Constraint Modelling with LLMs Using In-Context Learning

Kostis Michailidis, Dimos Tsouros, Tias Guns

European Research Council
Established by the European Commission

DTAI
Declarative Languages & Artificial Intelligence

KU LEUVEN

Constraint Modelling: An Example

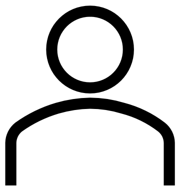
I am an actor with many offers.

Requirement 1: Only one movie per day.

Requirement 2: Maximize income.

Which movies to choose?

	Movie Title	Start Day	End Day
0	Tarjan of the Jungle	4	13
1	The Four Volume Problem	17	27
2	The President's Algorist	1	10
3	Steiner's Tree	12	18
4	Process Terminated	23	30
5	Halting State	9	16
6	Programming Challenges	19	25
7	Discrete Mathematics	2	7
8	Calculated Bets	26	31



Expert

Decision Variables:

$movies_0, movies_1, \dots, movies_8$

Domain:

$movies_i \in \{0, 1\}$

Constraints:

- $movies_0 + movies_2 \leq 1$
- $movies_0 + movies_3 \leq 1$
- $movies_0 + movies_5 \leq 1$
- [...]
- $movies_4 + movies_8 \leq 1$

Objective:

$$\max(\sum_{i=0}^{8} movies_i)$$

Source: http://www.hakank.org/minizinc/movie_scheduling.mzn
Source description: Steven S. Skiena's The Algorithm Design Manual

Constraint Modelling: An Example

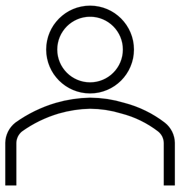
I am an actor with many offers.

Requirement 1: Only one movie per day.

Requirement 2: Maximize income.

Which movies to choose?

	Movie Title	Start Day	End Day
0	Tarjan of the Jungle	4	13
1	The Four Volume Problem	17	27
2	The President's Algorist	1	10
3	Steiner's Tree	12	18
4	Process Terminated	23	30
5	Halting State	9	16
6	Programming Challenges	19	25
7	Discrete Mathematics	2	7
8	Calculated Bets	26	31



Expert

```python
from cpmpy import *

# Data (title, start, end)
movies = [["Tarjan of the Jungle", 4, 13],
    ["The Four Volume Problem", 17, 27],
    ["The President's Algorist", 1, 10],
    ["Steiner's Tree", 12, 18],
    ["Process Terminated", 23, 30],
    ["Halting State", 9, 16],
    ["Programming Challenges", 19, 25],
    ["Discrete Mathematics", 2, 7],
    ["Calculated Bets", 26, 31]]
num_movies = len(movies)

# Decision Variables
selected_movies = boolvar(shape=num_movies)  # 1 if the movie is selected, 0 otherwise

# CP Model
model = Model()

# Non-overlapping movie schedules
for i in range(num_movies):
    for j in range(num_movies):
        # Check if the intervals overlap for each pair of movies
        if i != j and movies[i][2] > movies[j][1] and movies[j][2] > movies[i][1]:
            # Constraint: the movies cannot be selected together
            model += ~(selected_movies[i] & selected_movies[j])

# Objective: Maximize the number of selected movies
model.maximize(sum(selected_movies))

# Solve
model.solve()
print(selected_movies.value())
# Output:
# [False False False False False  True  True  True  True]
```

Source: http://www.hakank.org/minizinc/movie_scheduling.mzn
Source description: Steven S. Skiena's The Algorithm Design Manual

CP is powerful, but modelling requires expertise in:

