CLAIM LANGUAGE FOR PROVISIONAL PATENT SUPPORT

KK-2025-PROV-001: Grammar-Constrained Code Generation for Embedded Devices

---

> ***Note:*** *While provisional applications do not require formal claims, including claim-like language establishes clear priority support for the non-provisional filing. The following claims are drafted to maximize coverage of the disclosed embodiments.*

INDEPENDENT CLAIMS

Claim 1 — System Claim (Primary)

A system for generating and executing control programs on resource-limited embedded devices, the system comprising:

(a) a grammar-constrained language model configured to receive a natural language input describing a desired device behavior and to generate a sequence of tokens representing a program in a domain-specific language (DSL);

(b) a grammar-guided decoding module that maintains a parser state representing a partial derivation of the DSL grammar and that masks output logits of the language model at each generation step to permit only tokens corresponding to valid grammar continuations, thereby ensuring that the generated program is syntactically valid;

(c) a compiler configured to translate the generated DSL program into a bytecode representation comprising a sequence of instructions from a predefined instruction set; and

(d) a micro-interpreter executing on an embedded microcontroller, the micro-interpreter configured to execute the bytecode representation with bounded memory usage and deterministic timing characteristics.

Claim 2 — Method Claim (Primary)

A method for programming a resource-limited embedded device using natural language, the method comprising:

(a) receiving a natural language description of a desired device behavior;

(b) processing the natural language description through a grammar-constrained language model to generate tokens of a domain-specific language program, wherein generating each token comprises: (i) computing logits over a vocabulary, (ii) determining valid next tokens based on a current parser state and a context-free grammar of the domain-specific language, (iii) masking logits corresponding to invalid tokens, and (iv) sampling a token from the masked distribution;

(c) compiling the generated domain-specific language program into bytecode comprising instructions executable by a micro-interpreter;

(d) transmitting the bytecode to an embedded microcontroller; and

(e) executing the bytecode on the embedded microcontroller using a micro-interpreter having a fixed maximum memory footprint and bounded worst-case execution time per instruction.

Claim 3 — Apparatus Claim (Embedded Device)

An embedded computing apparatus comprising:

(a) a microcontroller having a processor, a program memory, and a data memory, wherein the data memory is less than 512 kilobytes;

(b) a micro-interpreter stored in the program memory, the micro-interpreter comprising: (i) an instruction decoder configured to decode bytecode instructions from a predefined instruction set, (ii) an evaluation stack having a fixed maximum depth, (iii) a variable storage region having a fixed maximum size, and (iv) an execution loop configured to dispatch instructions with deterministic cycle counts;

(c) a program storage region configured to store bytecode received from an external source; and

(d) one or more hardware interfaces controlled by execution of the bytecode, the hardware interfaces selected from the group consisting of: general-purpose input/output pins, analog-to-digital converters, pulse-width modulation outputs, serial communication interfaces, and sensor interfaces.

Claim 4 — Method Claim (Grammar-Guided Generation)

A method for generating syntactically valid programs using a language model, the method comprising:

(a) defining a context-free grammar specifying valid syntax of a target domain-specific language;

(b) initializing a parser state representing an empty derivation;

(c) for each generation step until a complete program is formed: (i) computing, by a transformer-based language model, a probability distribution over a token vocabulary, (ii) querying the parser state to determine a set of tokens that represent valid grammar continuations, (iii) setting probabilities of tokens not in the set of valid grammar continuations to zero or a negligible value, (iv) sampling a token from the modified probability distribution, and (v) updating the parser state based on the sampled token; and

(d) outputting the sequence of sampled tokens as a syntactically valid program.

Claim 5 — System Claim (Fine-Tuning Architecture)

A system for adapting a language model to generate domain-specific code, the system comprising:

(a) a pre-trained transformer language model having a plurality of weight matrices;

(b) a sparse low-rank adaptation (S-LoRA) module comprising, for each adapted weight matrix: (i) a first low-rank matrix having dimensions $d \times r$ where r is substantially less than d, (ii) a second low-rank matrix having dimensions $r \times d$, (iii) a sparsity mask defining which blocks of the low-rank matrices are active, wherein inactive blocks have zero-valued parameters; and

(c) a training procedure configured to jointly optimize the low-rank matrices and the sparsity mask to minimize a loss function over training examples comprising natural language descriptions and corresponding domain-specific language programs.

DEPENDENT CLAIMS — System Claims

Claim 6

The system of claim 1, wherein the grammar-constrained language model has fewer than 50 million parameters.

Claim 7

The system of claim 1, wherein the grammar-constrained language model has fewer than 10 million parameters and is quantized to 8-bit or 4-bit integer precision.

Claim 8

The system of claim 1, wherein the embedded microcontroller is selected from the group consisting of: ESP32, ESP8266, RP2040, ATmega328, ATmega2560, STM32F4, and SAMD21.

