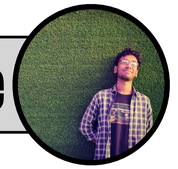
Decoding Code Hallucinations in LLMs

Dr. Antonio Mastropaolo

Instructor

Mr. Alvi Haque



Teaching Assistant

WILLIAM & MARY

CHARTERED 1693

Spring 2026

Decoding Code Hallucinations in LLMs

Hallucinations describe the scenario in which the output produced by the LLMs looks like it is fluent but factually incorrect and/or inconsistent

Decoding Code Hallucinations in LLMs

Hallucinations describe the scenario in which the output produced by the LLMs looks like it is fluent but factually incorrect and/or inconsistent

Input-Conflicting: Contradicts the prompt

Context-Conflicting: Inconsistent with previous output

Fact-Conflicting: Misaligned with real-world knowledge

Decoding Code Hallucinations in LLMs

Hallucinations describe the scenario in which the output produced by the LLMs looks like it is fluent but factually incorrect and/or inconsistent

Input-Conflcting: Contradicts the prompt

> ***Prompt:*** *Summarize the following article without mentioning any political figures:"*
>
> ***Article Snippet:*** *"The new healthcare policy aims to expand access to rural clinics. President Smith emphasized the importance of equitable coverage."*
>
> ***LLM Output:*** *"President Smith introduced a groundbreaking healthcare reform to improve rural access."*

Decoding Code Hallucinations in LLMs

Hallucinations describe the scenario in which the output produced by the LLMs looks like it is fluent but factually incorrect and/or inconsistent

Input-Conflcting: Contradicts the prompt

Prompt: *Summarize the following article without mentioning any political figures:"*

Article Snippet: *"The new healthcare policy aims to expand access to rural clinics. President Smith emphasized the importance of equitable coverage."*

LLM Output: *"President Smith introduced a groundbreaking healthcare reform to improve rural access."*

Decoding Code Hallucinations in LLMs

Hallucinations describe the scenario in which the output produced by the LLMs looks like it is fluent but factually incorrect and/or inconsistent

Context-Conflcting: Inconsistent with previous output

User: *“What’s the capital of Italy?”*

LLM: *“The capital of Italy is Rome.”*

User: *“And what about France?”*

LLM: *“The capital of France is Rome.”*

Decoding Code Hallucinations in LLMs

Hallucinations describe the scenario in which the output produced by the LLMs looks like it is fluent but factually incorrect and/or inconsistent

Context-Conflcting: Inconsistent with previous output

User: *"What's the capital of Italy?"*

LLM: *"The capital of Italy is Rome."*

User: *"And what about France?"*

LLM: *"The capital of France is Rome."*

Decoding Code Hallucinations in LLMs

Hallucinations describe the scenario in which the output produced by the LLMs looks like it is fluent but factually incorrect and/or inconsistent

Fact-Conflcting: Misaligned with real-world knowledge

Prompt: "Tell me about Albert Einstein."

LLM Output: *"Albert Einstein was a 21st-century physicist best known for inventing the telescope."*

Decoding Code Hallucinations in LLMs

Hallucinations describe the scenario in which the output produced by the LLMs looks like it is fluent but factually incorrect and/or inconsistent

Cool….but what about the code?

Decoding Code Hallucinations in LLMs

Hallucinations describe the scenario in which the output produced by the LLMs looks like it is fluent but factually incorrect and/or inconsistent

NLP Hallucination.	***Code Hallucination***
Input Conflicting	*Misimplements requirements*
Context Conflicting	*Uses wrong library/API call*
Fact Conflicting	*Incorrect variable reuse/logic*

