Generation of Robot Manipulation Plans Using Generative Large Language Models

7^{th} IEEE Conference on Robotic Computing 2023

Jan-Philipp Töberg & Philipp Cimiano
jtoeberg@techfak.uni-bielefeld.de

CITEC & CoAI JRC
Bielefeld University

December 12, 2023

Outline

Robots in Open World Situations

Goal

Enable cognitive robots to perform household tasks in varying situations

Robots in Open World Situations

Goal

Enable cognitive robots to perform household tasks in varying situations

Problem

Households are *open worlds* ⇒ unknown environments containing unknown objects and necessitating unknown actions

Robots in Open World Situations

Goal

Enable cognitive robots to perform household tasks in varying situations

Problem

Households are *open worlds* $\Rightarrow$ unknown environments containing unknown objects and necessitating unknown actions

Solutions [Din+23]

1. Acquiring knowledge via human-robot interaction
2. Dynamically building a knowledge base to assist a task planner
3. **Use generative Large Language Models (LLMs) as a task planner**

Generating Manipulation Plans

- Generation of manipulation plans is also done in related work (e.g. *Code as Policies* [Lia+23] & *ProgPrompt* [Sin+23]):

Generating Manipulation Plans

- Generation of manipulation plans is also done in related work (e.g. *Code as Policies* [Lia+23] & *ProgPrompt* [Sin+23]):
  - Plans are written in Python
  - Generation uses one-shot prompting
  - Interaction with the robot through manually developed framework
  - Promising results in simulation and on the real robot

Generating Manipulation Plans

- Generation of manipulation plans is also done in related work (e.g. *Code as Policies* [Lia+23] & *ProgPrompt* [Sin+23]):
  - Plans are written in Python
  - Generation uses one-shot prompting
  - Interaction with the robot through manually developed framework
  - Promising results in simulation and on the real robot
- Open questions:
  - How do LLMs perform on cognitive architectures?

Generating Manipulation Plans

- Generation of manipulation plans is also done in related work (e.g. *Code as Policies* [Lia+23] & *ProgPrompt* [Sin+23]):
  - Plans are written in Python
  - Generation uses one-shot prompting
  - Interaction with the robot through manually developed framework
  - Promising results in simulation and on the real robot
- Open questions:
  - How do LLMs perform on cognitive architectures?
  - How does the example in the prompt influence performance?

Cognitive Robot Abstract Machine (CRAM) [BMT10]

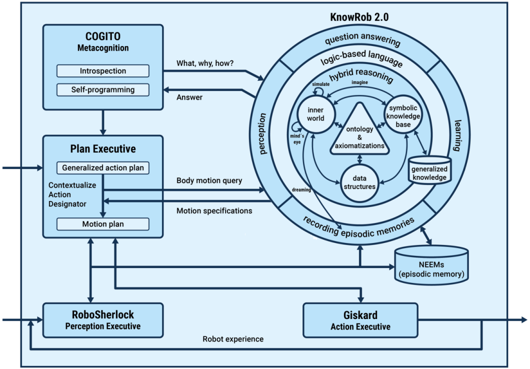


Figure: CRAM Architecture from [Ver+22]

- Hybrid cognitive architecture for autonomous robots

Cognitive Robot Abstract Machine (CRAM) [BMT10]

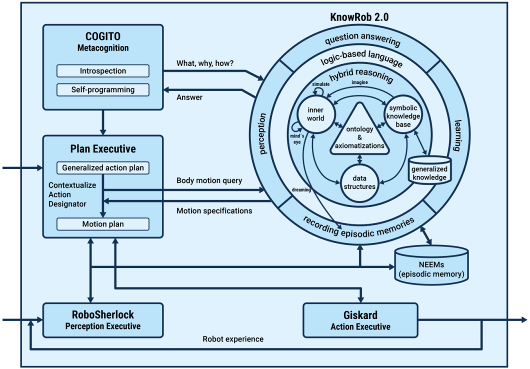


Figure: CRAM Architecture from [Ver+22]

