

LLMs with Language Control

Xin Zhang

Peking University

Last Lecture

Neurosymbolic Programs

Symbolic Programs

- Interpretable
- Verifiable
- Structured domain knowledge
- Data efficient

Neural Networks

- Scalable algorithms
- Flexible
- Handles messy data
- Easy to get started



Last Lecture: Neural-Symbolic Programming in PP

- Treating neural networks as part of probabilistic programs' inputs
 - DeepProblog [Manhaeve et al., NeurIPS'18]
 - Scallop [Li et al., PLDI'23]
- Key Challenge: End-to-end training

This lecture

- Also some kind of neursymbolic programming
- Neural part being large language models (LLMs)

Today's Way to Interact with LLMs

- Prompting with natural languages
- Problems
 - Natural languages are ambiguous
 - LLMs may not follow

Qwen 3

Mary didn't marry Bob because he was rich. Did she make the right choice?



Whether Mary made the right choice by not marrying Bob solely based on his wealth depends on many factors that aren't covered in your statement. Important aspects to consider include Mary's values, her feelings towards Bob, and the overall compatibility between them.

Deepseek V3

Mary didn't marry Bob because he was rich. Did she make the right choice?



The sentence "Mary didn't marry Bob because he was rich" is ambiguous and can be interpreted in two different ways, which affects whether Mary's choice was "right" or not. Here's the breakdown:

Interpretation 1: Mary did not marry Bob, and the reason was that he was rich.

- Meaning: Mary chose not to marry Bob *due to* his wealth (perhaps she didn't want to marry someone for money or had other personal reasons).
- Did she make the right choice? This depends on Mary's values. If she prioritized love, compatibility, or independence over financial security, then her choice could be seen as "right" for her. If she later regretted missing out on a loving relationship with Bob, it might be seen as "wrong." Without more context, we can't definitively say.

Interpretation 2: Mary married Bob, but not because he was rich (i.e., his wealth was not the reason).

Deepseek V3

"List three fruits in bullet points, each starting with '🍎'.



Here's a list of three fruits, each starting with the apple emoji (🍎):

- 🍎pple
- 🍎range
- 🍎ango

Let me know if you'd like any modifications! 😊

Kimi

Count from 1 to 100, but skip numbers that can be divided by 13



Sure, I'll count from 1 to 100, skipping numbers that are divisible by 13. Here we go:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 93, 94, 95, 96, 97, 98, 99, 100.

LLM with Language Control

- Reduce ambiguity
- Enforce additional constraints
 - Correctness
- Key ideas
 - Use programming languages to interact with LLMs
 - Force LLMs to output structured sentences

LMQL

Luca Beurer-Kellner, Marc Fischer, Martin T. Vechev: Prompting Is Programming: A Query Language for Large Language Models.
Proc. ACM Program. Lang. 7(PLDI): 1946-1969 (2023)

Overview of LMQL

- Combining natural language prompts with SQL and python
 - Allows strict controls on LLM outputs
 - Enables building frontends for LLMs easily
- <https://lmql.ai>

Showcases

```
"Say 'this is a test': [RESPONSE]" · where · len(TOKENS(RESPONSE)) < 10
```

```
sample(temperature=1.2)
```

```
"Say 'this is a test' [RESPONSE]" · where · len(TOKENS(RESPONSE)) < 25
```

```
if "test" not in RESPONSE:  
    "You did not say 'test', try again: [RESPONSE]" · where · \  
    · · · len(TOKENS(RESPONSE)) < 25  
else:  
    "Good job"
```

Showcases

```
# instructions + few-shot samples
"""
A list of good dad jokes. A indicates the punchline
Q: How does a penguin build its house?
A: Igloos it together.
Q: Which knight invented King Arthur's Round Table?
A: Sir Cumference.
"""

# generate a joke
"Q: [JOKE]\n" where len(TOKENS(JOKE)) < 120 and STOPS_AT(JOKE, "?")
"A: [PUNCHLINE]" where STOPS_AT(PUNCHLINE, "\n") and len(TOKENS(PUNCHLINE)) > 1
```

Showcases

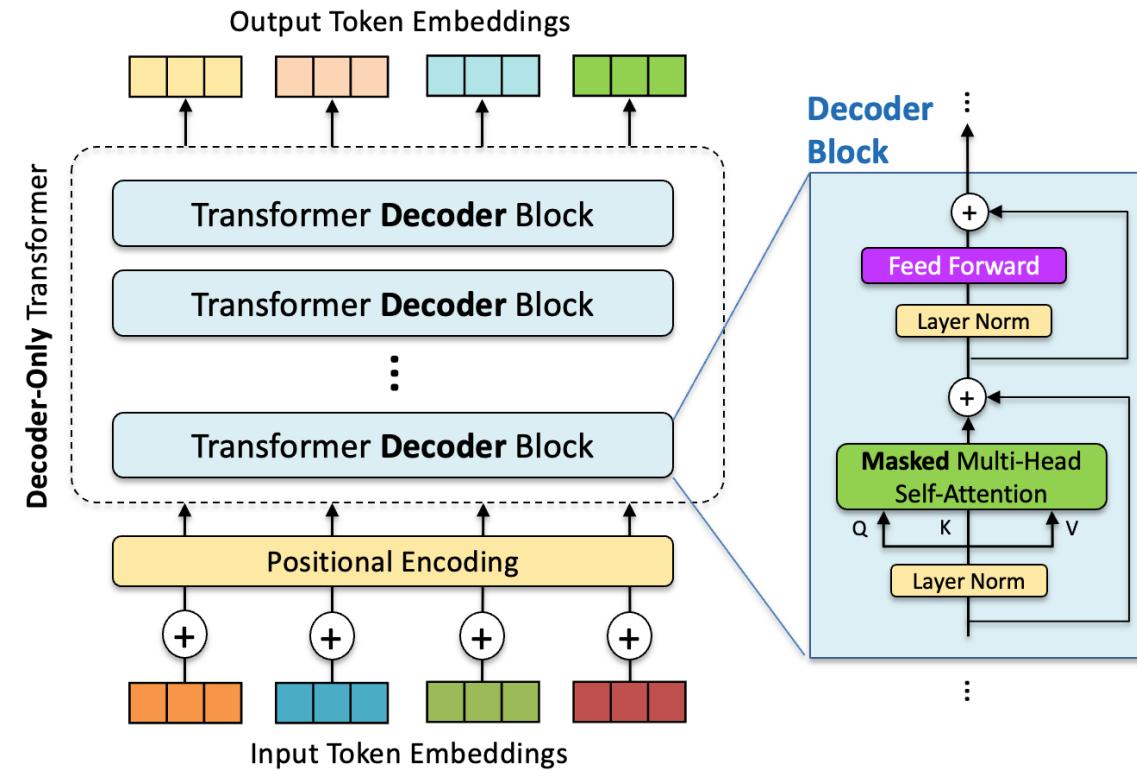
```
# specify a decoding strategy for the query
sample(temperature=0.8)

"A list of things not to forget when going to the beach: \n"
# use a loop to generate a list
for i in range(4):
    "- [THING] \n" where \
        THING in set(["Volleyball", "Sunscreen", "Bathing Suit"])
```

