Artificial Intelligence applied to code repair after code static analysis verification

18th ESA Workshop on Avionics, Data, Control and Software Systems ~ ADCSS2024

Ainhoa López (TAS) / David de Fitero (UAH)

Evolution of APR Methods and LLM Advances

- Traditional APR Methods
  - **Heuristic-based:** Uses predefined rules to generate patches (Saha et al., 2017)
  - **Constraint-based:** Reduces search space using logical restrictions (Xiong et al., 2017)
  - **Template-based:** Applies predefined correction patterns (J. Jiang et al., 2018)
- Modern ML Approaches
  - Frame APR as a translation task using **RNNs**, **LSTMs**, and **Transformers**
- Vulnerability Repair
  - **General-purpose tools** (e.g., Angelix (Mechtaev et al., 2016), concolic program repair (Shariffdeen et al., 2021)).
  - **Security-specific methods** (deductive reasoning, inductive reasoning, and static analysis)
- LLM developments in APR
  - **LLMs with completion engines** improve patch generation accuracy (Wei et al., 2023)
  - **Fine-tuned LLMs** surpass traditional APR tools (Huang et al., 2023; Silva et al., 2024; Xia et al., 2023)

Followed Approach

- Datasets
  - **CommitPackFt** capturing real-world bug fixes
  - **SonarQube rules dataset** based on code quality rules
- Fine-tuning
  - **QLoRA** (Dettmers et al., 2023): Combines LoRA (Low-Rank Adaptation) with 4-bit quantization for resource-efficient fine-tuning
  - **NEFTune** (Jain et al., 2023): Adds noise to embeddings for improved performance
- **Refinement after initial results**
  - Expanded training data
    - **Synthetic dataset** to augment coverage of SonarQube rules, providing more diverse training examples
    - Added **650 manual corrections** aligned with MISRA guidelines
  - Model upgrade - **Llama 3 - 8B**

Description of the Datasets

- CommitPackFT
  - Filtered version of the **CommitPack dataset**
  - Contains **high-quality commit messages** resembling natural language instructions
  - Paired with **before and after code** corresponding to the commit
- Synthetic Dataset
  - Generated using **Llama 3 - 70B** model
  - Seed examples provided with **code before/after changes** based on **SonarQube rules**
  - Model generated **new samples** following similar patterns from seeds
- SonarQube Dataset
  - Extracted from SonarQube analysis
  - Aligned with **MISRA C** standards
  - Includes **human-generated fixes** for detected issues
  - **Smallest dataset -** upsampled for balance

Training Process Overview

- Data Preprocessing
  - Input
    - Commit message or SonarQube rule
    - Source code with line numbers
  - Output
    - Line range of modification and the new lines to replace them
- Model & Techniques
  - CodeLlama / Llama 3 - 8B
  - Supervised Fine-tuning with QLoRA/LoRA and NEFTune
- Hardware
  - Trained on a NVIDIA H100