- Hybrid cognitive architecture for autonomous robots
- Tasks are vaguely described as high-level goals from which specific low-level motions are derived

Cognitive Robot Abstract Machine (CRAM) [BMT10]

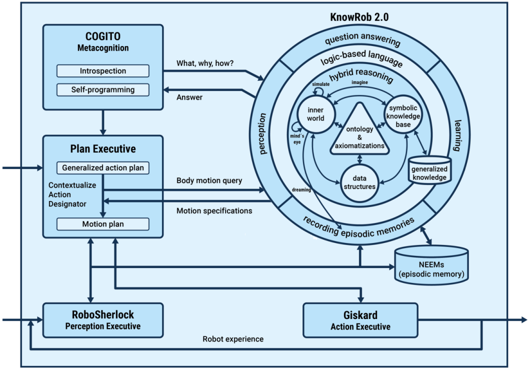


Figure: CRAM Architecture from [Ver+22]

- Hybrid cognitive architecture for autonomous robots
- Tasks are vaguely described as high-level goals from which specific low-level motions are derived
- Plans are written in Common LISP and are called *designators*

Experiment Setup

- 9 designators written by experts for 9 different actions (*Close, Halve, Hold, Open, Pick Up, Place Down, Pour, Slice & Wipe*)

Experiment Setup

- 9 designators written by experts for 9 different actions (*Close*, *Halve*, *Hold*, *Open*, *Pick Up*, *Place Down*, *Pour*, *Slice* & *Wipe*)
- For each action, use each other action as a reference → 72 results

Experiment Setup

- 9 designators written by experts for 9 different actions (*Close, Halve, Hold, Open, Pick Up, Place Down, Pour, Slice & Wipe*)
- For each action, use each other action as a reference → 72 results
- Repeat the experiment 5 times per model → 360 generated designators per model

Experiment Setup

- 9 designators written by experts for 9 different actions (*Close, Halve, Hold, Open, Pick Up, Place Down, Pour, Slice & Wipe*)
- For each action, use each other action as a reference → 72 results
- Repeat the experiment 5 times per model → 360 generated designators per model
- Use 3 different models / model versions:
  - `gpt-3.5-turbo-0301`
  - `gpt-3.5-turbo-0613`
  - `gpt-4-0613`

Prompt

The following LISP source code describes a CRAM designator for the action of "*[reference action]*", where the executing robot would be *[reference action description]*:
[reference designator]
Can you please take this example and create a new designator for the action "*[target action]*", where the robot should be *[target action description]*. Your answer should only include the designator and no additional text.

Prompt

The following LISP source code describes a CRAM designator for the action of "*[reference action]*", where the executing robot would be *[reference action description]*:
[reference designator]
Can you please take this example and create a new designator for the action "*[target action]*", where the robot should be *[target action description]*. Your answer should only include the designator and no additional text.

Example Descriptions

Close → Closing an arbitrary container
Pour → Pouring the content of one container into another
Wipe → Cleaning a surface using some kind of towel

Code Generation Metrics

- Machine Translation Metrics:
  - BLEU [Pap+01]
  - ROUGE-1 (R-1), ROUGE-2 (R-2) & ROUGE-L (R-L) [Lin04]
  - chrF [Pop15]

Code Generation Metrics

- Machine Translation Metrics:
  - BLEU [Pap+01]
  - ROUGE-1 (R-1), ROUGE-2 (R-2) & ROUGE-L (R-L) [Lin04]
  - chrF [Pop15]
- Code Generation Quality Metrics:
  - CodeBERTScore (CBS) [Zho+23]

Code Generation Metrics

- Machine Translation Metrics:
  - BLEU [Pap+01]
  - ROUGE-1 (R-1), ROUGE-2 (R-2) & ROUGE-L (R-L) [Lin04]
  - chrF [Pop15]
- Code Generation Quality Metrics:
  - CodeBERTScore (CBS) [Zho+23]
- Compilation Success

