- Source S2 claims that A causally influenced C in virtue of its causal power to do so.

So, unless specific contexts are specified, if Source S1 claims that `A -> C` and also that `B -> C`, this is neither a contradiction nor (by default) a claim that A-and-B jointly influenced C as a package. It is simply two separate claims.

3. The Filter Pipeline (Query Language)

Analysis is performed by passing a links table through a sequence of filters.

Syntax Rule FIL-PIPE: The Semantic Pipeline

The meaning of a result is defined by the cumulative semantic restrictions or transformations of the filters applied.

- **Syntax:** `Input |> Filter1 |> Filter2 |> Output`

- **Multi-line Syntax:**

```
Input
    |> Filter1
    |> Filter2
    |> Output
```

- **Processing:** Filters are applied sequentially. The output of `Filter N` becomes the input of `Filter N+1`.

Types of Filters

- **All filters take a Links table as input.** Most filters return a Links table (so they can be chained). **Output filters terminate the pipeline** by returning a derived view (table/summary) rather than a Links table.

In practice (as in the app), filter behaviour is often **multi-effect**. For example, a filter may rewrite labels *and* add tracking columns. So instead of forcing each filter into exactly one "type", we treat each filter as having an **effect signature**:

- **Row selection**: changes *which links* are included (drops/retains rows).
- **Label rewrite**: changes `Cause`/`Effect` labels (recoding/normalisation).
- **Column enrichment**: adds or recalculates columns (metadata/metrics/flags), typically to support later filtering or display.
- **Configuration**: changes display/formatting settings without changing the links table (app-specific; not formalised here as a core links-table transform).

Below, filters are grouped by their **primary intent**, and each rule declares its **Effects:** line.

Row-selection filters

Syntax Rule FIL-CTX: Context Filters

Reduces the evidentiary base based on Source metadata.

- **Effects:** row selection
- **Operation:** `filter_sources | <criteria...>`
- **Semantics:** Restrict the evidence base to links whose `Source_ID` is in the retained set of sources.

Syntax Rule FIL-FREQUENCY: Content Filters

Reduces the evidentiary base based on signal strength.

- **Effects:** row selection
- **Operation:** `filter_links | <criteria...>`
- **Semantics:** Retain only links meeting an evidence threshold (e.g., `min_source_count=2`).

Syntax Rule FIL-TOPO: Topological Filters

Retains links based on their position in a causal chain.

- **Effects:** row selection
- **Operation:** `trace_paths | from="<factor>" | to="<factor>" | <options...>`
- **Semantics:** "Retaining only mechanisms that connect *From* to *To*."

Label-rewrite filters

Semantics Rule FIL-ZOOM: The Zoom Filter (Hierarchical Syntax)

Extends the logic to handle nested concepts via a separator syntax.

- **Effects:** label rewrite
- **Syntax:** Factors may use the `;` separator (e.g., `General Concept; Specific Concept`).
- **Semantics:** `A; B` implies `B` is an instance or sub-component of `A`.
- **Operation:** `transform_labels | zoom_level=1`. If `zoom_level=1`, rewrite labels by truncating text after the first separator.
- **Inference:** At Zoom Level 1, `A; B` is treated logically as `A`.

Semantics Rule FIL-OPP: The Combine Opposites Filter (Bivalence Syntax)

Extends the logic to handle polarity/negation.

- **Effects:** label rewrite; column enrichment
- **Syntax:** Factors may use the `~` prefix (e.g., `~Employment`) or tag pairs.
- **Semantics:** `~A` is the negation of `A`.
- **Operation:** `combine_opposites`. Rewrites negative labels (e.g., `~Income`) to their positive counterparts (`Income`) and adds tracking columns such as `flipped_cause` and `flipped_effect`.
- **Inference:** Evidence for `~A -> ~B` is treated as corroborating evidence for `A -> B` (with flipped polarity).

Column-enrichment filters

Syntax Rule FIL-BUNDLE: The Bundling Filter

This filter aggregates co-terminal links (links with the same cause and effect) to calculate evidence metrics without reducing the row count. We normally think of it as being automatically applied after any other filter.

- **Effects:** column enrichment
- **Operation:** `bundle_links`
- **Definition (bundle object):** `Bundle(A, B) = all links L where L.Cause <mark> A AND L.Effect </mark> B`
- **Logic:** For every link `L`, identify the set of all links `S` where `S.Cause <mark> L.Cause` AND `S.Effect </mark> L.Effect`.

- **Transformation:** Appends new columns to the Links table:
- `Bundle`: For convenience, a text representation of the connection (e.g., "A -> B").
- `Citation_Count`: The total count of rows in set `S`. Represents volume of coding.
- `Source_Count`: The number of unique `Source_IDs` in set `S`. Represents breadth of evidence (consensus).
- **Lemma:** `Source_Count <= Citation_Count`.

