PROVISIONAL PATENT APPLICATION

TITLE OF THE INVENTION

Method and System for Semantic-Aware Control Flow in Programming Languages Using Large Language Models

---

INVENTOR(S) IDENTIFICATION

Inventor: [TO BE FILLED BY APPLICANT]
Address: [TO BE FILLED BY APPLICANT]
Citizenship: [TO BE FILLED BY APPLICANT]

WRITTEN DESCRIPTION (SPECIFICATION)

1. FIELD OF THE INVENTION

The present invention relates generally to the field of computer programming languages and software development methodologies, and more particularly to systems and methods for implementing control flow mechanisms in programming languages. Specifically, the invention pertains to leveraging large language models (LLMs) to enable semantic understanding and evaluation of natural language conditions within programming control flow structures at runtime, creating a new paradigm of AI-native programming languages.

2. BACKGROUND OF THE INVENTION

2.1 Traditional Control Flow Limitations

Programming languages have relied on boolean logic and explicit conditional statements since their inception. Traditional control flow structures such as if-then-else statements, switch cases, and loop conditions require programmers to translate human intent into rigid boolean expressions. This translation process introduces several limitations:

1. **Semantic Gap**: Developers must bridge the gap between human understanding ("user seems frustrated") and machine-executable logic (sentiment_score < 0.3)
2. **Context Loss**: Boolean conditions cannot capture nuanced contextual information
3. **Inflexibility**: Conditions must be explicitly defined at compile time
4. **Maintenance Burden**: As understanding evolves, countless boolean conditions must be updated

2.2 Current AI Integration Approaches

Existing approaches to integrating artificial intelligence with programming fall into three categories:

1. **AI-Assisted Code Generation**: Tools like GitHub Copilot generate code from natural language descriptions but don't change how programs execute
2. **Preprocessing Systems**: Sentiment analysis and intent recognition systems that convert natural language to structured data, which then feeds into traditional boolean logic
3. **Workflow Automation**: Platforms that use AI to extract information but still rely on conventional if-then-else structures

2.3 The Unmet Need

Despite advances in AI and natural language processing, no existing programming language or framework allows developers to write conditions in natural language that are evaluated semantically at runtime. Current systems maintain a strict separation between AI processing and program control flow, missing the opportunity for truly AI-native programming.

3. SUMMARY OF THE INVENTION

The present invention provides a revolutionary method and system for implementing semantic-aware control flow in programming languages, where natural language conditions are evaluated by large language models at runtime. This creates the first AI-native programming paradigm where LLMs become first-class citizens of program execution rather than external preprocessing tools.

Key innovations include:

1. **Runtime Semantic Evaluation**: Natural language conditions are evaluated dynamically during program execution
2. **Context-Aware Processing**: LLMs consider full conversational and execution context when evaluating conditions
3. **Protocol-Based Architecture**: Universal addressing system for contexts and resources
4. **Confidence Thresholding**: Probabilistic evaluation with configurable confidence levels
5. **Caching and Optimization**: Performance optimization for production environments

4. DETAILED DESCRIPTION OF THE INVENTION

4.1 System Architecture Overview

The invention comprises several interconnected components that work together to enable semantic-aware control flow:

```
┌─────────────────────────────────────────────────────────┐
│                   APPLICATION LAYER                       │
│  ┌─────────────┐  ┌──────────────┐  ┌──────────────┐  │
│  │  FlowMind   │  │   Workflow   │  │  Developer   │  │
│  │   Parser    │  │    Engine    │  │     API      │  │
│  └─────────────┘  └──────────────┘  └──────────────┘  │
└─────────────────────────────────────────────────────────┘
                            │
┌─────────────────────────────────────────────────────────┐
│                  SEMANTIC ENGINE LAYER                    │
│  ┌─────────────┐  ┌──────────────┐  ┌──────────────┐  │
│  │   Condition │  │   Context    │  │  Confidence  │  │
│  │  Evaluator  │  │   Manager    │  │   Scorer     │  │
│  └─────────────┘  └──────────────┘  └──────────────┘  │
└─────────────────────────────────────────────────────────┘
                            │
┌─────────────────────────────────────────────────────────┐
│                    LLM INTERFACE LAYER                    │
│  ┌─────────────┐  ┌──────────────┐  ┌──────────────┐  │
│  │    Prompt   │  │   Response   │  │   Caching    │  │
│  │   Builder   │  │    Parser    │  │   Layer      │  │
│  └─────────────┘  └──────────────┘  └───────────────┘  │
└────────────────────────────────────────────────────────┘
                            │
┌────────────────────────────────────────────────────────┐
│                   PROTOCOL LAYER                        │
│  ┌─────────────┐  ┌──────────────┐  ┌───────────────┐  │
│  │  Protocol   │  │   Resource   │  │     Auto      │  │
│  │  Registry   │  │   Resolver   │  │   Discovery   │  │
│  └─────────────┘  └──────────────┘  └───────────────┘  │
└────────────────────────────────────────────────────────┘
```