Action Similarity Metrics

- Wu-Palmer-Similarity (WuP) [WP94] between WordNet synsets [Mil95]

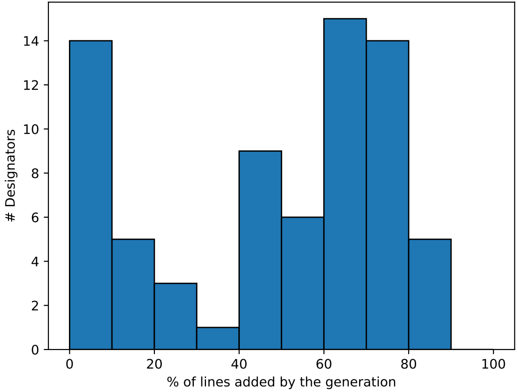
Action Similarity Metrics

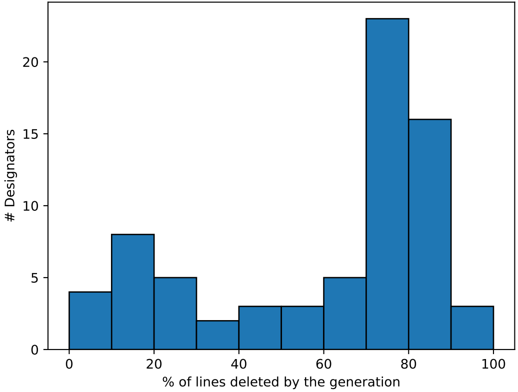
- Wu-Palmer-Similarity (WuP) [WP94] between WordNet synsets [Mil95]
- Cosine Similarity between GloVe embeddings [PSM14]

Action Similarity Metrics

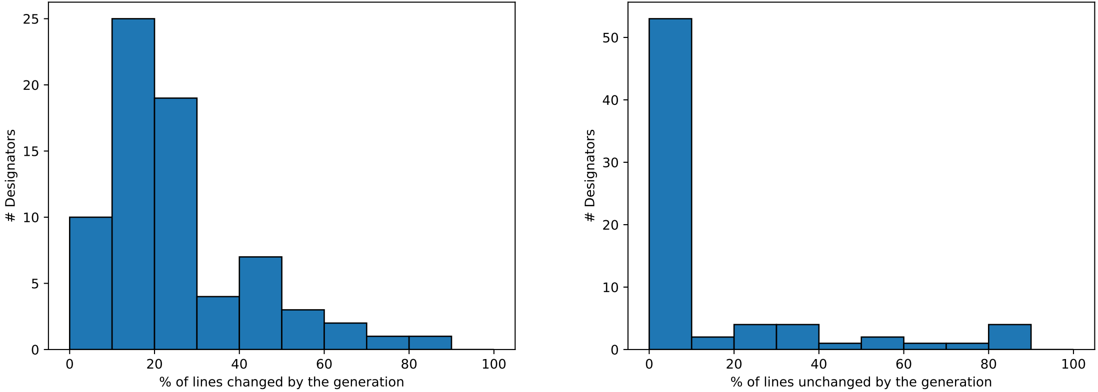
- Wu-Palmer-Similarity (WuP) [WP94] between WordNet synsets [Mil95]
- Cosine Similarity between GloVe embeddings [PSM14]
- Sensorimotor Distance (SMD) [WC22]