- Domain Knowledge
- Mathematical Skills
- Solver Formalisms

Making CP More Accessible

Modelling Tools

Constraint Acquisition

Users provide or classify (non-)solutions

From Natural Language

Textual descriptions of combinatorial problems

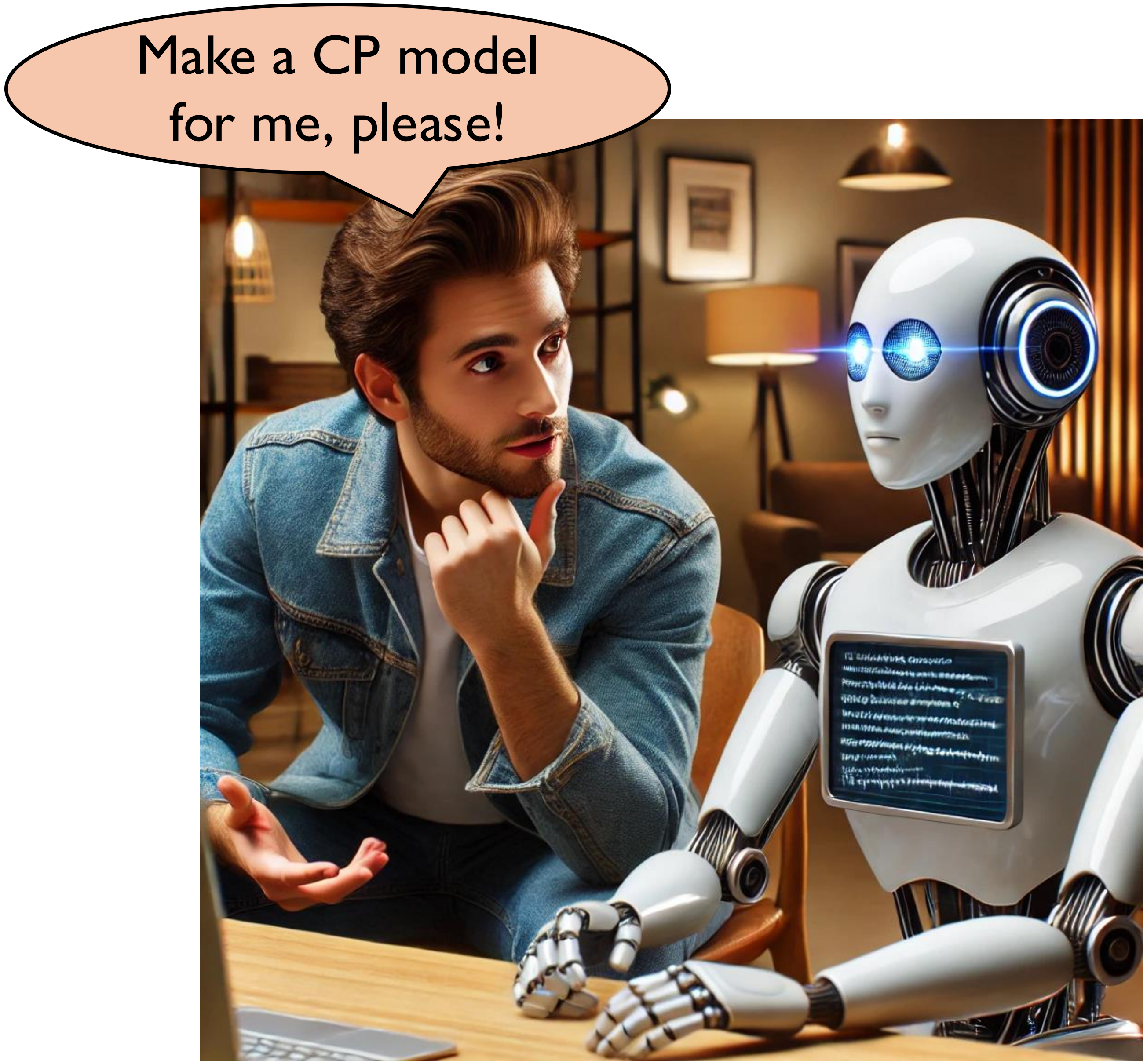
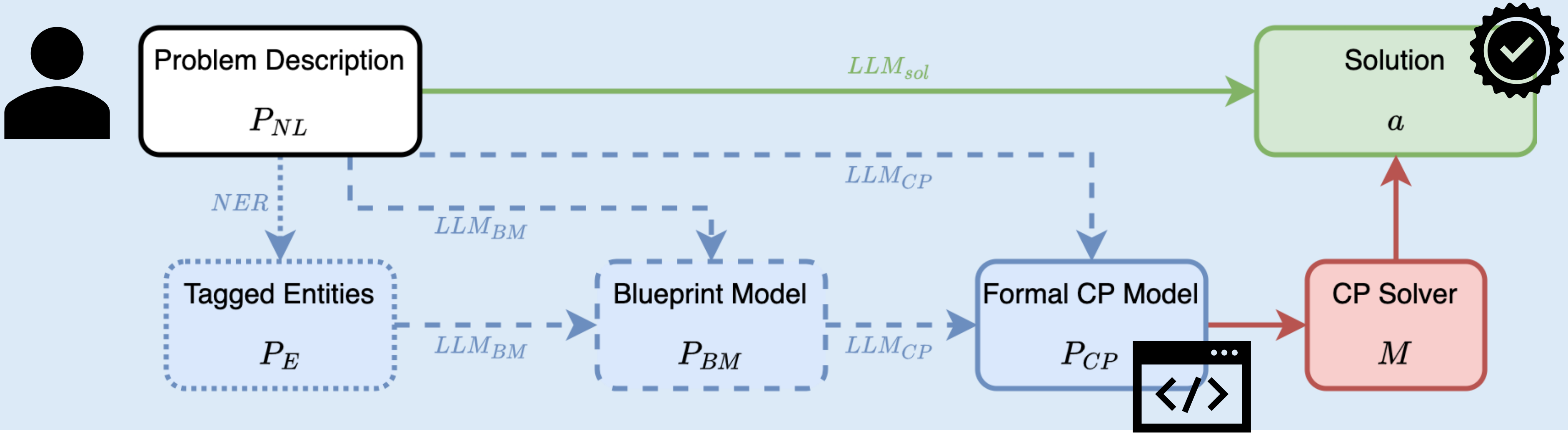


Image created by OpenAI's ChatGPT.

Can
Large Language Models
assist in
Constraint Modelling?

Methodology



Background: Large Language Models

$$LLM(Prompt) = Response$$

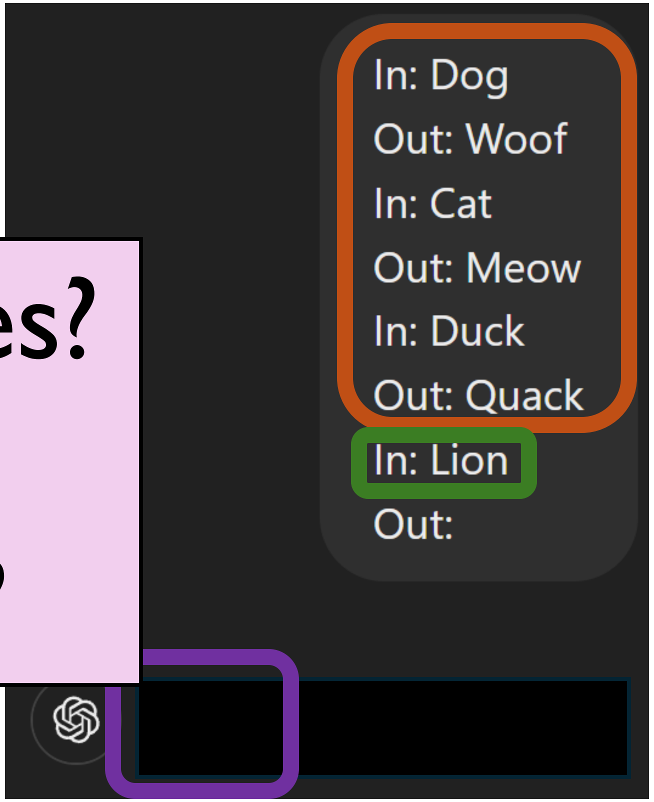
In-Context Learning (ICL):

An emergent ability of modern LLMs (>GPT-3).