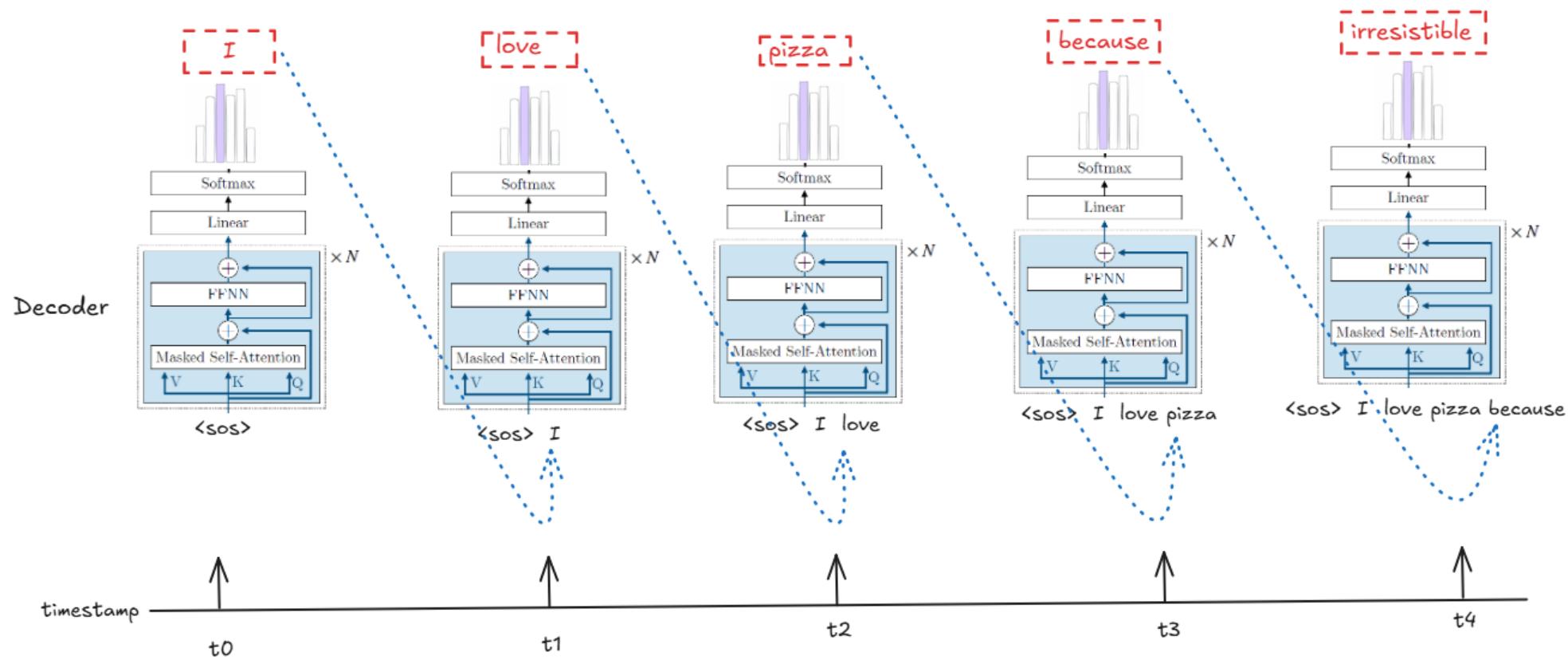
Showcases

```
"""
Write a summary of Bruno Mars, the singer:
{{
    "name": "[STRING_VALUE]",
    "age": [INT_VALUE],
    "top_songs": [
        "[STRING_VALUE]",
        "[STRING_VALUE]"
    ]
}}
"""
where STOPS_BEFORE(STRING_VALUE, "'") and \
    INT(INT_VALUE) and len(TOKENS(INT_VALUE)) < 2
```

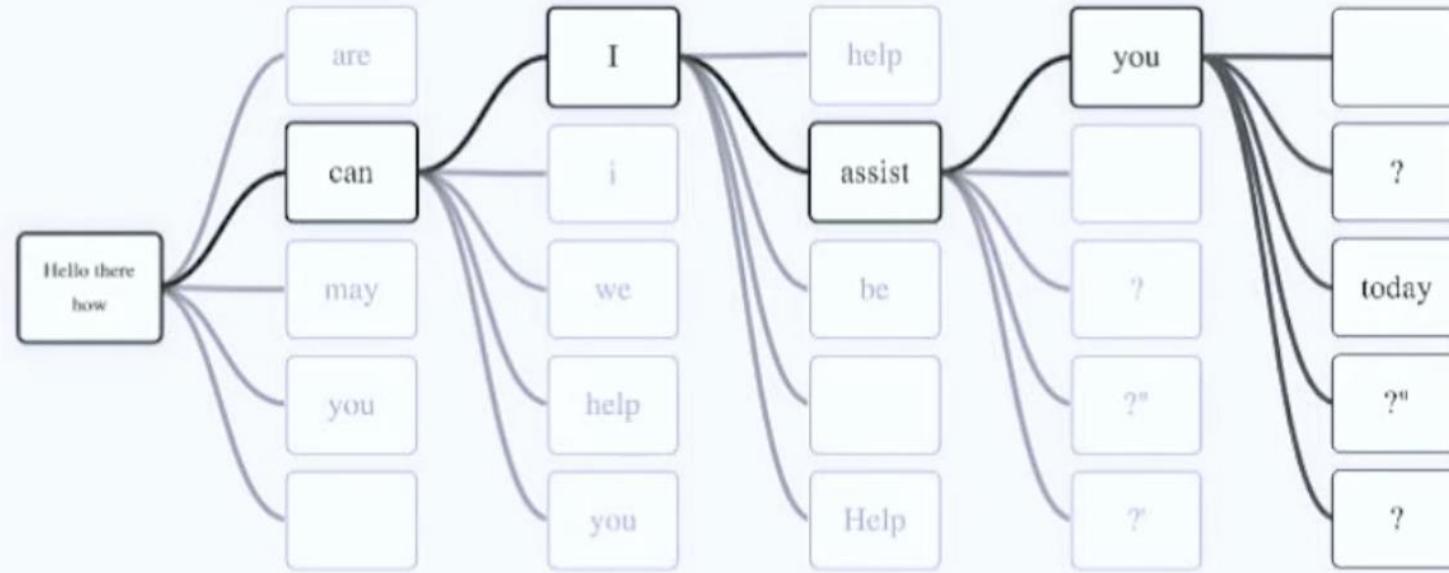
Background: LLMs



Background: LLMs



Background: LLMs



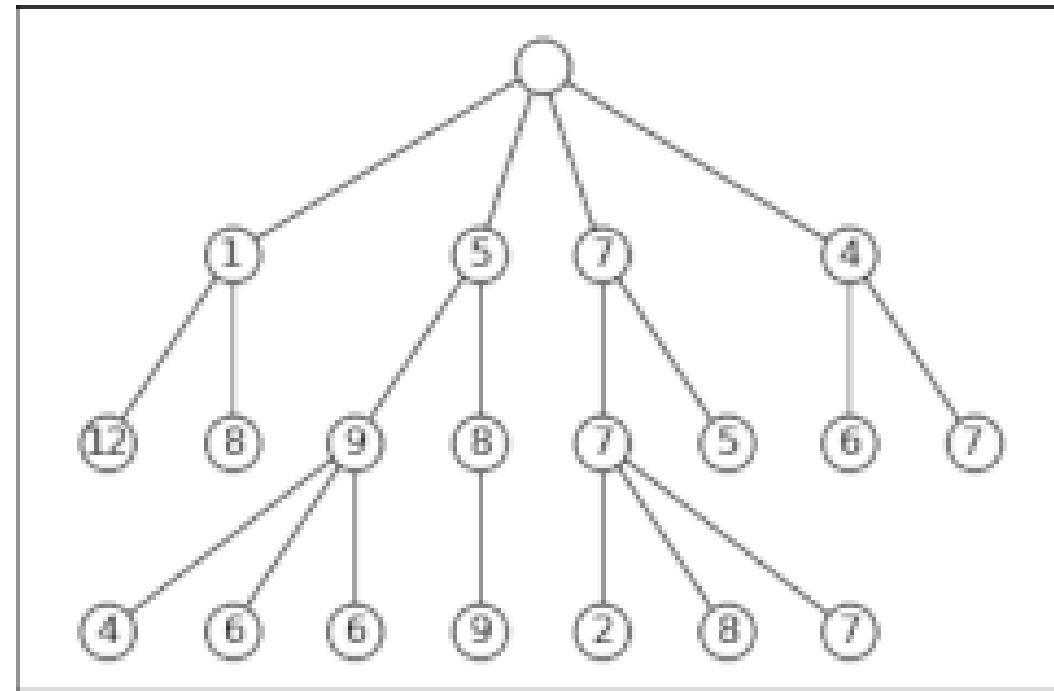
A **decoding algorithm** chooses the next token from the full vocabulary at each timestep (e.g. sample, argmax).

From Luca Beurer-Kellner's slides.

Background: LLMs

- How to get the next words
 - **Greedy decoding**: taking the word with the highest probability
 - **Sampling** (next token)
 - **Full encoding**: taking the sentence with the highest probability
 - **Beam search**

Background: Beam Search



https://en.wikipedia.org/wiki/Beam_search

Background: Masked Decoding

- Rules out certain tokens at certain positions

a mask $\mathbf{m} \in \{0, 1\}^{|\mathcal{V}|}$, $\mathbf{m} \odot \text{softmax}(\mathbf{z})$

Then normalize

Challenge 1: Interaction

(a) Manual Prompt

What is the circumference of the earth?

I believe the best person to answer this question is _____.

Indeed, _____ addressed this question:

Prompt 1

LM completion

Prompt 2

Challenge 2: Constraints & Token Representation

(b) GPT-2 completions after Prompt 1 :

- a physicist
- an astronomer
- a geologist
- Neal deGrasse Tyson
- William O'Malley, who has a PhD in Geodesy and is a professor at Colorado State University.
- the person having the knowledge and answer will probably have to refer to the relevant geophysics book and equations derived from that theory.
- a physicist, like Thomas Kugler at UC Irvine or one of the other physicists working with NASA ...
- a man named David
- actually Mother Earth herself?

