HOLY GRAIL 2.0: FROM NATURAL LANGUAGE TO CONSTRAINT MODELS

PTHG 2023

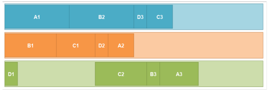
Dimos Tsouros[1], Hélène Verhaeghe[1], Serdar Kadıoğlu[2], Tias Guns[1]

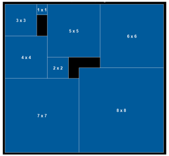
27 August 2023

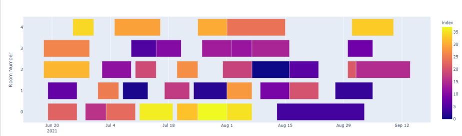
[1] KULeuven, Leuven, Belgium, *dimos.tsouros@kuleuven.be,helene.verhaeghe@kuleuven.be,tias.guns@kuleuven.be*
[2] AI Center of Excellence, Fidelity Investments, USA, Department of Computer Science, Brown University, USA, *serdar.kadioglu@fmr.com*

CONSTRAINT PROGRAMMING

- Constraint programming (CP)
  - Solving combinatorial problems in AI
  - Assignment problems, scheduling and timetabling, bioinformatics, vehicle routing and more



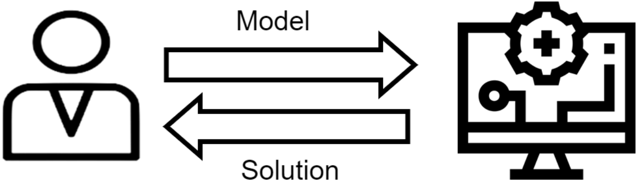




HOLY GRAIL OF COMPUTER PROGRAMMING

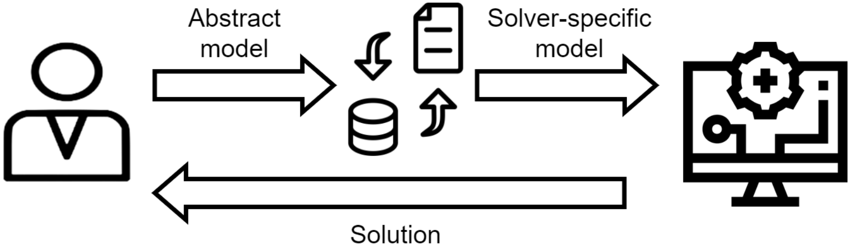
E. Freuder: ”Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it”

- Model + Solve paradigm



PROGRESS TOWARDS THE HOLY GRAIL

- Advantages in modeling languages and other tools
  - Unburden the user with technical details for each solver
  - Closer to the stated goal
  - Showed that CP has the capability to achieve the stated goal



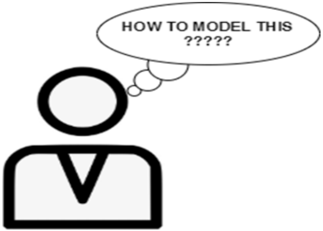
Abstract model

Solver-specific model

Solution

HOLY GRAIL 2.0: STATING THE PROBLEM IN NATURAL LANGUAGE

- Still a gap between the natural description of a problem and its formal formulation as an optimization problem
- Expertise is needed to model the problem as a constraint model



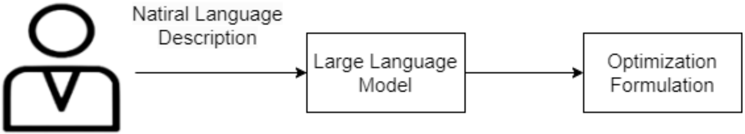
NATURAL LANGUAGE TO OPTIMIZATION MODELS

LARGE LANGUAGE MODELS

- Huge recent advancements using Large Language Models (LLMs)
  - Many succesfull LLMs (Bart, GPT etc.)
  - Applications like: Chatbots (ChatGPT), code generators (Codex) translators …
  - Can be specialized to specific applications using fine-tuning or few-shot learning
  - Can handle complex inputs, “seeing” connections between entities in paragraphs

LLMS FOR MODELING?

- Potential to use LLMs to process textual descriptions of constraint models!

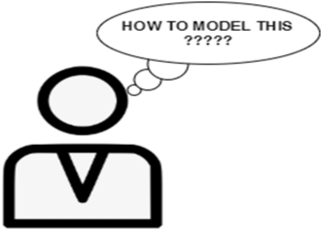


- Can work well for simpler well-defined problems
- Does it work well if not simple and not precisely defined?

MODELING IS NOT A SIMPLE TASK

Modeling is not a simple natural language processing task

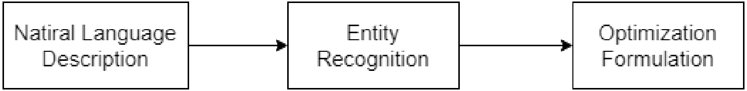
- Entity recognition in the textual description (Variables, constraints, objective)
- Formulation as an optimization problem
- Write code to modeling language?



