ICML
International Conference
On Machine Learning

Efficient Skill Grounding via Code Refactoring with Small Language Models

Sera Choi[1], Wonje Choi[1], Saehun Chun[1], Daehee Lee[1], Jooyoung Kim[1], Chaeun Lee[2], Honguk Woo[1]

[1]Dept. of Computer Science and Engineering · [2]Dept. of Systems Management Engineering, Sungkyunkwan University

SUNGKYUNKWAN UNIVERSITY
1398
성균관대학교

1. Introduction

Problem

- Existing skill representations do not **explicitly separate functional semantics from executable components**, incurring substantial **computational overhead** when deployment conditions change.
- Under deployment constraints, LLMs are impractical, while sLMs struggle with deployment-time skill grounding that requires **retraining the model** or **full skill regeneration**.

Key Idea

- Represent skills as **executable code**, structurally **separating semantic intent from execution bindings**.
- Ground skills via **localized code refactoring**, enabling inference-only grounding without retraining.

Contributions

- **RECENT framework:** Refactoring-centric agent framework for efficient skill grounding with sLMs.
- **Ontology-based reasoning:** Pre-execution grounding for embodiment gaps.
- **In-situ adaptation:** Execution-time patching for environmental variations.

2. Approach

Non-refactoring-centric vs. Refactoring-centric

- Existing CaP rewrites the entire skill on every mismatch → ~3k tokens per task.
- **RECENT preserves the skill structure and patches only execution bindings → ~1.1k tokens per task.**

Refactoring-centric Skill Grounding

- Embodiment gaps → **ontology-guided pre-execution editing** (API / parameter substitution).
- Environmental variations → **in-situ adaptation**, patching yet-to-be-executed code at runtime.

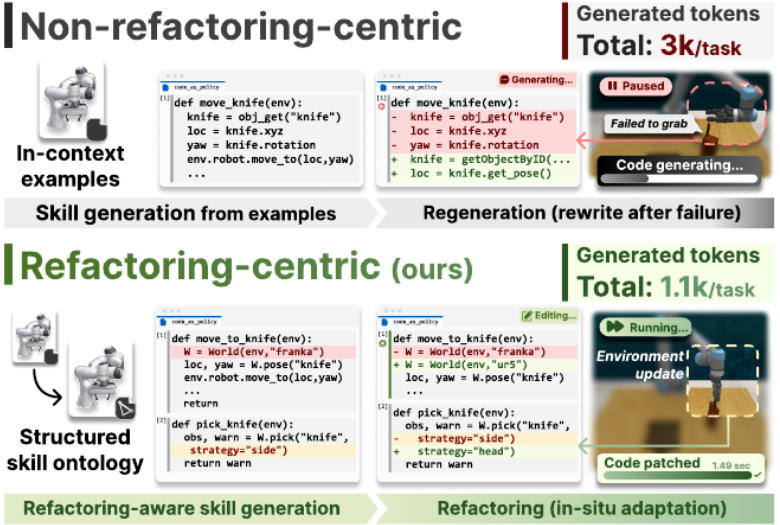

Figure 1. Existing skill grounding (top) and our refactoring-centric grounding (bottom).