LML Solution

(a) Manual Prompt

What is the circumference of the earth?

I believe the best person to answer this question is _____.

Indeed, _____ addressed this question:

Prompt 1

LM completion

Prompt 2

(c) LML query

What is the circumference of the earth? I believe the best person to answer this question is [EXPERT]

Indeed, {EXPERT} addressed this question: [ANSWER]

(d) LML constraint

`len(words(EXPERT)) <= 3 and stop_at(EXPERT, ".")`

(b) GPT-2 completions after **Prompt 1** :

- a physicist
- an astronomer
- a geologist
- Neal deGrasse Tyson
- William O'Malley, who has a PhD in Geodesy and is a professor at Colorado State University.
- the person having the knowledge and answer will probably have to refer to the relevant geophysics book and equations derived from that theory.
- a physicist, like Thomas Kugler at UC Irvine or one of the other physicists working with NASA ...
- a man named David
- actually Mother Earth herself?

LMQL Syntax

LMQL Program

```
<decoder> <query>
from <model>
[where <cond>]
[distribute <dist>]
```

```
<decoder> ::= argmax | beam(n=<int>) | sample(n=<int>)
<query> ::= <python_statement>+
<cond> ::= <cond> and <cond> | <cond> or <cond> | not <cond> | <cond_term>
          | <cond_term> <cond_op> <cond_term>
<cond_term> ::= <python_expression>
<cond_op> ::= < | > | = | in
<dist> ::= <var> over <python_expression>
```

The LMQL Runtime

Algorithm 1: Evaluation of a top-level string s

Input: string s , trace u , scope σ , language model f

- 1 **if** s contains [$\langle \text{varname} \rangle$] **then**
- 2 $s_{\text{pre}}, \text{varname}, s_{\text{post}} \leftarrow \text{unpack}(s)$
 // e.g. "a [b] c" \rightarrow "a ", "b", " c"
- 3 $u \leftarrow us_{\text{pre}}$ // append to trace
- 4 $v \leftarrow \text{decode}(f, u)$ // use the LM for the hole
- 5 $\sigma[\text{varname}] \leftarrow v$ // updated scope
- 6 $u \leftarrow uv$ // append to trace
- 7 **else if** s contains { $\langle \text{varname} \rangle$ } **then**
- 8 $\text{varname} \leftarrow \text{unpack}(s)$ // e.g. "{b}" \rightarrow "b"
- 9 $v \leftarrow \sigma[\text{varname}]$ // retrieve value from scope
- 10 $s \leftarrow \text{subs}(s, \text{varname}, v)$ // replace placeholder
 with value
- 11 $u \leftarrow us$ // append to trace
- 12 **else**
- 13 $u \leftarrow us$ // append to trace
- 14 **end**

Algorithm 2: Decoding

Input: trace u , scope σ , LM f

Output: decoded sequence v

- 1 $v \leftarrow \epsilon$
- 2 **while** True **do**
- 3 $\mathbf{m} \leftarrow \text{compute_mask}(u, \sigma, v)$
- 4 **if** $\wedge_i(m_i = 0)$ **then break**
- 5 $z \leftarrow 1/Z \cdot \mathbf{m} \odot \text{softmax}(f(uv))$
- 6 $t \leftarrow \text{pick}(z)$
- 7 **if** $t = \text{EOS}$ **then break**
- 8 $v \leftarrow vt$
- 9 **end**

The LMQL Runtime

```
1 argmax
2   "A list of things not to forget when "
3   "travelling:\n"
4   things = []
5   for i in range(2):
6     "- [THING]\n"
7     things.append(THING)
8   "The most important of these is [ITEM]."
9   from "EleutherAI/gpt-j-6B"
10  where
11    THING in ["passport",
12                  "phone",
13                  "keys", ...] // a longer list
14    and len(words(THING)) <= 2
```

The LMQL Runtime

line	update	state after update
1		$u = \epsilon$ $g = \{\}$
2	$s \leftarrow "A_list_of_things_not_to_forget_when"$ $u \leftarrow us$	$u = "A_list_of_things_not_to_forget_when"$ $g = \{\}$
3	$s \leftarrow "travelling:\n"$ $u \leftarrow us$	$u = "A_list_of_things_not_to_forget_when\ travelling\n"$ $g = \{\}$
4, $i = 0$	$s \leftarrow "-__ [THING]\n"$ $s_{pre}, varname, s_{post} \leftarrow "-__", THING, \n$ $u \leftarrow us_{pre}$ $v \leftarrow "sun_screen" = decode(f, u)$ $u \leftarrow uvs_{post}$ $g[varname] \leftarrow v$	$u = "A_list_of_things_not_to_forget_when\ travelling_\n -__sun_screen\n"$ $g = \{i = 0, THING = "sun_screen",$ $\quad things = ["sun_screen"]\}$
4, $i = 1$	$s \leftarrow "-__ [THING]\n"$ $s_{pre}, varname, s_{post} \leftarrow "-__", THING, \n$ $u \leftarrow us_{pre}$ $v \leftarrow "beach_towel" = decode(f, u)$ $u \leftarrow uvs_{post}$ $g[varname] \leftarrow v$	$u = "A_list_of_things_not_to_forget_when\ travelling_\n -__sun_screen\n -__beach_towel\n"$ $g = \{i = 1, THING = "beach_towel",$ $\quad things = ["sun_screen", "beach_towel"]\}$

Validation and Constraint Decoding

Algorithm 3: Naive Decoding with Constraints

Input: trace u , scope σ , language model f

Output: decoded sequence v

```
1 Function decode_step( $f, u, v$ )
2    $z \leftarrow \text{softmax}(f(uv))$ 
3    $\mathbf{m} \leftarrow \mathbf{1}^{|\mathcal{V}|}$ 
4   do
5      $t \leftarrow \text{pick}(1/z \cdot \mathbf{m} \odot z)$ 
6     if  $t \neq \text{EOS}$  then decode_step( $u, v, vt$ )
7     else if  $t = \text{EOS} \wedge \text{check}(u, vt)$  then
8       return  $v$ 
9     else  $\mathbf{m}[t] \leftarrow 0$ 
10    while  $\bigvee_i m_i = 1$ 
11    decode_step( $f, u, \epsilon$ )
```

Follow Semantics: Look Ahead

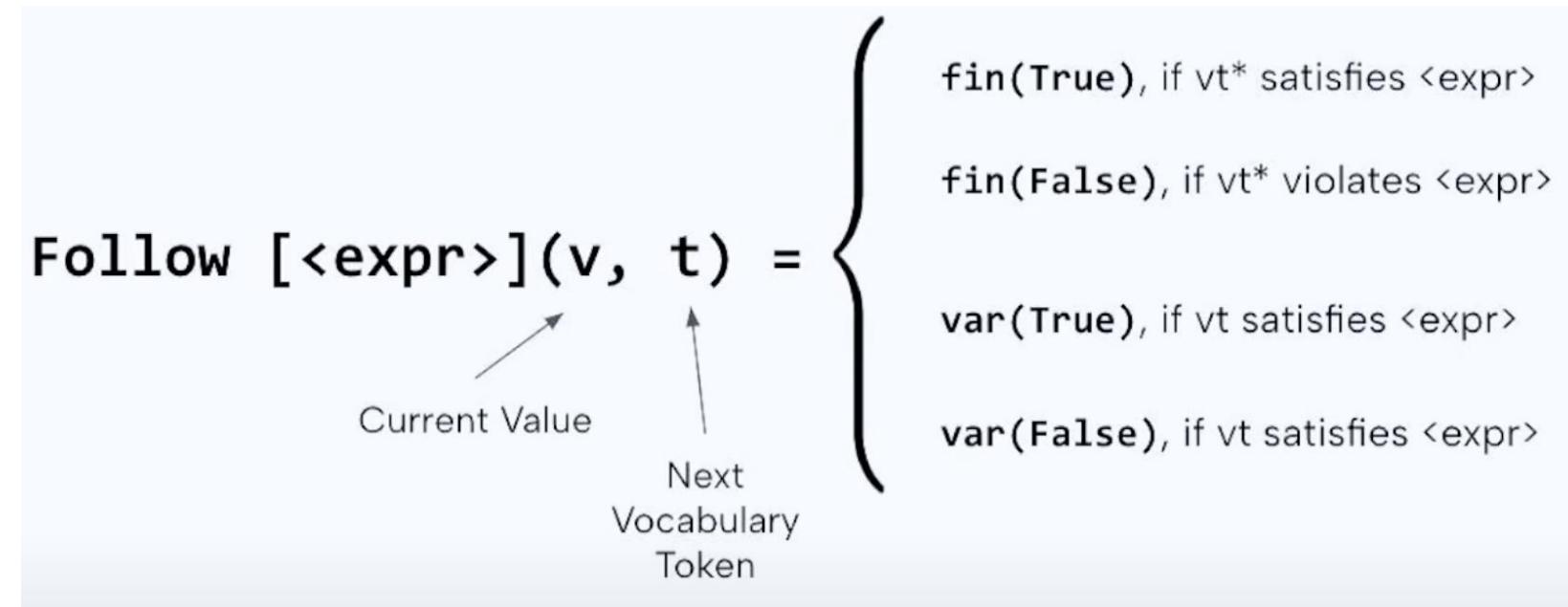
Follow [$\langle \text{expr} \rangle$](v, t) = $\left\{ \begin{array}{l} \text{True, if } vt \text{ satisfies } \langle \text{expr} \rangle \\ \text{False, if } vt \text{ violates } \langle \text{expr} \rangle \end{array} \right.$

Current Value Next Vocabulary Token

Problem: What if a constraint is only momentarily violated? e.g., `len(PUNCHLINE) > 10`

From Luca Beurer-Kellner's slides.

