LLMs Do It All: A Reality Check with Formal Concepts

Valentino Cocks, Arame Diop, Oscar Jimenez Flores, Yazmín Mendoza,
Marianne Huchard, Huaxi Yulin Zhang

"LLMs Do It All"
A Reality Check with Formal Concepts

Valentino Cocks, Arame Diop, Oscar Jimenez Flores, Yazmín Mendoza,
Marianne Huchard, Yulin (Huaxi) Zhang

Univ. Montpellier   CNRS   eproad   Univ. de Picardie Jules Verne

Table of Contents

Introduction
○●○○

Methodology
○○○○○○○○○○○

Results
○○○

Discussion
○○○○○

Perspectives
○○○

Context

Generated by ChatGPT 5, 12/08/2025

Introduction
OOOO

Methodology
OOOOOOOOOOO

Results
OOO

Discussion
OOOOO

Perspectives
OOO

Research Supervised Project (TER)
Master's Program in Skills Integration (MICo) at Montpellier University

LLM expert

FCA support

Introduction
○○○●

Methodology
○○○○○○○○○○○

Results
○○○

Discussion
○○○○○

Perspectives
○○○
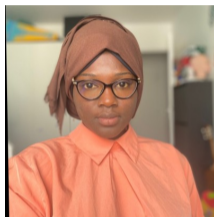
Context

Given a formal context, how effectively LLMs can identify the
associated formal concepts [Ganter and Wille, 1999]

Generated by ChatGPT 5, 13/08/2025

Introduction
OOOO

Methodology
●OOOOOOOOOO

Results
OOO

Discussion
OOOOO

Perspectives
OOO

Table of Contents

Introduction
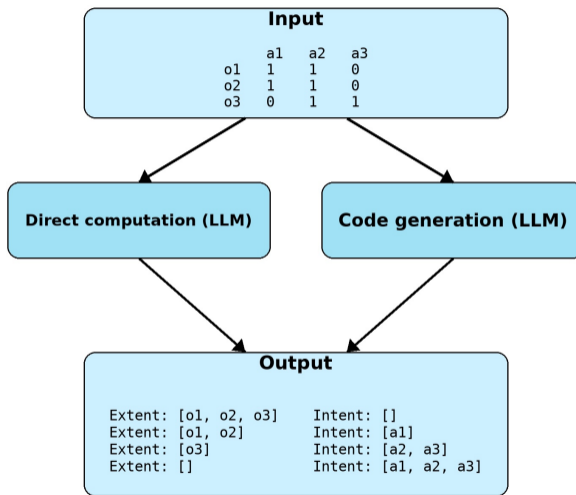0000
Methodology
0●00000000
Results
000
Discussion
00000
Perspectives
000

Approach


```
                          Input
                  a1   a2   a3
              o1   1    1    0
              o2   1    1    0
              o3   0    1    1
```

```
   Direct computation (LLM)          Code generation (LLM)
```

```
                          Output

   Extent: [o1, o2, o3]       Intent: []
   Extent: [o1, o2]           Intent: [a1]
   Extent: [o3]               Intent: [a2, a3]
   Extent: []                 Intent: [a1, a2, a3]
```

Generated by ChatGPT 5, 12/08/2025

Introduction
oooo

Methodology
oo●ooooooo

Results
ooo

Discussion
ooooo

Perspectives
ooo

LLM Selection

LMArena

- UC Berkeley
- Assess AI progress in real-world usage
- Community votes feed a public leaderboard
- Battles on: text-to-image, coding, search, math, etc.

https://lmarena.ai/leaderboard 13/08/2025

LLM Selection

LLMs versions of April 2025

- Thinking models
  - OpenAI's ChatGPT o3-mini-high
  - Google's Gemini 2.5 Pro
  - Anthropic's Claude 3.7 Sonnet for prompts requesting executable Java code
  - Anthropic's Claude 4 Sonnet for prompts not requesting for code
- Mistral AI's Mistral Large
- Deepseek Inc.'s DeepSeek-V3

Prompt Engineering

Investigated prompt techniques

- Zero-shot (single instruction; no examples)
- One-shot / few-shot (provide 1–k labeled examples)
- Role prompting (assign an explicit role or expertise)
- Contextual prompting (add background, constraints, definitions)
- Step-back prompting (restate the goal at a higher level, then solve)
- Chain of Thoughts (generate intermediate reasoning steps)
- Self-consistency (sample multiple reasoning paths and vote/select)
- Tree of Thoughts (branch, evaluate, prune; backtrack if needed)

https://www.kaggle.com/whitepaper-prompt-engineering.

Introduction
OOOO

Methodology
OOOOOOOOOOOO

Results
OOO

Discussion
OOOOO

Perspectives
OOO

Prompt Engineering (Example 1)

Zero-shot (ex. for direct computation)

In the field of formal concept analysis (FCA), given this .csv file with a context, I want you to extract and enumerate all lattice concepts. Put the list of concepts in a single line with this format: [[[], [o1, o2, o3, o4, o5]], [[a1], [o1, o3, o4]], [[a1, a2, a3, a5], [o4]]] without any quotation marks.

