LangGraph: Workflow Patterns for Agentic AI Systems

What is LangGraph?

LangGraph is an orchestration framework for building intelligent, stateful, and multi-step LLM workflows. It enables advanced features such as **parallelism**, **loops**, **branching**, **memory**, and **resumability** — making it ideal for agentic and production-grade AI applications.

LangGraph Overview

LangGraph models your logic as a **graph of nodes (tasks)** and **edges (routing)** instead of a linear chain. This allows dynamic, data-driven transitions between tasks — perfect for agentic systems that adapt in real time.

LangGraph Workflow Patterns

LangGraph supports several distinct workflow patterns for handling different AI orchestration scenarios. Each pattern represents a powerful design for combining reasoning, coordination, and modularity in LLM-driven systems.

Workflow 1: Prompt Chaining

Concept: Decompose a complex task into a sequence of smaller, manageable steps where each LLM call builds on the previous one.

When to Use

Ideal for tasks that can be cleanly broken down into fixed subtasks. This workflow trades off latency for higher accuracy — each step ensures correctness before proceeding.

Examples:

- Generate marketing copy, then translate it.
- Write an outline, verify its quality, then expand into a full document.

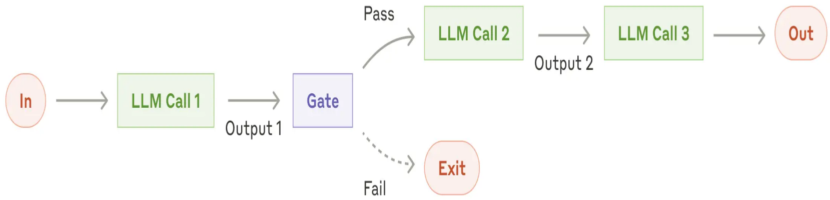

Figure 1: **Prompt Chaining Workflow**

Workflow 2: Routing

Concept: Classify incoming inputs and route them to specialized tasks or models optimized for each case.

When to Use

Effective for multi-domain systems where distinct categories require different handling. Improves efficiency by delegating simple queries to smaller models and complex ones to advanced models.

Examples:

- Route customer queries (refund, technical, general) to specialized handlers.
- Direct simple questions to lightweight models and hard ones to more powerful models.

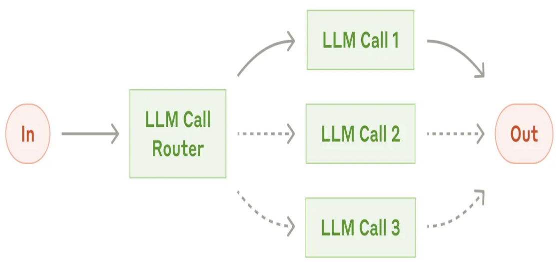

Figure 2: **Routing Workflow**

Workflow 3: Parallelization

Concept: Run multiple LLM calls simultaneously — either on independent subtasks (sectioning) or multiple trials (voting) — and then aggregate the outputs.

When to Use

Ideal for accelerating multi-part tasks or improving quality via consensus-based generation.

Examples:

- **Sectioning:** Divide a report into sections handled in parallel.
- **Voting:** Run multiple LLMs for review and select the best result.

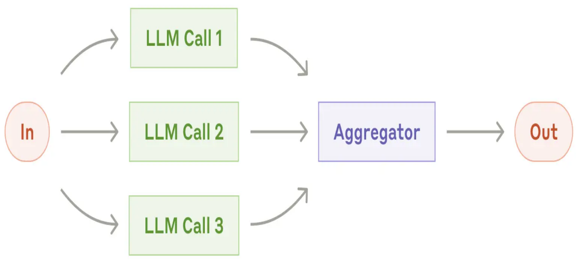

Figure 3: **Parallelization Workflow**

Workflow 4: Orchestrator-Workers

Concept: A central orchestrator dynamically plans, delegates, and coordinates subtasks among worker LLMs — synthesizing their outputs.

When to Use

Best for unpredictable or open-ended tasks where subtasks cannot be predefined — such as code generation or multi-source information gathering.

Examples:

- Autonomous coding agents modifying multiple files.
- AI research assistants gathering and analyzing data from multiple sources.

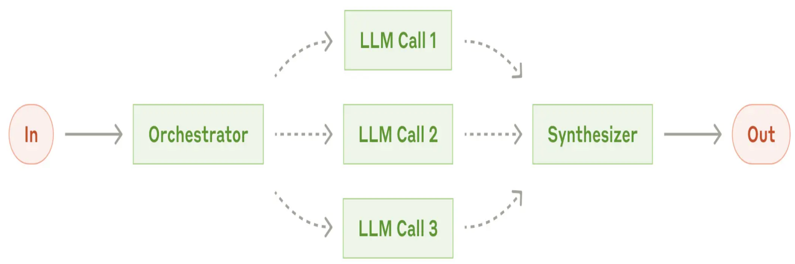

Figure 4: **Orchestrator–Workers Workflow**

Workflow 5: Evaluator-Optimizer

Concept: One LLM generates an output while another evaluates and provides feedback in an iterative improvement loop.

When to Use

Highly effective for refinement tasks with measurable evaluation criteria or feedback-driven improvement.

Examples:

- Iterative document polishing or literary translation.
- Multi-round search and summarization for research tasks.

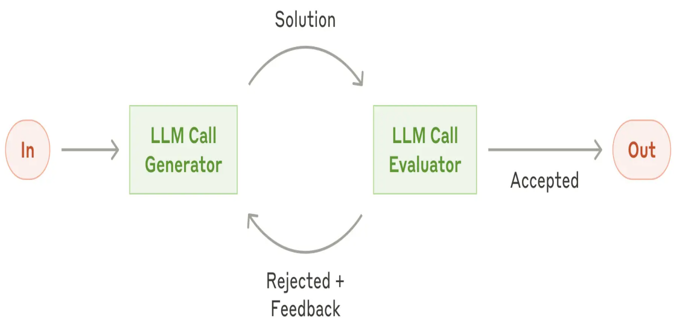

Figure 5: **Evaluator–Optimizer Workflow**

Summary

LangGraph Workflow Comparison

Workflow	**Best Use Case**
Prompt Chaining	Sequential reasoning and validation
Routing	Input classification and specialization
Parallelization	Independent or redundant task execution
Orchestrator–Workers	Dynamic task decomposition and synthesis
Evaluator–Optimizer	Iterative improvement and feedback loops

LangGraph unifies all these workflows — enabling flexible, reliable, and scalable agentic AI systems.