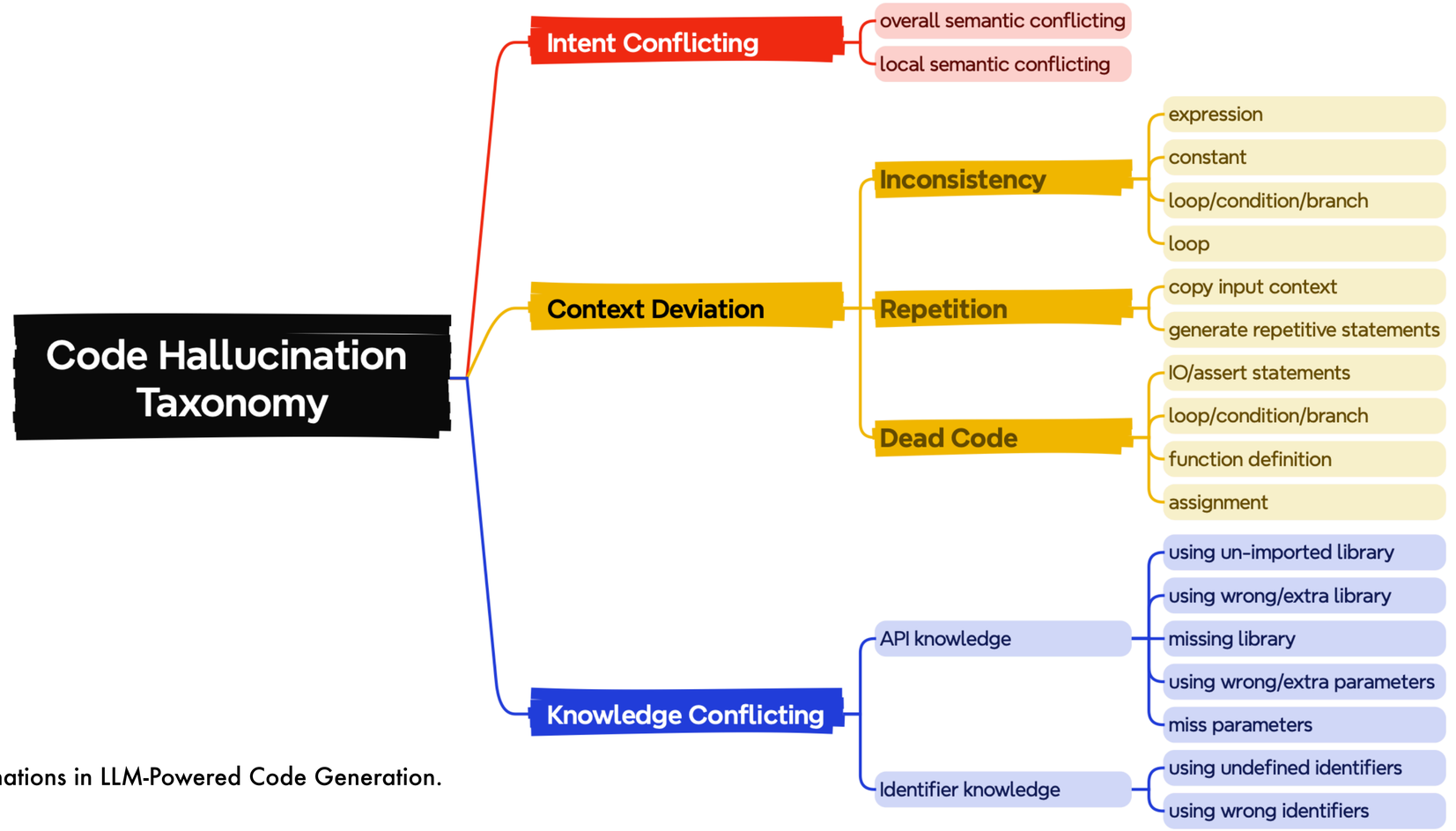
Decoding Code Hallucinations in LLMs

Hallucinations describe the scenario in which the output produced by the LLMs looks like it is fluent but factually incorrect and/or inconsistent

NLP Hallucination.	Code Hallucination
Input Conflicti	*nts*
	e wrong library/API call
Fact Conflicting	*Incorrect variable reuse/logic*

Common sense is an abstract concept

Decoding Code Hallucinations in LLMs



Exploring and Evaluating Hallucinations in LLM-Powered Code Generation.
Liu et al.