Manual Analysis - Added & Deleted Lines





Manual Analysis - (Un-)Changed Lines



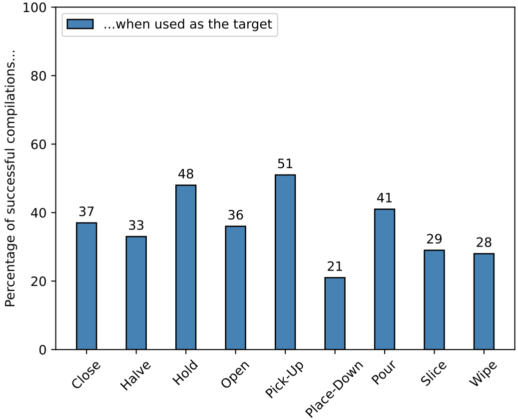
Metrics

Model	BLEU	R-1	R-2	R-L	chrF	CBS
`gpt-3.5-turbo-0301`	.595	.630	.527	.621	**.674**	.942
`gpt-3.5-turbo-0613`	.579	.614	.511	.612	.639	.940
`gpt-4-0613`	**.605**	**.631**	**.532**	**.623**	**.674**	**.945**

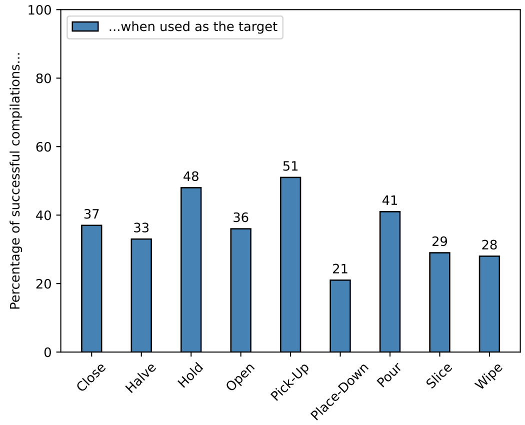
Compilation Success

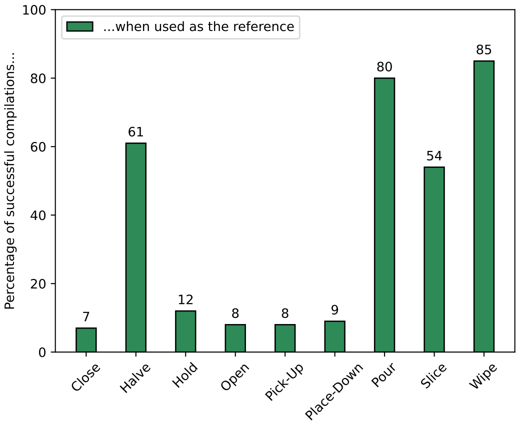
Model	Compiles	¬Compiles
`gpt-3.5-turbo-0301`	147 / 40,83%	213 / 59,17%
`gpt-3.5-turbo-0613`	101 / 28,06%	259 / 71,94%
`gpt-4-0613`	139 / 38,61%	221 / 61,39%
Σ	387 / 35,83%	693 / 64,17%

Compilation Success per Action



Compilation Success per Action





Action Similarity and Generation Correlation

Hypothesis

The **higher** the similarity between the reference and the target action, the **better** the generated designator

Action Similarity and Generation Correlation

Hypothesis

The **higher** the similarity between the reference and the target action, the **better** the generated designator

Testing

Calculate the Spearman Rank Correlation ρ between the code generation metrics and the action similarity metrics
Expectation: positive, significant ($p \leq 0.05$) correlations

Action Similarity and Generation Quality Correlation

Metric	WuP [WP94] ρ	WuP [WP94] p	GloVe [PSM14] ρ	GloVe [PSM14] p	SMD [WC22] ρ	SMD [WC22] p
BLEU [Pap+01]	**-.248**	**.000**	**-.282**	**.000**	**-.200**	**.000**
ROUGE-1 [Lin04]	-.086	.104	**-.270**	**.000**	**-.355**	**.000**
ROUGE-2 [Lin04]	**-.141**	**.008**	**-.264**	**.000**	**-.395**	**.000**
ROUGE-L [Lin04]	-.082	.122	**-.264**	**.000**	**-.358**	**.000**
chrF [Pop15]	**-.188**	**.000**	**-.296**	**.000**	**-.241**	**.000**
CBS [Zho+23]	-.101	.056	**-.215**	**.000**	**-.279**	**000**
Lines of Code	**-.287**	**.000**	**-.336**	**.000**	**-.204**	**.000**
Comp. Succ.	**-.278**	**.000**	**-.166**	**.002**	.007	.898