3. RECENT: Refactoring-centric Skill Grounding

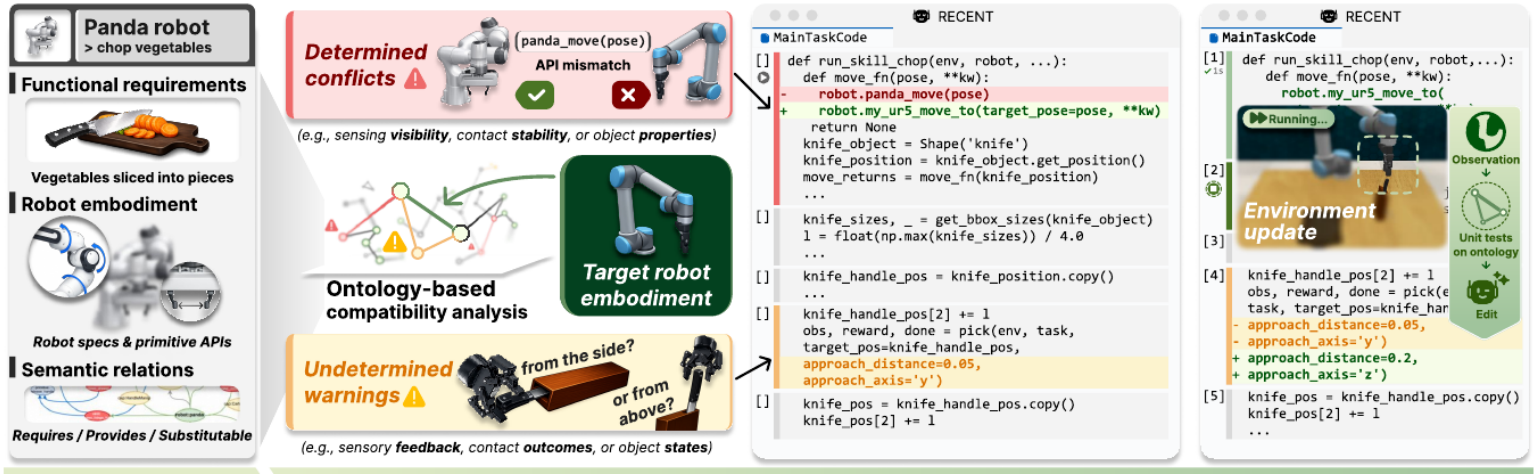

Figure 2. Overview of RECENT: (i) offline skill repository, (ii) ontology-based reasoning, (iii) in-situ adaptation.

(i) Offline skill repository

Reusable skill code is generated using an LLM based on a skill ontology and validated on a reference robot to support sLM-based deployment-time grounding.

(ii) Ontology-based reasoning

Ontology-guided structured reasoning identifies localized edit targets for determined conflicts, enabling an sLM to resolve them before execution through FIM-based edits.

(iii) In-situ adaptation

Undetermined warnings are detected by runtime unit tests and resolved by FIM-based localized patching, enabling uninterrupted execution.

4. Evaluation Settings

Skill Grounding Scenarios

- **Kinematic variations:**
  Panda → UR5, Sawyer (CoppeliaSim)
- **End-effector variations:**
  parallel-jaw → 2F-85, vacuum gripper (Genesis)
- All scenarios add dynamic, partially observable conditions (e.g., object pose, uncertainty, sensing noise)

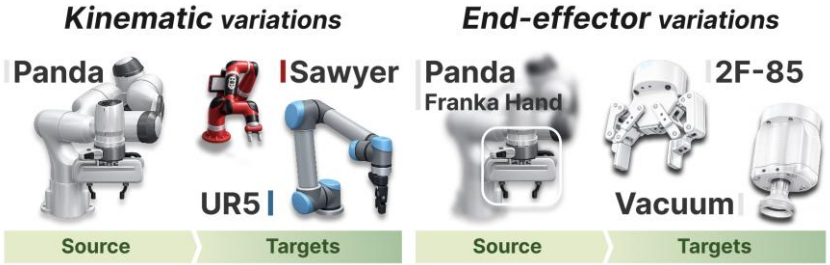

Figure 3. Evaluation settings: kinematic (left) and end-effector (right) variations.

Tasks, Baselines & Metrics

- Long-horizon tasks: up to **54 sequential API calls** each.
- **sLM baselines** (Qwen2.5-Coder-7B): CaP variants, Embodied code agents, Program repair agents
- **LLM-based upper bound** (GPT-5.2-Codex): CaP-Codex
- **Metrics:** SR, GC (↑); GO, EI, IT (↓)

5. Evaluation Performance

Experiment results

- RECENT achieves the best results among sLM-based methods and matches LLM-based CaP-Codex.
- Compared with the distilled sLM-based CaP baseline, RECENT improves SR by **+58.8 pp**.
- Grounding overhead is reduced by **99.1 pp**, showing the efficiency of localized code refactoring.
- Compared with LLM-based CaP-Codex, RECENT shows only **6.6 pp** lower SR under deployment constraints.

Method	SR (%) ↑	GC (%) ↑	GO (%) ↓	EI (count) ↓	IT (s) ↓
CaP (sLM)	19.5	38.7	90.7	3.95	40.6
CaP-CodeV-R1 (distilled sLM)	19.0	38.6	104.3	3.45	30.6
RepairAgent (strongest baseline)	47.1	69.0	55.0	1.43	67.5
RECENT (ours, sLM)	**77.8**	**85.8**	**5.2**	**0.71**	**2.1**
CaP-Codex (LLM, GPT-5.2-Codex)	*71.2*	*84.1*	*63.0*	*0.92*	*9.1*

Table 1. Performance averaged over Kinematic and End-effector variations (5 seeds). ↑ higher is better, ↓ lower is better.

6. Conclusion

A Refactoring-centric Framework: RECENT

- Represents skills as executable code that separates invariant semantic intent from embodiment- and environment-specific execution bindings.
- Enables sLMs to ground skills through localized code refactoring rather than full regeneration at deployment time.

Takeaways

- Significantly outperforms distilled sLM-based CaP in success rate, grounding efficiency, and execution stability.
- Matches LLM-based CaP performance while operating under on-device deployment constraints.

Future Direction

- Extend RECENT to ontology-level capability expansion and persistent repository evolution for lifelong skill learning across evolving embodiments.