1. What examples?
2. How many?
3. In what order?

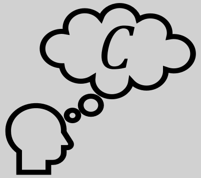
- Task Learning:

$$LLM(\boxed{C} \oplus \boxed{Q}) \approx \boxed{R}$$



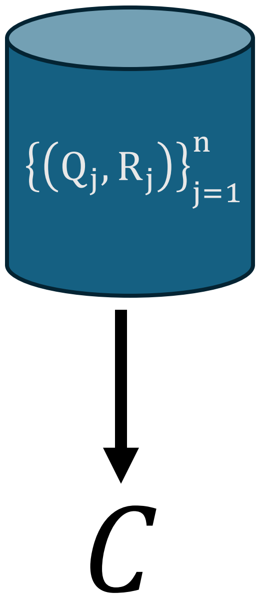
https://chatgpt.com/share/431e9c84-1fe4-412e-aa91-1f9f00063e9d

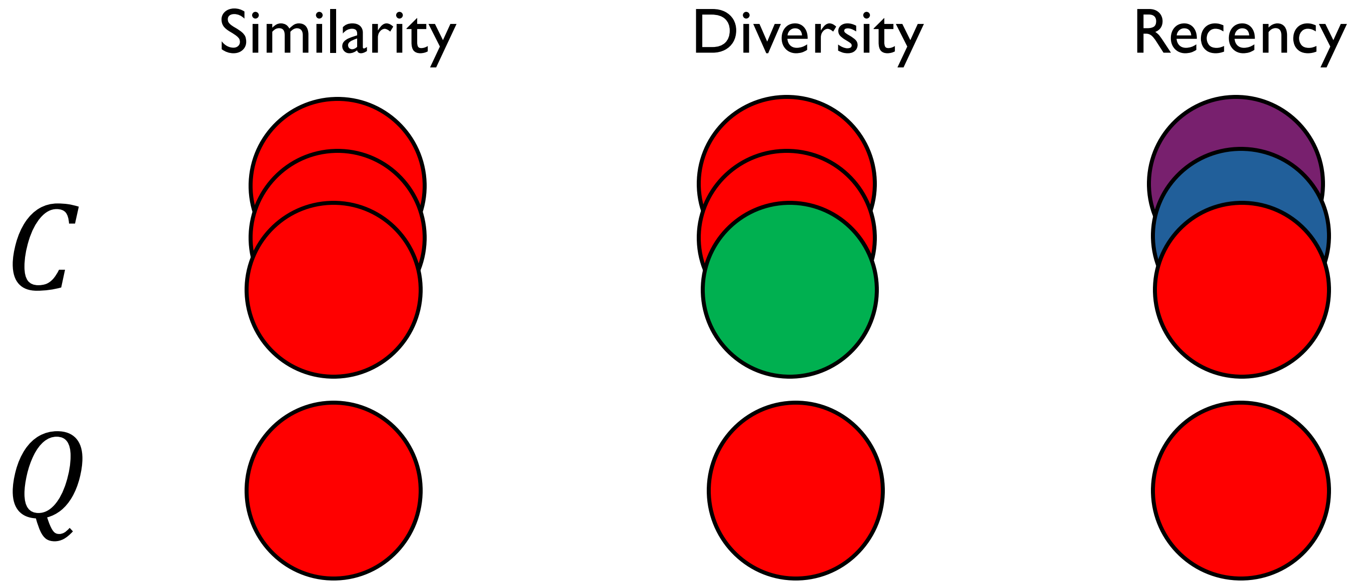
How to choose C?