Action Similarity and Generation Quality Correlation

Metric	WuP [WP94] ρ	WuP [WP94] p	GloVe [PSM14] ρ	GloVe [PSM14] p	SMD [WC22] ρ	SMD [WC22] p
BLEU [Pap+01]	**-.248**	**.000**	**-.282**	**.000**	**-.200**	**.000**
ROUGE-1 [Lin04]	-.086	.104	**-.270**	**.000**	**-.355**	**.000**
ROUGE-2 [Lin04]	**-.141**	**.008**	**-.264**	**.000**	**-.395**	**.000**
ROUGE-L [Lin04]	-.082	.122	**-.264**	**.000**	**-.358**	**.000**
chrF [Pop15]	**-.188**	**.000**	**-.296**	**.000**	**-.241**	**.000**
CBS [Zho+23]	-.101	.056	**-.215**	**.000**	**-.279**	**000**
Lines of Code	**-.287**	**.000**	**-.336**	**.000**	**-.204**	**.000**
Comp. Succ.	**-.278**	**.000**	**-.166**	**.002**	.007	.898

`gpt-3.5-turbo-0301`

$\Rightarrow$ All significant correlations are negative ($n = 360$)

Action Similarity and Generation Quality Correlation

Metric	WuP [WP94] ρ	WuP [WP94] p	GloVe [PSM14] ρ	GloVe [PSM14] p	SMD [WC22] ρ	SMD [WC22] p
BLEU [Pap+01]	**.117**	**.026**	**.235**	**.000**	-.013	.799
ROUGE-1 [Lin04]	**.117**	**.026**	**.183**	**.000**	-.015	.773
ROUGE-2 [Lin04]	.103	.051	**.156**	**.003**	-.010	.856
ROUGE-L [Lin04]	**.115**	**.029**	**.183**	**.000**	-.017	.748
chrF [Pop15]	**.114**	**.030**	**.194**	**.000**	-.023	.668
CBS [Zho+23]	.079	.133	**.153**	**.004**	.010	.846
Lines of Code	-.008	.880	.093	.078	-.034	.517
Comp. Succ.	**-.105**	**.047**	**-.129**	**.014**	-.084	.111

Action Similarity and Generation Quality Correlation

Metric	WuP [WP94] ρ	WuP [WP94] p	GloVe [PSM14] ρ	GloVe [PSM14] p	SMD [WC22] ρ	SMD [WC22] p
BLEU [Pap+01]	**.117**	**.026**	**.235**	**.000**	-.013	.799
ROUGE-1 [Lin04]	**.117**	**.026**	**.183**	**.000**	-.015	.773
ROUGE-2 [Lin04]	.103	.051	**.156**	**.003**	-.010	.856
ROUGE-L [Lin04]	**.115**	**.029**	**.183**	**.000**	-.017	.748
chrF [Pop15]	**.114**	**.030**	**.194**	**.000**	-.023	.668
CBS [Zho+23]	.079	.133	**.153**	**.004**	.010	.846
Lines of Code	-.008	.880	.093	.078	-.034	.517
Comp. Succ.	**-.105**	**.047**	**-.129**	**.014**	-.084	.111

`gpt-3.5-turbo-0613`

$\Rightarrow$ All significant correlations (except for compilation success) are positive ($n = 360$)

Action Similarity and Generation Quality Correlation

	WuP [WP94]		GloVe [PSM14]		SMD [WC22]	
Metric	ρ	p	ρ	p	ρ	p
BLEU [Pap+01]	-.061	.250	**-.133**	**.011**	**-.104**	**.050**
ROUGE-1 [Lin04]	.039	.464	**-.126**	**.017**	**-.240**	**.000**
ROUGE-2 [Lin04]	.013	.803	**-.115**	**.029**	**-.256**	**.000**
ROUGE-L [Lin04]	.039	.464	**-.118**	**.026**	**-.246**	**.000**
chrF [Pop15]	-.012	.822	**-.136**	**.010**	**-.142**	**.007**
CBS [Zho+23]	-.001	.992	**-.127**	**.016**	**-.203**	**.000**
Lines of Code	**-.146**	**.006**	**-.256**	**.000**	**-.166**	**.002**
Comp. Succ.	**-.195**	**.000**	-.037	.483	.019	.712