Decoding Code Hallucinations in LLMs

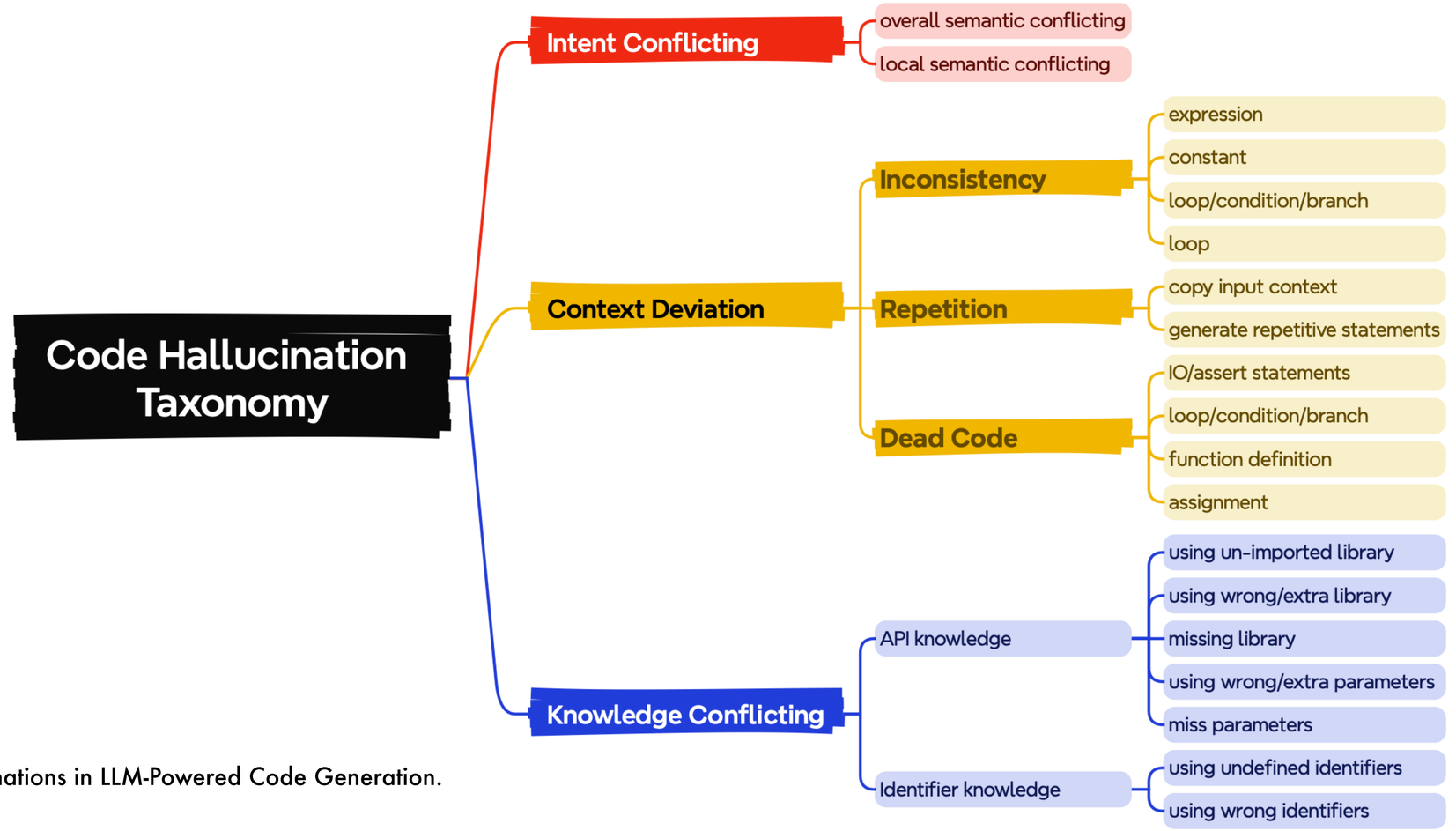
- **Intent Conflicting**
  - overall semantic conflicting
  - local semantic conflicting
- **Inconsistency**

```
def string_xor(a: str, b: str) -> str:
    """ Input are two strings a and b
    consisting only of 1s and 0s.
    Perform binary XOR on these inputs
    and return result also as a string.
    >>> string_xor('010', '110')
    '100'
    """
    # generated by CodeRL
    a, b = a.strip('0'), b.strip('0')
    res = ''
    if a!= b:
     res = '0' + res
    if a:
     res = '1' + res
    if b:
     res = '0' + res
    return res
```

