Marriage Problem Consultation On Predicate Task With X-Prolog 1.0

Frank Appiah

April 6, 2021

Marriage Problem Consultation On Predicate Task With X-Prolog 1.0

Prof Frank Appiah AKA FAEng PhD
appiahnsiahfrank@gmail.com
2.04.2021

Abstract. This report is an investigation reference on letter combinatorics showing the predicate sentences in Prolog syntax and some series of consult sentence case example in both textual and Pictorial forms.

Keywords. s-index, sentence, problem solving, words, Tableaux, predicate.

1 INTRODUCTION

Letter combinatorics is about sentences or phrases and counting problems. It is logical structured and involves discrete operations like subtraction, addition and multiplication. It is about alphanumeric labeling of sentences or phrases and proofing of combinatorial enumerations. The theory of combinatorics of sentences or phrases or words is called Letter Combinatorics (LC) with 8 bulletin requirements. A Marriage Problem (MP) made upof 5 sentences is used in the exploit of letter combinatorics. A generating function is calculated for MP to handle constraints of arrangement /selection and the combinatorial enumerations of MP. The predicate sentences are made from [5].

2 MARRIAGE PROBLEM (MP EXAMPLE)

(1) Damn it.
(2) What's wrong?

(3) It is a combination of 46 letters.
(4) Akua will not marry you.
(5) Pokua will not marry you.

The MP sentences are represented as predicates with each word captured in the predicate sentence, mpsentence. The following are the predicate sentences for the MP example :

1. mpsentence(damn, it).
2. mpsentence(what's, wrong).
3. mpsentence(it, is, a, combination, of, 46, letters).
4. mpsentence(akua, will, not, marry, you).
5. mpsentence(pokua, will, not, marry, you).

General predicate : mpsentence (word_1, word_n), n > 1.

The next predicate is to determine if a sentence is a question or not. There is only one question in all the five sentences. It is represented as mpsentenceask predicate sentence. This will take on two passing values of sentence number and an indicator of a question or not. The following question stances are:

1. mpsentenceask(1,no).
2. mpsentenceask(2, yes).
3. mpsentenceask(3, no).
4. mpsentenceask(4, no).
5. mpsentenceask(5, no).

General Predicate : mpsentenceask (sentence _no, response).

The number of words of a sentence is now represented with mpwordsize predicate sentences. . The following details are as follows :
1. mpwordsize(1, 2).
2. mpwordsize(2, 2).
3. mpwordsize(3, 6).
4. mpwordsize(4, 5).
5. mpwordsize(5, 5).

This predicate took its arguments to be the sentence number and the number of words. General predicate is represented as:
General Predicate : mpwordsize (sentence_no, word_number).
Further details on negation sentences are looked at. This will have the predicate sentence, mpnegation. This is explicitly sentences with a not word.
The problem solution are as follows :
1. mpnegation(1, no).
2. mpnegation(2, no).
3. mpnegation(3,no).

4. mpnegation(4, yes).
5. mpnegation(5, yes).

General Predicate : mpnegation (sentence _no, response).

MP example has only two negation statements in total. Statements like "damn it" creates a feeling of regret or disappointment. What's wrong did create sudden worry but does not bringthe negation that is not interesting. The predicate sentence is represented as mpregret. These are as follows :

1. mpregret(1, yes).
2. mpregret(2, no).
3. mpregret(3, no).
4. mpregret(4, no).
5. mpregret(5, no).

General Predicate : mpregret (sentence _no, response).

mpworry is the predicate sentence for sudden worry. These includes the following :

- mpworry(1, no).
- mpworry(2, yes).
- mpworry(3, no).
- mpworry(4, no).
- mpworry(5, no).

General Predicate : mpworry (sentence _no, response).

The problem solver took on statement 3 to bring out an approach. The predicate for this will be mpsolver. The knowledge needed to be programmed are as follows:

1. mpsolver(1, no).
2. mpsolver(2, no).
3. mpsolver(3, yes).
4. mpsolver(4, no).
5. mpsolver(5, no).