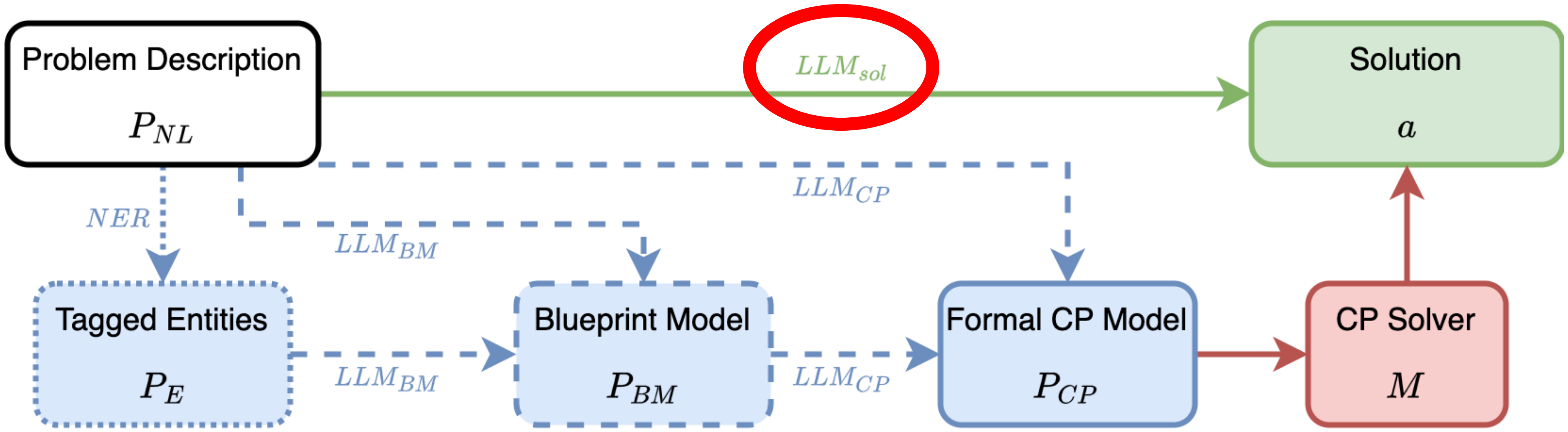
Baseline: Static/Random

Retrieval-Augmented ICL

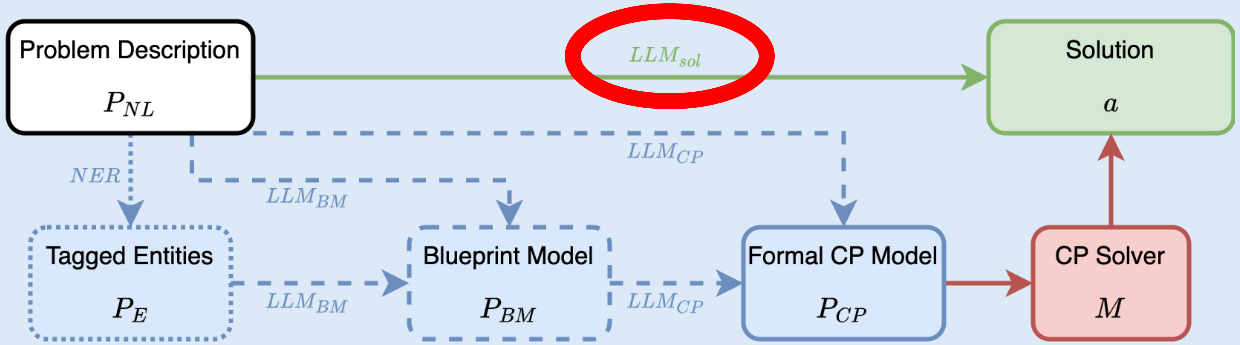




1. LLMs as solvers



1. LLMs as solvers



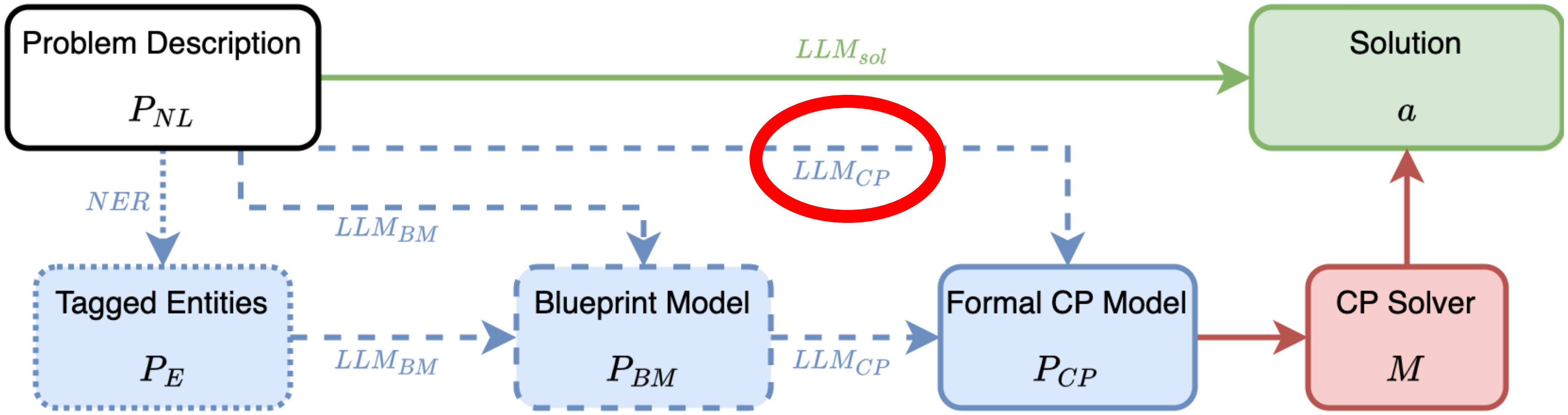
Solution generated with Chain-of-Thought

$$\boxed{a} = LLM(C \oplus Q)$$

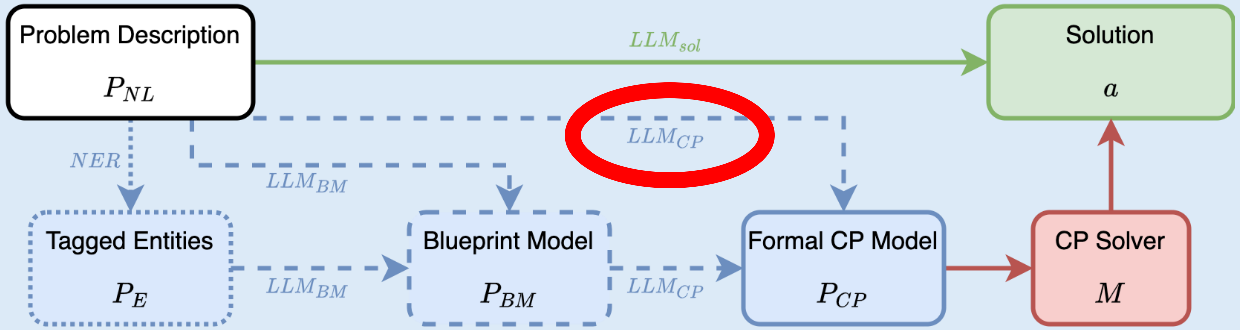
j	Q_j (Problem Desc)	R_j (CoT solution)
1	A popular actor… Which movies to choose?	Let's think step by step… Thus, the final solution is: $\{movies_0: 0,\ movies_1: 0,\ \ldots\}$
…	…	…
n	…	…

$$\{(\mathrm{Q_j}, \mathrm{R_j})\}_{\mathrm{j}=1}^{\mathrm{n}}$$

2. LLMs as modellers



2. LLMs as modellers



Generation of CP models:

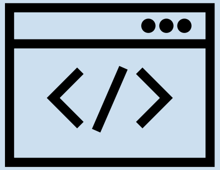
$$P_{CP} = LLM(C \oplus Q)$$

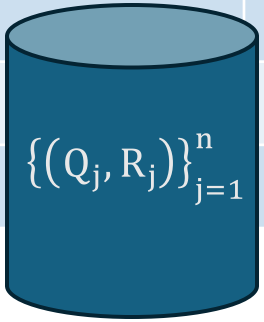
$$\boxed{\boldsymbol{a = M(P_{CP})}}$$

j	Q_j (Problem Desc)	R_j (CP Model)

