Specifying Weak Constraints in Processable English

Rolf Schwitter

Rolf.Schwitter@mq.edu.au

Overview

- Processable English ($PENG^{ASP}$)
- $PENG^{ASP}$ System and Architecture
- A Simple Specification in $PENG^{ASP}$
- Strong and Weak Constraints in Answer Set Programming
- Specifying Weak Constraints in $PENG^{ASP}$

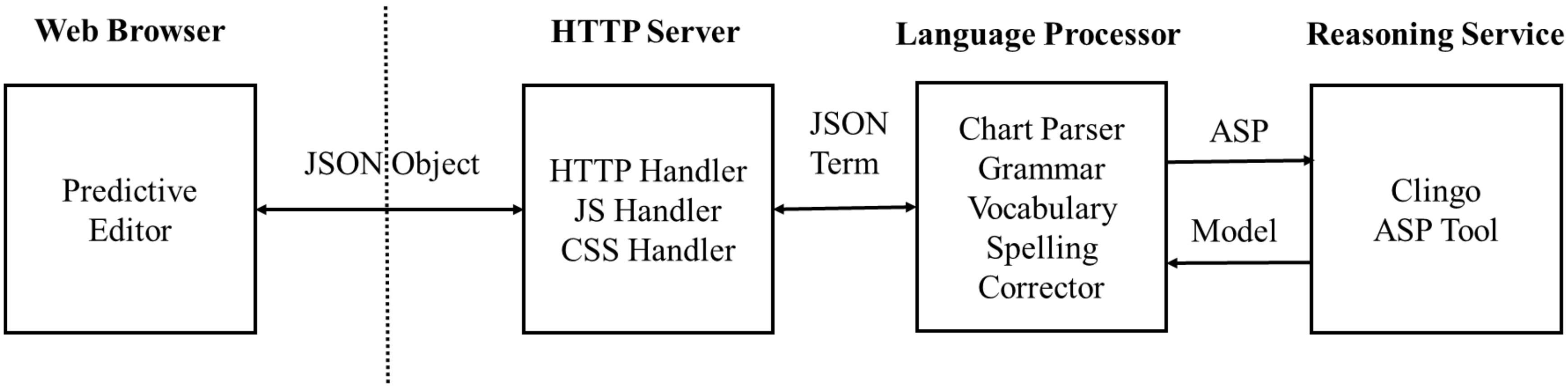
Processable English (PENG^{ASP})

- Processable English (PENG^{ASP}) is a controlled natural language that serves as a high-level specification language for Answer Set Programs (ASP).
- ASP is a form of declarative programming.
- ASP has its roots in: knowledge representation, logic programming, deductive databases, and constraint solving.
- The language processor of the PENG^{ASP} system translates a specification in controlled natural language into an executable ASP program and vice versa.

