“逻辑习题判别系统”使用说明

Introduction

本项目是一个“**逻辑习题判断系统**”，用于帮助教师批改习题，亦可帮助学生检查自己的答案的正确性。

该系统目前只支持**命题逻辑**，可处理三种类型的题目：

- 给定一个自然语言命题，写出相应的逻辑命题；
- 给定一个逻辑命题，写出相应的主析取（合取）范式；
- 给定前提和结论，写出论证过程。

针对上述三种题目，该系统提供了三种功能：

- **判断两个逻辑命题是否等价**，即验证用户给出的答案是否与标准答案等价；
- **判断一个逻辑命题是否为另一个逻辑命题的主析取（合取）范式**，即验证用户给出的答案是否为给定命题的主析取（合取）范式；
- **判断一个推理论证过程是否正确**，即验证用户给出的论证过程的正确性。

Usage

基本用法

该系统是一个命令行程序，目前只在**win10平台**下进行过测试。系统的使用过程大致为：

1. 用户创建一个文本文件，并在其中写入输入信息；
2. 用户执行程序`logic`，并将文本文件的路径作为参数；
3. 系统读取文本文件，按照选定的功能进行处理，并返回结果。

基本的使用方法如下:

```
Usage: logic command [path]

logic equiv path         Check if the given two propositions are equivalent
logic pcnf path          Check if the 2nd proposition is the principal
conjunctive normal form of the 1st one
logic pdnf path          Check if the 2nd proposition is the principal
disjunctive normal form of the 1st one
logic validate path      Check if the given proof is valid
logic help               Output this message
```

其中，

- `command`参数表示选定的功能，如果为`equiv`、`pdnf`、`pcnf`和`validate`中的一个，那么还需要在后面附加`path`参数；
- `path`参数表示**作为系统输入的文本文件的路径**；
- `equiv`功能用于**判断两个命题是否等价**；
- `pdnf`功能用于**判断第二个命题是否为第一个命题的主析取范式**；
- `pcnf`功能用于**判断第二个命题是否为第一个命题的主合取范式**；
- `validate`功能**将文本文件中的内容视为一个推理论证过程，并判断其正确性。**
- `help`功能用于**输出（上面的）帮助信息。**

使用约束

系统对符号、连接词等做出了相应的约束，如下：

- 命题变元符号
  - 小写字母`a,b,c,...,z`
  - 大写字母`A,B,...,Z`，但不包括`T`和`F`
- 布尔常量
  - `T`和`F`
- 连接词
  - 析取：`\/`
  - 合取：`/\`
  - 非：`~`
  - 蕴含：`->`
  - 等价：`<->`

在书写命题时，有以下注意点：

- 析取和合取符号是由`/`和`\`构成的；
- 按照系统设计优先级从高到低依次为`~`、`/\`、`\/`、`->`、`<->`，如果出现违反优先级的解析结果（BUG），请通过邮箱进行反馈。同时为了可读性，**建议使用括号区分命题层次**，以防出现意料之外的错误；
- 系统使用空格（包括换行）来区分多个命题，同时会忽略命题中符号之间的空格，所以可以恰当使用空格来增加可读性。

论证过程约束

推理论证的格式

系统对论证过程的格式做出了要求，请按照下面的格式书写论证过程：

```
Premises: 前提 （即n个命题，使用逗号分隔）
Conclusion: 结论 （1个命题）

Proof:
编号. 前提（即a个命题，使用逗号分隔） |- 结论 （1个命题） [规则 编号 编号]
编号. 前提（即b个命题，使用逗号分隔） |- 结论 （1个命题） [规则 编号 编号]
......
编号. 前提（即c个命题，使用逗号分隔） |- 结论 （1个命题） [规则 编号 编号]
```

针对以上格式，有以下注意点：

- 原则上，以上格式中出现空格的地方表示着**此处空格数量任意**，除了以下情况：
  - 某些要素是一个整体，**不可以在它们内部加空格**，如`Premises:`、`编号.`等，不能写为`Premises :`、`编号 .`
- `编号.`表示每一个步骤的编号，从`1.`开始数起，方便之后的步骤使用规则时进行指代。

  但要注意，**系统会按照各个步骤的书写顺序为其编号，而不会根据用户书写的`编号.`来为每一个步骤编号**。这就意味着，如果书写了错误的编号，只会影响用户书写推理论证的过程，而不会影响系统的解析。如：

  将第三步的`3.`写为`10.`，系统仍将其视为`3.`，只不过用户之后在使用规则时可能会用错编号。
- 前提包含了至少一个命题，如果为空，那么会出现解析错误。

  一般情况下不会出现前提为空的情况，但**如果真的遇到了，请为前提都加了一个与论证过程无关的命题**。这样既避免了前提为空的情况，也能保证推理论证过程的正确性不变。
- `[规则 编号 编号]`表示一个步骤中使用的推导规则。