RECENT WORKS

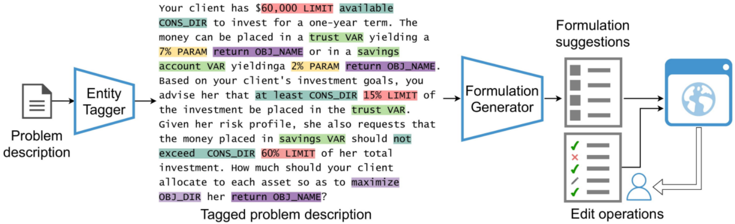
Many recent works have started to focus on Natural Language for Optimization (NL4OPT)

- Extract the optimization formulation from textual descriptions
- Different than many NLP tasks:
  - input text can be unstructured and ambiguous, especially when it describes many constraints of different types
- Mainly breaking the problem into 2 subtasks: Entity Recognition and Formulation



RECENT WORKS

- The problem of NL4OPT was firstly proposed in 2022[1],
  - The 2 subtasks were formally presented
  - The first dataset for these problems was introduced



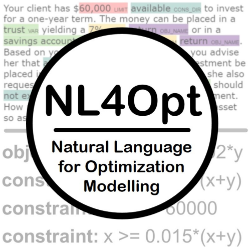
Your client has $60,000 LIMIT available CONS_DIR to invest for a one-year term. The money can be placed in a trust VAR yielding a 7% PARAM return OBJ_NAME or in a savings account VAR yieldinga 2% PARAM return OBJ_NAME. Based on your client's investment goals, you advise her that at least CONS_DIR 15% LIMIT of the investment be placed in the trust VAR. Given her risk profile, she also requests that the money placed in savings VAR should not exceed CONS_DIR 60% LIMIT of her total investment. How much should your client allocate to each asset so as to maximize OBJ_DIR her return OBJ_NAME?

- LLMs were exploited to tackle the 2 subtasks
  - a pre-trained transformer is used (XLM-RoBERTa), fine-tuned for entity recognition,
  - a BART-based model is used for prompt-based formulation,

[1] Ramamonjison et al., Augmenting Operations Research with Auto-Formulation of Optimization Models from Problem Descriptions, EMNLP 2022

RECENT WORKS

- Based on the work from[2], a competition on NL4OPT took place in NeurIPS 2022[3].
  - Several participants
  - 1 challenge for each subtask
  - Good results in both subtasks
  - Focusing on LP problems



Your client has $60,000 LIMIT available CONS_DIR to invest
for a one-year term. The money can be placed in a
trust VAR yielding a 7% return OBJ_NAME or in a
savings accoun return OBJ_NAME.
Based on y you advise
her that stment be
placed i she also
reques should
not ex ment.
How sset
so as

obj 2*y
cons (x+y)
constrai 60000
constraint: x >= 0.015*(x+y)

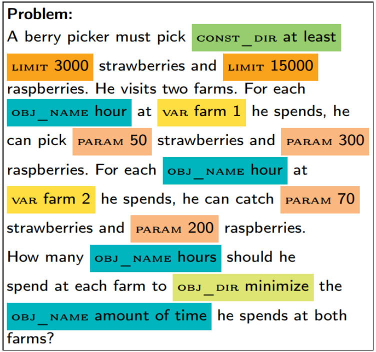
NEURAL INFORMATION PROCESSING SYSTEMS

[2] Ramamonjison et al., Augmenting Operations Research with Auto-Formulation of Optimization Models from Problem Descriptions, EMNLP 2022

[3] Ramamonjison et al., NL4Opt Competition: Formulating Optimization Problems Based on Their Natural Language Descriptions, NeurIPS 2022

ENTITY RECOGNITION (NER4OPT)

- Recognize optimization entities in textual descriptions[4]
- Optimization entities:
  - Parameters
  - Variables
  - Constraint direction
  - Objective direction
  - Objective



Problem:
A berry picker must pick CONST_DIR at least LIMIT 3000 strawberries and LIMIT 15000 raspberries. He visits two farms. For each OBJ_NAME hour at VAR farm 1 he spends, he can pick PARAM 50 strawberries and PARAM 300 raspberries. For each OBJ_NAME hour at VAR farm 2 he spends, he can catch PARAM 70 strawberries and PARAM 200 raspberries. How many OBJ_NAME hours should he spend at each farm to OBJ_DIR minimize the OBJ_NAME amount of time he spends at both farms?

[4] Dakle et al. ”Ner4Opt: Named Entity Recognition for Optimization Modelling from Natural Language”, CPAIOR 2023

ENTITY RECOGNITION (NER4OPT)

Differs from the standard named entity recognition (NER) in important ways stemming from the optimization context

- Different entities
- Same entities with different semantic theme
- Multi-sentence word problem with high-level of compositionality, ambiguity, variability
- Ner4Opt must be domain agnostic and generalize to new instances and applications
- Different view-points
- Extremely limited training data. Even human annotation requires expertise. Must operate on low-resource regime

[4] Dakle et al. ”Ner4Opt: Named Entity Recognition for Optimization Modelling from Natural Language”, CPAIOR 2023