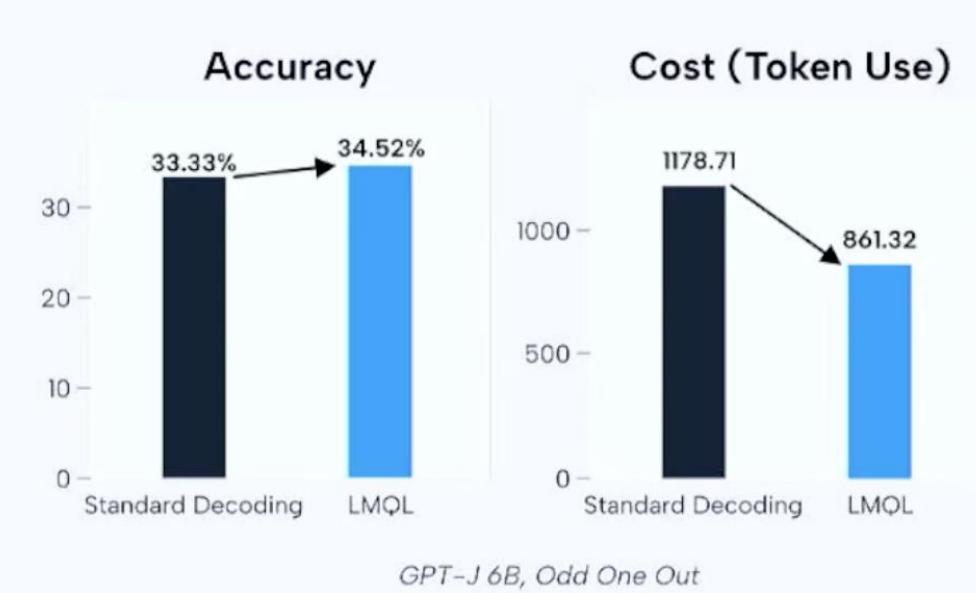
Follow and Final Semantics



From Luca Beurer-Kellner's slides.

Evaluation

- Constraining **maintains or improves LLM accuracy** on benchmark tasks
- **Early termination and logit masking** yields a **20-80% cost savings** compared to chunk-wise decoding with backtracking.



From Luca Beurer-Kellner's slides.

Conclusion

- One of the first works to use programming languages to control the interaction with LLMs
- Key idea is straightforward: Eagerly pruning outputs that do not satisfy the constraints
- More to see: <https://lmql.ai/>

Grammar-Constrained Decoding (GCD)

Saibo Geng, Martin Josifoski, Maxime Peyrard, Robert West: Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning. EMNLP 2023: 10932-10952

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, Loris D'Antoni: Grammar-Aligned Decoding. NeurIPS 2024

Motivation

- Previously we saw LMQL can enforce relatively simple constraints on the output
- To describe more structured outputs, we can use grammars!

OpenAI Already Supports JSON Outputs

Structured Outputs

 Copy page

Ensure responses adhere to a JSON schema.

Try it out

Try it out in the [Playground](#) or generate a ready-to-use schema definition to experiment with structured outputs.

 Generate

Introduction

JSON is one of the most widely used formats in the world for applications to exchange data.

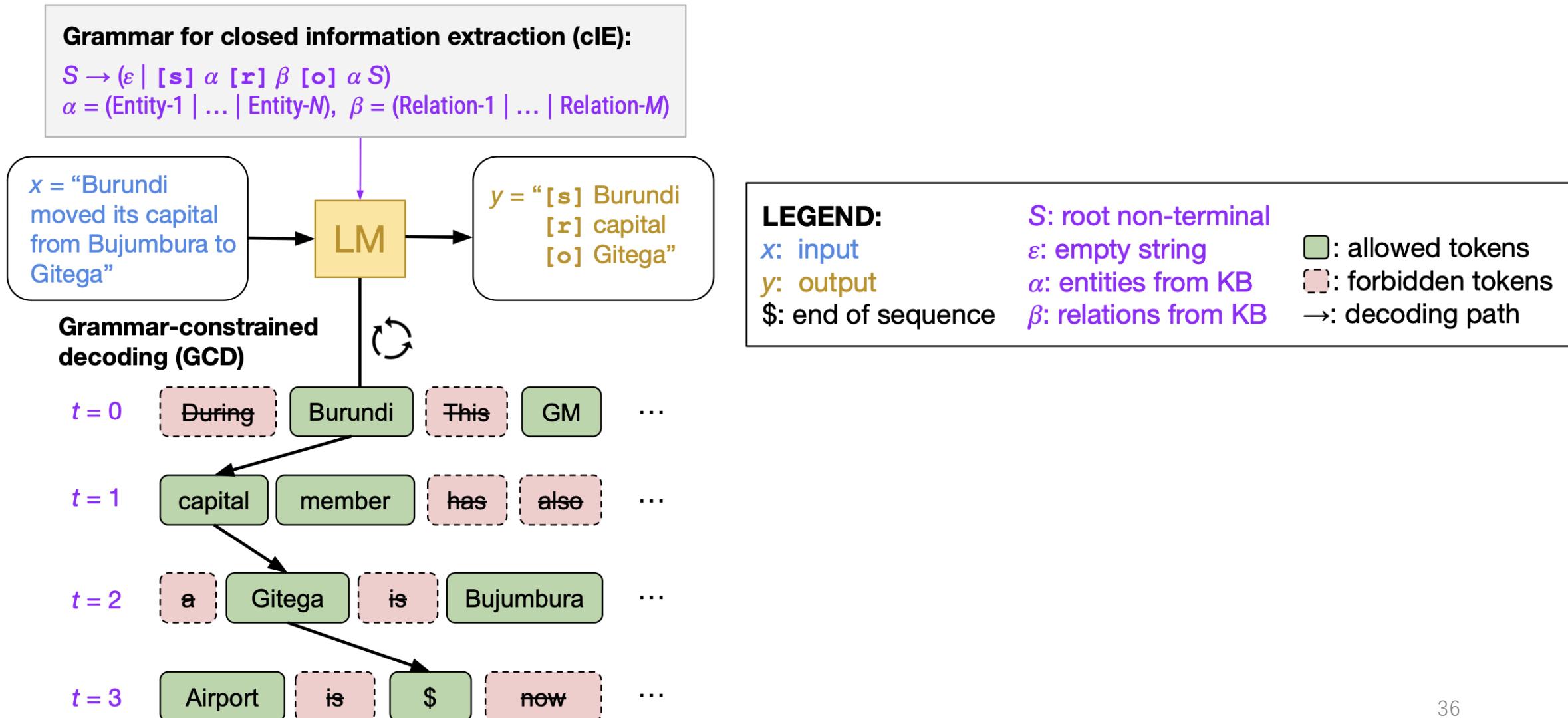
Structured Outputs is a feature that ensures the model will always generate responses that adhere to your supplied [JSON Schema](#), so you don't need to worry about the model omitting a required key, or hallucinating an invalid enum value.

Some benefits of Structured Outputs include:

- 1 **Reliable type-safety:** No need to validate or retry incorrectly formatted responses
- 2 **Explicit refusals:** Safety-based model refusals are now programmatically detectable
- 3 **Simpler prompting:** No need for strongly worded prompts to achieve consistent formatting

In addition to supporting JSON Schema in the REST API, the OpenAI SDKs for [Python](#) and [JavaScript](#) also make it easy to define object schemas using [Pydantic](#) and [Zod](#) respectively. Below, you can see how to extract information from unstructured text that conforms to a schema defined in code.

A More Complex Example



How to Implement GCD?