Action Similarity and Generation Quality Correlation

Metric	WuP [WP94] ρ	WuP [WP94] p	GloVe [PSM14] ρ	GloVe [PSM14] p	SMD [WC22] ρ	SMD [WC22] p
BLEU [Pap+01]	-.061	.250	**-.133**	**.011**	**-.104**	**.050**
ROUGE-1 [Lin04]	.039	.464	**-.126**	**.017**	**-.240**	**.000**
ROUGE-2 [Lin04]	.013	.803	**-.115**	**.029**	**-.256**	**.000**
ROUGE-L [Lin04]	.039	.464	**-.118**	**.026**	**-.246**	**.000**
chrF [Pop15]	-.012	.822	**-.136**	**.010**	**-.142**	**.007**
CBS [Zho+23]	-.001	.992	**-.127**	**.016**	**-.203**	**.000**
Lines of Code	**-.146**	**.006**	**-.256**	**.000**	**-.166**	**.002**
Comp. Succ.	**-.195**	**.000**	-.037	.483	.019	.712

gpt-4-0613

$\Rightarrow$ All significant correlations are negative ($n = 360$)

Action Similarity and Generation Quality Correlation

Hypothesis

The **higher** the similarity between the reference and the target action, the **better** the generated designator

Action Similarity and Generation Quality Correlation

Hypothesis

The **higher** the similarity between the reference and the target action, the **better** the generated designator

Results

- Action similarity negatively correlates with generation quality in the old ChatGPT and the GPT-4 model
- Action similarity positively correlates with generation quality in the new ChatGPT model
- significant correlations with the compilation success are always negative

$\Rightarrow$ Using a similar action as a reference **decreases** the chance of compiling successfully

Limitations

- Most metrics (BLEU, chrF & ROUGE) are for evaluating machine translation tasks

Limitations

- Most metrics (BLEU, chrF & ROUGE) are for evaluating machine translation tasks
- CodeBERTScore is not optimised for Common LISP

Limitations

- Most metrics (BLEU, chrF & ROUGE) are for evaluating machine translation tasks
- CodeBERTScore is not optimised for Common LISP
- Sensorimotor Distance is susceptible to semantic inaccuracy

Limitations

- Most metrics (BLEU, chrF & ROUGE) are for evaluating machine translation tasks
- CodeBERTScore is not optimised for Common LISP
- Sensorimotor Distance is susceptible to semantic inaccuracy
- No fine-tuning of the models due to limited sample size ($n = 9$)

Conclusion

How do LLMs perform on cognitive architectures (CRAM)?

- All analysed LLMs achieve solid results on the code generation metrics

Conclusion

How do LLMs perform on cognitive architectures (CRAM)?

- All analysed LLMs achieve solid results on the code generation metrics
- GPT-4 slightly outperforms ChatGPT

Conclusion

How do LLMs perform on cognitive architectures (CRAM)?

- All analysed LLMs achieve solid results on the code generation metrics
- GPT-4 slightly outperforms ChatGPT
- Only $\sim$ 36% of designators compiled successfully

Conclusion

How do LLMs perform on cognitive architectures (CRAM)?

- All analysed LLMs achieve solid results on the code generation metrics
- GPT-4 slightly outperforms ChatGPT
- Only $\sim$ 36% of designators compiled successfully
- The older ChatGPT version achieves the highest compilation success rate

Conclusion

How do LLMs perform on cognitive architectures (CRAM)?