Claim 9

The system of claim 1, wherein the micro-interpreter has a memory footprint of less than 16 kilobytes of program memory and less than 4 kilobytes of data memory.

Claim 10

The system of claim 1, wherein the micro-interpreter performs no dynamic memory allocation during program execution.

Claim 11

The system of claim 1, wherein the bytecode instruction set comprises instructions for: - pushing and popping values on an evaluation stack, - loading and storing values in variable

storage, - reading sensor values, - setting output pin states, - conditional branching, and - timed delay operations.

Claim 12

The system of claim 1, wherein the domain-specific language comprises constructs for: - event-driven conditional statements triggered by sensor readings, - periodic timed actions, and - pin or actuator control statements.

Claim 13

The system of claim 1, further comprising a verification module configured to analyze the generated DSL program to determine: - maximum stack depth, - maximum memory usage, - worst-case execution time, and - absence of unbounded loops.

DEPENDENT CLAIMS — Method Claims

Claim 14

The method of claim 2, wherein the grammar-guided decoding is performed incrementally token-by-token without backtracking.

Claim 15

The method of claim 2, wherein the context-free grammar is represented as a parsing table enabling O(1) lookup of valid next tokens given a parser state.

Claim 16

The method of claim 2, wherein transmitting the bytecode to the embedded microcontroller comprises transmitting over one of: a serial connection, a WiFi connection, or a Bluetooth connection.

Claim 17

The method of claim 2, further comprising: - storing a previously generated bytecode program on the embedded microcontroller, and - replacing the previously generated bytecode program with newly generated bytecode without restarting the microcontroller.

Claim 18

The method of claim 4, wherein the domain-specific language is a language for embedded device control comprising primitives for sensor reading, actuator control, and timed operations.

Claim 19

The method of claim 4, wherein the language model is a decoder-only transformer model with fewer than 50 million parameters.

Claim 20

The method of claim 4, further comprising fine-tuning the language model using sparse low-rank adaptation (S-LoRA) on a dataset of natural language descriptions paired with corresponding DSL programs.

DEPENDENT CLAIMS — Apparatus Claims

Claim 21

The apparatus of claim 3, wherein the micro-interpreter executes each instruction in a bounded number of processor cycles, enabling determination of worst-case execution time.

Claim 22

The apparatus of claim 3, wherein the program storage region stores bytecode received over a wireless communication interface.

Claim 23

The apparatus of claim 3, wherein the evaluation stack has a maximum depth of 32 elements or fewer.

Claim 24

The apparatus of claim 3, wherein the micro-interpreter implements a cooperative scheduling model allowing multiple concurrent behavioral rules to execute in time-sliced fashion.

Claim 25

The apparatus of claim 3, further comprising a safety module configured to: - prevent write operations to restricted hardware pins, - enforce rate limits on output state changes, and - bounds-check sensor values before use in computations.

DEPENDENT CLAIMS — S-LoRA Claims

Claim 26

The system of claim 5, wherein the sparsity mask is learned during training using a differentiable mask selection procedure.

Claim 27

The system of claim 5, wherein the sparsity mask defines block-structured sparsity with blocks of size at least 4×4.

Claim 28

The system of claim 5, wherein the rank r of the low-rank matrices is 16 or less.

Claim 29

The system of claim 5, wherein the sparse low-rank adaptation parameters are quantized to 8-bit or 4-bit integer precision for deployment.

Claim 30

The system of claim 5, wherein the training procedure comprises: - a first phase training the low-rank matrices with full density, and - a second phase introducing and optimizing the sparsity mask.

ALTERNATIVE EMBODIMENT CLAIMS

Claim 31 — On-Device Generation

A system according to claim 1, wherein the grammar-constrained language model executes on the embedded microcontroller, and wherein the grammar-constrained language model is a quantized model with fewer than 10 million parameters stored in flash memory of the embedded microcontroller.

Claim 32 — FPGA Implementation

An apparatus according to claim 3, wherein the micro-interpreter is implemented as a hardware state machine in a field-programmable gate array (FPGA) rather than as software executing on a processor.

Claim 33 — Multi-Grammar Support

A system according to claim 1, further comprising: - a grammar repository storing a plurality of domain-specific language grammars corresponding to different device domains, and - a grammar selection module configured to select an appropriate grammar based on the detected intent of the natural language input.

Claim 34 — Formal Verification

A method according to claim 2, further comprising: - prior to transmitting the bytecode to the embedded microcontroller, performing formal verification of the bytecode to prove absence of stack overflows, memory access violations, and timing constraint violations.

Claim 35 — Streaming Pipeline DSL

A system according to claim 1, wherein the domain-specific language comprises constructs for defining streaming data pipelines with multiple concurrent data sources, transformation stages, and output sinks.