Minimalist Propositional Logic Resolution (Theorem) Prover

DESCRIPTION

This file describes a propositional logic theorem prover module that is small, has a minimalist API and is designed to be easy to use. The source file name for the prover module is:

```
PLResolutionProver.py
```

For use cases examples and test cases see:

```
Provertestprogram.py
```

The prover uses propositional resolution as it's inference mechanism. For more information on propositional logic and the resolution rule of inference see the Stanford Introduction to Logic on line course on symbolic logic.

http://logic.stanford.edu/intrologic/homepage/index.html

The knowledgebase (KB) is represented in conjunctive normal form (a conjunction if disjunctions). Each disjunctive clause is represented as python set of positive or negative propositions (where propositions are represented as positive and negative integers:

Example: The disjunctive clause (A or B or not C) is equivalently represented as the following python set in the knowledgebase:
conjunctiveClause = {1,2,-3}

Internally the knowledge base is represented as a python list of python sets (again, the python sets are the conjunctive clauses:

Example: KB = (A or B or not C) and (C or D or E) is represented as: KB = [{1,2,-3}, {3,4,5}]

The prover is small, but it has few optimizations for speed.

LICENCE

This file is part of minimalist propositional logic resolution prover. It may be used for free under the terms of the licence below:

API

getKB()

Returns the knowledge base of the PLResolutionProver object as a list of lists representing the knowledgebase in conjunctive normal form (inner lists each representing a disjunctive clause)

Example:

```
>>>#KB = (A or C or E) and (not E or G or . . .
>>>print   myPLResolutionProver.getKB()
[[1, 3, 5], [-5, 7, -6], [63, -23, 49], . . . . .
```

setKB(KBInput)

Takes a list of lists as an input argument representing a knowledgebase in conjunctive normal form (inner lists each representing a disjunctive clause).

Example:

```
>>>myPLResolutionProver.setKB([1,2],[-2,3]) # Set the content of the
knowlegebase.
>>>print   myPLResolutionProver.getKB()  # Print content of knowledgebase.
[[1,2],[-2,3]]
```

Example:

```
>>>#To empty the knowlegebase type the following:
```

```
>>>myPLResolutionProver.setKB([]) # Set the content of the knowlegebase to an
empty list
```

---

addToKBAndResolve(disjunctiveClause)

Add a disjunctive clause to the knowledgebase and apply resolution inference rule between each the new clause and each relevant clause in the KB to derive new clauses. If new clauses are formed then resolve the newly formed clauses with the relevant parts KB as well.

Example:

```
>>>myPLResolutionProver.setKB([-1,2],[-2,3]) # Set the content of the
knowlegebase.
>>>print   myPLResolutionProver.addToKBAndResolve([-3]) # Add and resolve.
>>>print   myPLResolutionProver.getKB()  # Print content of knowledgebase.
[[-1,2], [-2,3], [-3], [-2], [-1]]
```

---

addToKB(disjunctiveClause)

Add a disjunctive clause to the knowledgebase and without applying resolution inference rule to each the new clause.

Example:

```
>>>myPLResolutionProver.setKB([-1,2],[-2,3]) # Set the content of the
knowlegebase.
>>>print   myPLResolutionProver.addToKB([-3]) # Add and resolve.
>>>print   myPLResolutionProver.getKB()  # Print content of knowledgebase.
[[-1,2], [-2,3], [-3]]
```