- All analysed LLMs achieve solid results on the code generation metrics
- GPT-4 slightly outperforms ChatGPT
- Only $\sim$ 36% of designators compiled successfully
- The older ChatGPT version achieves the highest compilation success rate

How does the example in the prompt influence performance?

- Action similarity negatively influences compilation success rate

Conclusion

How do LLMs perform on cognitive architectures (CRAM)?

- All analysed LLMs achieve solid results on the code generation metrics
- GPT-4 slightly outperforms ChatGPT
- Only $\sim$ 36% of designators compiled successfully
- The older ChatGPT version achieves the highest compilation success rate

How does the example in the prompt influence performance?

- Action similarity negatively influences compilation success rate
- Action similarity negatively influences code generation quality for GPT-4 & Old ChatGPT

Conclusion

How do LLMs perform on cognitive architectures (CRAM)?

- All analysed LLMs achieve solid results on the code generation metrics
- GPT-4 slightly outperforms ChatGPT
- Only $\sim$ 36% of designators compiled successfully
- The older ChatGPT version achieves the highest compilation success rate

How does the example in the prompt influence performance?

- Action similarity negatively influences compilation success rate
- Action similarity negatively influences code generation quality for GPT-4 & Old ChatGPT
- Action similarity positively influences code generation quality for new ChatGPT

Future Work

- Use other generative LLMs that focus on source code generation

Future Work

- Use other generative LLMs that focus on source code generation
- Repeat the experiment for Python version of CRAM (PyCRAM [Dec+23])

Future Work

- Use other generative LLMs that focus on source code generation
- Repeat the experiment for Python version of CRAM (PyCRAM [Dec+23])
- Generate PyCRAM designators from CRAM designators

Future Work

- Use other generative LLMs that focus on source code generation
- Repeat the experiment for Python version of CRAM (PyCRAM [Dec+23])
- Generate PyCRAM designators from CRAM designators
- Simulate the successfully compiled designators

Thank you for your attention!

Questions?

References I

[BMT10] Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. "CRAM - A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments". In: *Proceedings of the 2nd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*. Ed. by Ren C. Luo and Hajime Asama. Taipei, Taiwan: IEEE, 2010, pp. 1012–1017. ISBN: 978-1-4244-6674-0. DOI: 10.1109/IROS.2010.5650146.

[Dec+23] Jonas Dech et al. *PyCRAM*. Institute for Artificial Intelligence, Bremen University. GitHub, 2023. URL: https://github.com/cram2/pycram.

[Din+23] Yan Ding et al. "Integrating Action Knowledge and LLMs for Task Planning and Situation Handling in Open Worlds". In: *Autonomous Robots 2023 Special Issue on Large Language Models in Robotics* (2023). DOI: 10.48550/ARXIV.2305.17590. (Visited on 10/05/2023).

References II

[Lia+23] Jacky Liang et al. "Code as Policies: Language Model Programs for Embodied Control". In: *40th IEEE International Conference on Robotics and Automation (ICRA)*. London, UK: IEEE, 2023, pp. 9493–9500. DOI: 10.1109/ICRA48891.2023.10160591. (Visited on 02/22/2023).

[Lin04] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, 2004, pp. 74–81.

[Mil95] George A Miller. "WordNet: A Lexical Database for English". In: *Communications of the ACM* 38.11 (1995), pp. 39–41. DOI: 10.1145/219717.219748.

References III

[Pap+01] Kishore Papineni et al. "BLEU: A Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2001, p. 311. DOI: 10.3115/1073083.1073135. (Visited on 04/11/2023).

[Pop15] Maja Popović. "chrF: Character n-Gram F-score for Automatic MT Evaluation". In: *Proceedings of the 10th Workshop on Statistical Machine Translation*. 2015, pp. 392–395.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. (Visited on 06/01/2023).