Conflicting

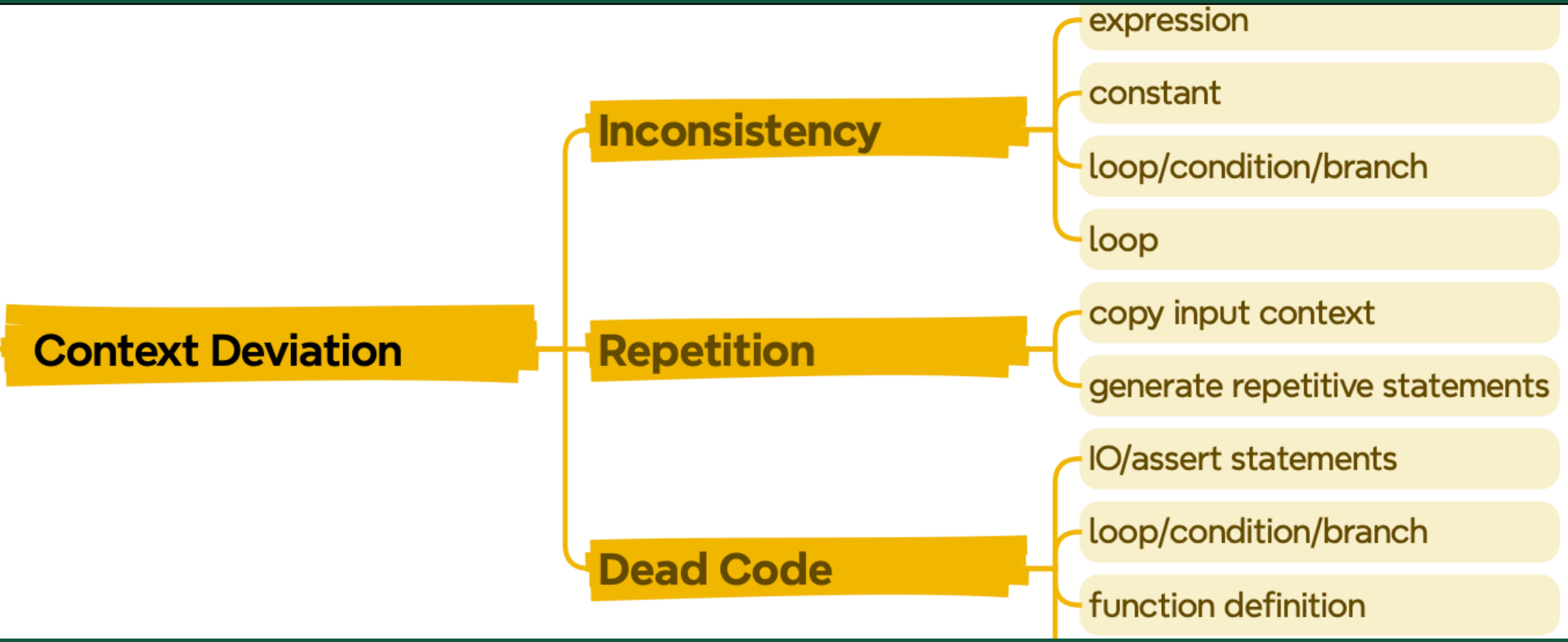
```
    # reference solution
    def xor(i, j):
        if i == j:
            return '0'
        else:
            return '1'
    return ''.join(xor(x, y) for x, y
            in zip(a, b))
```