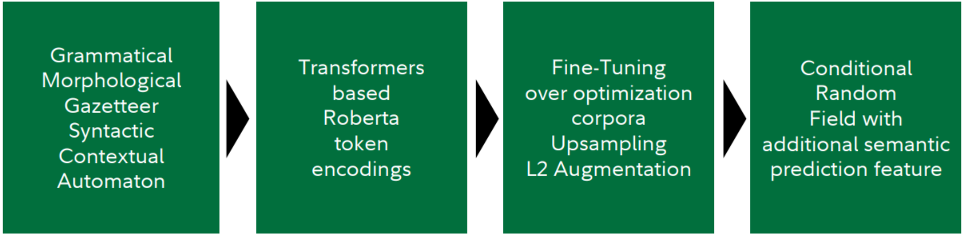
HYBRID METHOD FOR NER4OPT

Hybrid Modeling

Feature Engineering + Feature Learning

Feature engineering might be brittle but helps build apriori information

Feature learning brings semantic representations but struggles with long-range dependency



Grammatical
Morphological
Gazetteer
Syntactic
Contextual
Automaton

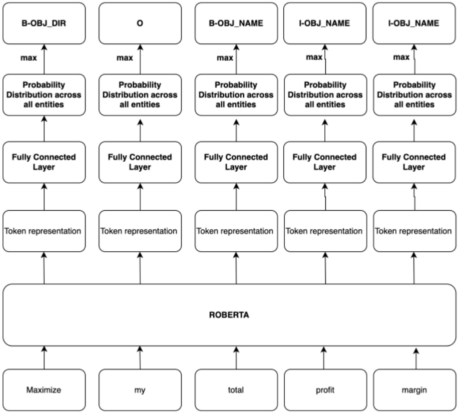
Transformers based Roberta token encodings

Fine-Tuning over optimization corpora
Upsampling
L2 Augmentation

Conditional Random Field with additional semantic prediction feature

[4] Dakle et al. ”Ner4Opt: Named Entity Recognition for Optimization Modelling from Natural Language”, CPAIOR 2023

FORMULATING NER4OPT AS TOKEN CLASSIFICATION



Maximize my total profit margin

❑ Token classification problem with encoders

❑ Roberta embeddings with 1024 dimensions

❑ A fully-connected layer of size 1024 learns to map token level embeddings into named-entity-labels

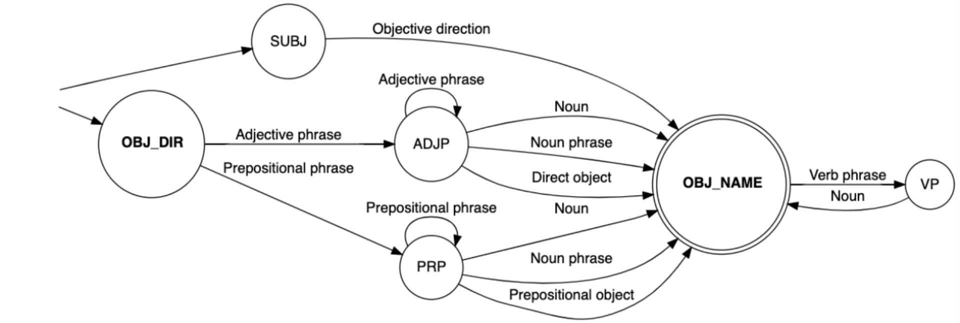
❑ Followed by softmax activation function to output dimension of 1 x 13

❑ Minimize training loss with cross-entropy loss

[4] Dakle et al. ”Ner4Opt: Named Entity Recognition for Optimization Modelling from Natural Language”, CPAIOR 2023

REGULAR AUTOMATON FOR NAME EXTRACTION

Objective name and variables have many similarities To avoid confusion, automatons were created



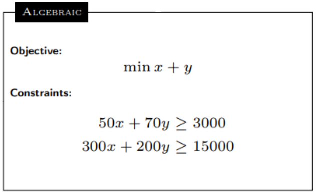
profit SUBJ to be maximized OBJ_DIR

maximize OBJ_DIR the total monthly ADJP profit NOUN

[4] Dakle et al. ”Ner4Opt: Named Entity Recognition for Optimization Modelling from Natural Language”, CPAIOR 2023

FORMULATION PROBLEM

- How are the entities connected with each other?
  - Where are the parameters used?
  - What variables are involved in each constraint?
  - What is the exact mathematical formulation of the constraints and objective?



ALGEBRAIC

Objective:

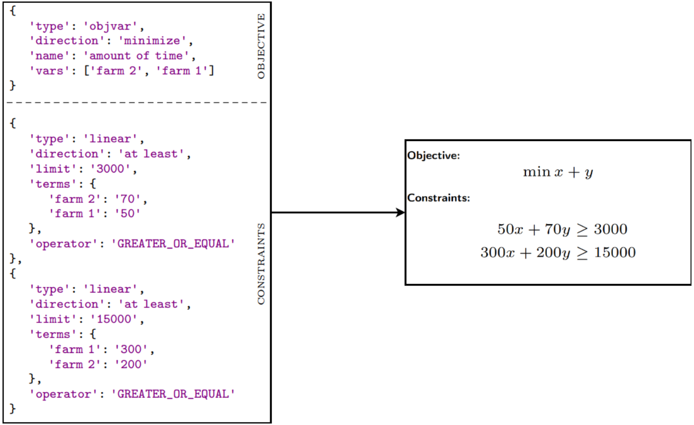
$$\min x + y$$

Constraints:

$$50x + 70y \geq 3000$$
$$300x + 200y \geq 15000$$

FORMULATION PROBLEM: SPLIT TO 2 SUBTASKS AGAIN

- First find what variables/parameters participate in the constraints
- Then extract the mathematical formulation



```
{
   'type': 'objvar',
   'direction': 'minimize',
   'name': 'amount of time',
   'vars': ['farm 2', 'farm 1']
}
------------------------------
{
   'type': 'linear',
   'direction': 'at least',
   'limit': '3000',
   'terms': {
      'farm 2': '70',
      'farm 1': '50'
   },
   'operator': 'GREATER_OR_EQUAL'
},
{
   'type': 'linear',
   'direction': 'at least',
   'limit': '15000',
   'terms': {
      'farm 1': '300',
      'farm 2': '200'
   },
   'operator': 'GREATER_OR_EQUAL'
}
```

Objective:

$$\min x + y$$

Constraints:

$$50x + 70y \geq 3000$$
$$300x + 200y \geq 15000$$

FORMULATING THE OPTIMIZATION PROBLEM

- Augment the input problem description by encapsulating the named entities in corresponding tags
- Prompt-guided input using Bart-base and Bart-Large models
- Extract either one constraint at a time[5] [6][7] or all-at-once[8] [9]

[5] He et al., Linear programming word problems formulation using ensemble crf-NER labeler and t5 text generator with data augmentations, arXiv 2022

[6] Ning, Y. et al., A novel approach for auto-formulation of optimization problems. arXiv, 2023

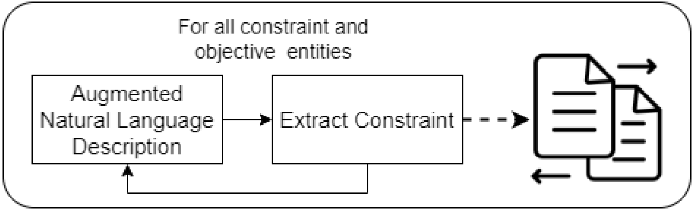
[7] Ramamonjison et al., Augmenting Operations Research with Auto-Formulation of Optimization Models from Problem Descriptions, EMNLP 2022

[8] Jang, S., Tag embedding and well-defined intermediate representation improve auto-formulation of problem description. arXiv, 2022

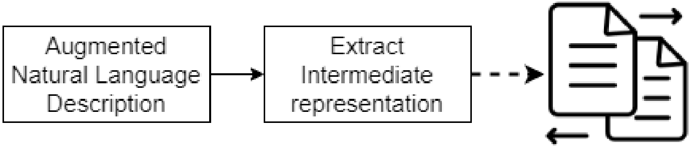
[9] Gangwar et al., Highlighting Named Entities in Input for Auto-Formulation of Optimization Problems, arXiv, 2022

FORMULATING THE OPTIMIZATION PROBLEM

- Extract one constraint at a time

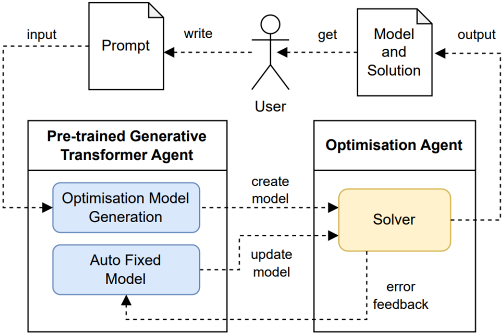

For all constraint and objective entities
Augmented Natural Language Description
Extract Constraint

- Extract whole model all-at-once


Augmented Natural Language Description
Extract Intermediate representation

WORKS ON CP

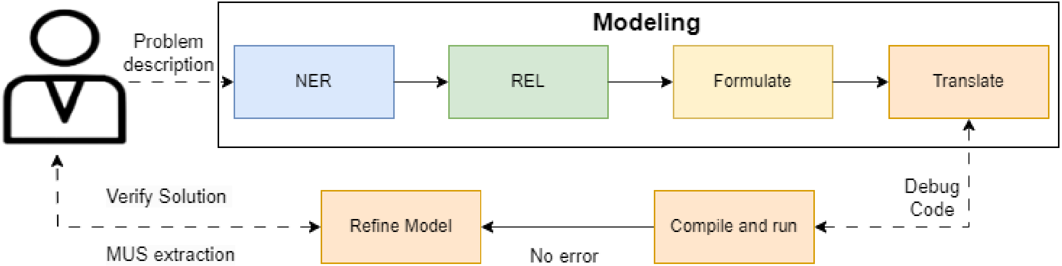
- All mentioned works focus on LP problems
- An approach modeling problems in Minizinc was recently developed[10]
- Extract whole model all-at-once + auto-debugging