- Directly prompt LLMs?

code-davinci-002

Exhaustively express the information from the sentence in a form of {subject, relation, object} triplets. Triplets should cover all the information from the text, but no more.

Text:

Folk Songs for Far Out Folk is an album by Fred Katz originally released on Warner Bros.

Triplets:

{Fred Katz, album, Folk Songs for Far Out Folk}
{Fred Katz, album, Folk Songs for Far Out Folk}
{Fred Katz, album, Folk Songs for Far Out Folk}
{Fred Katz, album, Folk Songs for Far Out Folk}
{Fred Katz, album, Folk Songs for Far Out Folk}
{Fred Katz, album, Folk Songs for Far Out Folk}
{Fred Katz, album, Folk Songs for Far Out Folk}

GPT 3.5

Martin Josifoski, Marija Sakota, Maxime Peyrard, Robert West: Exploiting Asymmetry for Synthetic Training Data Generation: SynthIE and the Case of Information Extraction. EMNLP 2023: 1555-1574

How to Implement GCD?

- Use in-context-learning?

code-davinci-002

Exhaustively express the information from the sentence in a form of {subject, relation, object} triplets. Triplets should cover all the information from the text, but no more.

Text:

Aarhus airport serves the city of Aarhus.

Triplets:

{"subject": Aarhus_Airport, "relation": place served by transport hub, "object": Aarhus}

Text:

Arsène Lupin, Gentleman Burglar is the first collection of stories by Maurice Leblanc recounting the adventures of Arsène Lupin.

Triplets:

{"subject": Arsène_Lupin,_Gentleman_Burglar, "relation": author, "object": Maurice_Leblanc} {"subject": Arsène_Lupin, "relation": creator, "object": Maurice_Leblanc}

Text:

Folk Songs for Far Out Folk is an album by Fred Katz originally released on Warner Bros.

Triplets:

{"subject": Folk_Songs_for_Far_Out_Folk, "relation": artist, "object": Fred_Katz} {"subject": Folk_Songs_for_Far_Out_Folk, "relation": label, "object": Warner_Bros.}

GPT 3.5

Martin Josifoski, Marija Sakota, Maxime Peyrard, Robert West: Exploiting Asymmetry for Synthetic Training Data Generation: SynthIE and the Case of Information Extraction.
EMNLP 2023: 1555-1574

How to Implement GCD?

- Training?
- Finetuning is possible (De Cao et al. 2021 and Josifoski et al. 2022)
 - Training has a cost
 - Needs data

The GCD Framework in EMNLP 2023

- Using a parser to judge whether the sentence is correct
- Using in-context learning to boost success rate by LLMs

The GCD Framework in EMNLP 2023

Assume parsing works

Parsing let us know if a sentence is valid according to a grammar.
It provides a `IsSentenceValid` function: `str -> bool`.

How GCD works

The high-level algorithm of GCD is:

- ① Given an existing sentence(not necessarily complete) s
- ② Get a probability distribution over the next token $P(w_i|s)$
- ③ For each candidate token w_i in the distribution:
 - ④ Check if the sentence $s + w_i$ is valid according to the parser
 - ⑤ If valid, add w_i to the whitelist
 - ⑥ sample from the whitelist
- ⑦ Repeat until the sentence is complete

GCD Supports a lot of NLP Tasks

(1) Closed information extraction: see Fig. 1

(2)* Entity disambiguation: $S \rightarrow \ell m [\alpha] r$, where ℓ is left context of mention m , r is right context, and α is disjunction of candidate entities for mention m

(3)* Constituency parsing: $S \rightarrow B_{0,0}; B_{i,j} \rightarrow [\alpha (B_{i,j+1} | C_{i,j+1}) | C_{i,j} \rightarrow x_i (C_{i+1,j} | E_{i+1,j})]; C_{n,j} \rightarrow E_{n,j}; E_{i,j+1} \rightarrow] (E_{i,j} | B_{i,j}); E_{n,j+1} \rightarrow] E_{n,j}; E_{n,0} \rightarrow \varepsilon$, where $\alpha = (S | NP | VP | ...)$

(4)* Coreference resolution: $S_i \rightarrow x_i [(x_1 | \dots | x_n | \perp)] S_{i+1}; S_n \rightarrow \varepsilon$, where \perp means “no referent”

(5)* Part-of-speech tagging: $S_i \rightarrow x_i [(NOUN | VERB | ADJ | \dots)] S_{i+1}; S_n \rightarrow \varepsilon$

(6)* Dependency parsing: $S_i \rightarrow x_i [(ROOT | NSUBJ | DOBJ | \dots) (x_1 | \dots | x_n | \perp)] S_{i+1}; S_n \rightarrow \varepsilon$, where \perp means “no head”

(7)* Word sense disambiguation: $S_i \rightarrow x_i [\alpha_i] S_{i+1}; S_n \rightarrow \varepsilon$, where α_i is the disjunction of all WordNet glosses of word x_i

(8)* Phrase chunking: $S \rightarrow B_0; B_i \rightarrow [C_i; B_n \rightarrow \varepsilon; C_i \rightarrow x_i (C_{i+1} | \alpha)] B_{i+1}; C_n \rightarrow \alpha]$, where $\alpha = (NP | VP | PP | \dots)$

(9)* Semantic role labeling: Same as phrase chunking, but with $\alpha = (\text{TARGET} | \text{ARG0} | \text{ARG1} | \dots)$

(10)* Entity linking: Same as phrase chunking, but with α the disjunction of all KB entity names (or \perp for “no entity”)

(11)* CCG parsing: Same as constituency parsing, but with syntactic types (e.g., $(S \setminus NP)/NP$) instead of constituent labels. Extra constraints ensure that nodes have at most two children and that syntactic types combine correctly.

(12)* Question answering: $S \rightarrow [q] [A]; A \rightarrow (\varepsilon | \alpha A)$, where q is the question and α the disjunction of all vocabulary words

(13)* Extractive summarization: $S \rightarrow (\varepsilon | [\alpha] S)$, where α is the disjunction of all sentences from input x

(14)* Semantic parsing with λ -calculus: A logical form is a rooted tree, generated by a context-free grammar

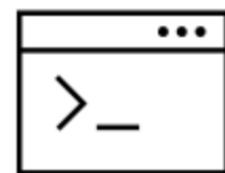
Experiment Results

Method	Precision	Recall	F1
<i>Weakly supervised</i>			
GenIE T5-base	49.6 ± 0.3	26.8 ± 0.2	34.8 ± 0.2
<i>Few-shot unconstrained</i>			
LLaMA-7B	10.2 ± 0.5	14.3 ± 0.7	11.9 ± 0.5
LLaMA-13B	10.3 ± 0.6	17.0 ± 0.9	12.9 ± 0.6
LLaMA-33B	14.1 ± 1.0	23.1 ± 1.4	17.5 ± 1.0
Vicuna-7B	12.5 ± 0.2	16.7 ± 0.1	14.3 ± 0.2
Vicuna-13B	13.4 ± 0.2	15.2 ± 0.2	14.4 ± 0.2
<i>Few-shot constrained</i>			
LLaMA-7B	27.9 ± 0.6	20.2 ± 0.5	23.5 ± 0.5
LLaMA-13B	36.2 ± 0.7	26.5 ± 0.5	30.6 ± 0.5
LLaMA-33B	39.3 ± 0.9	33.2 ± 0.8	36.0 ± 0.7
Vicuna-7B	25.4 ± 0.5	15.8 ± 0.3	19.5 ± 0.3
Vicuna-13B	38.7 ± 1.0	19.8 ± 0.8	26.1 ± 0.8

Compared to LMQL

- Uses grammars rather than SQL constraints to limit the outputs
- Uses an incremental parser
 - Can check partial sentence