  当规则只需要一个之前的步骤时，请使用`[规则 编号]`；

  当规则不需要前面的步骤时，请使用`[规则]`。
- 请不要随意调换前提中多个命题的顺序，这样会影响推理论证过程的正确性。
- 最后一步中，前提必须与`Premises`完全相同，结论必须与`Conclusion`完全相同。
- 目前可以在论证过程中加注释。
  - 注释以`\\`开头，之后的所有内容都属于注释，直到换行；
  - **注释必须用换行结束**，这意味着如果在文件的最后一行添加了注释，**请务必加上换行符**。

例子

以下是一个合法且正确的推导过程：

```
Premises:  p -> (q -> s), ~r \/ p, q
Conclusion:  r -> s

Proof:
1.  p -> (q -> s), ~r \/ p, q, r    |-  r              [prem]
2.  p -> (q -> s), ~r \/ p, q, r    |-  ~r \/ p        [prem]
3.  p -> (q -> s), ~r \/ p, q, r    |-  r -> p         [orToImply 2]
4.  p -> (q -> s), ~r \/ p, q, r    |-  p              [imple 3 1]
5.  p -> (q -> s), ~r \/ p, q, r    |-  p -> (q -> s)  [prem]
6.  p -> (q -> s), ~r \/ p, q, r    |-  q -> s         [imple 4 5]
7.  p -> (q -> s), ~r \/ p, q, r    |-  q              [prem]
8.  p -> (q -> s), ~r \/ p, q, r    |-  s              [imple 6 7]
9.  p -> (q -> s), ~r \/ p, q       |-  r -> s         [impli 8]
```

下面是添加了注释的合法的推导过程（**注意最后一行使用了注释，所以必须换行**）：

```
Premises:  p -> (q -> s), ~r \/ p, q    \\ comment
Conclusion:  r -> s                     \\ comment

\\comment

Proof:

\\comment
\\comment

1.  p -> (q -> s), ~r \/ p, q, r    |-  r       [prem]                \\
comment
2.  p -> (q -> s), ~r \/ p, q, r    |-  ~r \/ p     [prem]
3.  p -> (q -> s), ~r \/ p, q, r    |-  r -> p      [orToImply 2]
4.  p -> (q -> s), ~r \/ p, q, r    |-  p       [imple 3 1]
5.  p -> (q -> s), ~r \/ p, q, r    |-  p -> (q -> s)   [prem]
6.  p -> (q -> s), ~r \/ p, q, r    |-  q -> s      [imple 4 5]         \\
comment
7.  p -> (q -> s), ~r \/ p, q, r    |-  q       [prem]
8.  p -> (q -> s), ~r \/ p, q, r    |-  s       [imple 6 7]
9.  p -> (q -> s), ~r \/ p, q   |-  r -> s      [impli 8]               \\
comment

\\comment
```

支持的规则

> 详情请参阅另一个文档`rule.pdf`。

功能说明

equiv

> 该功能要求文本文件中包含**两个**命题。具体用法可参照以下例子：
>
> 1. 将以下文本保存为文件`True.txt`
>
>    ```
>    A \/ (B-> C)
>    ~B \/ A \/ C
>    ```
>
> 2. 执行程序并获取结果
>
>    ```
>    PS F:\LogicSystem> .\logic.exe equiv .\test_file\equiv\True.txt
>    True: The given proposition IS equivalent.
>    ```

pcnf

> 该功能在使用方法上与`pdnf`相似，要求文本文件中包含**两个**命题。具体用法可参照以下例子：
>
> 1. 将以下文本保存为文件`conTrue.txt`
>
>    ```
>    (~(A->B)) /\ ((~B) <-> C)
>    ((~A \/ (~B \/ ~C)) /\ ((~A \/ (~B \/ C)) /\ ((~A \/ (B \/ C)) /\ ((A \/ (~B
>    \/ ~C)) /\ ((A \/ (~B \/ C)) /\((A \/ (B \/ ~C)) /\ (A \/ (B \/ C))))))))
>    ```
>
> 2. 执行程序并获取结果
>
>    ```
>    PS F:\毕业论文\LogicSystem> .\logic.exe pcnf .\test_file\norm\conTrue.txt
>    TRUE: The 2nd proposition IS the principal conjunctive normal form of the 1st proposition.
>    ```

validate

> 该功能要求文本文件中包含**符合规范的论证过程**。具体用法可参照以下例子：
>
> 1. 将以下文本保存为文件`True_1.txt`

```
Premises:  p -> (q -> s), ~r \/ p, q
Conclusion:  r -> s

Proof:
1.  p -> (q -> s), ~r \/ p, q, r    |-  r              [prem]
2.  p -> (q -> s), ~r \/ p, q, r    |-  ~r \/ p        [prem]
3.  p -> (q -> s), ~r \/ p, q, r    |-  r -> p         [orToImply 2]
4.  p -> (q -> s), ~r \/ p, q, r    |-  p              [imple 3 1]
5.  p -> (q -> s), ~r \/ p, q, r    |-  p -> (q -> s)  [prem]
6.  p -> (q -> s), ~r \/ p, q, r    |-  q -> s         [imple 4 5]
7.  p -> (q -> s), ~r \/ p, q, r    |-  q              [prem]
8.  p -> (q -> s), ~r \/ p, q, r    |-  s              [imple 6 7]
9.  p -> (q -> s), ~r \/ p, q       |-  r -> s         [impli 8]
```