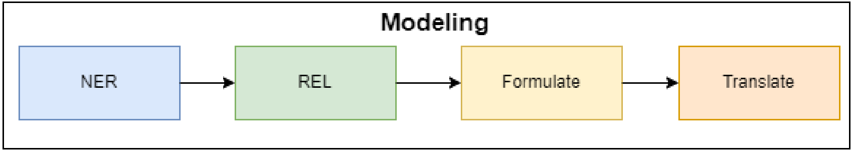
[10] Almonacid, Towards an automatic optimisation model generator assisted with generative pre-trained transformer. arXiv, 2023

OUTLINE OF OUR PROJECT

FRAMEWORK


Modeling
Problem description
NER
REL
Formulate
Translate
Debug Code
Verify Solution
MUS extraction
Refine Model
No error
Compile and run

SUBMODULES EXPLAINATION

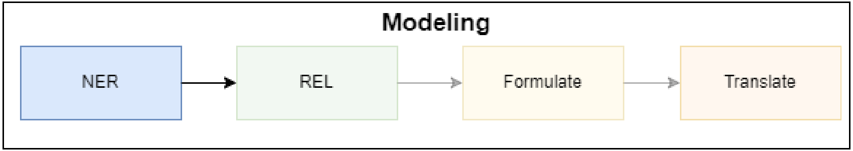


- **NER**: Entity recognition
- Problem formulation in 2 subtasks
  - **REL**: Entity relations recognition
  - **Formulate**: Final formulation
- **Translate** to a modeling language

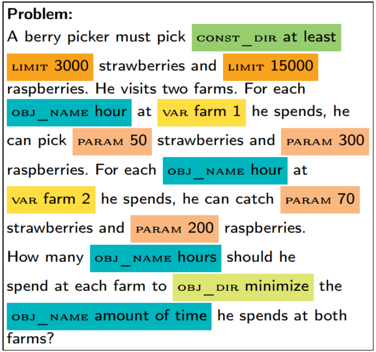
- Any from the existing methods can be used in the first 3 steps
- LLMs have been shown to be effective in code-writing in well-predefined tasks[a]

[a] Xu et al., A systematic evaluation of large language models of code. International Symposium on Machine Programming, 2022

NER



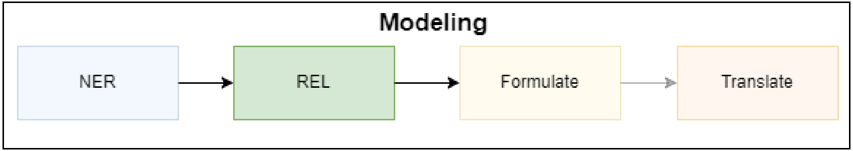
- The NER4OPT[a] task described earlier
- Recognize optimization entities in textual descriptions



Problem:
A berry picker must pick CONST_DIR at least LIMIT 3000 strawberries and LIMIT 15000 raspberries. He visits two farms. For each OBJ_NAME hour at VAR farm 1 he spends, he can pick PARAM 50 strawberries and PARAM 300 raspberries. For each OBJ_NAME hour at VAR farm 2 he spends, he can catch PARAM 70 strawberries and PARAM 200 raspberries. How many OBJ_NAME hours should he spend at each farm to OBJ_DIR minimize the OBJ_NAME amount of time he spends at both farms?

[a] Dakle et al. ”Ner4Opt: Named Entity Recognition for Optimization Modelling from Natural Language”, CPAIOR 2023

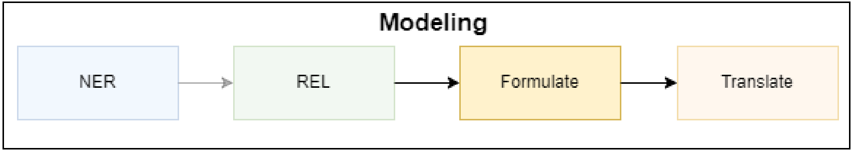
REL



- First task of problem formulation: Find entities relations
  - Input: Entities from the NER step and problem description
  - Output: List of other entities participating in each entity

- Variables: <Hours at Farm 1> <50 strawberries/hour, 300 raspberries/hour>, <Hours at Farm 2> <70 strawberries/hour, 200 raspberries/hour>
- Constraints: <Strawberry Constraint> <3000 strawberries, Hours at Farm 1, Hours at Farm 2>, <Raspberry Constraint> <15000 raspberries, Hours at Farm 1, Hours at Farm 2>
- Objective Function: <Total Time> <Hours at Farm 1, Hours at Farm 2>
- Objective Direction: Minimize

FORMULATE



Formulate: Final formulation

Parameters:

- S = 3000 (minimum strawberries to be picked)
- R = 15000 (minimum raspberries to be picked)
- S1 = 50 (strawberries per hour at farm 1)
- R1 = 300 (raspberries per hour at farm 1)
- S2 = 70 (strawberries per hour at farm 2)
- R2 = 200 (raspberries per hour at farm 2)

Variables:

- H1 (hours spent at farm 1)
- H2 (hours spent at farm 2)

Domains:

- H1, H2 >= 0

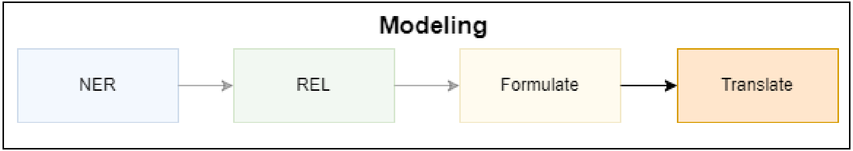
Constraints:

- S1*H1 + S2*H2 >= S (strawberry constraint)
- R1*H1 + R2*H2 >= R (raspberry constraint)

Objective Function:

- Minimize (H1 + H2)

TRANSLATE

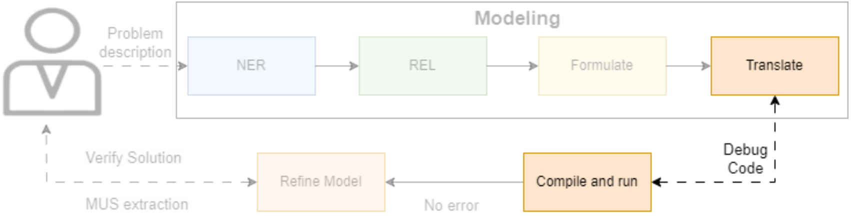


- **Translate** to a modeling language

```
from cpmpy import *

# Parameters
S = 3000
R = 15000
S1 = 50
R1 = 300
S2 = 70
R2 = 200
```

```
# Variables
H1 = intvar(0, int(S/S1 + R/R1)) # hours spent at farm 1
H2 = intvar(0, int(S/S2 + R/R2)) # hours spent at farm 2

# Constraints
model = Model([S1*H1 + S2*H2 >= S, # strawberry constraint
               R1*H1 + R2*H2 >= R]) # raspberry constraint

# Objective Function
model.minimize(H1 + H2)

# Solve the model
model.solve()
```

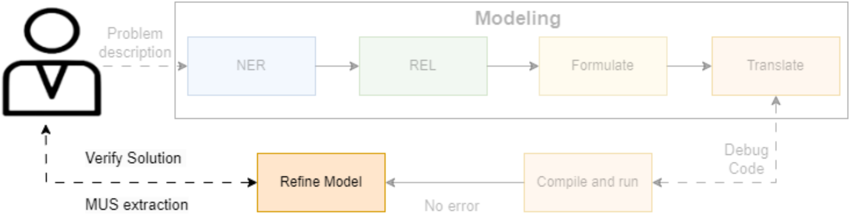
AUTOMATIC DEBUGGING



- After the modeling process is completed, we have the code of the model in a modeling language
- It has been shown that LLMs can be exploited for bug-fixing with good results [11]
- In this step, the code from the previous step is compiled and run.
  - If it outputs a bug it returns to the translation step, using the bug description returned in the prompt
  - Stops when the code runs normally

[11] Sobania et al., An analysis of the automatic bug fixing performance of chatgpt, arXiv, 2023

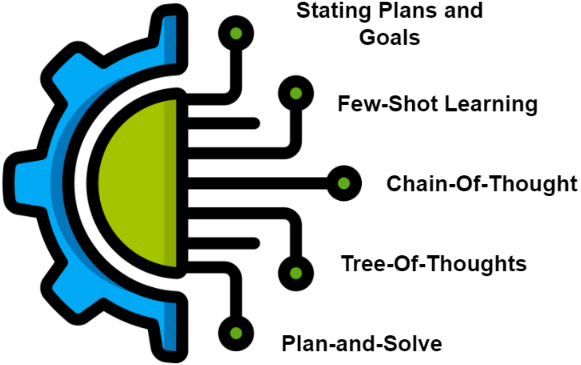
REFINE MODEL



- Present the final model and potential solution(s) to the user
- In case the problem is unsatisfiable extract MUSes and refine the model interactively
- In case the model is incomplete, refine the model using constraint acquisition to complete it with the missing constraints

LEVERAGING LLMS

- Leveraging LLMs in all submodules
- LLMs
  - Work on token-level
  - Predict sequentially the next token.
  - Prompt: User input
- Prompt design plays a crucial role
- **Prompt Engineering**



LEVELS OF ABSTRACTION

- Various levels of abstraction in the problem description
  - Problem and its categorization (knapsack, tsp etc.) stated clearly, describing the variables and constraints.

    > I wish to solve a Knapsack problem, where I have 5 items, and so 5 boolean variables to tell me which items are picked. The weights of my items are 2, 3, 7, 4, and 1 with their utilities being 2, 3, 1, 2, and 3. The limit of weight is 10.

  - Omit the problem category but still state it clearly, describing the variables and constraints.

    > I have 5 items, with 5 boolean variables showing if they are picked. Their weights are 2, 3, 7, 4, and 1 with their utilities being 2, 3, 1, 2, and 3. The total weight I can carry is 10. I want to maximize the utility of the objects I take.