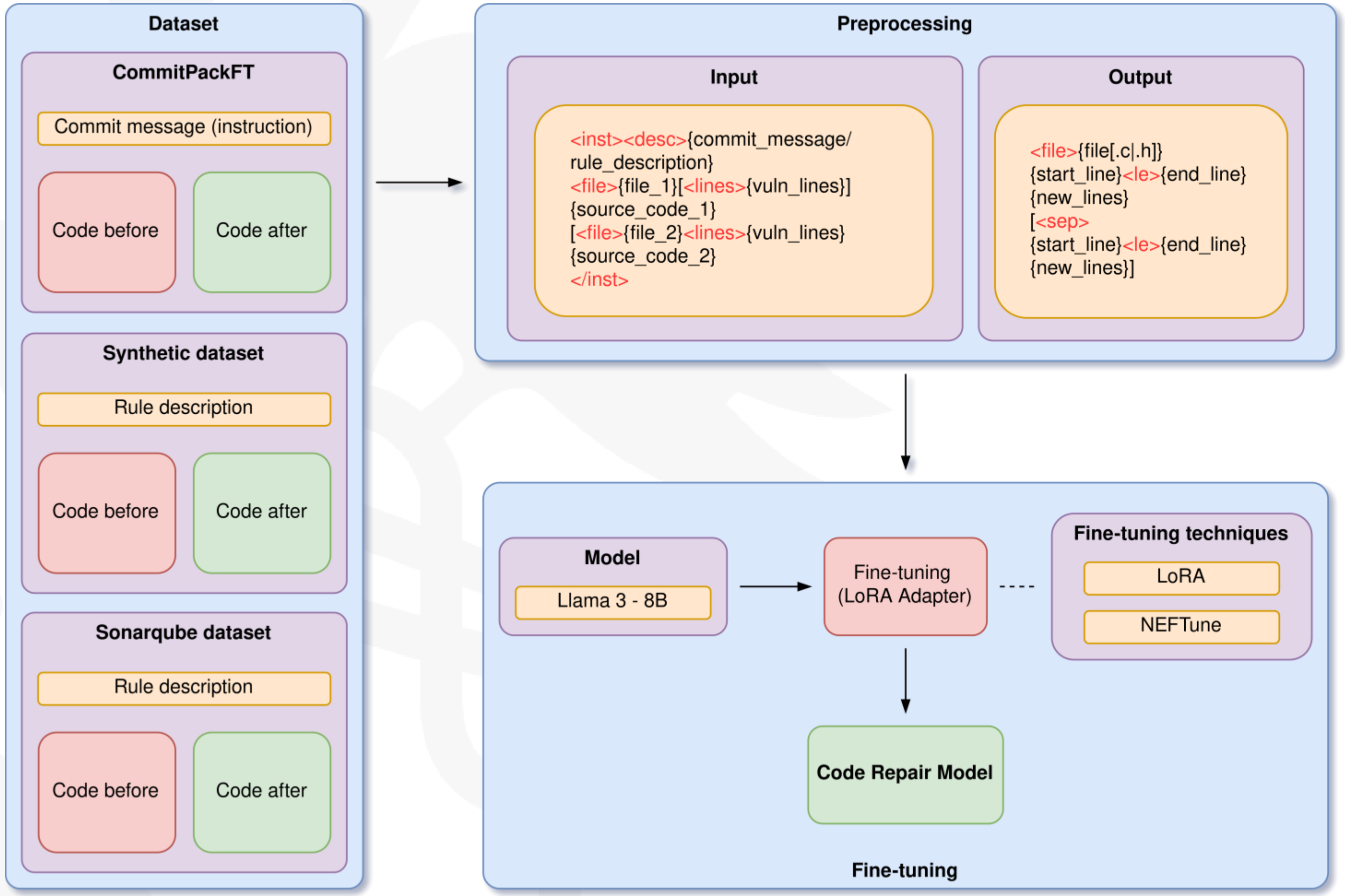
Input

```
<inst><desc>Uppercase the literal suffix.
<file>file.c<lines>424
420         if (dds_corr_magnitude > algo_wrapper_cmcu_2_man_cfg->fw_algo_fdir_cfg_4.dyn_fdir_dis_thr) {
421             algo_fdir_cycles = algo_wrapper_cmcu_2_man_cfg->fw_algo_fdir_cfg_4.dyn_fdir_dis_time;
422         }
423
424         if (algo_fdir_cycles > 0u) {
425             algo_fdir_cycles = algo_fdir_cycles - 1u;
426             (void) bsp_gpio_set_low(BSP_GPIO_FCA_EN);
427         }
428 </inst>
```

Output

```
<file>file.c
423<le>426
    if (algo_fdir_cycles > 0U) {
        algo_fdir_cycles = algo_fdir_cycles - 1U;
```

Approach diagram



{OPEN}

Refinement techniques

- Tailored contextual input
  - Use **tree-sitter** to reduce model input
  - Adjust context size based on the specific rule
- Additional Instructions
  - Adds specific instructions to certain rules to improve repair efficacy and ensure adherence to coding standards
- Simultaneous Fixing
  - Utilizes **batch inference** to accelerate processing when multiple repair rules are provided
- Quantization with EETQ
  - **Int8 quantization** reduces memory use with minimal accuracy loss
- Inference Engine
  - Uses **Text Generation Inference** (TGI) for fast and efficient model inference

Experiments

Manual human evaluation for the results provided for the project CO2M product ICU.

