IBM Research

Representing Prompting Patterns with PDL: Compliance Agent Case Study

Mandana Vaziri Louis Mandel Yuji Watanabe Hirokuni Kitahara Martin Hirzel Anca Sailer

Prompt engineering is hard
How does Prompt Declaration Language (PDL) help?

Prompts at the forefront

PDL written in YAML

Single declarative language with control structures, and functions for pattern reuse

Few orthogonal features

Composition of LLMs & code

PDL abstracts away the plumbing necessary for such compositions

Supports a wide variety of model providers and models, based on LiteLLM

Implicit accumulation of messages

PDL keeps track of the context (list of messages) implicitly, making programs much less verbose

Support for chat APIs

```yaml
description: tool use
defs:
  search:
    description: Wikipedia search    # Tool definition
    function:
      topic:
        type: string
        description: Topic to search
    return:
      lang: python                   # Python code block
      code: |
        import warnings, wikipedia
        warnings.simplefilter("ignore")
        try:
          result = wikipedia.summary("${ topic }")
        except wikipedia.WikipediaException as e:
          result = str(e)
text:
- role: system                       # System prompt
  content: |
    You are a helpful AI assistant with access to the
    following tools. If a tool does not exist in the
    provided list of tools, notify the user that you
    do not have the ability to fulfill the request.
  contribute: [context]
- role: tools                        # Tools prompt
  content:
    text: ${ [ search.signature ] } # Using the function signature
  contribute: [context]
- "What is the circumference of planet Earth?\n"   # Query
- def: actions
  model: ollama_chat/granite3.3:8b
  parser: json
  spec: [{ name: string, arguments: { topic: string }}]   # Type checking
- "\n"
- if: ${ actions[0].name == "search" }  # Conditional
  then:
    call: ${ search }                  # Tool calling
    args:
      topic: ${ actions[0].arguments.topic }
```

Intrinsics

PDL is based on granite-io and supports the following intrinsics:
thinking, hallucinations answerability, certainty citations, query-rewrite

Automated parallelization

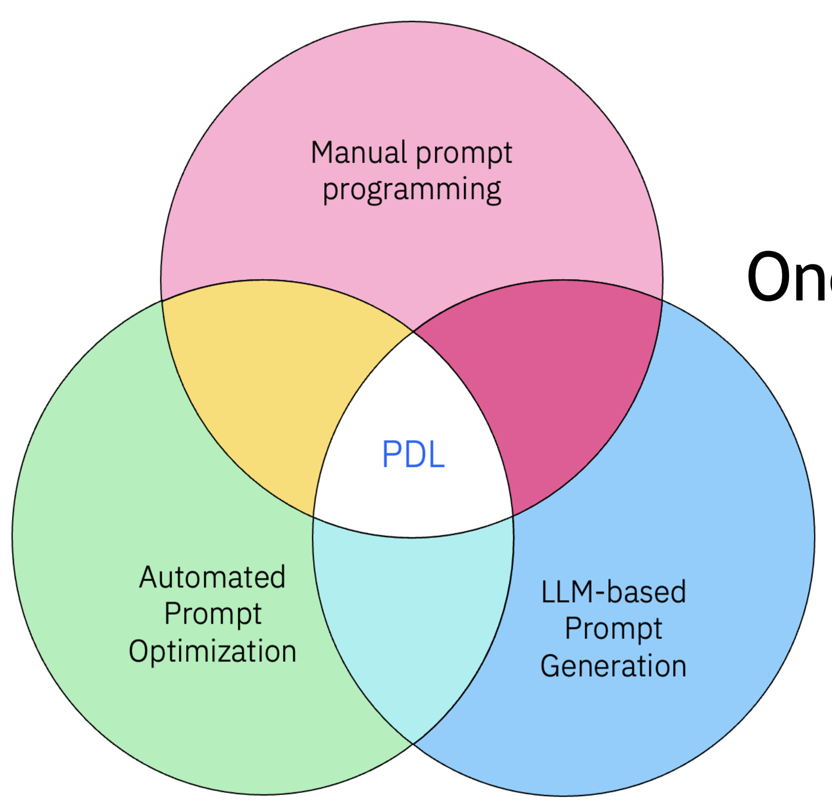
In PDL, all model calls are asynchronous and will be executed in parallel in the absence of data dependencies

Automated Prompt Optimization

AutoPDL starts with a PDL program with variables, a domain specification, and a dataset. It optimizes prompting patterns and few-shots. Paper at AutoML'25

Type checking

PDL provides type checking of both input and output of models. Types feed seamlessly into constrained decoding