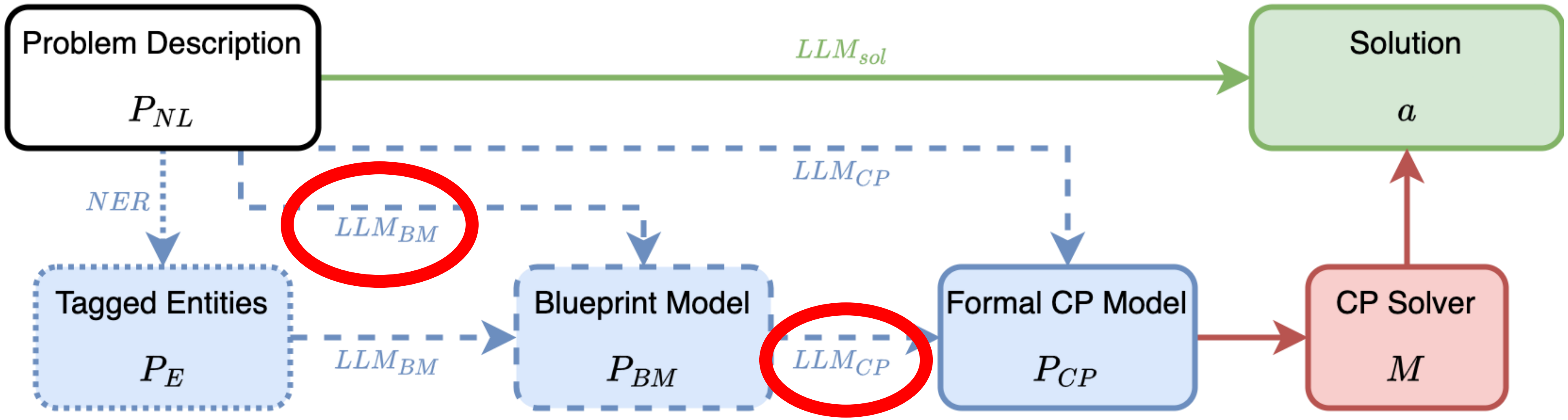
1	A popular actor…
	
	 Which movies to choose?
		from cpmpy import *
		
		 # Data
		 movies = [{"title": "Tarjan of the Jungle", "interval": [4, 13]}, …]
		 # Constraints
		 …
		 # Solve
		 model.solve()

…	…	…
n	…	…

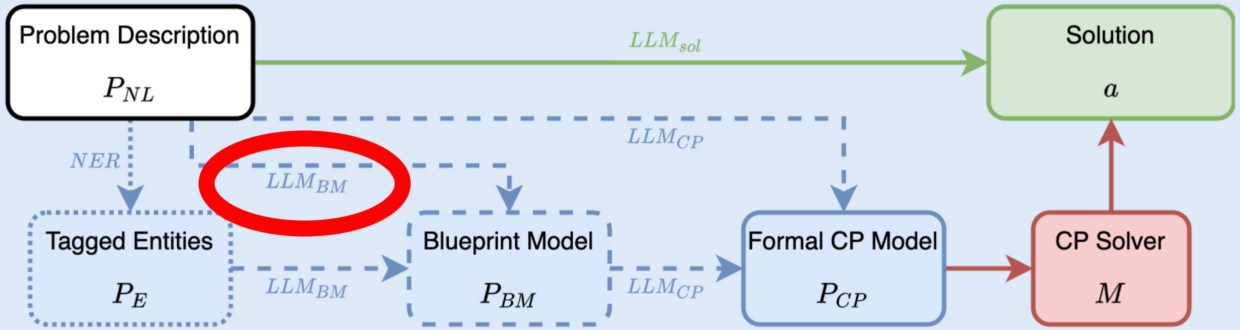




3. Blueprint Model



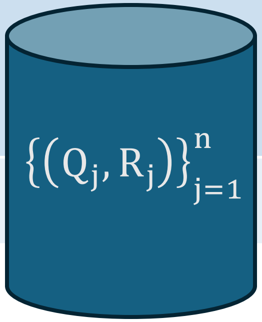
3. Blueprint Model



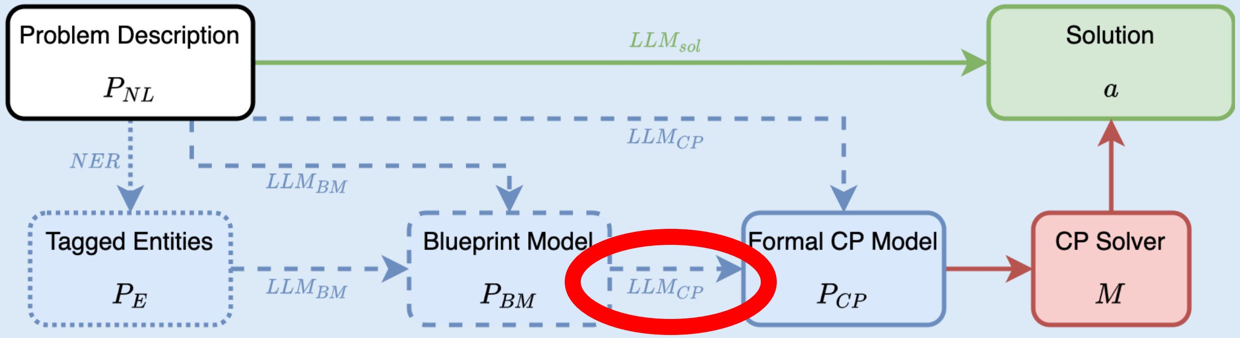
Decompose Model Generation:

$$P_{BM} = LLM(C \oplus Q)$$

j	Q_j (Problem Desc)	R_j (Blueprint Model)
1	A popular actor… Which movies to choose?	# Decision Variables: Define binary variables to represent whether the movie is selected (1) or not (0). The variable 'selected_movies' is an array of size equal to the number of movies (9 in this case). # Domains: Each variable in 'selected_movies' has a domain of [0, 1] where 1 represents the movie is selected and 0 represents the movie is not selected. # Constraints: Non-overlapping: Ensure that the intervals of selected movies do not overlap. This can be represented by iterating over the movies and for each pair of selected movies (i, j), add a constraint that the intervals of the movies do not overlap. # Objective: Maximize the number of selected movies, which is equivalent to maximizing the sum of the 'selected_movies' array.
…		…



3. Blueprint Model



Decompose Model Generation:

$$P_{BM} = LLM(C \oplus Q)$$

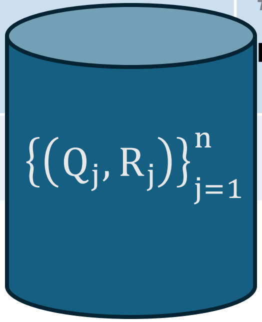
$$P_{CP} = LLM(C \oplus Q \oplus P_{BM})$$