- **Method:** CodeLlama, fine-tuned on CommitPackFT dataset + own dataset and refined model Llama3
- **Input dataset to the model:** 21 components of CO2M HDSW ICU:
  - JSON with source code (in the example: "01_00d_irqmp.c")
  - JSON with header file
  - JSON structure with information on the issues detected: rule ID, rule description, line of code where issue is detected


```
{
    "file_name": "01_00d_irqmp.c",
    "code": "/*==========================================================\n *              THALES ALENIA SPACE SPAIN\n *==========================================================\n * The copyright to the computer program(s) herein is the\n * property of Thales Alenia Space Spain, Madrid.\n * The program(s) shall be used and/or copied only with the\n * written permission of Thales Alenia Space Spain or in\n * accordance with the terms and conditions stipulated in\n * the agreement/contract under which the program(s) have\n * been supplied\n *==========================================================*/\n/**\n * \\file 01_00c_irqmp.c\n * \\date 15/01/2015\n * \\author Thales Alenia Space Espana\n */\n\n/*==============================================\n * IMPLEMENTED INTERFACES\n *==============================================*/\n#include <01_00d_irqmp.h>\n\n/*==============================================\n * USED INTERFACES\n *==============================================*/\n\n/*==============================================\n * PRIVATE DEFINED CONSTANT #define\n *==============================================*/\n#define IRQMP_IRQ_NUM_MIN\t\t1\n#define IRQMP_IRQ_NUM_MAX\t\t31\n\n/* Level, force and clear registers work on basic interrupts */\n#define IRQMP_LEVEL_MASK\t\t0x0000FFFEU\n#define IRQMP_FORCE_MASK\t\t0x0000FFFEU\n#define IRQMP_CLEAR_MASK\t\t0x0000FFFEU\n\n/** Multiprocessor Status Register NCPU bits */\n#define IRQMP_STATUS_NCPU_OFFSET \t28\n#define IRQMP_STATUS_NCPU_MASK \t\t(0xFU << IRQMP_STATUS_NCPU_OFFSET)\n\n/** Multiprocessor Status Register BA bit */\n#define IRQMP_STATUS_BA_MASK \t\t(1U << 27)\n\n/** Multiprocessor Status Register EIRQ bits position */\n#define IRQMP_STATUS_EIRQ_OFFSET \t16\n#define IRQMP_STATUS_EIRQ_MASK \t\t(0xFU << IRQMP_STATUS_EIRQ_OFFSET)\n#define IRQMP_BCAST_MASK \t\t0x0000FFFEU\n#define IRQMP_PROC_IRQ_MASK_MASK \t0xFFFFFFFEU\n\n/*==============================================\n * STATIC GLOBAL VARIABLE\n *==============================================*/\n\n/*==============================================\n * IMPLEMENTED INTERFACES\n *==============================================*/\nvoid irqmp_set_irq_level_high(irqmp_t *u, unsigned int irq)\n{\n\t/* Set corresponding bit in level register */\n\tu->level |= ((1U << irq) & IRQMP_LEVEL_MASK);\n}\n\nvoid irqmp_set_irq_level_low(irqmp_t *u, unsigned int irq)\n{\n\t/* Clear corresponding bit in level register */\n\tu->level &= ~((1U << irq) & IRQMP_LEVEL_MASK);\n}\n\nunsigned int irqmp_is_irq_pending(const irqmp_t *u, unsigned int irq)\n{\n\t/* Check corresponding bit in pending register */\n\treturn u->pending & (1U << irq);\n}\n\nvoid irqmp_force_irq(irqmp_t *u, unsigned int cpu, unsigned int irq)\n{\n\tif (cpu < IRQMP_MAX_SUPPORTED_PROCS) {\n\t\tif (cpu == 0) {\n\t\t\t/* Use force register for CPU 0 */\n\t\t\tu->force |= ((1U << irq) & IRQMP_FORCE_MASK);\n\t\t}\n\t\telse {\n\t\t\t/* Use force[cpu] register for other CPUs */\n\t\t\tu->proc_force[cpu] |= ((1U << irq) & IRQMP_FORCE_MASK);\n\t\t}\n\t}\n}\n\nvoid irqmp_clear_irq(irqmp_t *u, unsigned int irq)\n{\n\t/* Set corresponding bit in clear register */\n\tu->clear |= ((1U << irq) & IRQMP_CLEAR_MASK);\n}\n\nunsigned int irqmp_get_nbr_cpus(const irqmp_t *u)\n{\n\t/* Number of CPUs is specified in the status register */\n\treturn ((u->status & IRQMP_STATUS_NCPU_MASK) >> IRQMP_STATUS_NCPU_OFFSET)\n\t\t\t+ 1;\n}\n\nunsigned int irqmp_get_eirq_id(const irqmp_t *u, unsigned int cpu)\n{\n\treturn u->proc_ext_ack[cpu] & 0x1FU;\n}\n\nint irqmp_is_broadcast_available(const irqmp_t *u)\n{\n\t/* Broadcast available is specified in the status register */\n\tif (u->status & IRQMP_STATUS_BA_MASK) {\n\t\treturn 1;\n\t}\n\n\treturn 0;\n}\n\nunsigned int irqmp_get_extended_irq_nbr(const irqmp_t *u)\n{\n\t/* Extended IRQ number is specified in the status register */\n\treturn (u->status & IRQMP_STATUS_EIRQ_MASK) >> IRQMP_STATUS_EIRQ_OFFSET;\n}\n\nint irqmp_cpu_is_power_down(const irqmp_t *u, unsigned int cpu)\n{\n\t/* CPU state is specified in the status register */\n\tif (u->status & (1U << cpu)) {\n\t\treturn 1;\n\t}\n\n\treturn 0;\n}\n\nint irqmp_cpu_is_runnig(const irqmp_t *u, unsigned int cpu)\n{\n\t/* CPU state is specified in the status register */\n\tif (u->status & (1U << cpu)) {\n\t\treturn 0;\n\t}\n\n\treturn 1;\n}\n\nvoid irqmp_cpu_start(irqmp_t *u, unsigned int cpu)\n{\n\t/* Set corresponding bit in status register */\n\tu->status |= (1U << cpu);\n}\n\nvoid irqmp_set_broadcast(irqmp_t *u, unsigned int irq)\n{\n\t/* Set corresponding bit in broadcast register */\n\tu->broadcast |= ((1U << irq) & IRQMP_BCAST_MASK);\n}\n\nvoid irqmp_set_no_broadcast(irqmp_t *u, unsigned int irq)\n{\n\t/* Clear corresponding bit in broadcast register */\n\tu->broadcast &= ~((1U << irq) & IRQMP_BCAST_MASK);\n}\n\nvoid irqmp_enable_irq(irqmp_t *u, unsigned int cpu, unsigned int irq)\n{\n\tif (cpu < IRQMP_MAX_SUPPORTED_PROCS) {\n\t\t/* Set corresponding bit in proc mask register */\n\t\tu->proc_mask[cpu] |= ((1U << irq) & IRQMP_PROC_IRQ_MASK_MASK);\n\t}\n}\n\nvoid irqmp_disable_irq(irqmp_t *u, unsigned int cpu, unsigned int irq)\n{\n\tif (cpu < IRQMP_MAX_SUPPORTED_PROCS) {\n\t\t/* Clear corresponding bit in proc mask register */\n\t\tu->proc_mask[cpu] &= ~((1U << irq) & IRQMP_PROC_IRQ_MASK_MASK);\n\t}\n}\n\n"
}
```



```
[
    {
        "rule_id": "c:S813",
        "rule_description": "Replace this use of \"unsigned int\" with a typedef.",
        "lines": [
            58,
            64,
            70,
            76,
            90,
            96,
            103,
            108,
            118,
            124,
            134,
            144,
            150,
            156,
            162,
            170
        ]
    },
    {
        "rule_id": "c:S109",
        "rule_description": "Assign this magic number 0x1FU to a well-named variable, and use the variable instead.",
        "lines": [
            105
        ]
    }
]
```