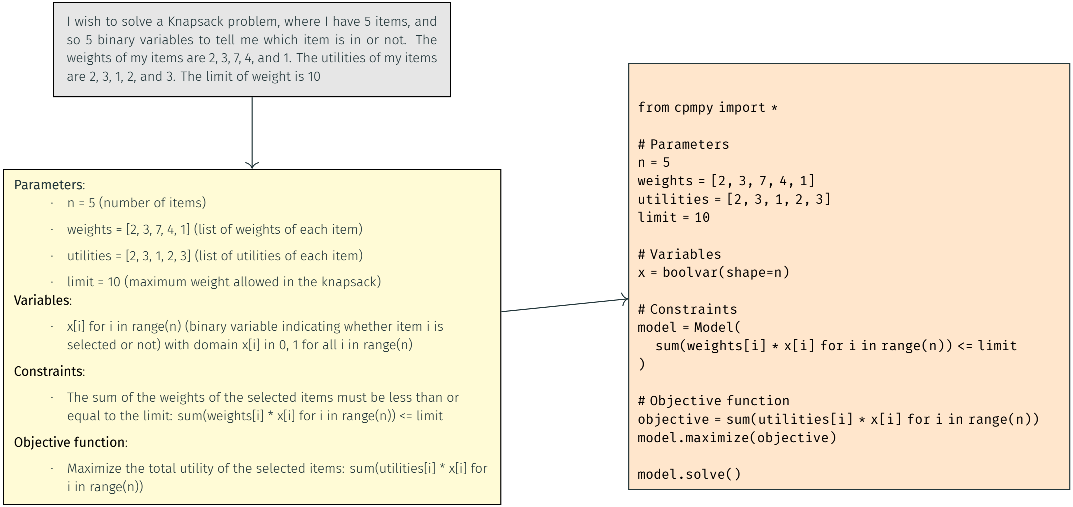
  - Omit also the use of modeling lexical (variable, constraint), stating the problem as non-expert.

    > I want to go on vacation. The airport only allows 10 kg for my luggage. I have 5 items, with their weights being 2, 3, 7, 4, and 1 and their utilities being 2, 3, 1, 2, and 3. What items to pick?

  - Most abstract form, where parameters can be implicit

    > I want to go on vacation. The airport allows 10 kg for my luggage. I have 5 items: my ski combination (7kg), some warm clothes (4kg), boots (3kg), a book for hiking (1kg), and an umbrella (2kg). As I'm going hiking, I think my boots and my book are really important, while the ski combination would not help me well. What items should I get with me?

EXAMPLE



I wish to solve a Knapsack problem, where I have 5 items, and so 5 binary variables to tell me which item is in or not. The weights of my items are 2, 3, 7, 4, and 1. The utilities of my items are 2, 3, 1, 2, and 3. The limit of weight is 10

Parameters:

- n = 5 (number of items)
- weights = [2, 3, 7, 4, 1] (list of weights of each item)
- utilities = [2, 3, 1, 2, 3] (list of utilities of each item)
- limit = 10 (maximum weight allowed in the knapsack)

Variables:

- x[i] for i in range(n) (binary variable indicating whether item i is selected or not) with domain x[i] in 0, 1 for all i in range(n)

Constraints:

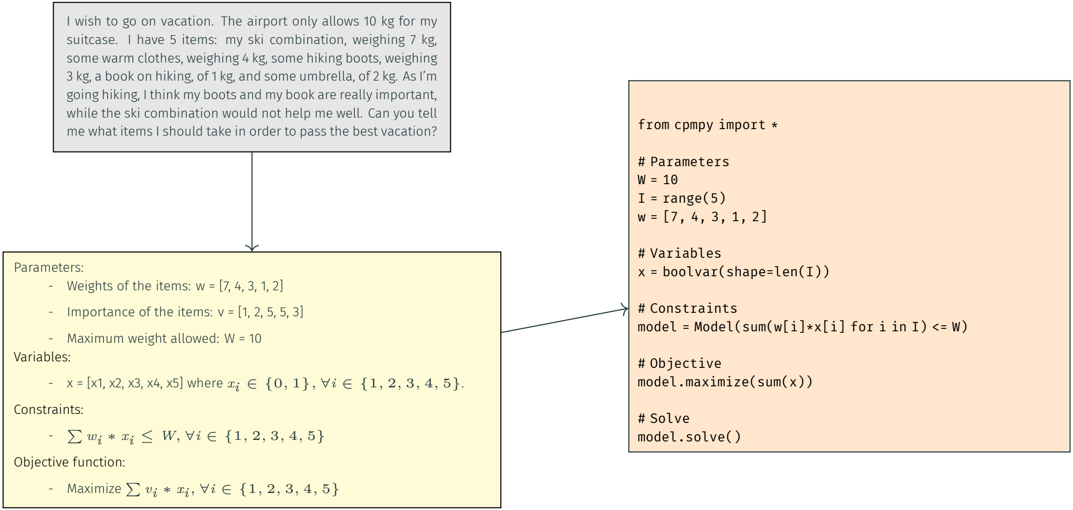
- The sum of the weights of the selected items must be less than or equal to the limit: sum(weights[i] * x[i] for i in range(n)) <= limit

Objective function:

- Maximize the total utility of the selected items: sum(utilities[i] * x[i] for i in range(n))

```
from cpmpy import *

# Parameters
n = 5
weights = [2, 3, 7, 4, 1]
utilities = [2, 3, 1, 2, 3]
limit = 10

# Variables
x = boolvar(shape=n)

# Constraints
model = Model(
   sum(weights[i] * x[i] for i in range(n)) <= limit
)

# Objective function
objective = sum(utilities[i] * x[i] for i in range(n))
model.maximize(objective)

model.solve()
```

EXAMPLE: NON-EXPERT DESCRIPTION



I wish to go on vacation. The airport only allows 10 kg for my suitcase. I have 5 items: my ski combination, weighing 7 kg, some warm clothes, weighing 4 kg, some hiking boots, weighing 3 kg, a book on hiking, of 1 kg, and some umbrella, of 2 kg. As I'm going hiking, I think my boots and my book are really important, while the ski combination would not help me well. Can you tell me what items I should take in order to pass the best vacation?