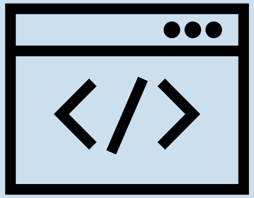
$$\boxed{a = M(P_{CP})}$$

j	Q_j (Problem Desc $\oplus$ Blueprint Model)	R_j (CP Model)

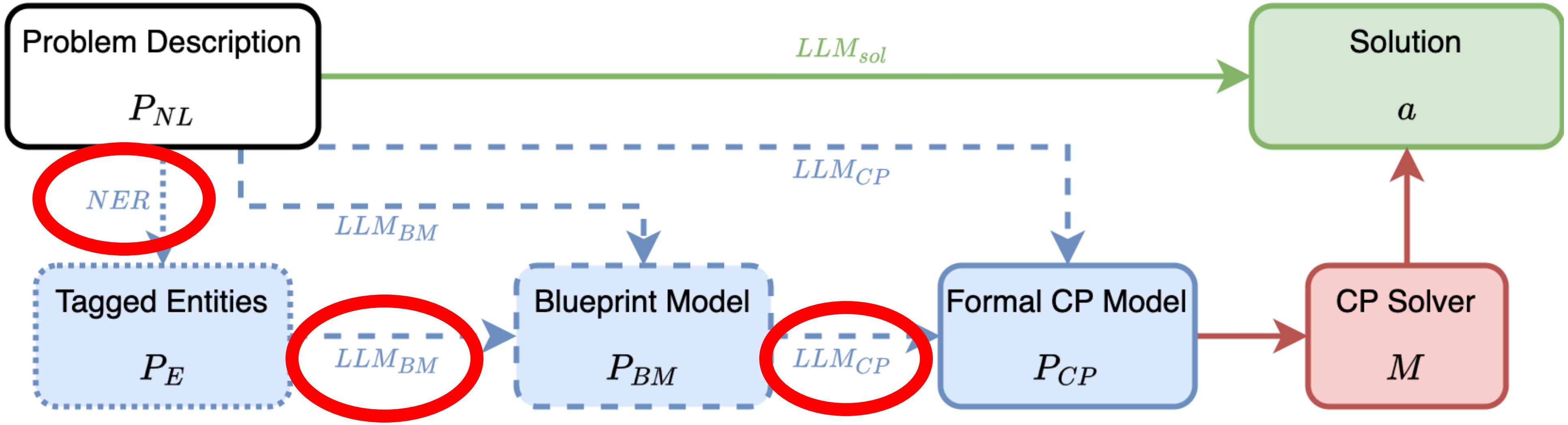
1	A popular actor…
	Which movies to choose?
	<Blueprint Model>
		from cpmpy import *
		
		# Data
		movies = [{"title": "Tarjan of the Jungle", "interval": [4, 13]}, …]
		# Constraints
		…
		# Solve
		model.solve()

…	…	…

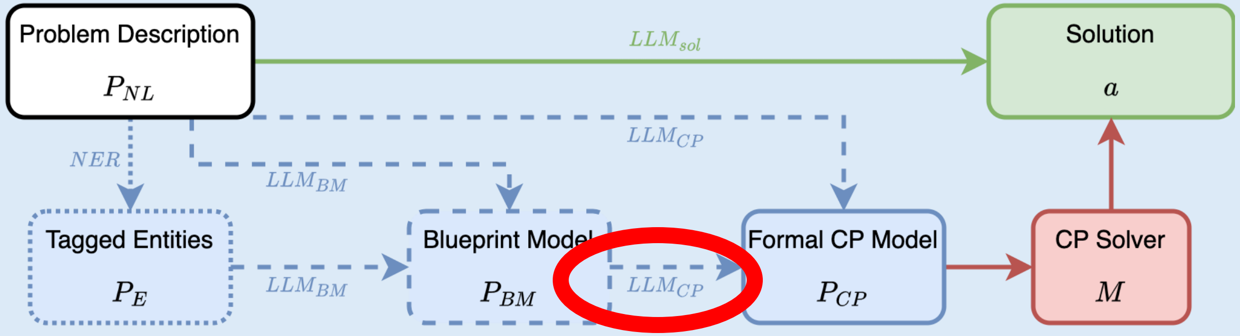




4. Named Entity Recognition



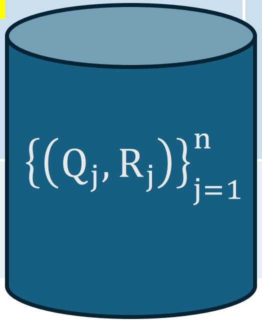
4. Named Entity Recognition



j	Q_j (Problem Desc)	R_j (CP Model)

1	A popular actor…
	
	Which movies to choose?
	
	CONST (36-41): only one
	OBJ_DIR (44-49): maximize
	…
	
	<Blueprint Model>
		from cpmpy import *
		
		# Data
		movies = [{"title": "Tarjan of the Jungle", "interval": [4, 13]}, …]
		# Constraints
		…
		# Solve
		model.solve()

…	…	…



Evaluation Metrics

Declaration Accuracy

Model Accuracy

Solution Accuracy

```
# Decision Variables
selected_movies = boolvar(shape=num_movies)  # 1 if the movie is selected, 0 otherwise

# CP Model
model = Model()

# Non-overlapping movie schedules
for i in range(num_movies):
    for j in range(num_movies):
        # Check if the intervals overlap for each pair of movies
        if i != j and movies[i][2] > movies[j][1] and movies[j][2] > movies[i][1]:
            # Constraint: the movies cannot be selected together
            model += ~(selected_movies[i] & selected_movies[j])

# Objective: Maximize the number of selected movies
model.maximize(sum(selected_movies))

# Solve
model.solve()
print(selected_movies.value())
# Output:
# [False False False False False  True  True  True  True]
```

#Errors: CP Model could not be run

Experiments: Datasets

- **NL4Opt**[1]: Linear Optimisation Problems
- **LGPs**[2]: Logic Grid Puzzles

Homogeneous

- **Mixed**: 18 diverse problems drawn from a CP Modelling course.

New

[1] Rindranirina Ramamonjison, Timothy Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, et al. Nl4opt competition: Formulating optimization problems based on their natural language descriptions.