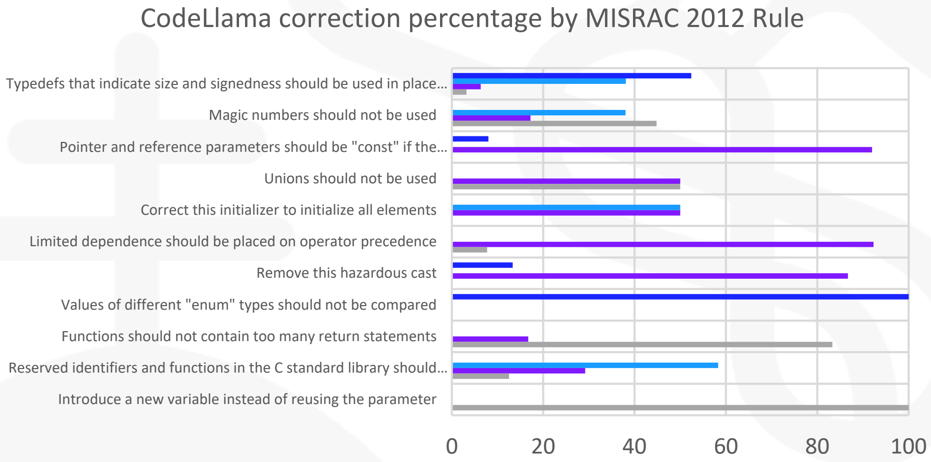

- **Output dataset:** JSON structure with rule description and fixed code (source code and header file)