Zero-shot (ex. for code generation)

Generate a Java program that computes all formal concepts from a given context in Formal Concept Analysis (FCA). The input is a CSV file where rows represent objects, columns represent attributes, and cell values indicate whether an object has an attribute (e.g., 'X' for presence, empty for absence). The CSV file path should be hardcoded for use in Eclipse IDE. The program should: Read the CSV file from a fixed path (e.g., "C:/input.csv"); Parse the context into a binary matrix (objects × attributes); Implement the NextClosure algorithm (or Ganter's algorithm) to compute all formal concepts; Print each formal concept as a pair (extent, intent), where extent is the set of objects and intent is the set of attributes; Use only standard Java libraries (no external dependencies). (…)

Introduction
oooo

Methodology
ooooooo●oooo

Results
ooo

Discussion
ooooo

Perspectives
ooo

Prompt Engineering (Example 2)

Tree of Thoughts (ex. for code generation)

▶Step 0: Initial Setup
Thought A0: "I will parse CSV with BufferedReader + String.split."
Thought B0: "I will use Scanner + delimiter regex."
Thought C0: "I will use java.nio.file + Streams API."
Evaluate A0 vs B0 vs C0 (compare simplicity, performance, ease of use)
Prune all but the best parsing strategy

▶Step 1: Data Structures for Context
Thought A1: "Store incidence in boolean[][] ."
Thought B1: "Store in List<BitSet>."
Thought C1: "Store in Map<String, Set<String>."
Evaluate memory and access patterns
Prune to one
(...)

Introduction
oooo

Methodology
oooooooo●ooo

Results
ooo

Discussion
ooooo

Perspectives
ooo

Prompt Engineering (Example 2 follow up)

Tree of Thoughts (ex. for code generation)

(...)
▶Step 2: Concept Generation Algorithm
Thought A2: "Use NextClosure (Ganter's) algorithm."
Thought B2: "Use Close-by-One (CbO) algorithm."
Thought C2: "Use brute-force double closure (all subsets)."
Evaluate time complexity and clarity
Prune to one
▶Step 3: Code Structure and API
Thought A3: "One monolithic main() method."
Thought B3: "Split into FormalContext, FormalConcept, Algorithm classes."
Evaluate maintainability and readability
Prune to one
▶Step 4: Final Implementation
Based on the surviving branch choices, write the complete Java code. Ensure it: Reads the CSV
correctly; Computes all formal concepts; Prints each in the required format.

Benchmarks

Synthetic contexts (randomly generated)

- Increasing size (from 2×3 to 50×50)

- Incidence probability $p = 0.5$

- https://github.com/OskrJF/TER-M1---Code-and-Test-sets.git

Real-world contexts

- Selection of UCI Machine Learning Repository https://archive.ics.uci.edu/

- Binarized by A. Gutierrez for FCA4J evaluation

- Available at https://gite.lirmm.fr/gutierre/fca4j-benchmark

	#objects	#attributes	#incidence	Density	#concepts
AVERAGE	373	1539	17094	0,248	118778
MIN	19	9	207	0,00456	91
MAX	3720	12960	103681	0,4181	806032

Evaluation Metrics

Ground-truth

Concept lattices computed with FCA4J `https://www.lirmm.fr/fca4j`

Metrics

- precision (correctness): how many of the predicted concepts are correct

$$\text{Precision} = \frac{\text{concepts found by LLM}}{\text{concepts found by LLM} + \text{pairs found by LLM that are not concepts}}$$

- recall (completeness): how many of the concepts were found

$$\text{Recall} = \frac{\text{concepts found by LLM}}{\text{concepts found by LLM} + \text{concepts in ground truth not found by LLM}}$$

- F1 measure: Harmonic mean of precision and recall

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Process

Steps

- A program adapts FCA4J dot output to get the ground truth concept list
- In LLM produced code
  - no modification of concept computation
  - no quality check [Yetistiren et al., 2023]
  - some modification of output format for comparison with FCA4J adapted output
- A program compares the set of concepts returned by LLM code to FCA4J adapted output (exact match required for full success)
- A program compares runtimes
  - Runtime is acceptable within a factor of 10 of `fca4j` on the same datasets
  - No hard threshold imposed on peak memory

Introduction
OOOO

Methodology
OOOOOOOOOOO

Results
●OO

Discussion
OOOOO

Perspectives
OOO

Table of Contents

Introduction
oooo

Methodology
ooooooooooo

Results
o●o

Discussion
ooooo

Perspectives
ooo

Direct Computation (Small contexts $\leq 15 \times 15$)

General and mixed models: DeepSeek-V3, Mistral Large, ChatGPT-o4

- F1 20% $\longrightarrow$ 100%
- heterogeneous results, no direct correlation to context complexity

Advanced reasoning models: Claude 4 Sonnet, Gemini 2.5 Pro

- F1 $\geq$ 80%
- Near-perfect results with 15×15 with: Role prompting, CoT, Step-back, Self-consistency, etc.
- A few observed hallucinations with creation of new objects.