Decoding Code Hallucinations in LLMs

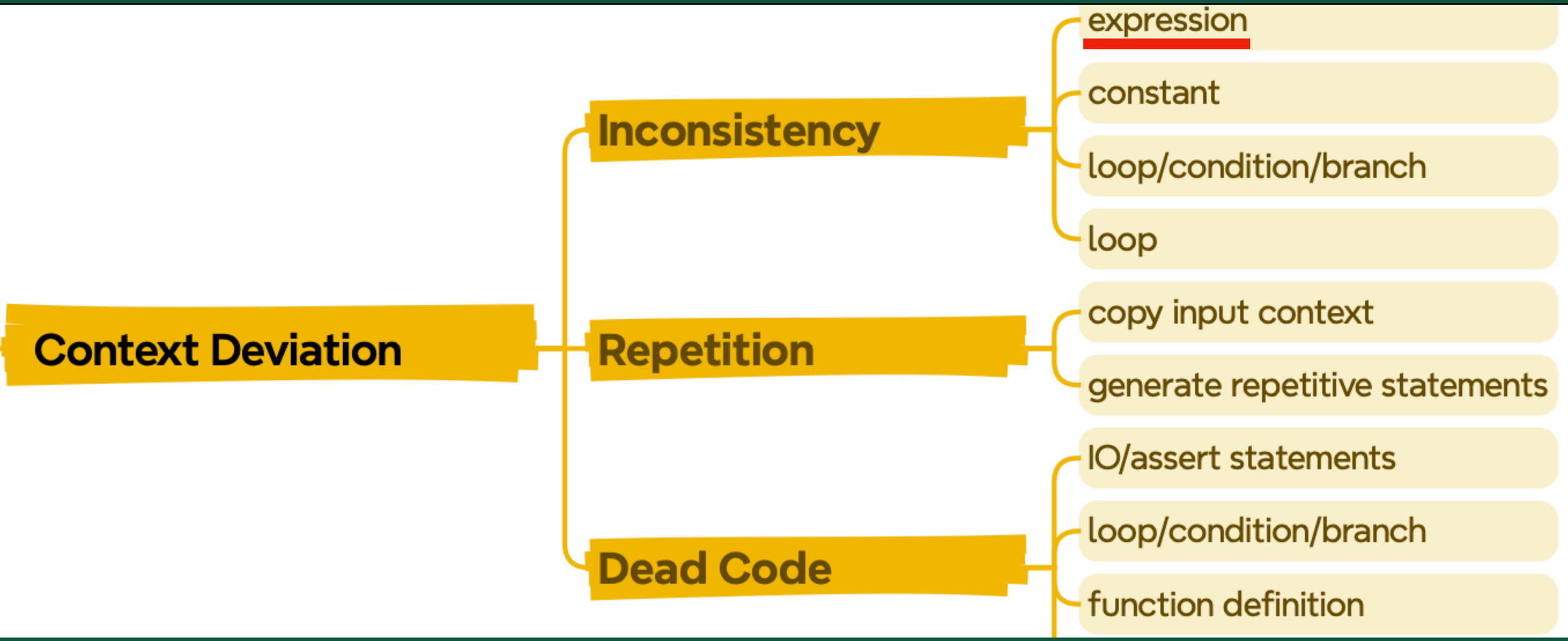


Exploring and Evaluating Hallucinations in LLM-Powered Code Generation.
Liu et al.

Decoding Code Hallucinations in LLMs



Decoding Code Hallucinations in LLMs



Decoding Code Hallucinations in LLMs

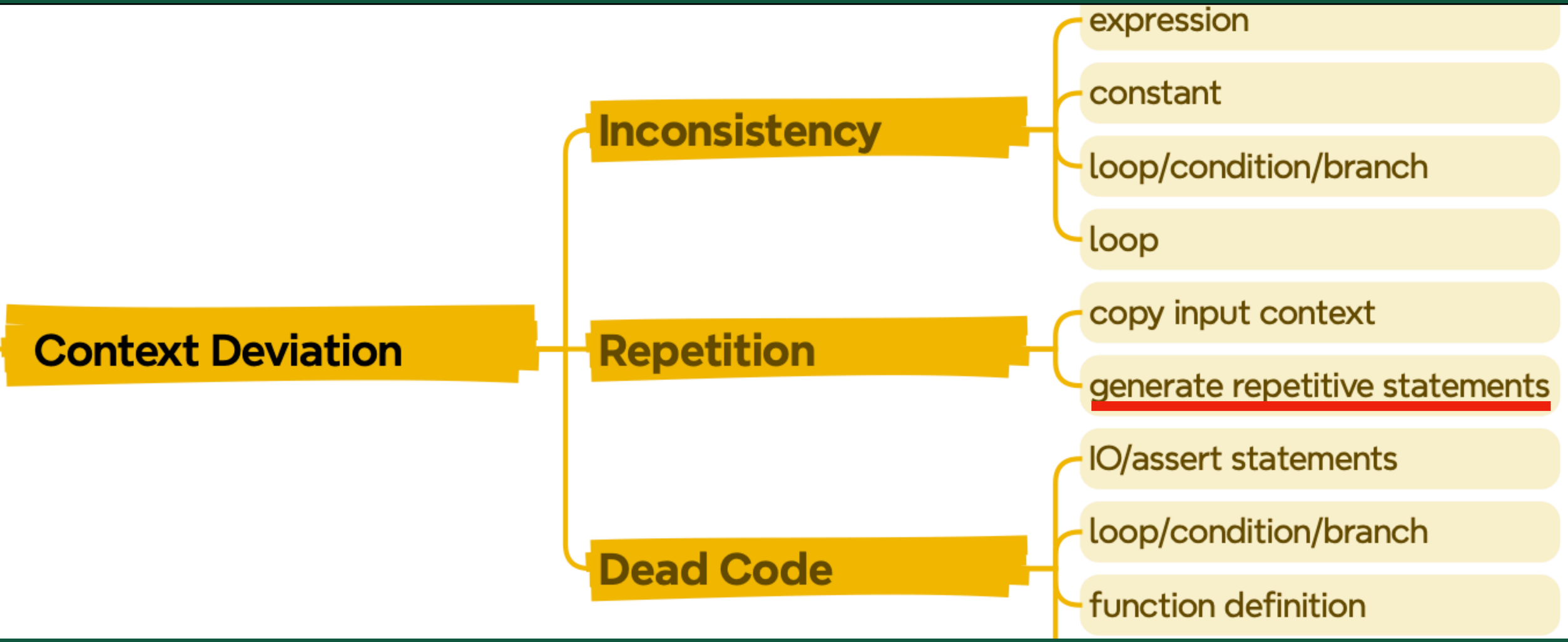
- expression
- constant
- loop/condition/branch
- loop
- copy input context
- generate repetitive statements
- IO/assert statements
- loop/condition/branch

```
def filter_by_substring(strings: List[str], substring: str)
-> List[str]:
    """ Filter an input list of strings only for
        ones that contain given substring
    >>> filter_by_substring([], 'a')
    []
    >>> filter_by_substring(['abc', 'bacd', 'cde',
        'array'], 'a')
    ['abc', 'bacd', 'array']
    """
    # generated by CodeRL
    return [string for string in strings if
            string.startswith(substring)]

    # reference solution
    return [x for x in strings if substring in x]
```