General Predicate : mpsolver (sentence _no, response).

The third round tried to bring out a solution in the context of problem solving. The 4 and 5 statements are involved with names of female sex. These are Akua and Pokua. The fact base for this representation is captured with predicate sentences, mpnamsex. These will include the following :

- mpnamsex(1, no).
- mpnamsex(2, no).
- mpnamsex(3, no).
- mpnamsex(4, yes).

● mpnamsex(5, yes).

General Predicate : mpnamsex (sentence _no, response).

It will be smart to know of the exact names involved. mpname predicate will be used to storefacts of name information. These includes the following sentences:

1. mpname(1, people).
2. mpname(2, object).
3. mpname(3, thing).
4. mpname(4, person).
5. mpname(5, person).

General Predicate : mpname (sentence _no, response).

This predicate captures a person's fact to the database. The assertions are as follows :
● mpperson(1, noname).
● mpperson(2, noname).
● mpperson(3, noname).
● mpperson(4. Akua).
● mpperson(2, Pokua).

General Predicate : mpperson (sentence _no, response).

The name information brings out the predicate concepts that includes mpstate that combines the words people, person, object and thing to the sentences.
The following statements are made:
● mpstate(1, 'Damn it on people').
● mpstate(2, 'What's wrong with you').
● mpstate(3, 'The thing is a combination of 46 letters')
● mpstate(4, 'A person will not marry you').
● mpstate(5, 'A person will not marry you').

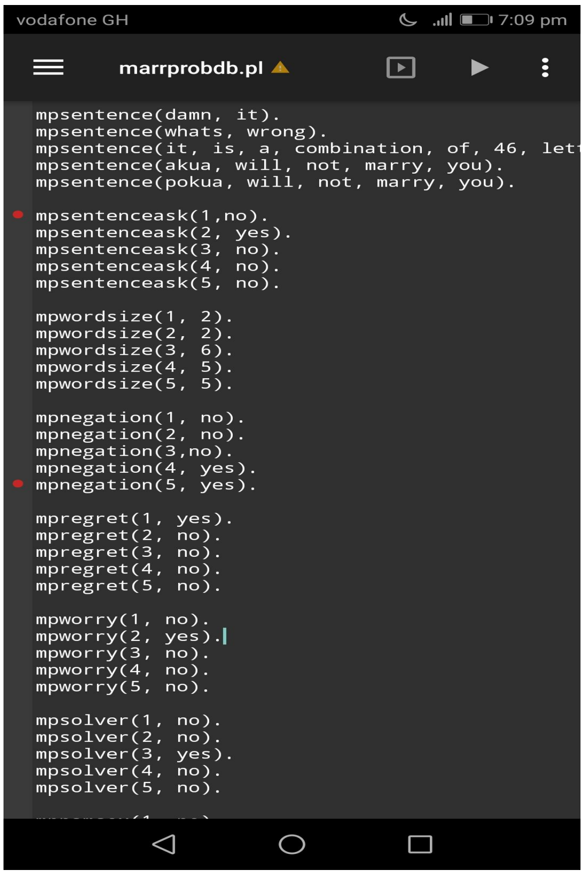
General Predicate : mpstate(sentence _no, response).

The Joy of predicates on 5 Secondary sentences is done in conclusion remarks.
Finally, the s-index predicate sentences are enumerated below :
1. sindex(1, 1, 6, 2).
2. sindex(2, 1, 10, 2).
3. sindex(3, 1, 27, 7).
4. sindex (4, 1, 19, 5).
5. sindex(5, 1, 20, 5).

