User Manual

PIIL is an implementation of propositional partial information ionic logic. Given a set of formulas in propositional partial information ionic logic it computes model scheme using analytic tableau method.

This program has to be run from command line as follows:

java -jar piil.jar inputfile [outputfile]

Input file should follow the following grammar:

Input	:=	(Comment | Sentence)*
Sentence	:=	Turnstile? Formula (Seperator Formula)*
Seperator	:=	NewLine | ';'
Comment	:=	C-style multi-line comment
Formula	:=	Ion | '(' Formula ')' | Unary_Con Formula | Formula Binary_Con Formula | 'bot (' Fromula ')' | Prop_Var | true | false
Ion	:=	'*' Digit? '(' Formula ',' Formula ')'
Unary_Con	:=	- | ~ | ~'
Binary_Con	:=	/\ | \/ | ! | ->
Digit	=	[0-8]
Prop_Var	:=	[a-zA-Z0-9_]+
Turnstile	:=	T | NT | PT | NPT

Input file can only contain ASCII characters. But output file will contain Unicode characters. So a Unicode compatible software has to be used to view the output file.

Some operators used in PIIL are not available in ASCII. Therefore the following alternative representation must be followed:

(implication := -> or >) , (and := & or /\) , (or := | or \/) , (negation := -) , (not_true := ~) , (not_potentially_true := ~'), (interjunction := !) , (bottom function := bot) , (True Turnstile := T) , (Not True Turnstile := NT) , (Potentially True Turnstile := PT) , (Not Potentially True Sign := NPT) , (*0 := Diamondsuite) , (*1 := Heartsuite) , (*2 := Circle) , (*3 = Spadesuite) , (*4 := Clubsuite) , (*5 := Blackfly) , (*6 := Spadesuite-twin) , (*7 := Clubsuite-twin) , (*8 := Butterfly), (* := Generic ionic operator).

Example:

Now we will show how to use this program using an example.
Step1: Create an input file, say test.in, and type in the following lines.

```
/* tweety is a penguine.
penguines are bird
birds typically fly unless they are penguines.
does tweety fly?
using butterfly ion.
*/
```

```
/*
p = tweety is a penguine
b = tweety is a bird
f = tweety flies
*/

T p                    /* tweety is a penguine */
T p -> b               /* if tweety is a penguine then tweety is a bird */
T b -> *8(f & -p, f)   /* if tweety is a bird then if it is acceptable that
tweety flies and tweety is not a penguine then tweety flies (in a soft
sense) */
```

Step 2: From command line run the following command:

```
java -jar piil.jar test.in test.out
```

Step 3: Open test.out using a unicode aware software, for example a browser, to see the result generated by this program. In this case this output file will contain the following:

```
Input: [⊨ p, ⊨ (p -> b), ⊨ (b -> ⋈((f & -(p)), f))]

2 models found.
<{⊨ b,⊨ p}, {j2 ⊮ f}, {}>
<{⊨ b,⊨ p}, {j2 ⊨ p}, {}>
```