One representation
Many uses

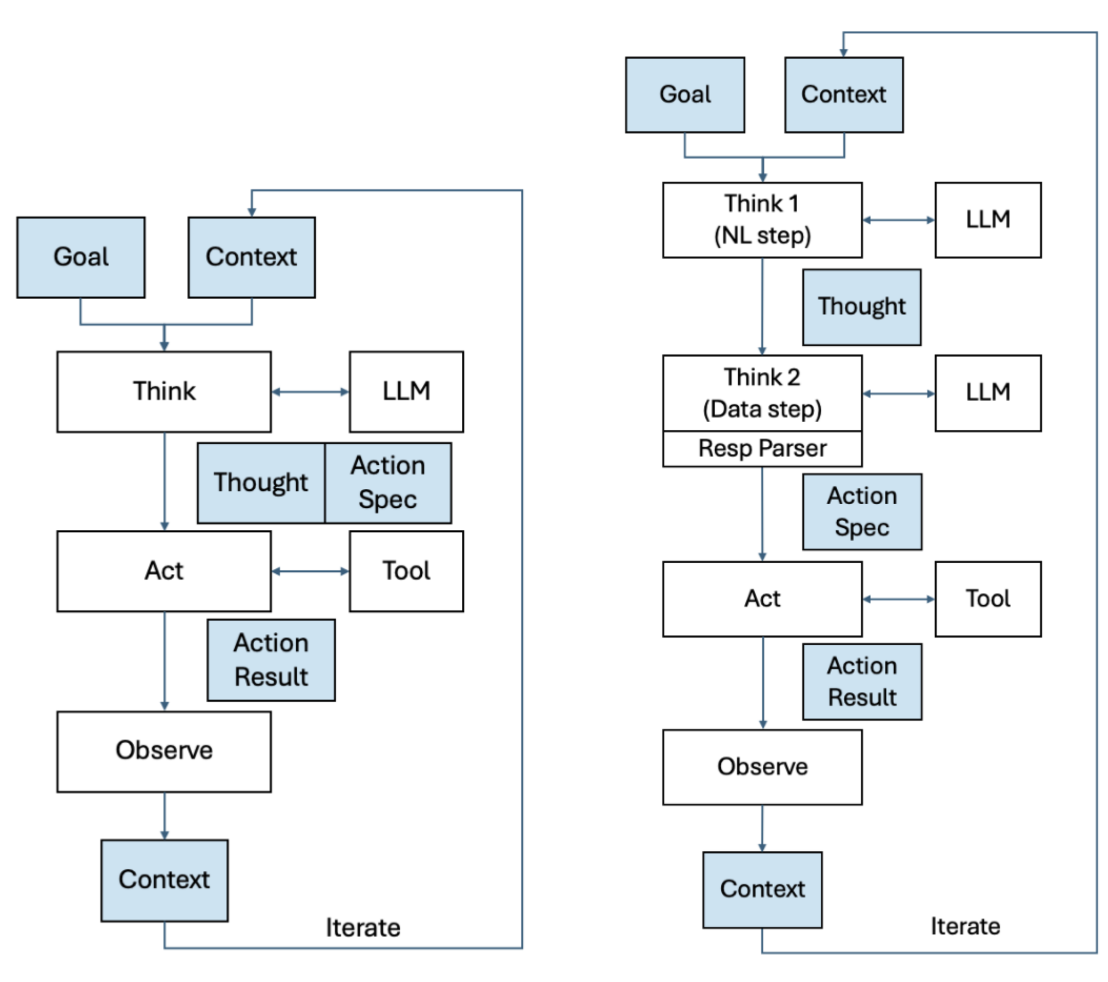
IT Compliance

IT compliance tasks require specialized expertise because of complex standards and internal organizational policies. Automation tools exist for routine operations, but policy assessment remains largely a manual task.

What is CISO Agent?

The CISO Agent provides automated support to IT teams. When it receives new regulatory requirements the agents: analyzes their content, identifies target systems, generates and deploys any scripts, validates outcomes, and provides comprehensive posture reporting.