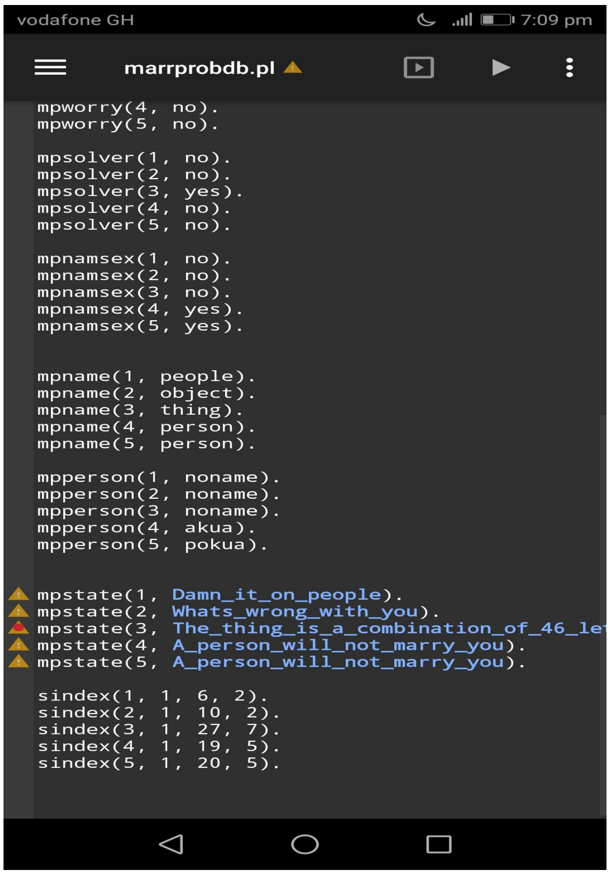
General Predicate : sindex (sentence _no, min_letter, max_letter, word_count).

3 Marriage Problem Database

This logic programming is being done on a Android mobile device with Xprolog application installed. The database file is marrprobdb.pl. The predicate sentences discussed earlier have similar constructs of prolog syntax and Semantics.