We measure importance using two distinct metrics:

- **Citation Count:** The total number of times a link or factor was mentioned across the entire project. This counts every single row in the data.
- *Technical:* `citation_count`
- **Source Count (or Number of People):** The number of *unique* sources (people or documents) that mentioned a link or factor. This avoids double-counting if one person repeats the same point multiple times.
- *Technical:* `source_count`

Output filters

Output Rule OUT-FACTORS: Factors table view

Returns a Factors table (one row per factor) derived from a Links table (typically after `FIL-BUNDLE`).

- **Operation:** `factors_view`
- **Semantics:** Aggregate over the set of factor labels appearing anywhere in `Cause` or `Effect`, and compute per-factor summaries (e.g., role metrics).

Output Rule OUT-MAP: Graphical map view

Returns a graphical network view of the current Links table.

- **Operation:** `map_view`
- **Semantics (data):**
- **Nodes:** Factors (labels appearing in `Cause` or `Effect`).
- **Edges: Bundles** (one edge per `Cause -> Effect` pair), built from the **current filtered/transformed labels** (so the map reflects Zoom/Combine-Opposites/etc.).
- **Bundling:** If bundle-level columns are not already present, the map view implicitly applies `FIL-BUNDLE` to compute bundle metrics (e.g., `Citation_Count`, `Source_Count`) used for labels and styling.
- **Semantics (presentation):** The view includes a formatting configuration that maps derived metrics to visual encodings (e.g., edge width by citation/source count; edge colour/arrowheads by mean sentiment; node colour by outcomeness; node/label sizes by frequency), plus a legend summarising the current view and applied filters.

Definition Rule MET-NODE: Factor Role Metrics

These metrics describe the topological role of a factor.

- **In-Degree (incoming citations):** Count of incoming links (times this factor appears as an Effect).

- *Technical:* `citation_count_in`
- **Out-Degree (outgoing citations):** Count of outgoing links (times this factor appears as a Cause).
- *Technical:* `citation_count_out`
- **Outcomeness:** `In-Degree / (In-Degree + Out-Degree)`.
- *Interpretation:* A score nearing 1 indicates an Outcome; a score nearing 0 indicates a Driver.

4. Example Queries

Example A: The "Drivers" Query *Question: What do female participants say are the main drivers of Income?*

```
Result = ProjectData
  |> filter_sources | Gender="Female"         // Rule FIL-CTX
  |> trace_paths | to="Income" | steps=1      // Rule FIL-TOPO
  |> filter_links | min_citations=2           // Rule FIL-FREQUENCY
```

Example B: The "Mechanism" Query *Question: Is there valid narrative evidence that Training leads to Better Yields?*

```
Result = ProjectData
  |> transform_labels | zoom_level=1                 // Rule FIL-ZOOM
  |> trace_paths | from="Training" | to="Yield" | thread_tracing=TRUE   // Rule FIL-TOPO + Rule I
```

5 Causal Inference?

Inference Rule INF-EVID: Evidence is not effect size

We quantify **evidence strength**, not **causal effect strength**.

- **Observation:** `Link | Cause="A" | Effect="B"` appears 10 times.
- **Inference:** There are 10 pieces of evidence (10 coded mentions) for the claim `A -> B`.
- **Invalid inference:** The influence of A on B is 10 times stronger than a link appearing once.

Inference Rule INF-FACT: Factual Implication

If we observe `Link | Cause="A" | Effect="B" | Source_ID="S1"`:

- **Deduction:** `Source S1 claims A happened/exists`.
- **Deduction:** `Source S1 claims B happened/exists`.

Inference Rule INF-THREAD: Thread Tracing (Valid Transitivity)

We can infer a long causal chain (indirect influence) only if one source provides every step.

- **Logic:** `Link | Cause="A" | Effect="B" | Source_ID="S1"` AND `Link | Cause="B" | Effect="C" | Source_ID="S1"` => Valid path `A -> B -> C`.

Inference Rule INF-CTX: The Context Rule (The Transitivity Trap)

We cannot infer causal chains by stitching together different sources without checking context.

- **Logic:** `Link | Cause="A" | Effect="B" | Source_ID="S1"` AND `Link | Cause="B" | Effect="C" | Source_ID="S2"` => INVALID path `A -> C`, unless S1 and S2 share the same `Context`.

Appendix A: AI Extensions

These filters extend the core logic using probabilistic AI models (Embeddings or Clustering).

Semantics Rule FIL-SOFT: The Soft Recode Filter

Extends logic using semantic similarity (vector embeddings).

- **Operation:** `soft_recode | magnets="<...>" | similarity_threshold="<...>"`
- **Inference:** If `Label A` is similar to `Magnet M` (> threshold), treat `A` as `M`.

Semantics Rule FIL-AUTO: The Auto Recode Filter

Extends logic using unsupervised clustering.

- **Operation:** `auto_recode | target_clusters=K`
- **Inference:** Factors group together based on inherent semantic proximity into `K` emergent themes.