Computation time

≤ 1 MIN (contexts $\leq 15 \times 15$)

Code generation

"Either it barely passes or it falls apart."

Termination

Many unfinished executions
E.g. no finished execution for Chain of Thoughts in chat GPT o3-mini-high, very low success in Deepseek V3

High LLM computation time

SECs..2 DAYs for LLMs
MSs..5 MINs for FCA4J

IR metrics

Often close to 1 for the vast majority of finished executions
Very low for a few cases

Introduction
○○○○

Methodology
○○○○○○○○○○○

Results
○○○

Discussion
●○○○○

Perspectives
○○○

Table of Contents

Current performance of LLMs

Difficult to determine whether reasoning ("thinking") models implicitly produced code for direct computation, even when not instructed to do so.

No formal guarantee of correctness, although completed runs yield encouraging results.

High computation time, with frequent crashes.

Results can be highly prompt-sensitive, sometimes exhibiting chaotic variation.

Worth pursuing as future work: the code-generation approach.

Introduction
○○○○

Methodology
○○○○○○○○○○○

Results
○○○

Discussion
○○●○○

Perspectives
○○○

Other issues of LLMs

🔒 Risk of data exposure (security and confidentiality).

🌿 Environmental cost (energy use and carbon footprint).

🔀 Non-deterministic, probabilistic behavior.

⚖️ On real-world instances, the model may be swayed by domain priors rather than strictly adhering to the required combinatorial computation.

⏳ Unpredictable, hard-to-control computation time.

💵 Monetary cost.

🚫 Limited interpretability of processes and results.

Introduction
OOOO

Methodology
OOOOOOOOOOO

Results
OOO

Discussion
OOOOO

Perspectives
OOO

Usage of LLMs by unfamiliar users

Challenges:

⚠ Prompt design is nontrivial; small wording changes can flip outcomes.

🔀 Model/version instability and non-determinism hinder reproducibility.

How LLMs can help:

💡 Propose and critique prompt variants; add concise few-shot examples; propose hyperparameters.

🖧 Turn goals into structured templates (role, context, constraints, tests).

Introduction
○○○○

Methodology
○○○○○○○○○○○

Results
○○○

Discussion
○○○○●

Perspectives
○○○

Help provided by LLMs to FCA unfamiliar users

As of today, Algorithms remain essential:

⚙️ Algorithms are still required for exhaustive and reliable extraction of formal concepts, especially on medium/large contexts.

What LLMs are genuinely good at:

📖 Explaining FCA notions (contexts, derivation operators, closure) with context-specific examples.

🏷️ Summarising and labelling concepts (naming extents/intents, generating natural-language interpretations).

🧩 Prototyping glue code: parsers, small utilities, basic visualization (e.g., Hasse diagram scaffolding).

🛢️ Data preparation: normalising/merging attributes, spotting issues in CSVs.

💬 Producing didactic walkthroughs of algorithms (e.g., step-by-step NextClosure).

Introduction
OOOO

Methodology
OOOOOOOOOOO

Results
OOO

Discussion
OOOOO

Perspectives
●OO

Table of Contents

Introduction
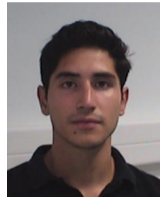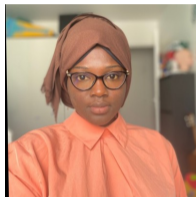OOOO

Methodology
OOOOOOOOOOO

Results
OOO

Discussion
OOOOO

Perspectives
O●O

Perspectives

- ↻ Regularly reproduce the experiments (periodic reruns and reproducibility logs).
- ⚙ Systematically study the impact of generation parameters (e.g., temperature and maximum token length).
- ▤ Train or adapt models on a curated corpus of Formal Concept Analysis (FCA) documents and code.
- ⤳ Deepen and fully automate the analysis of results.
- ❓ Revisit the broader question: within data analysis, what is the role of Formal Concept Analysis (FCA)?

Introduction
○○○○

Methodology
○○○○○○○○○○○

Results
○○○

Discussion
○○○○○

Perspectives
○○●

¡Vă mulțumesc pentru atenție!

https://github.com/OskrJF/TER-M1---Code-and-Test-sets/

Introduction
oooo
Methodology
ooooooooooo
Results
ooo
Discussion
ooooo
Perspectives
oo●

📄 Ganter, B. and Wille, R. (1999).
Formal Concept Analysis - Mathematical Foundations.
Springer.

📄 Yetistiren, B., Özsoy, I., Ayerdem, M., and Tüzün, E. (2023).
Evaluating the code quality of ai-assisted code generation tools: An empirical
study on github copilot, amazon codewhisperer, and chatgpt.
CoRR, abs/2304.10778.