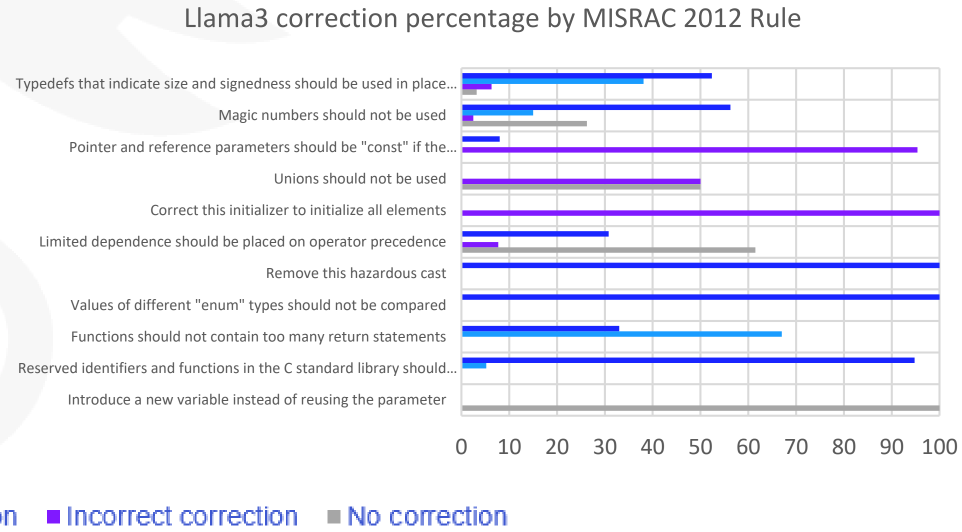
ThalesAlenia Space — A Thales / Finmeccanica Company

Universidad de Alcalá

Experiments

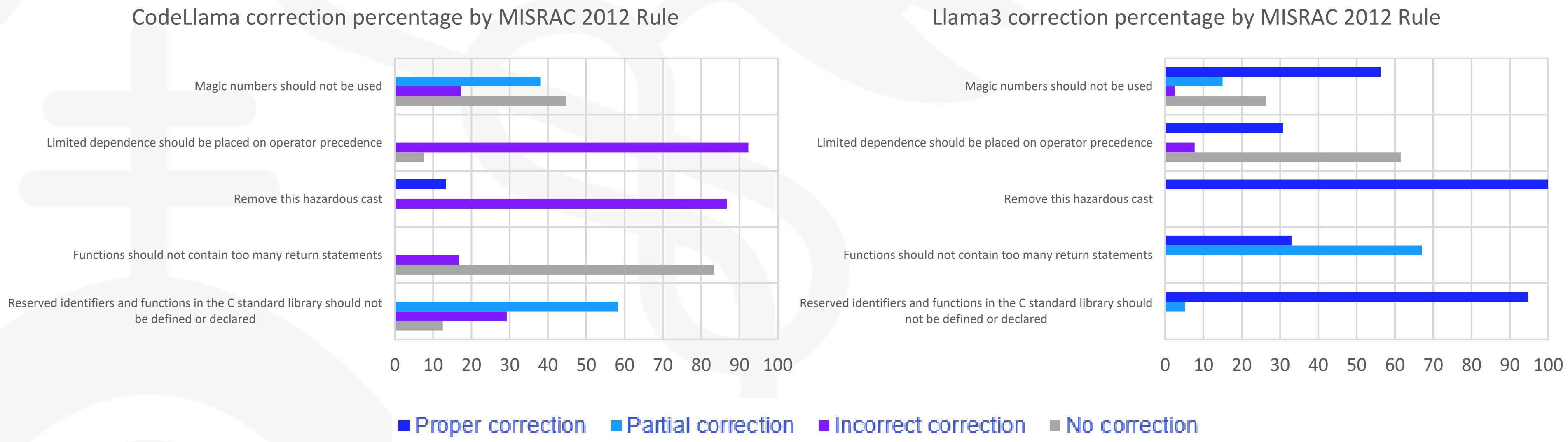
- Precision metrics
  - Results broken down for each rule: comparison CodeLlama vs Llama3
    - Variation in correction effectiveness
      - CodeLlama showed that the model's ability to handle different rules varied considerably.
      - Llama3 increased the proper and partial corrections but still shows variability in its correction ability





Experiments

- Precision metrics
  - Results broken down for each rule: comparison CodeLlama vs Llama3
    - Variation in correction effectiveness
      - CodeLlama showed that the model's ability to handle different rules varied considerably.
      - Llama3 increased the proper and partial corrections but still shows variability in its correction ability