Chief Information Security Officer (CISO) Compliance Agent Case Study



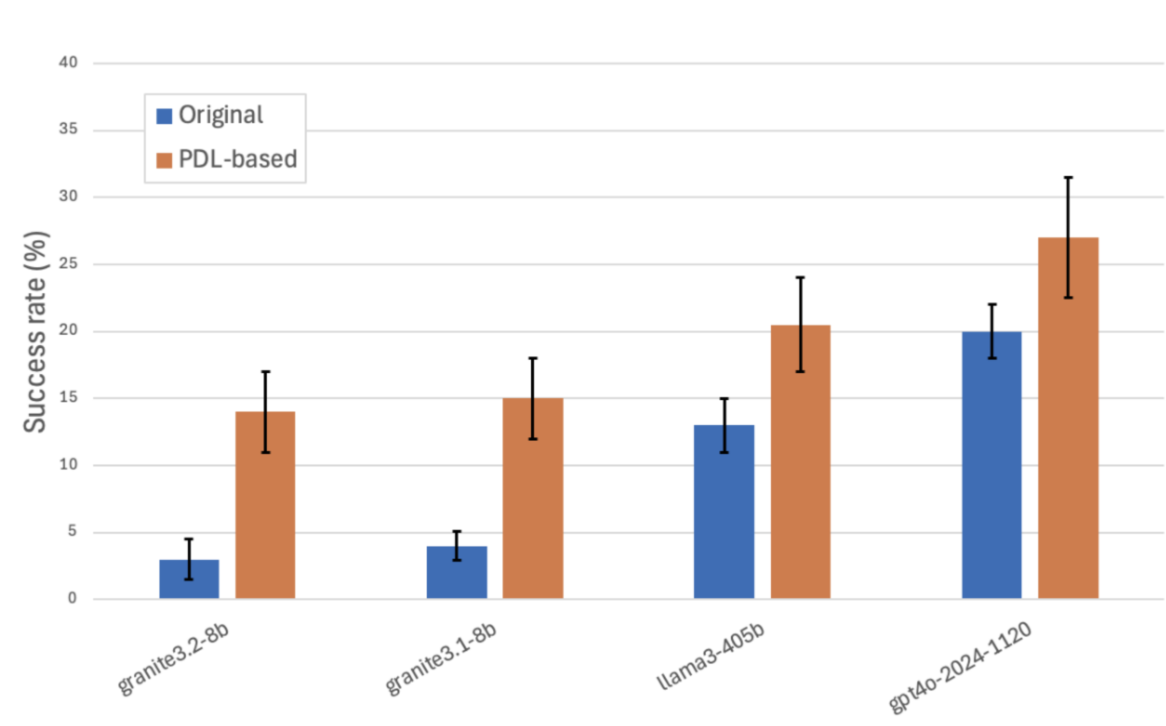
(a) Original ReAct (b) PDL-based

Prompt pattern customization with PDL

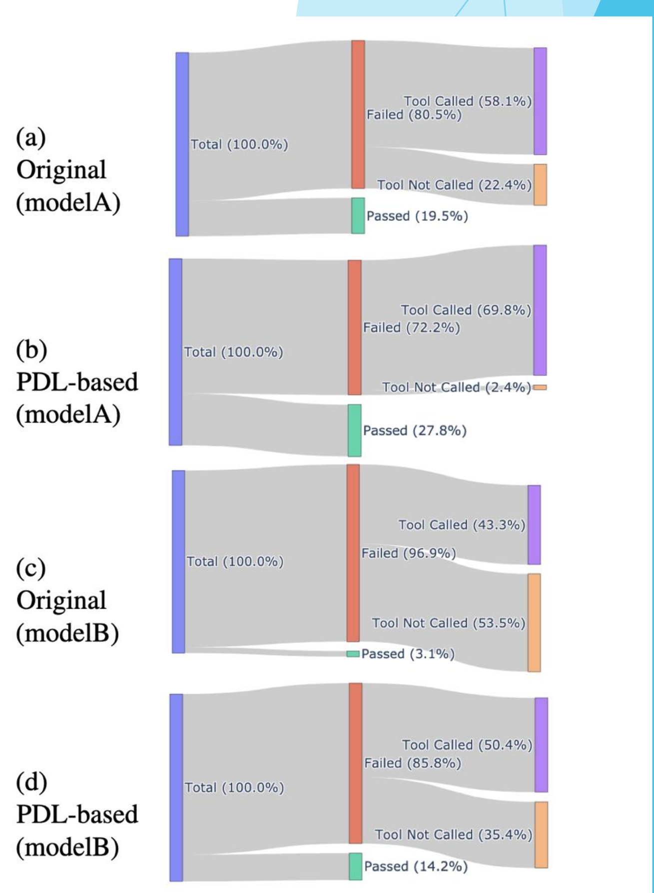
We compare two implementations of the CISO Agent: with CrewAI (a) and with PDL (b). As a framework CrewAI does not provide sufficient flexibility to customize the prompting pattern. With PDL, the Think step in the ReAct loop is split in 2 steps: first generate a thought and then generate an action spec. PDL also allowed to easily add a specialized parser for the second step.

4x better performance

The PDL implementation demonstrates consistent improvements across all models, with particularly dramatic gains in smaller models like granite3.2-8b, achieving 4 times better performance. The tool call success rate improved significantly.



Performance evaluation results using IT-Bench comparing CrewAI (blue) and PDL-based (orange) implementations



Tool Call Success Rate Comparison
model A: gpt4o-2024-11-20
model B: granite3.2-8b-instruct