LangChain Components: Chains and Pipelines

Introduction to Chains

Concept

A **Chain** in LangChain is a structured sequence of components that process data step by step — connecting user input, prompt templates, language models, memory, and output parsers. Each chain represents a logical unit of work, similar to a small pipeline in a data flow system.

Chains help in:

- Automating multi-step reasoning.
- Combining tools, retrievers, and models.
- Making LLM responses structured and context-aware.

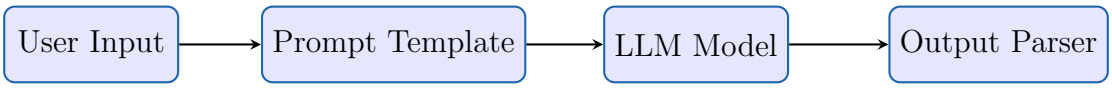

Figure 1: Basic Chain Flow: From Input to Processed Output

Sequential Chains (Pipelines)

Concept

A **Sequential Chain**, or **Pipeline**, links multiple chains so that the output of one becomes the input of the next. This forms a linear, ordered reasoning flow.

Use Cases:

- Summarize a document, then translate it.
- Extract entities, then classify their sentiment.

- Generate outlines, then expand each section.

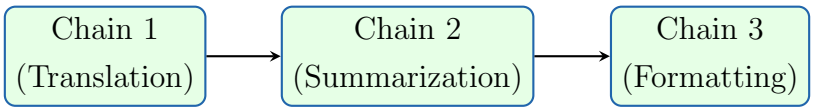

Figure 2: Sequential Chain (Pipeline): Step-by-step execution flow.

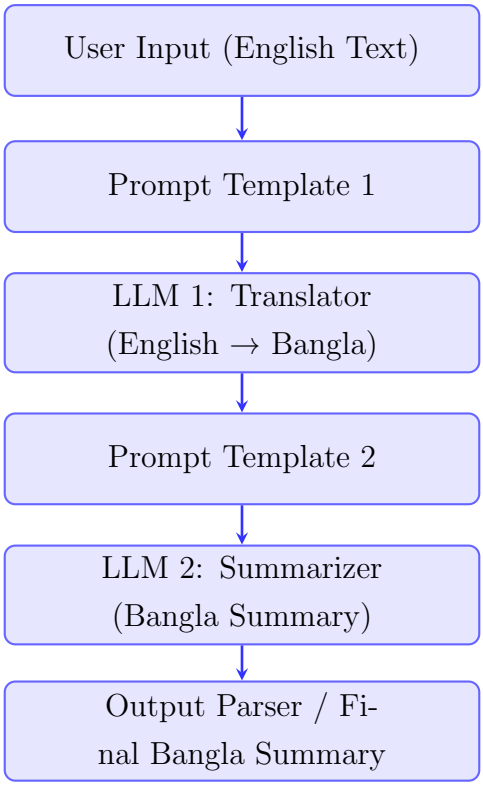

Figure 3: Sequential Chain: English-to-Bangla Translation Followed by Bangla Summarization (Top-to-Bottom Layout)

Each chain runs in sequence, passing intermediate outputs forward until the final result is produced.

Parallel Chains

Concept

Parallel Chains execute multiple independent chains simultaneously using the same input. They are ideal for tasks that can run concurrently.

Use Cases:

- Generating summary, keywords, and sentiment at once.

- Running multiple LLMs for comparison or ensemble voting.
- Multi-perspective document analysis.

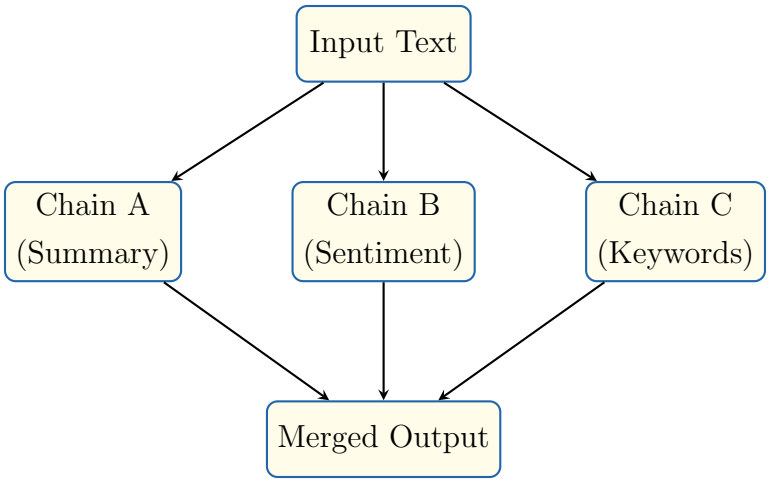

Figure 4: Parallel Chains: Multiple branches processing the same input simultaneously.

Conditional Chains

Concept

A **Conditional Chain** makes decisions dynamically — routing input to different sub-chains depending on conditions like input type, length, or confidence score.

Use Cases:

- Route long vs. short documents to different summarizers.
- Detect input language and choose corresponding model.
- Handle structured vs. unstructured input differently.

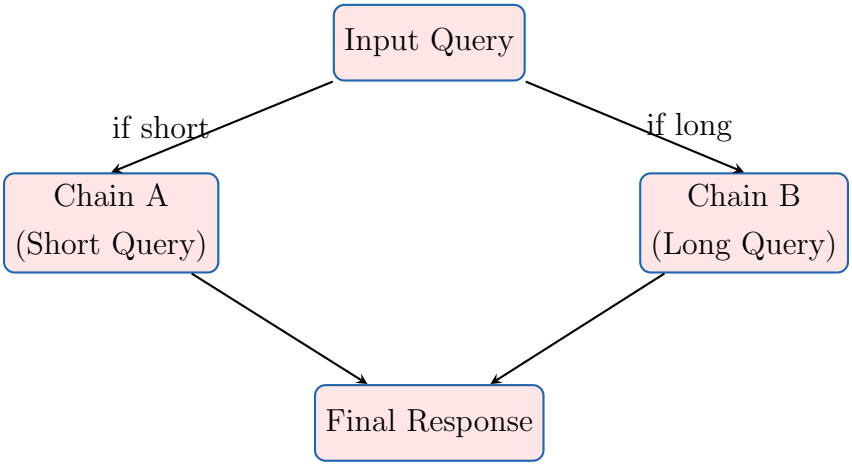

Figure 5: Conditional Chain: Input routed to different branches based on logic.

Comparative Summary

Chain Type	Purpose	Ideal Use Case
Simple Chain	Single-step task	Direct input → output
Sequential Chain	Stepwise pipeline	Multi-stage reasoning
Parallel Chain	Concurrent branches	Independent subtasks
Conditional Chain	Smart routing	Context-based decision

Conclusion

LangChain's chain system forms the **core workflow engine** of intelligent LLM applications. By composing sequential, parallel, and conditional chains, developers can build adaptive and scalable pipelines for:

- Complex question answering
- Retrieval-augmented generation (RAG)
- Decision-making and reasoning workflows

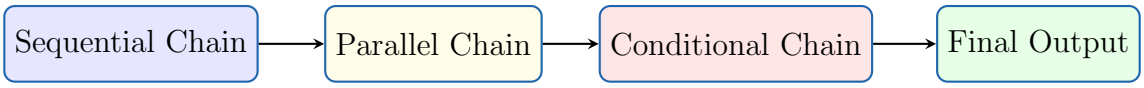

Figure 6: Unified Workflow: Combining different chains for adaptive reasoning.