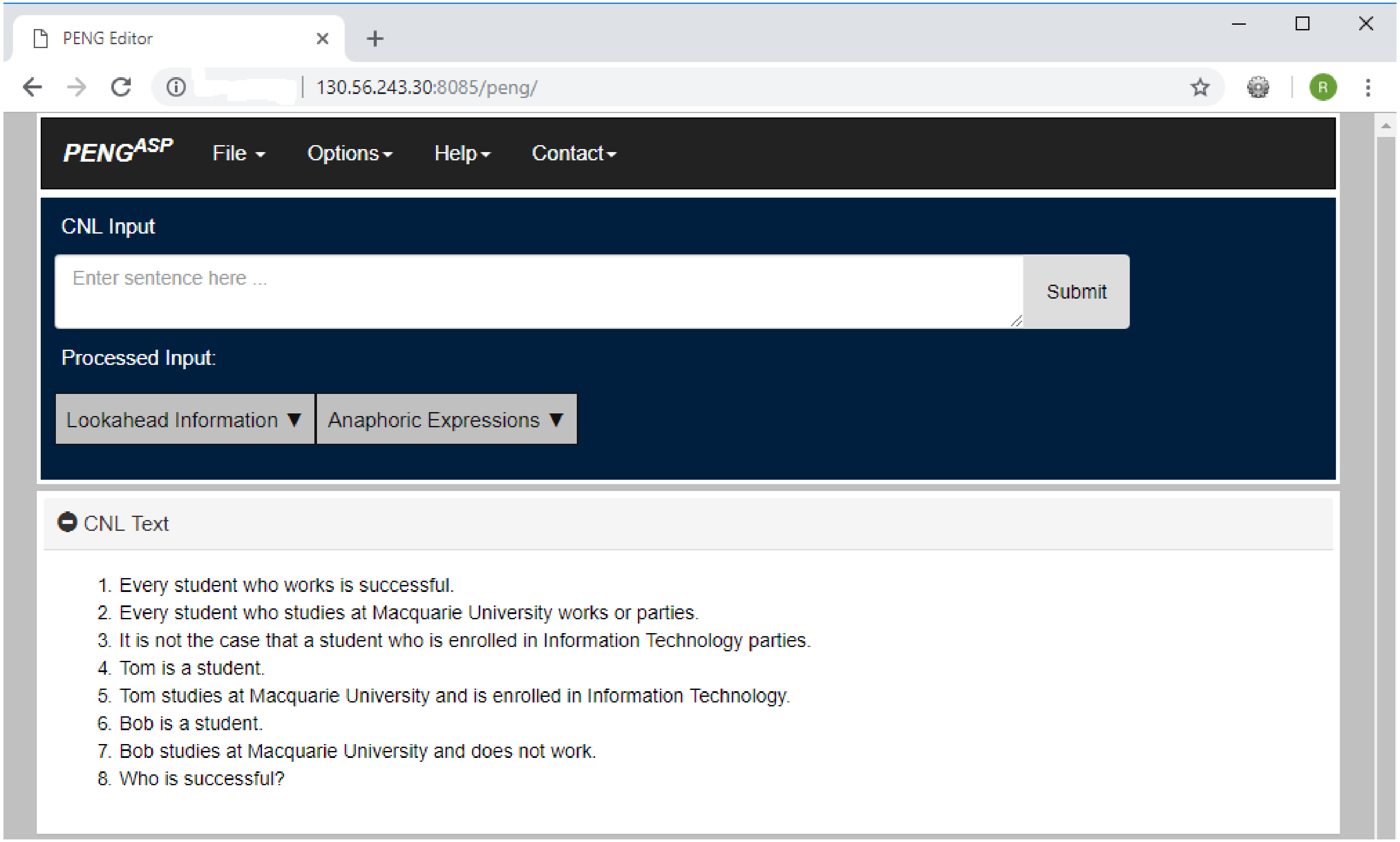
The PENG^{ASP} System

- A predicative text editor is used to guide the writing of a specification in controlled natural language.
- The specification is parsed incrementally during the writing process with the help of a chart parser.
- All natural language processing tasks occur in parallel:
  - anaphoric expressions are resolved,
  - look-ahead information is generated, and
  - an executable ASP is generated.
- New content words can be defined on the fly.

PENG^{ASP} Architecture



PENG^{ASP} at Work



Answer Set Program

Answer Set Program   Background Theory: OFF

```
prop(A,successful) :- class(A,student), pred(A,work).

pred(B,work) ; pred(B,party) :- class(B,student), pred(B,C,study_at), named(C,macquarie_university).

:- class(D,student), prop(D,E,enrolled_in), named(E,information_technology), pred(D,party).

named(1,tom).

class(1,student).

pred(1,2,study_at).

named(2,macquarie_university).

prop(1,3,enrolled_in).

named(3,information_technology).

named(4,bob).

class(4,student).

pred(4,2,study_at).

-pred(4,work).

answer(named(F,G)) :- named(F,G), prop(F,successful).
```

Answer Sets

Answer Sets

```
clingo version 5.3.0
Reading from asp.lp
Solving...
Answer: 1
-pred(4,work) class(1,student) class(4,student) named(1,tom) named(2,macquarie_university)
named(3,information_technology) named(4,bob) pred(1,2,study_at) pred(4,2,study_at) prop(1,
3,enrolled_in) pred(1,work) pred(4,party) prop(1,successful) answer(named(1,tom))
SATISFIABLE

Models       : 1
Calls        : 1
Time         : 0.001s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000s
```

Strong and Weak Constraints

- ASP supports strong constraints and weak constraints.
- Strong constraints weed out answer sets.
- Weak constraints should be satisfied if possible, they rank answer sets.
- Strong constraint:

```
:- Literal.
```

- Weak constraint:

```
:~ Literal. [Weight@Level, Term]
```

Weak Constraints

- Weak constraints can be weighted according to their importance.
- In the presence of weights, best answer sets minimize the sum of the weights of the violated weak constraints.
- Weak constraints can also be prioritized.
- The violation of the constraints of the highest priority level is minimised first, followed by the lower priority levels in descending order.