Inconsistent

Decoding Code Hallucinations in LLMs



Decoding Code Hallucinations in LLMs

- **Inconsistency**
  - expression
  - constant
  - loop/condition/branch
  - loop
- **Repetition**
  - copy input context
  - generate repetitive statements
- **Dead Code**
  - IO/assert statements
  - loop/condition/branch
  - function definition

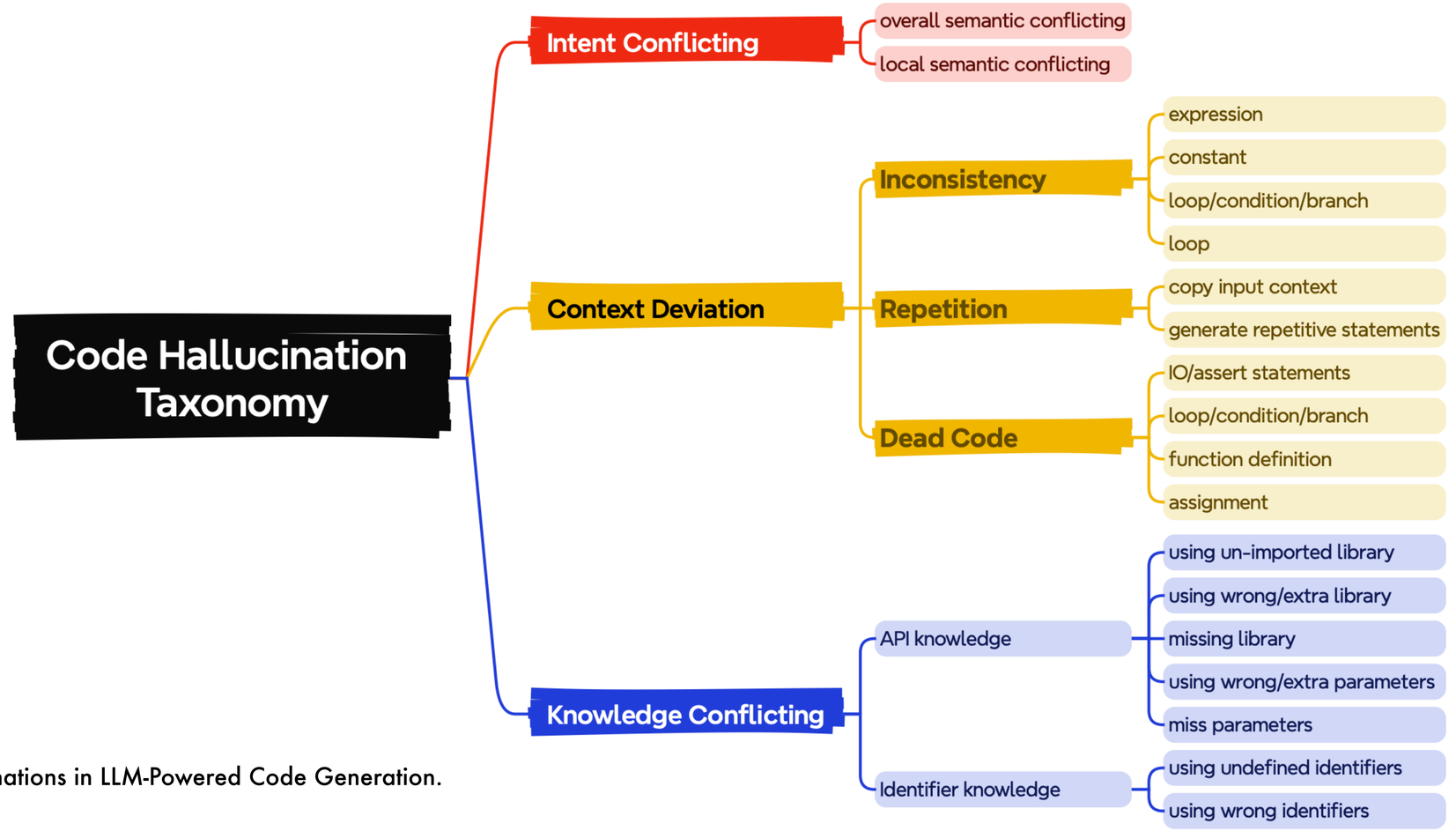
```python
def largest_prime_factor(n: int):
    """Return the largest prime factor of n.
       Assume n > 1 and is not a prime.
    >>> largest_prime_factor(13195)
    29
    >>> largest_prime_factor(2048)
    2
    """
```

```python
# reference solution
def is_prime(k):
    if k < 2:
        return False
    for i in range(2, k - 1):
        if k % i == 0:
            return False
    return True
largest = 1
for j in range(2, n + 1):
    if n % j == 0 and is_prime(j):
        largest = max(largest, j)
return largest
```

```python
# generated by CodeGen
def largest_prime_factor(n: int):
    if n < 2:
        return 1
    if n % 2 == 0:
        return 2
    if n % 3 == 0:
        return 3
    if n % 5 == 0:
        return 5
```