4.2 FlowMind Language Syntax

The invention introduces FlowMind, a YAML-based language extension that enables semantic control flow:

Traditional Approach:

```yaml
- analyze_sentiment: user_input
- calculate: sentiment_score
- if: sentiment_score < 0.3
  then:
    - escalate_to_human
```

FlowMind Semantic Approach:

```yaml
- if: "user seems frustrated with the response"
  confidence: 0.8
  then:
    - escalate_to_human
  else:
    - continue_conversation
```

4.3 Semantic Evaluation Process

The semantic evaluation process follows these steps:

1. **Condition Extraction**: The FlowMind parser identifies semantic conditions in the workflow
2. **Context Assembly**: Relevant context is gathered from the execution environment
3. **Prompt Construction**: A specialized prompt is built for the LLM evaluation
4. **LLM Evaluation**: The condition is evaluated by the LLM with context
5. **Confidence Scoring**: The response is analyzed for confidence level
6. **Flow Decision**: Control flow branches based on evaluation result

4.4 Protocol-Based Context System

The invention includes a revolutionary protocol-based addressing system:

```
agent://cortisol_guardian
file://contexts/custom/my-agent.yaml
markdown://prompts/system-base.md
script://generators/dynamic-personality.js
```

This system enables:

- Universal resource addressing
- Auto-discovery of contexts
- User-defined protocol handlers
- Infinite extensibility

4.5 Implementation Details

4.5.1 Condition Evaluator Algorithm

```javascript
async function evaluateSemanticCondition(condition, context) {
  // Check cache first
  const cacheKey = hashCondition(condition, context);
  if (cache.has(cacheKey)) {
    return cache.get(cacheKey);
  }

  // Build evaluation prompt
  const prompt = buildEvaluationPrompt(condition, context);

  // Query LLM
  const response = await llm.evaluate(prompt);

  // Parse response and extract confidence
  const result = parseEvaluation(response);

  // Cache result
  cache.set(cacheKey, result, TTL);

  return result;
}
```

4.5.2 Context Assembly Process

```javascript
async function assembleContext(protocols) {
  const contexts = [];

  for (const protocol of protocols) {
    const handler = registry.getHandler(protocol);
    const content = await handler.load(protocol);
    contexts.push(content);
  }

  return mergeContexts(contexts);
}
```

4.6 Performance Optimizations

1. **Intelligent Caching**: Results cached based on condition and context hash
2. **Batch Evaluation**: Multiple conditions evaluated in single LLM call

3. **Confidence Shortcuts**: Skip evaluation for high-confidence cached results
4. **Fallback Mechanisms**: Graceful degradation when LLM unavailable

5. EMBODIMENTS OF THE INVENTION

5.1 Customer Service Workflow

```
name: customer-support-flow
flow:
  - if: "customer is asking about a previous issue"
    then:
      - load: agent://memory-specialist
      - retrieve_context: customer_history

  - if: "customer seems satisfied with the explanation"
    confidence: 0.7
    then:
      - mark_resolved
    else:
      - if: "issue requires technical expertise"
        then:
          - escalate: technical_support
```

5.2 Development Assistant

```
name: code-review-assistant
flow:
  - analyze: submitted_code

  - if: "code has potential security vulnerabilities"
    confidence: 0.9
    then:
      - flag_for_security_review
      - generate_security_report

  - if: "code style matches team conventions"
    then:
      - approve_style_check
    else:
      - suggest_improvements
```