Weak Constraints

- The weak constraint:

  ```
  :~ pred(I, N, W). [W@L, I]
  ```

  can be expressed alternatively as a minimize statement:

  ```
  #minimize { W@L, I : pred(I, N, W) }
  ```

- Minimize statements can be interpreted as maximize statements with inverse weights:

  ```
  #maximize { -W@L, I : pred(I, N, W) }.
  ```

Scenario in Full Natural Language

We want to choose one among five available accommodations. These accommodations are identified via their names (Amora, Grace, Metro, Posh and Adina), and each accommodation owns a certain number of stars and costs a certain amount of money.

Of course, the more stars an accommodation has, the more it costs per night. Additionally, we know that the motel Posh is located on a main street, which is why we expect its rooms to be noisy.

Avoiding noise is very important to us, minimizing the cost per star is less important, and maximizing the star rating is least important to us.

Reconstructed Scenario in PENG^{ASP}

1. Choose either the accommodation Amora or Grace or Metro or Posh or Adina.
2. The hotel Amora owns five stars and costs 170 dollars.
3. The hotel Grace owns four stars and costs 140 dollars.
4. The hotel Metro owns three stars and costs 90 dollars.
5. The guesthouse Adina owns two stars and costs 60 dollars.
6. The motel Posh that is located on the main street owns three stars and costs 75 dollars.

Reconstructed Scenario in PENG^{ASP}

7. If an accommodation is located on a main street then the accommodation is noisy.
8. If an accommodation costs N dollars and owns M stars then N / M is the cost per star of the accommodation.
9. If an accommodation owns N stars then N is the star rating of the accommodation.

Reconstructed Scenario in PENG^{ASP}

10. Minimizing that an accommodation is noisy has the high priority H.
11. Minimizing the cost per star of an accommodation has the medium priority M.
12. Maximizing the star rating of an accommodation has the low priority L.
13. The high priority is 3.
14. The medium priority is 2.
15. The low priority is 1.

Sentence 1

Choose either the accommodation Amora or Grace or Metro or Posh or Adina.

```
1 { class(X, accommodation) :
    named(X, (amora ; grace ; metro ; posh ; adina)) } 1.
```

Sentences 2-5: 4

The hotel Metro owns three stars and costs 90 dollars.

```
class(7, hotel).
named(7, metro).
pred(7, 8, own).
data_prop(8, pos_int(3), cardinal).
class(8, star).
pred(7, 9, cost).
data_prop(9, pos_int(90), cardinal).
class(9, dollar).
```

Sentence 6

The motel Posh that is located on the main street owns three stars and costs 75 dollars.

```
class(13, motel).
named(13, posh).
prop(13, 14, located_on).
class(14, main_street).
pred(13, 8, own).
pred(13, 15, cost).
data_prop(15, pos_int(75), cardinal).
class(15, dollar).
```

Sentence 6: Relative Clause

The motel Posh that is located on the main street owns three stars and costs 75 dollars.

```
class(13, motel).
named(13, posh).
prop(13, 14, located_on).
class(14, main_street).
pred(13, 8, own).
pred(13, 15, cost).
data_prop(15, pos_int(75), cardinal).
class(15, dollar).
```

Sentence 6: Anaphoric Reference

The motel Posh that is located on the main street owns three stars and costs 75 dollars.

```
class(13, motel).
named(13, posh).
prop(13, 14, located_on).
class(14, main_street).
pred(13, 8, own).
pred(13, 15, cost).
data_prop(15, pos_int(75), cardinal).
class(15, dollar).
```

Sentence 7

If an accommodation is located on a main street then the hotel is noisy.

```
prop(X, noisy) :-
  class(X, accommodation),
  prop(X, Y, located_on),
  class(Y, main_street).
```

Sentence 8

If an accommodation costs N dollars and owns M stars then N / M is the cost per star of the accommodation.

```
rel(N/M, X, cost_per_star) :-
   class(X, accommodation),
   pred(X, Y, cost),
   data_prop(Y, pos_int(N), cardinal),
   class(Y, dollar),
   pred(X, Z, own),
   data_prop(Z, pos_int(M), cardinal),
   class(Z, star).
```