References IV

[Sin+23] Ishika Singh et al. “ProgPrompt: Generating Situated Robot Task Plans Using Large Language Models”. In: *40th IEEE International Conference on Robotics and Automation (ICRA)*. London, UK: IEEE, May 2023, pp. 11523–11530. ISBN: 9798350323658. DOI: 10.1109/ICRA48891.2023.10161317. (Visited on 08/15/2023).

[Ver+22] David Vernon et al. “Action Selection and Execution in Everyday Activities: A Cognitive Robotics and Situation Model Perspective”. In: *TopiCS*. Everyday Activities 14.2 (2022), pp. 344–362. DOI: 10.1111/tops.12569. (Visited on 10/17/2022).

[WC22] Cai Wingfield and Louise Connell. “Sensorimotor Distance: A Grounded Measure of Semantic Similarity for 800 Million Concept Pairs”. In: *Behav Res* (Sept. 2022). ISSN: 1554-3528. DOI: 10.3758/s13428-022-01965-7. (Visited on 07/11/2023).

References V

[WP94] Zhibiao Wu and Martha Palmer. "Verb Semantics and Lexical Selection". In: *Proceedings of ACL 94*. arXiv, 1994. DOI: 10.48550/ARXIV.CMP-LG/9406033. (Visited on 05/17/2023).

[Zho+23] Shuyan Zhou et al. "CodeBERTScore: Evaluating Code Generation with Pretrained Models of Code". In: *Deep Learning for Code (DL4C) Workshop at the 11th International Conference on Learning Representations (ICLR)*. Kigali, Rwanda, 2023. DOI: 10.48550/ARXIV.2302.05527. (Visited on 04/11/2023).

Zero-Shot Prompting

```
(defaction slice-food (food-object)
    :precondition (and (is-sliceable food-object)
        (not (is-sliced food-object)))
    :effect (and (is-sliced food-object)
        (small-slice (make-slice food-object))
        (big-slice (make-slice food-object)))
    :body (progn (cut-food-object food-object)))
```

Incremental Refinement through Interaction

Task

- Refine a generated designator through interacting with ChatGPT (`gpt-3.5-turbo-0613`)

Incremental Refinement through Interaction

Task

- Refine a generated designator through interacting with ChatGPT (`gpt-3.5-turbo-0613`)
- In each step, tell ChatGPT one mistake and prompt it to fix it without just telling "add this line"

Incremental Refinement through Interaction

Task

- Refine a generated designator through interacting with ChatGPT (`gpt-3.5-turbo-0613`)
- In each step, tell ChatGPT one mistake and prompt it to fix it without just telling "add this line"
- Chosen example: *Pick-Up* based on *Close*
    - 1 added line (needs to be removed)
    - 8 missing lines (need to be added)
    - 14 changed lines (not all need changing)

Refinement Experience

- 12 rounds of dialogue necessary for all these changes

Refinement Experience

- 12 rounds of dialogue necessary for all these changes
- When prompted to remove 1 variable from the header, ChatGPT (additionally) added 5 correct lines at a fitting place

Refinement Experience

- 12 rounds of dialogue necessary for all these changes
- When prompted to remove 1 variable from the header, ChatGPT (additionally) added 5 correct lines at a fitting place
- When asked about this, ChatGPT does not provide an answer and removes the 5 lines

Refinement Experience

- 12 rounds of dialogue necessary for all these changes
- When prompted to remove 1 variable from the header, ChatGPT (additionally) added 5 correct lines at a fitting place
- When asked about this, ChatGPT does not provide an answer and removes the 5 lines
- After 9 rounds, ChatGPT started to add comments to its changes

Refinement Experience

- 12 rounds of dialogue necessary for all these changes
- When prompted to remove 1 variable from the header, ChatGPT (additionally) added 5 correct lines at a fitting place
- When asked about this, ChatGPT does not provide an answer and removes the 5 lines
- After 9 rounds, ChatGPT started to add comments to its changes

Summary

⇒ ChatGPT could successfully refine the designator

⇒ ChatGPT is not reliable since it introduces unforeseen changes

⇒ This demonstration worked well because the final result was known beforehand