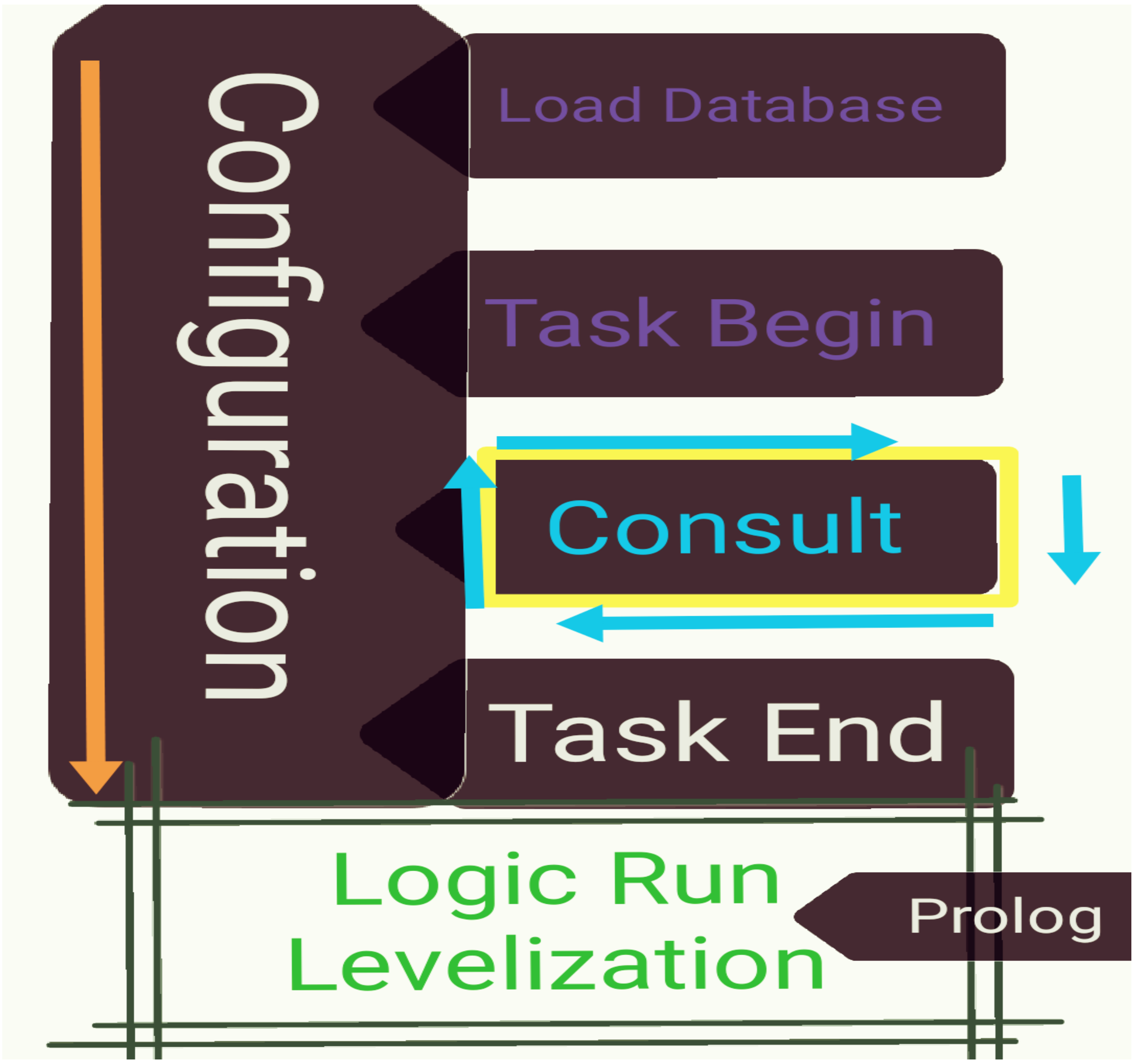
```
mpsentence(damn, it).
mpsentence(whats, wrong).
mpsentence(it, is, a, combination, of, 46, let
mpsentence(akua, will, not, marry, you).
mpsentence(pokua, will, not, marry, you).

mpsentenceask(1,no).
mpsentenceask(2, yes).
mpsentenceask(3, no).
mpsentenceask(4, no).
mpsentenceask(5, no).

mpwordsize(1, 2).
mpwordsize(2, 2).
mpwordsize(3, 6).
mpwordsize(4, 5).
mpwordsize(5, 5).

mpnegation(1, no).
mpnegation(2, no).
mpnegation(3,no).
mpnegation(4, yes).
mpnegation(5, yes).

mpregret(1, yes).
mpregret(2, no).
mpregret(3, no).
mpregret(4, no).
mpregret(5, no).

mpworry(1, no).
mpworry(2, yes).
mpworry(3, no).
mpworry(4, no).
mpworry(5, no).

mpsolver(1, no).
mpsolver(2, no).
mpsolver(3, yes).
mpsolver(4, no).
mpsolver(5, no).
```



```
mpworry(4, no).
mpworry(5, no).

mpsolver(1, no).
mpsolver(2, no).
mpsolver(3, yes).
mpsolver(4, no).
mpsolver(5, no).

mpnamsex(1, no).
mpnamsex(2, no).
mpnamsex(3, no).
mpnamsex(4, yes).
mpnamsex(5, yes).


mpname(1, people).
mpname(2, object).
mpname(3, thing).
mpname(4, person).
mpname(5, person).

mpperson(1, noname).
mpperson(2, noname).
mpperson(3, noname).
mpperson(4, akua).
mpperson(5, pokua).


mpstate(1, Damn_it_on_people).
mpstate(2, Whats_wrong_with_you).
mpstate(3, The_thing_is_a_combination_of_46_le
mpstate(4, A_person_will_not_marry_you).
mpstate(5, A_person_will_not_marry_you).

sindex(1, 1, 6, 2).
sindex(2, 1, 10, 2).
sindex(3, 1, 27, 7).
sindex(4, 1, 19, 5).
sindex(5, 1, 20, 5).
```