[2] Arindam Mitra and Chitta Baral. Learning to automatically solve logic grid puzzles.

LLMs as CP solvers? No.
LLMs as CP modellers? Promising.

Dataset	Method	Accuracy (%)
NL4Opt	CoT Solving	11.46
	CP Modelling	81.31
LGPs	CoT Solving	9.36
	CP Modelling	57.00
Mixed	CoT Solving	16.67
	CP Modelling	50.00

Static 4-shot ICL, with gpt-3.5-turbo-0125

Intermediate Representations

Configuration: gpt-3.5-turbo-0125, 4-shot static ICL.

Dataset	Method	#Err	Sol. (%)	Decl. (%)	Model (%)
NL4Opt	CP	7	81.31	87.81	79.24
	+ BM	8	84.43	89.93	82.01
	+ NER	8	85.47	88.60	80.62
LGPs	CP	11	57.00	80.45	55.00
	+ BM	18	58.00	70.69	58.00
	+ NER	20	54.00	67.77	50.00

Linear Opt. Problems

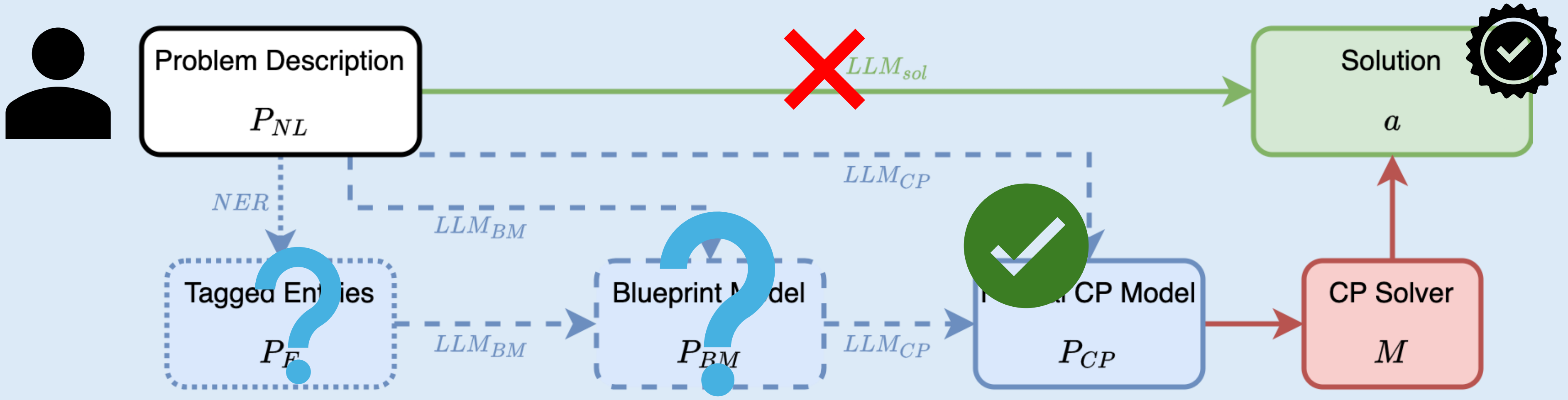
For more complex problems:

BM: Larger Context

NER: Variable Misclassification

Dataset	#Shots	CP	+ BM
Mixed	2	33.33%	16.67%
	4	50.00%	50.00%
	8	50.00%	44.44%
	12	55.56%	50.00%

Configuration: gpt-3.5-turbo-0125, static ICL, Solution Accuracy.


Problem Description
P_{NL}
LLM_{sol}
Solution
a
LLM_{CP}
NER
LLM_{BM}
CP Model
P_{CP}
CP Solver
M
LLM_{BM}
LLM_{CP}

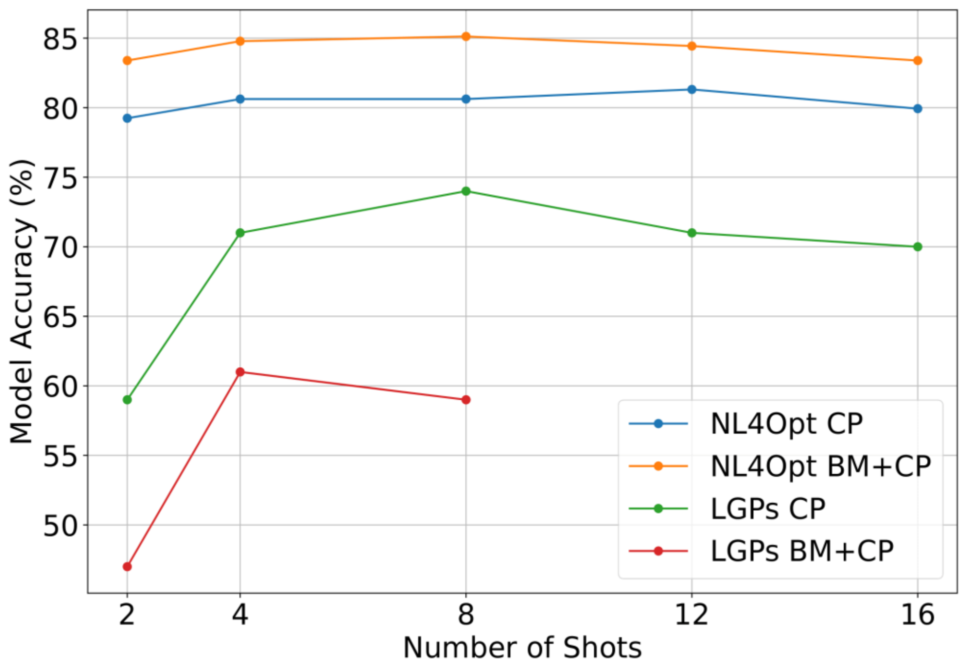
Retrieval-Augmented ICL? Yes.