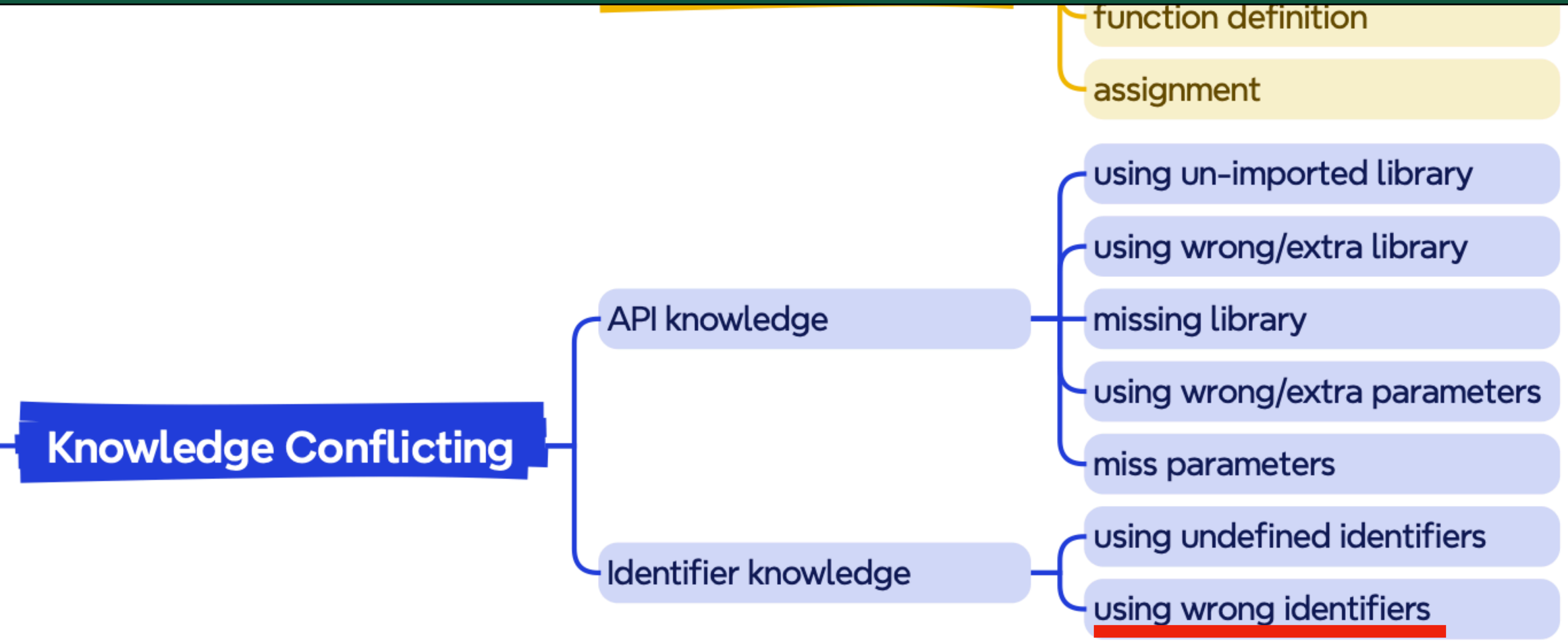
Repeated within the generation

Decoding Code Hallucinations in LLMs



Exploring and Evaluating Hallucinations in LLM-Powered Code Generation.
Liu et al.