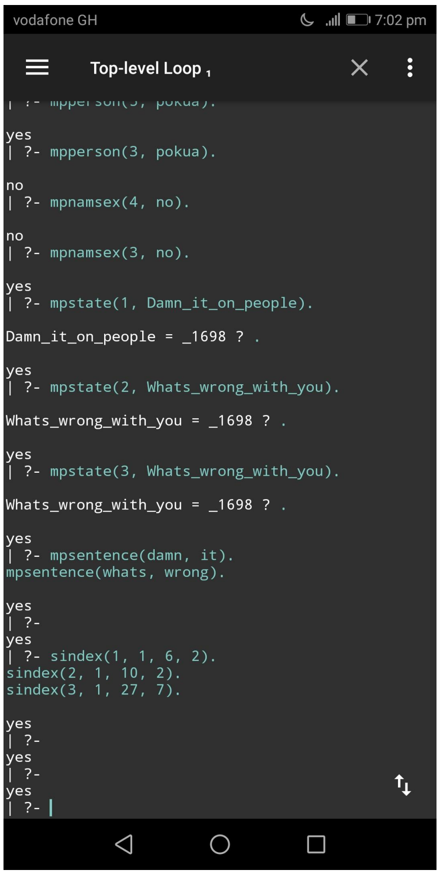
4 Consultation

This work reports on loading a prolog file in the cause of consulting predicates stored in a prolog database. The logic Run levelization on the logic program is shown in the figure below. Afterall, I will now show the consultation process on loading a database ready for consulting.

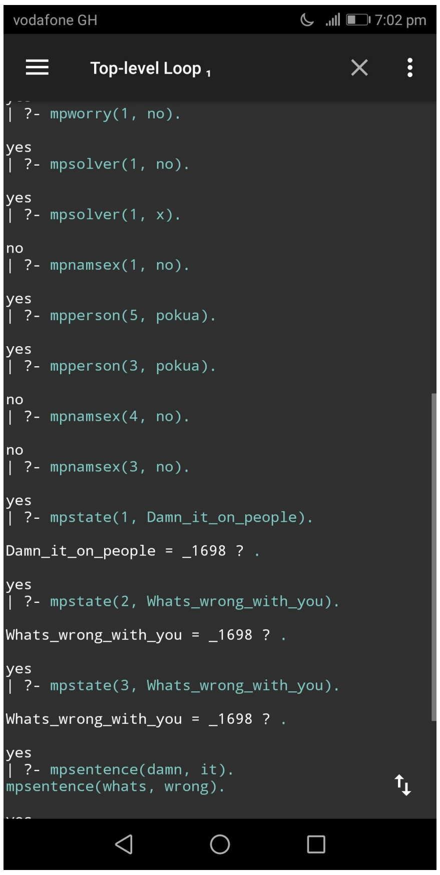


```
| ?- ['/storage/emulated/0/Download/marrprobdb.pl'].
message(informational,
[task_begin(consult),file_name('/storage/emulated/0/Download/marrprobdb.pl')]).
[style_warning(singleton_variable,A_person_will_not_marry_you),
file_name('/storage/emulated/0/Download/marrprobdb.pl'), line_number(67),
char_start(1243), char_end(1270)]).
message(informational,
[task_end(consult),file_name('/storage/emulated/0/Download/marrprobdb.pl')]).

yes
| ?- mpworry(1, no).

yes
| ?- mpsolver(1, no).
```