5.3 Content Moderation System

```
name: content-moderator
flow:
  - if: "content appears to violate community guidelines"
    confidence: 0.85
    then:
```

```
    - if: "violation seems unintentional"
      then:
        - send_warning
      else:
        - remove_content
        - notify_moderators
```

6. TECHNICAL ADVANTAGES

1. **Intuitive Programming**: Developers write conditions as they think
2. **Reduced Complexity**: Eliminates complex boolean logic trees
3. **Adaptive Behavior**: Systems learn and improve over time
4. **Context Awareness**: Full situational understanding in decisions
5. **Maintainability**: Natural language is self-documenting
6. **Flexibility**: Conditions can evolve without code changes

7. INDUSTRIAL APPLICABILITY

The invention has broad applications across industries:

1. **Enterprise Automation**: Complex business rules in natural language
2. **Healthcare**: Patient interaction systems with nuanced understanding
3. **Education**: Adaptive learning systems that understand student needs
4. **Finance**: Risk assessment with contextual understanding
5. **Gaming**: NPCs with realistic conversational behaviors
6. **IoT**: Smart home systems that understand user intent

8. VARIATIONS AND EXTENSIONS

1. **Multi-Modal Evaluation**: Extend to images, audio, video
2. **Distributed Evaluation**: Multiple LLMs for consensus
3. **Domain-Specific Models**: Specialized LLMs for vertical markets
4. **Hybrid Approaches**: Combine semantic and traditional logic
5. **Real-Time Learning**: Conditions that adapt based on outcomes

9. DRAWINGS DESCRIPTION

Figure 1: System Architecture Diagram showing the interaction between FlowMind Parser, Semantic Engine, LLM Interface, and Protocol Layer

Figure 2: Semantic Evaluation Flow Chart illustrating the step-by-step process from condition extraction to flow decision

Figure 3: Protocol-Based Context Assembly showing how different protocol handlers resolve and load contexts

Figure 4: Performance Optimization Architecture depicting caching layers, batch processing, and fallback mechanisms

Figure 5: Example Workflow Execution demonstrating semantic condition evaluation in a customer service scenario

---

CLAIMS (Informal - for provisional filing)

While formal claims are not required for provisional applications, the following informal claims outline the scope of protection sought:

1. A method for implementing semantic-aware control flow in programming languages, comprising:
    - Parsing source code to identify natural language conditional statements
    - Assembling contextual information relevant to condition evaluation
    - Submitting the natural language condition and context to a large language model
    - Receiving and interpreting the model's evaluation with confidence scoring
    - Directing program execution flow based on the semantic evaluation result
2. The method of claim 1, further comprising a protocol-based context assembly system for universal resource addressing
3. The method of claim 1, wherein semantic evaluations are cached based on condition and context signatures
4. A system for semantic-aware programming, comprising:
    - A parser for identifying natural language conditions in code
    - A semantic engine for preparing and submitting conditions to LLMs
    - A protocol registry for extensible context handling
    - A confidence scoring mechanism for probabilistic flow control
5. The system of claim 4, further comprising auto-discovery mechanisms for self-organizing contexts
6. A programming language extension enabling natural language conditions in control flow structures
7. The language extension of claim 6, supporting confidence thresholds for probabilistic branching
8. A computer-readable medium containing instructions for implementing semantic-aware control flow
9. The medium of claim 8, wherein the instructions enable runtime evaluation of natural language conditions
10. A method for optimizing semantic condition evaluation through intelligent caching and batch processing

---

ABSTRACT

A method and system for implementing semantic-aware control flow in programming languages where natural language conditions are evaluated by large language models at runtime. The invention enables developers to write control flow conditions in natural language (e.g., "if user seems frustrated") that are dynamically evaluated during program execution. The system includes a FlowMind parser for identifying semantic conditions, a semantic engine for orchestrating LLM evaluation, a protocol-based context assembly system for universal resource addressing, and performance optimizations including intelligent caching. This creates the first AI-native programming paradigm where LLMs become first-class citizens of program execution, fundamentally transforming how software responds to human intent and context.

End of Provisional Patent Application