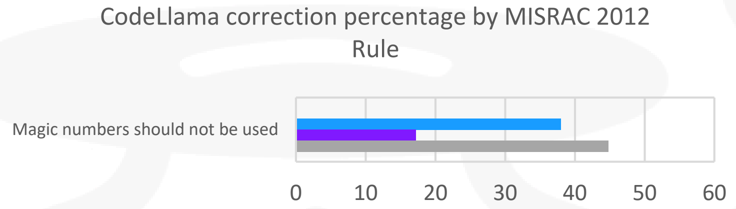


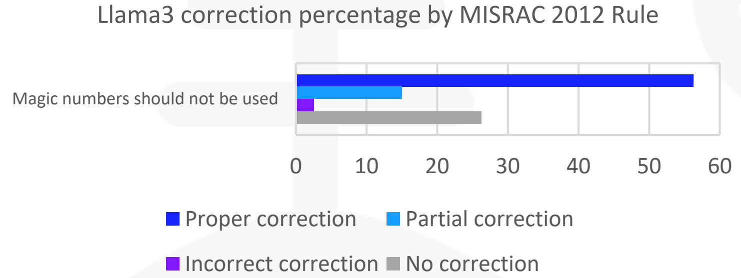
{OPEN}

Experiments

- Precision metrics: Variation in correction effectiveness
  - Improvement of Llama3 wrt CodeLlama

  Example: "Magic numbers should not be used"





CodeLlama partial solution

```c
#include <stddef.h>
#include <memctrl.h>
#include <01_054_ftmctrl.h>
#include <01_054_ftmctrl_hw_desc.h>
#include <01_054_ftmctrl_prom_io.h>
#include <01_054_ftmctrl_ram.h>
#include <01_054_ftmctrl_edac.h>
#include <01_054_ftmctrl_rs.h>

static int32_t ftmctrl_init_prom(ftmctrl_t *u, const mctrl_bus_t *bus)
{
    int32_t ret = 0;

    /* Configure bus width */
    switch (bus->width) {
    case PROM_WIDTH_8:
        /* Sets the data width of the PROM area ("00"=8) */
        ftmctrl_prom_set_width_8(u);
        break;
    case PROM_WIDTH_16:
        /* Sets the data width of the PROM area ("01"=16) */
        ftmctrl_prom_set_width_16(u);
        break;
    case PROM_WIDTH_32:
        /* Sets the data width of the PROM area ("10"=32) */
        ftmctrl_prom_set_width_32(u);
        break;
    default:
        ret = -1;
        break;
    }
```

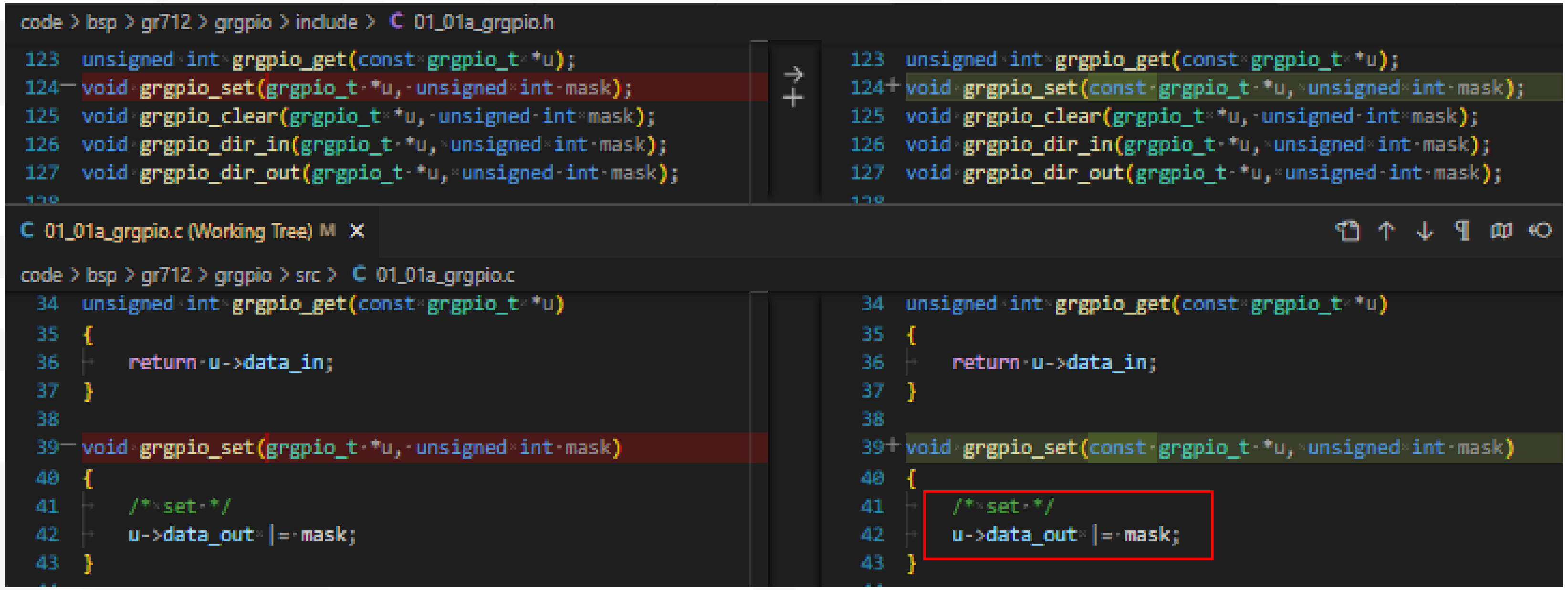
Llama3 correct solution

```c
#include <stddef.h>
#include <memctrl.h>
#include <01_054_ftmctrl.h>
#include <01_054_ftmctrl_hw_desc.h>
#include <01_054_ftmctrl_prom_io.h>
#include <01_054_ftmctrl_ram.h>
#include <01_054_ftmctrl_edac.h>
#include <01_054_ftmctrl_rs.h>

#define PROM_WIDTH_8     8U
#define PROM_WIDTH_16    16U
#define PROM_WIDTH_32    32U
static int32_t ftmctrl_init_prom(ftmctrl_t *u, const mctrl_bus_t *bus)
{
    int32_t ret = 0;

    /* Configure bus width */
    switch (bus->width) {
    case PROM_WIDTH_8:
        /* Sets the data width of the PROM area ("00"=8) */
        ftmctrl_prom_set_width_8(u);
        break;
    case PROM_WIDTH_16:
        /* Sets the data width of the PROM area ("01"=16) */
        ftmctrl_prom_set_width_16(u);
        break;
    case PROM_WIDTH_32:
        /* Sets the data width of the PROM area ("10"=32) */
        ftmctrl_prom_set_width_32(u);
        break;
    default:
        ret = -1;
        break;
    }
```