Parameters:

- Weights of the items: w = [7, 4, 3, 1, 2]
- Importance of the items: v = [1, 2, 5, 5, 3]
- Maximum weight allowed: W = 10

Variables:

- x = [x1, x2, x3, x4, x5] where $x_i \in \{0, 1\}, \forall i \in \{1, 2, 3, 4, 5\}$.

Constraints:

- $\sum w_i * x_i \leq W, \forall i \in \{1, 2, 3, 4, 5\}$

Objective function:

- Maximize $\sum v_i * x_i, \forall i \in \{1, 2, 3, 4, 5\}$

```
from cpmpy import *

# Parameters
W = 10
I = range(5)
w = [7, 4, 3, 1, 2]

# Variables
x = boolvar(shape=len(I))

# Constraints
model = Model(sum(w[i]*x[i] for i in I) <= W)

# Objective
model.maximize(sum(x))

# Solve
model.solve()
```

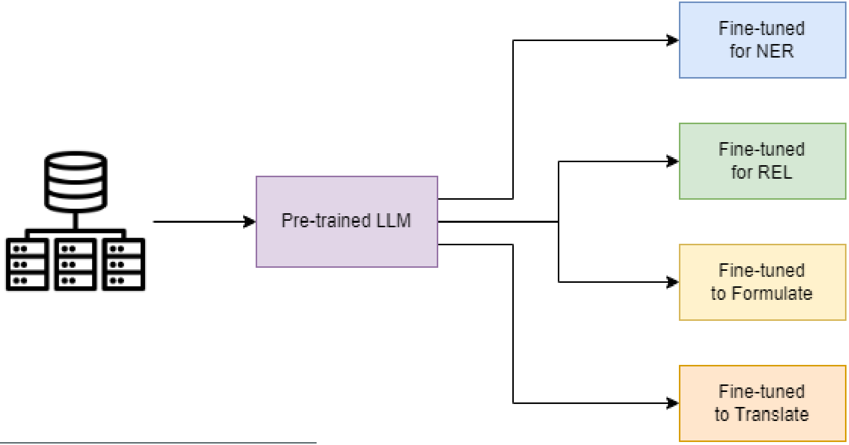
FUTURE WORK AND SUMMARY

FUTURE WORK

- Exploit more CP and optimization domain knowledge in prompt-tuning, in-context learning, and fine-tuning
- Create a dataset for CP problems
  - Input-Ouput for each submodule of the system
  - Different levels of abstraction
  - Different problem types
  - Including all constraint types

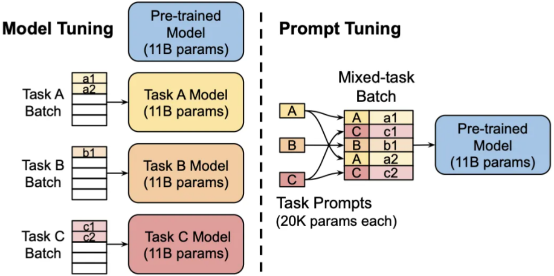
FUTURE WORK

- Fine-tune LLMs[12] for each subtask



[12] Dodge et al., Finetuning pre-trained language models: Weight initializations, data orders, and early stopping, arXiv, 2020

- Soft prompt-tuning is an alternative to fine-tuning LLMs[13]
- Keeping the pre-trained LLM frozen (using the same model for all tasks)
- Only learn a small task-specific (soft) prompt, consisting of k tunable tokens that are prepended to the input text


Model Tuning
Pre-trained Model (11B params)
Task A Batch
a1
a2
Task A Model (11B params)
Task B Batch
b1
Task B Model (11B params)
Task C Batch
c1
c2
Task C Model (11B params)
Prompt Tuning
Mixed-task Batch
A
B
C
A a1
C c1
B b1
A a2
C c2
Pre-trained Model (11B params)
Task Prompts (20K params each)

[13] Lester et al.,The Power of Scale for Parameter-Efficient Prompt Tuning, arXiv, 2021

SUMMARY

- We are closer to the stated goal for the Holy Grail
  - Modeling languages, Constraint Acquisition ...
- Gap from natural language description to CP model still exists
- Advancements in Large Language Models boost NLP
- Recent work on LP shows great potential
- Outlined our project on a framework for ”Natural Language to CP models”
- Use of LLMs and techniques from NL4OPT
- Lots of future work ...

DEMONSTRATION

Time for a live demonstration!