Problem with GCD: Probability Distortion



Generate a binary string that ends with 1

Prompt

\mathcal{G}

Context-

Free

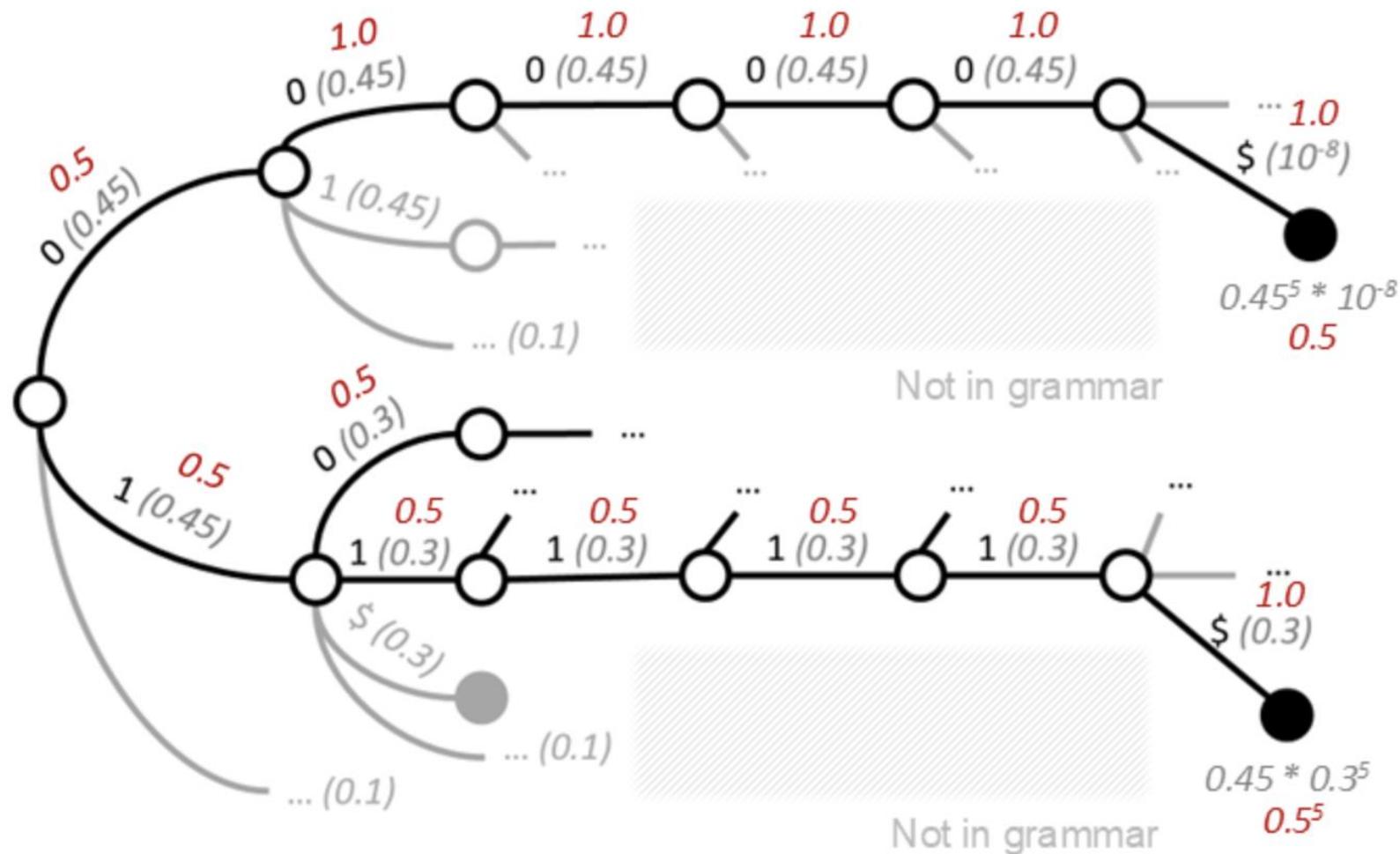
Grammar

$S ::= 00000 \mid 1A_2$
 $A_i ::= 0A_{i+1} \mid 1A_{i+1}, \text{ for } i = 2, 3, 4$
 $A_5 ::= 0 \mid 1$

{ 00000, 10000, 10001, ..., 11110, 11111 }

Assuming the LLM has an equal probability to generate 0 and 1, what is $P(11111)$?

Problem with GCD: Probability Distortion



Problem: Grammar-Aligned Decoding

w	$P(w)$	$P_{GAD}(w)$
AA	$0.2 * 10^{-9}$	
0 0 0 0 0	$0.45^5 * 10^{-8}$	$2.78 * 10^{-8}$
0 0 0 0 1	0.45^6	
...	...	
1 0 0 0 0	$0.45^5 * 10^{-8}$	∞ $2.78 * 10^{-8}$
...
1 1 1 1 1	0.45^6	0.125
...	...	

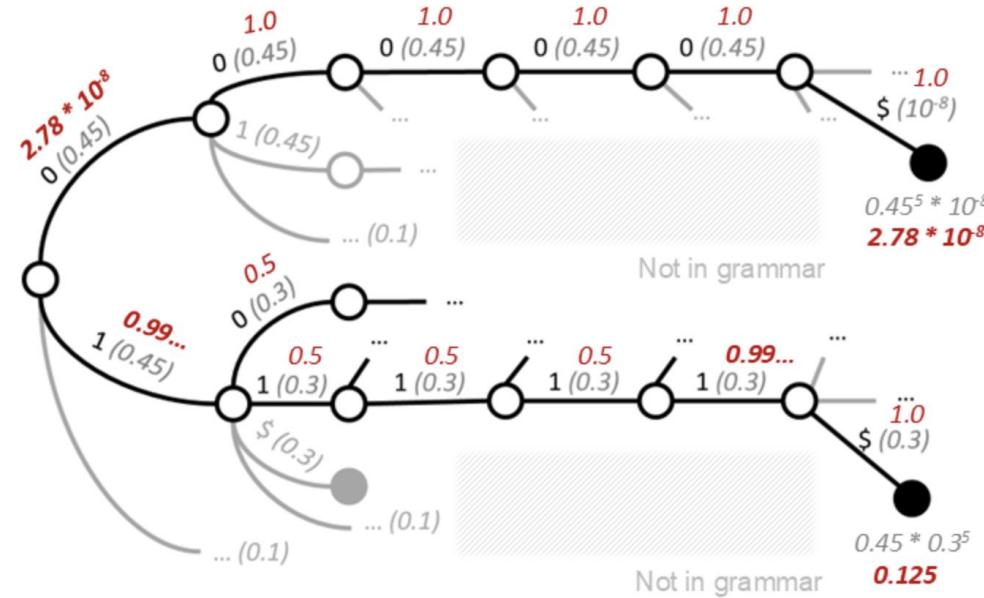
Problem: Grammar-Aligned Decoding

Given a model distribution P and a context-free grammar \mathcal{G} ,

grammar-aligned decoding (GAD) is the task of sampling from the distribution P_{GAD} that is *proportional to P* , but restricted in a grammar.

$$Q^{P,\mathcal{G}}(w) = \frac{\mathbb{1}[w \in \mathcal{L}(\mathcal{G})] \cdot P(w)}{\sum_{w'} \mathbb{1}[w' \in \mathcal{L}(\mathcal{G})] \cdot P(w')}$$

Next-token Conditional with Expected Future Grammaticality (EFG)

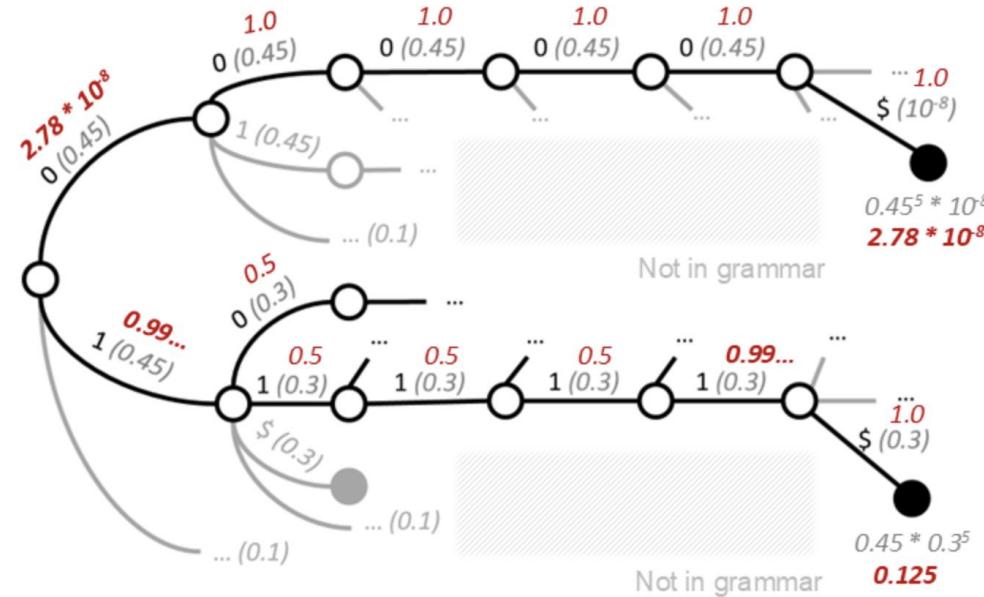


Intractable!

$$P_{GAD}(w_i | w_{1:i-1}) \propto P(w_i | w_{1:i-1}) \cdot \mathbb{E}_{P(w_{i+1:n} | w_{1:i})}[1[w \in \mathcal{L}(\mathcal{G})]]$$

In GAD, next-token conditionals are adjusted by the total probabilities of all possible continuations, called *expected future grammaticality (EFG)*.