Sentence 9

If an accommodation owns N stars then N is the star rating of the accommodation.

```
rel(N, X, star_rating) :-
  class(X, accommodation),
  pred(X, Y, own),
  data_prop(Y, pos_int(N), cardinal),
  class(Y, star).
```

Sentence 10

Minimizing that an accommodation is noisy has the high priority H.

```
#minimize { 1@H,
            X : class(X, accommodation),
                prop(X, noisy),
                prop(P, high),
                class(P, priority),
                data_prop(P, pos_int(H), nominal) }.
```

Sentence 11

Minimizing the cost per star of an accommodation has the medium priority M.

```
#minimize { X@M,
            Y : rel(X, Y, cost_per_star),
                class(Y, accommodation),
                prop(P, medium),
                class(P, priority),
                data_prop(P, pos_int(M), nominal) }.
```

Sentence 12

Maximizing the star rating of an accommodation has the low priority L.

```
#maximize { X@L,
            Y : rel(X, Y, star_rating),
                class(Y, accommodation),
                prop(P, low),
                class(P, priority),
                data_prop(P, pos_int(L), nominal) }.
```

Sentences 13-15: 13

The high priority is 3.

```
prop(16, high).
class(16, priority).
data_prop(16, pos_int(3), nominal).
```

Evaluation

```
:~ prop(13, noisy), class(13, accommodation). [1@3, 13]

:~ class(1,  accommodation). [34@2,  1]
:~ class(4,  accommodation). [35@2,  4]
:~ class(7,  accommodation). [30@2,  7]
:~ class(10, accommodation). [30@2, 10]
:~ class(13, accommodation). [25@2, 13]

:~ class(1,  accommodation). [-5@1,  1]
:~ class(4,  accommodation). [-4@1,  4]
:~ class(7,  accommodation). [-3@1,  7]
:~ class(10, accommodation). [-2@1, 10]
:~ class(13, accommodation). [-3@1, 13]
```

Evaluation

```
:~ prop(13, noisy), class(13, accommodation). [1@3, 13]

:~ class(1,  accommodation). [34@2,  1]
:~ class(4,  accommodation). [35@2,  4]
:~ class(7,  accommodation). [30@2,  7]
:~ class(10, accommodation). [30@2, 10]
:~ class(13, accommodation). [25@2, 13]

:~ class(1,  accommodation). [-5@1,  1]
:~ class(4,  accommodation). [-4@1,  4]
:~ class(7,  accommodation). [-3@1,  7]
:~ class(10, accommodation). [-2@1, 10]
:~ class(13, accommodation). [-3@1, 13]
```

Evaluation

```
:~ prop(13, noisy), class(13, accommodation). [1@3, 13]

:~ class(1,  accommodation). [34@2,  1]
:~ class(4,  accommodation). [35@2,  4]
:~ class(7,  accommodation). [30@2,  7]
:~ class(10, accommodation). [30@2, 10]
:~ class(13, accommodation). [25@2, 13]

:~ class(1,  accommodation). [-5@1,  1]
:~ class(4,  accommodation). [-4@1,  4]
:~ class(7,  accommodation). [-3@1,  7]
:~ class(10, accommodation). [-2@1, 10]
:~ class(13, accommodation). [-3@1, 13]
```

Evaluation

```
:~ prop(13, noisy), class(13, accommodation). [1@3, 13]

:~ class(1,  accommodation). [34@2,  1]
:~ class(4,  accommodation). [35@2,  4]
:~ class(7,  accommodation). [30@2,  7]
:~ class(10, accommodation). [30@2, 10]
:~ class(13, accommodation). [25@2, 13]

:~ class(1,  accommodation). [-5@1,  1]
:~ class(4,  accommodation). [-4@1,  4]
:~ class(7,  accommodation). [-3@1,  7]
:~ class(10, accommodation). [-2@1, 10]
:~ class(13, accommodation). [-3@1, 13]
```

Solution

```
% named(7, metro) ... class(7, hotel) ...
% class(7, accommodation)
% Optimization: 0 30 -3
% OPTIMUM FOUND
% Time : 0.004s
```

Take-Home Messages

- PENG^{ASP} is human-understandable and machine-processable controlled natural language.
- PENG^{ASP} can bridge the gap between English and a formal languages like ASP.
- PENG^{ASP} can be used to specify not only strong constraints but also weak constraints in controlled language.
- Weak constraints rank answer sets.
- Future research: PENG^{ASP} for smart contracts, for dynamic domains and for conceptual modelling.