2. 执行程序并获取结果

```
PS F:\LogicSystem> .\logic.exe validate .\test_file\validate\True_1.txt
TRUE: The proof in ".\test_file\validate\True_1.txt" IS valid.
```

FAQ

"ERROR: need two propositions, but found 0." 是什么意思?

> 这种情况一般是发生在对文本文件成功解析之后，发现文本文件中**没有命题**（此时文本文件一般为空）。

"ERROR: need two propositions, but found 1." 是什么意思?

> 这种情况一般是发生在对文本文件成功解析之后，发现文本文件中只包含了**一个命题**。在使用`equiv`、`pcnf`和`pdnf`功能时，需要两个命题。

“Error ... parsing ...” 是什么意思?

> 出现这两个词时，代表解析文本出现了错误。请检查文本内容是否符合规范。

"The final step does NOT fit the form that deducing "Conclusion" from "Premises"." 是什么意思?

> 这个错误信息指论证过程的最后一步不是“由给定前提推导出结论”的形式。请检查最后一步，确保前提与前面的`Premises`完全相同（包括命题的顺序），同时确保结论与前面的`Conclusion`完全相同。

""...": cannot deduce the given proposition." 是什么意思?

> 这个错误的原因一般为“**无法对编号对应的步骤使用指定的推导规则**”或“**使用规则推导出来的结果与当前步骤的结论不同**”。
>
> - 比如，`imple`规则要求作为参数的两个步骤的**前提完全相同（包括顺序）**，同时其中一个步骤的结论必须为**蕴含命题**，另一个步骤的结论为**该蕴含命题的前件**，只要使用规则时有任何一点不符合，就会引起该错误。
> - 又比如，使用规则推导出`A`，但当前步骤的结论为`B`，此时也会触发错误。

""...": found invalid parameter(s)." 是什么意思?

> 这个错误的原因一般是规则指定的**编号**非法，比如：
>
> - 当前为第三步，但是规则使用了编号`5`。

""...": need ... parameters, but found ...." 是什么意思?

> 这个错误的原因一般是规则指定的**编号**数量有误，比如：
>
> - 当前规则需要使用之前的两个步骤，但只给了一个编号。

"Unknown rule: "..."" 是什么意思?

> 这个错误是指给定的规则无法识别。
>
> - 请查阅规则表并检查规则拼写是否有误；
> - 如果无误，请检查规则与编号是不是**有且仅有一个空格**，多余的空格会被识别为规则的一部分。

规范制定得太麻烦了，为什么不做得更容易写?

> 目前的规范是在保证系统的解析功能正常工作的前提下，最大程度地考虑了用户友好这一点，从而制定出来的。
>
> 当然，如果你有更好的想法，也欢迎来信与我探讨实现的可能性。

为什么我明明按照说明文档去写了，还是出现了错误?

> - 首先，不妨仔细阅读说明文档，确保没有出现一些细节上的错误；
> - 其次，这种情况有可能是说明文档有问题，亦有可能是系统设计上的失误，欢迎来信向我反馈。

有一个证明题使用目前支持的规则完全无法推导，或者缺少了某些规则使得部分推导变得很麻烦，有办法解决吗？

> 该系统支持的推导规则集是有一定的扩展能力的。所以，遇到这种情况时，不妨向我反馈，告知规则内容和使用场景，我会尽可能地去丰富规则集。

Feedback

> 本系统的实现工作是由个人完成的，而个人精力有限，无法做到全方位的测试，故而难免出现一些BUG或者设计失误。
>
> 如果遇到了问题或者有什么建议，欢迎通过邮箱`guojn25@mail2.sysu.edu.cn`来联系我，我会尽快在第一时间回复。
>
> 可以反馈的情况**包括但不限于**：
>
> - 本文档说明不到位的地方；
> - 出现了BUG，如出现与系统设计相反的结果；
> - 有助于提升系统的建议，如一些规则表中没有但又十分常用的规则；
> - 在阅读了本文档以及询问了他人之后仍然无法解决的问题，比如一个你认为正确但系统判误的论证过程。
>
> > 这种情况有可能是论证过程本身有误或不符合系统设计，不妨问问他人的意见。当然也可以向我反馈，但我可能无法及时回复，从而耽误你的时间。