Experiments

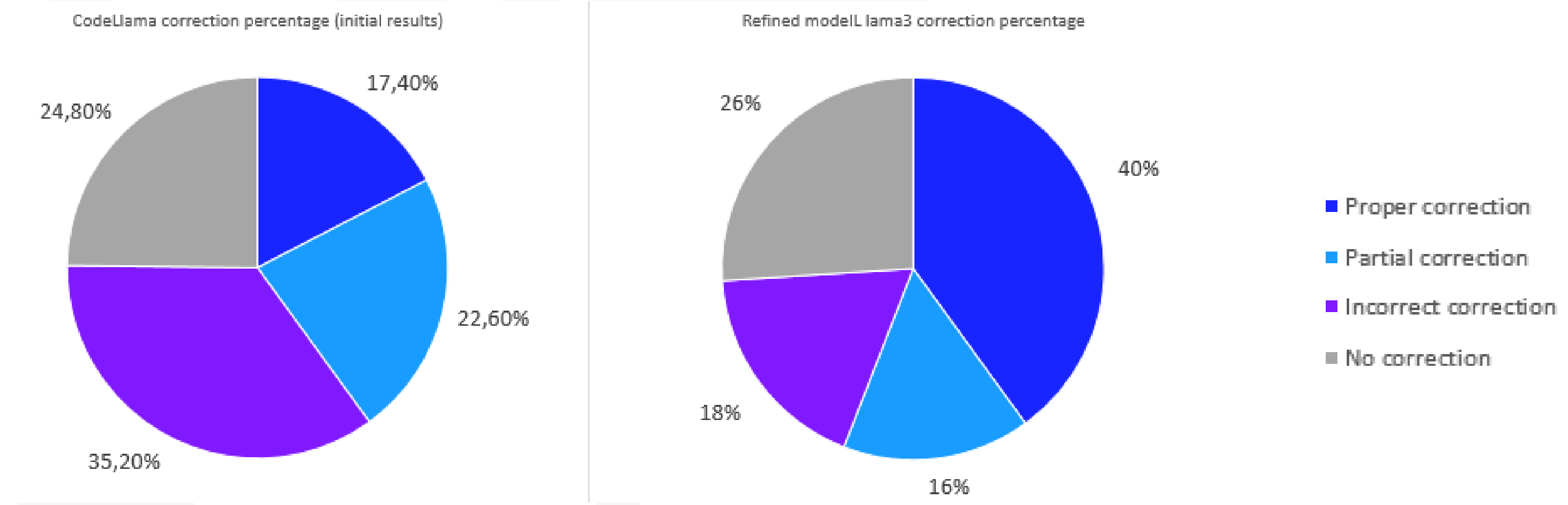
- Precision metrics: False positive and code context
  - Still a weakness in the refined model

  Example:"Pointer and reference parameters should be 'const' if the corresponding object is not modified": 92% of corrections were incorrect.