Next-token Conditional with Expected Future Grammaticality (EFG)

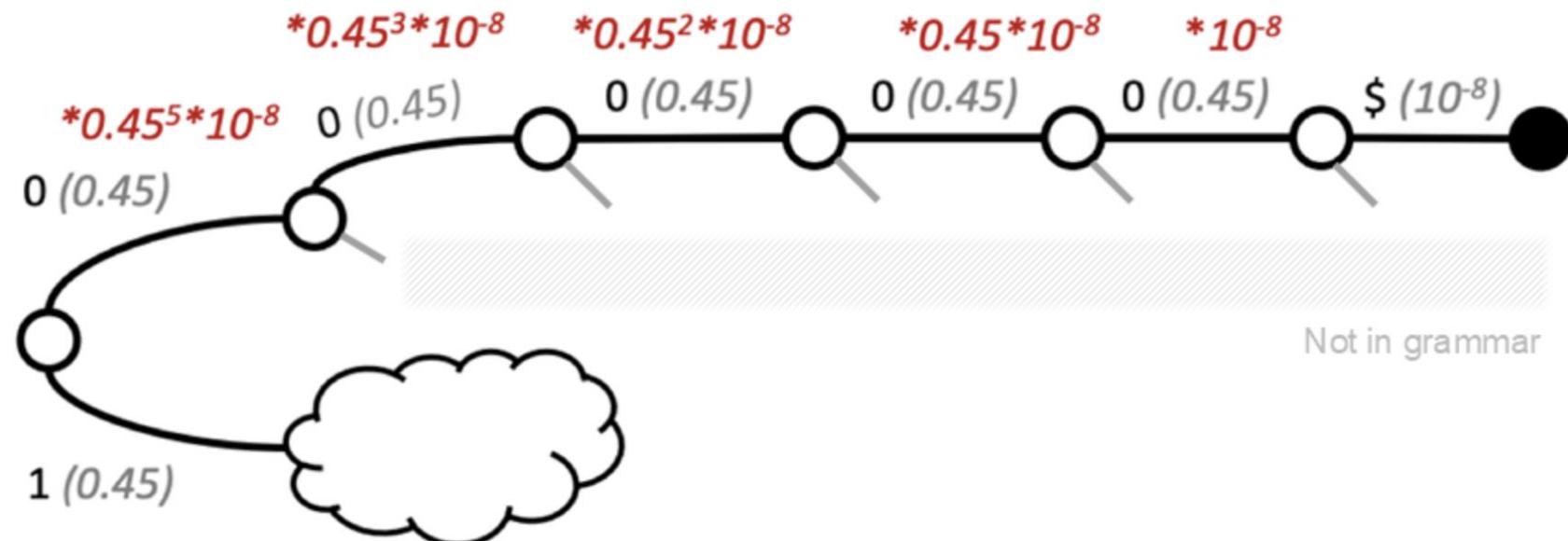


Intractable!

$$P_{GAD}(w_i | w_{1:i-1}) \propto P(w_i | w_{1:i-1}) \cdot \mathbb{E}_{P(w_{i+1:n} | w_{1:i})}[1[w \in \mathcal{L}(\mathcal{G})]]$$

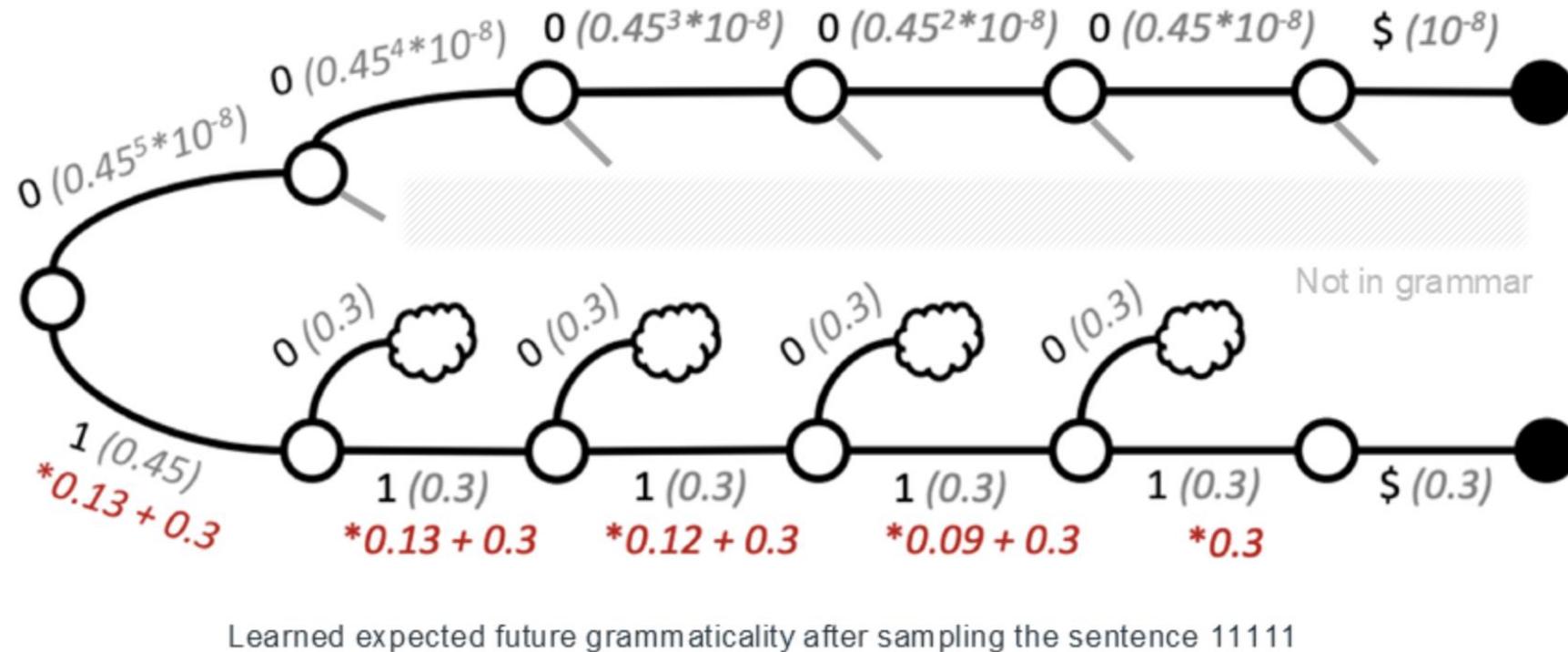
In GAD, next-token conditionals are adjusted by the total probabilities of all possible continuations, called *expected future grammaticality (EFG)*.

Algorithm: Adaptive Sampling with Approximate Expected Futures (ASAp)



Learned expected future grammaticality after sampling the sentence 00000

Algorithm: Adaptive Sampling with Approximate Expected Futures (ASAp)



Evaluations

KL Divergence

$$D_{KL}(Q \parallel P_{GAD}) = \sum Q(x) \log \frac{Q(x)}{P_{GAD}(x)}$$

A measure of how much a probability distribution Q is different from the ideal probability distribution P_{GAD}