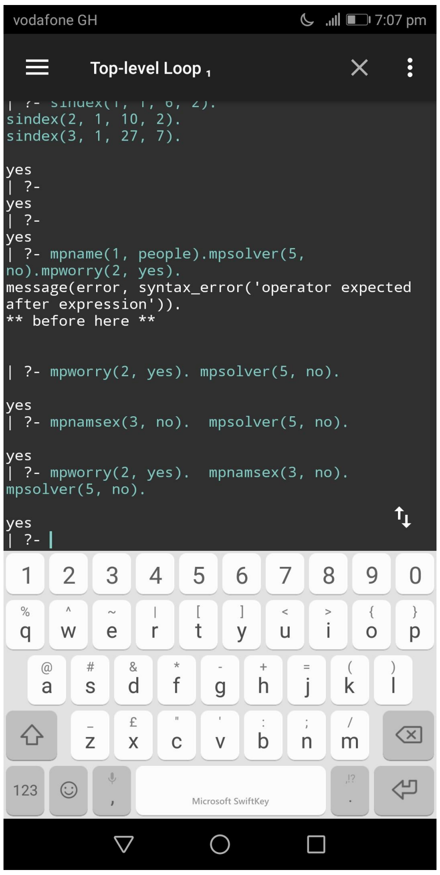
```
yes
| ?- mpsolver(1, x).

no
| ?- mpnamsex(1, no).

yes
| ?- mpperson(5, pokua).

yes
| ?- mpperson(3, pokua).

no
```



```
yes
| ?- mpperson(3, pokua).

no
| ?- mpnamsex(4, no).

no
| ?- mpnamsex(3, no).

yes
| ?- mpstate(1, Damn_it_on_people).

Damn_it_on_people = _1698 ? .

yes
| ?- mpstate(2, Whats_wrong_with_you).

Whats_wrong_with_you = _1698 ? .

yes
| ?- mpstate(3, Whats_wrong_with_you).

Whats_wrong_with_you = _1698 ? .

yes
| ?- mpsentence(damn, it).
mpsentence(whats, wrong).

yes
| ?-
yes
| ?- sindex(1, 1, 6, 2).
sindex(2, 1, 10, 2).
sindex(3, 1, 27, 7).

yes
| ?-
yes
| ?-
yes
| ?-
```



```
| ?- mpworry(1, no).

yes
| ?- mpsolver(1, no).

yes
| ?- mpsolver(1, x).

no
| ?- mpnamsex(1, no).

yes
| ?- mpperson(5, pokua).

yes
| ?- mpperson(3, pokua).

no
| ?- mpnamsex(4, no).

no
| ?- mpnamsex(3, no).

yes
| ?- mpstate(1, Damn_it_on_people).

Damn_it_on_people = _1698 ? .

yes
| ?- mpstate(2, Whats_wrong_with_you).

Whats_wrong_with_you = _1698 ? .

yes
| ?- mpstate(3, Whats_wrong_with_you).

Whats_wrong_with_you = _1698 ? .

yes
| ?- mpsentence(damn, it).
mpsentence(whats, wrong).
```



```
sindex(2, 1, 10, 2).
sindex(3, 1, 27, 7).

yes
| ?-
yes
| ?-
yes
| ?- mpname(1, people).mpsolver(5,
no).mpworry(2, yes).
message(error, syntax_error('operator expected
after expression')).
** before here **


| ?- mpworry(2, yes). mpsolver(5, no).

yes
| ?- mpnamsex(3, no).  mpsolver(5, no).

yes
| ?- mpworry(2, yes).  mpnamsex(3, no).
mpsolver(5, no).

yes
| ?-
```

Further Reading.

(1) Appiah Frank. Letter Combinatorics : Theory on counting problems. EPSRC UK Turing AI Letter. 2020
(2) Appiah Frank. Letter Combinatorics : Theory on counting problems. Marriage Problem. Mendeley Publication. 2020
(3) Hooper, Joan B. On assertive predicates.In Syntax and Semantics volume 4, pp. 91-124. Brill, 1975.
(4) Yoon, Youngeun. Total and partial predicates and the weak and strong interpretations. Natural language semantics 4, no. 3 (1996): 217-236.
(5) Frank Appiah. Representative Artificials On LetterCombinatorics Case With Predicate Sentences. Easychair No 4556. November 13, 2020.