Decoding Code Hallucinations in LLMs



Decoding Code Hallucinations in LLMs

- function definition
- assignment
- using un-imported library
- using wrong/extra library
- missing library
- using wrong/extra parameters
- miss parameters
- using undefined identifiers
- using wrong identifiers

```
def longest(strings: List[str]) -> Optional[str]:
    """ Out of list of strings, return the longest
        one.Return the first one in case of multiple
        strings of the same length. Return None in case
        the input list is empty.
    >>> longest([])

    >>> longest(['a', 'b', 'c'])
    'a'
    >>> longest(['a', 'bb', 'ccc'])
    'ccc'
    """
    if not strings:
        return None
# generated by CodeRL
max_len_idx = 0
max_len_str = None
for idx, curr_str in enumerate(strings):
    if len(curr_str) > max_len_len_str:
        max_len_str = curr_str
        max_len_idx = idx
# reference solution
maxlen = max(len(x) for x in strings)
    for s in strings:
        if len(s) == maxlen:
            return s
```

Decoding Code Hallucinations in LLMs

Mitigation Approaches

1. ***Precision is Key:*** Craft Precise and constraint-aware prompts!
   "Only select from the bag of tasks provided as input"
2. ***Add Context:*** Simply ensure you make use of In-Context Learning whenever possible
3. ***Log probs:*** Extract the likelihood of the predicted word(s)
We can frame this value(s) in terms of confidence that measures how "confident" is the model when predicting Y, given X
4. ***LLMs as Judge:*** Simply ask e.g., Gemini to evaluate ChatGPT's prediction

Decoding Code Hallucinations in LLMs

Mitigation Approaches

1. ***Precision is Key:*** Craft Precise and constraint-aware prompts!
   "Only select from the bag of tasks provided as input"
2. ***Add Context:*** Simply ensure you make use of In-Context Learning whenever possible
3. ***Log probs:*** Extract the likelihood of the predicted word(s)
   We can frame this value(s) in terms of confidence that measures how "confident" is the model when predicting Y, given X
4. ***LLMs as Judge:*** Simply ask e.g., Gemini to evaluate ChatGPT's prediction

Decoding Code Hallucinations in LLMs

Mitigation Approaches

1. *Precision is Key:* Craft Preicise and constraint-aware prompts! *"Only select from the bag of tasks provided as input"*
2. *Add Context:* Simply ensure you make use of In-Context Learning whenever possible
3. *Log probs:* Extract the likelihood of the predicted word(s) We can frame this value(s) in terms of confidence that measures how "confident" is the model when predicting Y, given X
4. *LLMs as Judge:* Simply ask e.g., Gemini to evaluate ChatGPT's prediction

Decoding Code Hallucinations in LLMs

Mitigation Approaches

1. ***Precision is Key:*** Craft Precise and constraint-aware prompts!
   "Only select from the bag of tasks provided as input"
2. ***Add Context:*** Simply ensure you make use of In-Context Learning whenever possible
3. ***Log probs:*** Extract the likelihood of the predicted word(s)
We can frame this value(s) in terms of confidence that measures how "confident" is the model when predicting Y, given X
4. ***LLMs as Judge:*** Simply ask e.g., Gemini to evaluate ChatGPT's prediction

```
{
  "model": "gpt-4o-mini",
  "messages": [...],
  "logprobs": true,
  "top_logprobs": 5
}
```

Decoding Code Hallucinations in LLMs

Mitigation Approaches

1. *Precision is Key:* Craft Precise and constraint-aware prompts! *“Only select from the bag of tasks provided as input”*
2. *Add Context:* Simply ensure you make use of In-Context Learning whenever possible
3. *Log probs:* Extract the likelihood of the predicted word(s) We can frame this value(s) in terms of confidence that measures how “confident” is the model when predicting Y, given X
4. *LLMs as Judge:* Simply ask e.g., Gemini to evaluate ChatGPT’s prediction