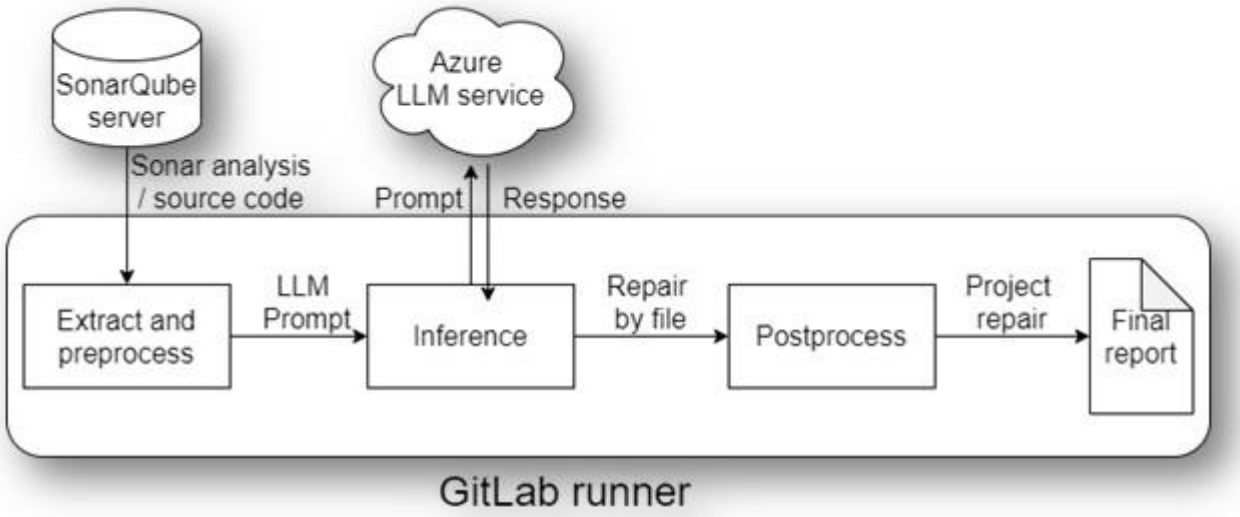
Experiments

- Precision metrics: overall results
  - Fully accurate corrections: increased up to 40% --> reduces manual workload and saves developer's time
  - Partial corrections: slightly decrease to 15,9% --> provides useful insights to guide developers in resolving complex issues
  - Incorrect corrections: dropped to 18,1% --> increased model reliability and less time spent reviewing errors

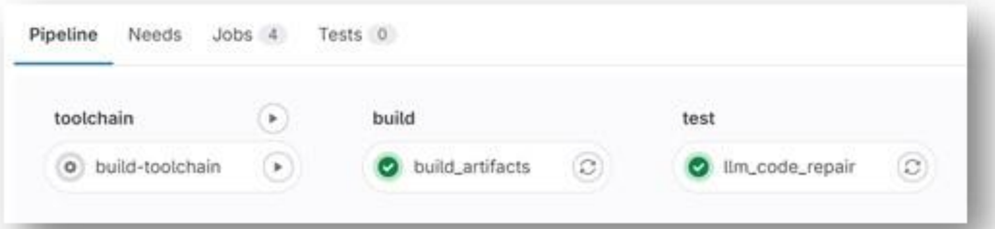


{OPEN}

Final Deployment



GitLab



{OPEN}

Conclusions

- Still not a full replacement for developers
  - Despite its improvements, the model cannot yet replace developers entirely. Human oversight is still essential for more intricate, context-specific coding decisions and fine-tuning.
- Better code quality and compliance
  - The system's improved alignment with coding standards like MISRA ensures that the code is not only more reliable but also compliant with industry regulations. This contributes to long-term maintainability and reduces the risk of defects.
- Increased confidence in automated suggestions
  - The sharp decrease in incorrect corrections means developers can rely more on the model's suggestions, reducing time spent double-checking or correcting faulty outputs, thus streamlining the entire review process.
- Support for complex issue resolution
  - While full corrections have increased, the value of partial corrections lies in guiding developers through complex or edge-case issues, providing a foundation upon which they can make finer adjustments with confidence.