- ✓Relevance
- ✓Diversity
- ✓Recency

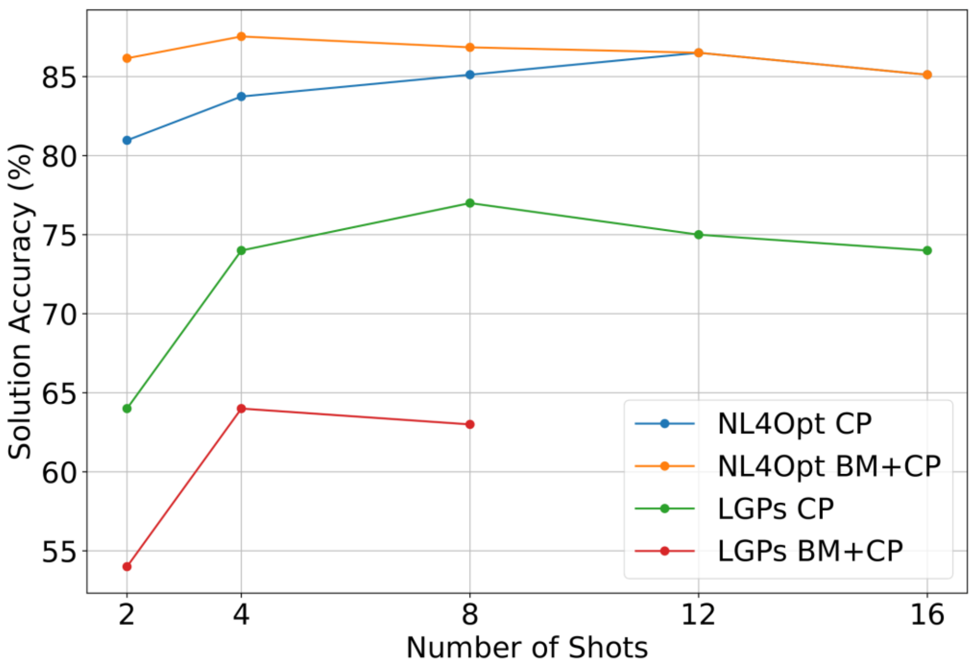
Retrieval	LGPs Sol. (%)	LGPs Decl. (%)	NL4Opt Sol. (%)	NL4Opt Decl. (%)
Static	57.00	80.45	81.31	87.81
Random	66.00	76.75	77.16	85.60
Similarity	68.00	85.34	85.12	87.72
R-Similarity	72.00	89.83	83.39	88.16
Diversity	66.00	83.09	84.08	87.99
R-Diversity	74.00	87.98	83.74	87.10
Random w/ Last-Similar	76.00	89.96	83.74	88.34

CP modelling. Config: gpt-3.5-turbo-0125, R-MMR $\lambda = 0.5$.

The more examples the better? Not exactly.



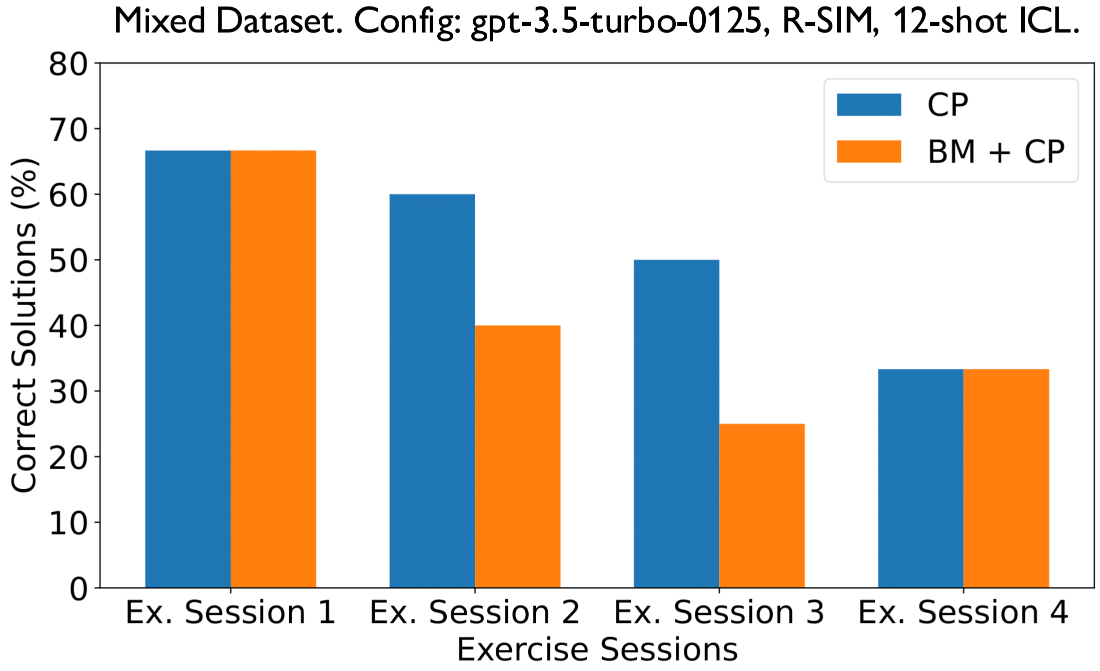
Config: gpt-3.5-turbo-0125, R-MMR (λ = 0.5)



Adding in-context examples improves **up to a threshold.**

What about diverse problems?

- Few examples to retrieve from.
- More complex problems.



<u>Evaluation Challenge:</u>

Choice of Decision Variables is free => ~~Model Accuracy.~~

Prompt LLMs to generate code that prints the solution in specific format => **Solution Accuracy**.

Summary

1. An LLM-based NL2CP framework
2. Intermediate Problem Representation
3. Evaluation of RAG strategies
4. Model Comparison Metrics
5. Initiated & Augmented Datasets

Key Conclusions

LLMs as CP solvers? No
LLMs as CP modellers? **Promising**

Retrieval-Augmented In-Context
Learning **boosts** performance

! Modelling more complex problems
remains **challenging**

! **Missing** Evaluation Datasets:
From natural language to CP models

Future Work

Main:

- High-Quality NL-CP Evaluation Dataset.
- Integrate coding techniques.
- Interactive modelling system.

Other:

- Beyond ICL: Supervised Fine-Tuning
- Large external data.

Demo

https://chatopt.cs.kuleuven.be

Credits: Thomas Sergeys

Thank you!

https://chatopt.cs.kuleuven.be