Expectations

$$\mathbb{E}_{P(w)}[P(w)] = \sum P(w)$$

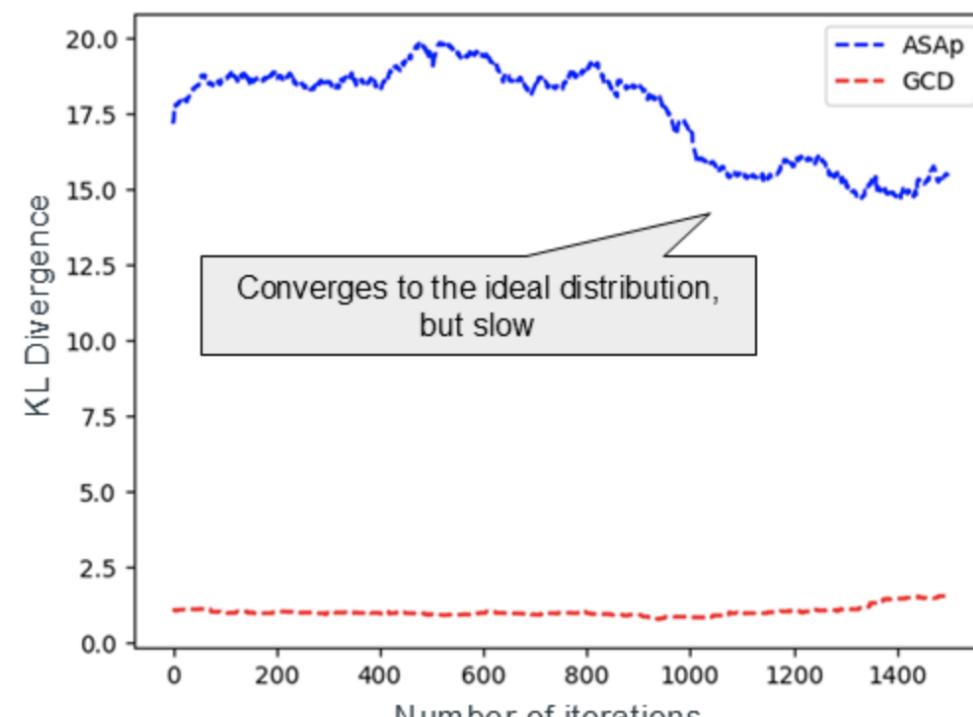
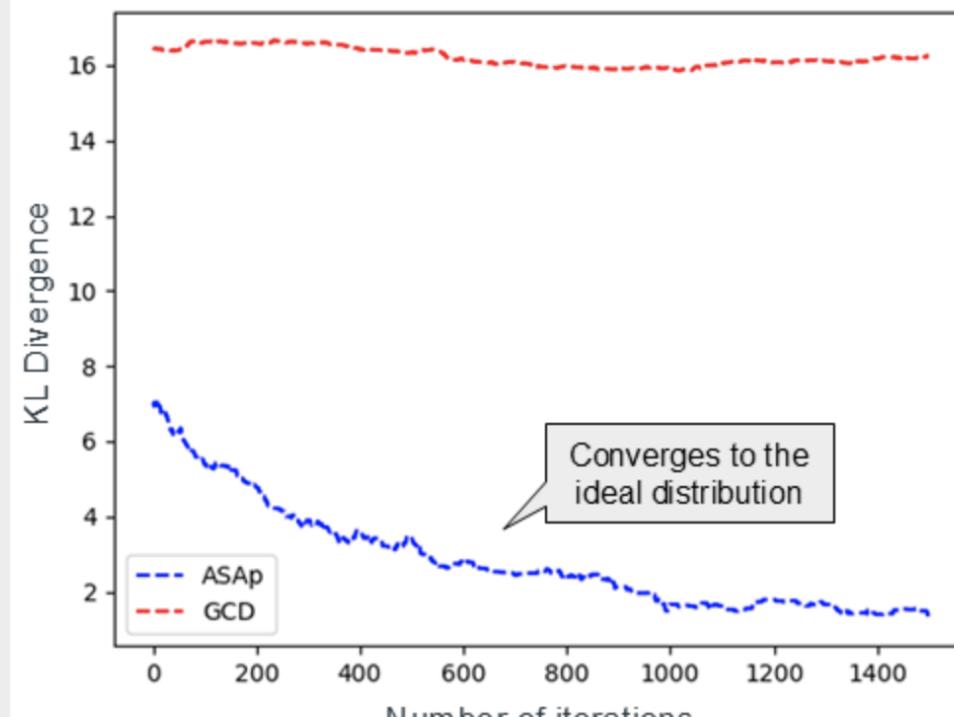
How close is the expectation of probability distribution Q to the ideal distribution P_{GAD}



Synthesize a program in the grammar
that satisfies the specification

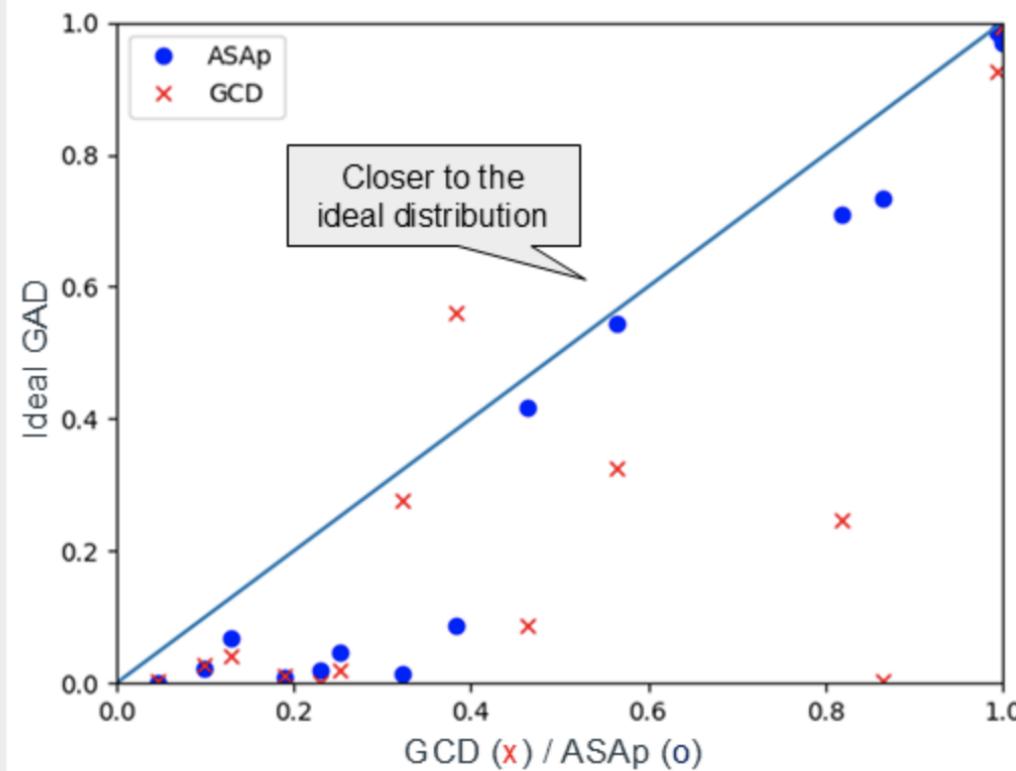
Evaluations

KL divergence between ASAp/GCD and the ideal GAD

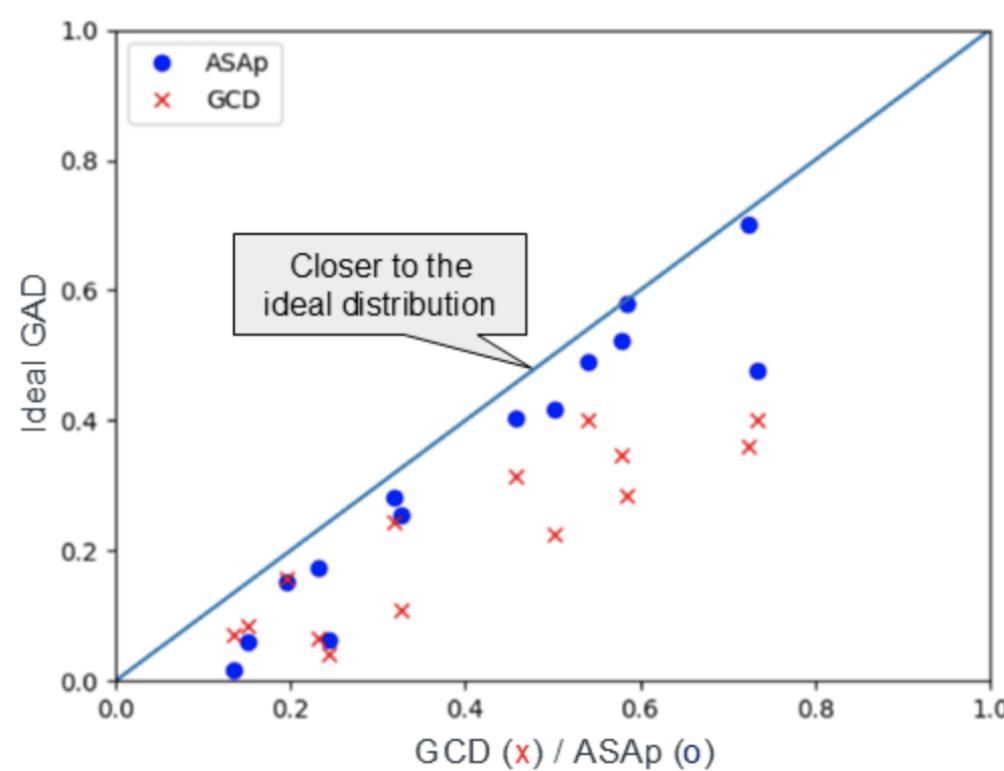


Evaluations

Expectations of ASAp/GCD vs. expectations of ideal GAD



Strings with Linear Integer Arithmetic



Loop Invariant Generation with Bit-vector Arithmetic

Final Notes

- LLMs are becoming smarter, but it is unrealistic to trust it to generate fully-structured outputs without mistakes
- LLMs with language control is still an active and relatively undeveloped field.