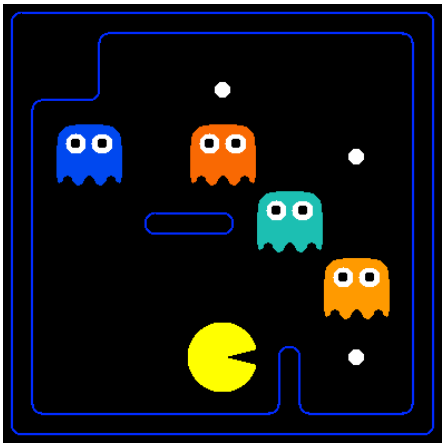
课程作业 2：逻辑与经典规划



任务概述

在这次课程作业中，你将使用/编写 Python 函数来生成逻辑语句用以描述吃豆人世界，包括使用 `Expr` 数据类型来表示和实现命题逻辑语句和函数，以及使用 SAT（可满足性问题）求解器 pycosat 来解决与规划相关的逻辑推断任务，例如，生成动作序列以到达目标位置并吃掉所有豆子。这里，SAT 求解器以一个逻辑表达式作为输入，返回一个满足该表达式的模型（如果存在这样的模型），即，满足该表达式的逻辑符号的真值赋值。

作业的程序包位于 QQ 群文件：课程作业\logic.zip。需要预先安装 pycosat（SAT 求解器 PicoSAT 的 Python 包装器）：

```
pip install picosat
```

或者

```
pip3 install pycosat
```

对于 conda

```
conda install pycosat
```

关于 `Expr` 类的介绍请参考：

https://inst.eecs.berkeley.edu/~cs188/sp24/projects/proj3/#the-expr-class

其中需要特别注意的是，**尽可能使用 `conjoin` 和 `disjoin` 函数来分别创建合取和析取表达式。**

命题符号命名规则请参考：

https://inst.eecs.berkeley.edu/~cs188/sp24/projects/proj3/#prop-symbol-names-important

其中需注意 **`PropSymbolExpr` 构造函数的用法。**

所有你自己的算法实现都是位于 **`logicPlan.py`** 文件中相应任务的函数下面。

作业内容

任务 1：逻辑语句表示（Logic Warm-up）

使用 `Expr` 数据类型来表示命题逻辑语句。请在 `logicPlan.py` 中实现以下函数：

- `sentence1()`：创建一个 `Expr` 实例来表示以下三个句子全为真这一命题。 注意：不要做任何逻辑上的简化，直接按照给定顺序将句子放入一个列表，并返回列表的合取式。

$$A \vee B$$

$$\neg A \leftrightarrow (\neg B \vee C)$$

$$\neg A \vee \neg B \vee C$$

- `sentence2()`：创建一个 `Expr` 实例来表示以下四个句子全为真这一命题。 同样，不要做任何逻辑上的简化，直接按照给定顺序将句子放入一个列表，并返回列表的合取式。

$$C \leftrightarrow (B \vee D)$$

$$A \rightarrow (\neg B \wedge \neg D)$$

$$\neg(B \wedge \neg C) \rightarrow A$$

$$\neg D \rightarrow C$$

- `sentence3()`：使用 `PropSymbolExpr` 构造函数创建符号`'PacmanAlive_0'`、`'PacmanAlive_1'`、`'PacmanBorn_0'`和`'PacmanKilled_0'`。然后，在此基础上创建一个 `Expr` 实例，将以下三个英语句子按给定顺序编码为一个命题逻辑语句，并且不进行任何逻辑上的简化（`PropSymbolExpr(str, a1, a2, a3, a4, time=a5)`创建表达式 `str[a1,a2,a3,a4]_a5`，其中 `str` 是一个字符串变量）：

  1. *Pacman is alive at time 1 if and only if he was alive at time 0 and he was not killed at time 0 or he was not alive at time 0 and he was born at time 0.*
  2. *At time 0, Pacman cannot both be alive and be born.*
  3. *Pacman is born at time 0.*

- `findModelUnderstandingCheck()`：
  1. 理解 `findModel(sentence)`函数内部如何运作，并在 Python 中打开一个交互式会话并运行：
     ```
     from logicPlan import *
     findModel(sentence1())
     findModel(sentence2())
     findModel(sentence3())
     ```
     来使用 `findModel` 测试之前实现的 `sentence1()`、`sentence2()`和 `sentence3()`。查看结果是否符合你的预期。
  2. 根据上述内容，填写 `findModelUnderstandingCheck`，使其返回在允许小写变量的情况下 `findModel(Expr('a'))`返回的内容。这里只需直接给出输出结果即可，除了已给定的之外，无需使用 `findModel` 或者 `Expr`。

- `entails(premise, conclusion)`：当且仅当 `premise` 蕴含 `conclusion` 时返回 `True`。 提示：这里可以借助 `findModel` 函数，并考虑使用反证法。

- `plTrueInverse(assignments, inverse_statement)`：给定赋值 `assignments`，当且仅当表达式 `inverse_statement` 的非为 `True` 时返回 `True`。这里可以借助 `pl_true` 函数。

可以通过运行：

```
python autograder.py -q q1
```

来测试和调试你的代码。

任务 2：逻辑函数实现（Logic Workout）

请在 `logicPlan.py` 中实现以下三个函数（如需要创建合取和析取表达式时，尽可能使用 `conjoin` 和 `disjoin` 函数）：

- `atLeastOne(literals)`：返回单个 CNF 表达式(`Expr`)，仅当输入列表中至少有一个表达式为真时，该表达式才为真。每个输入表达式都是一个文字。
- `atMostOne(literals)`：返回单个 CNF 表达式(`Expr`)，仅当输入列表中至多有一个表达式为真时，该表达式才为真。 每个输入表达式都是一个文字。提示：可以借助 `itertools.combinations` 函数。
- `exactlyOne(literals)`：使用 `atLeastOne` 和 `atMostOne` 返回单个 CNF 表达式(`Expr`)，仅当输入列表中恰好有一个表达式为真时，该表达式才为真。每个输入表达式都是一个文字。

这三个函数返回的 `Expr` 必须是合取范式(CNF)。注意：在函数实现中**不可以**使用 `to_cnf` 函数，或任何辅助函数 `logic.eliminate_implications`、`logic.move_not_inwards` 和 `logic.distribute_and_over_or`。

在后面的任务中实现规划智能体时，不要在你构建的知识库中运行 `to_cnf` 函数，因为 `to_cnf` 有时会使你的逻辑表达式变得更长。在后面的任务中，可以重用你所实现的 `atLeastOne(.)`、`atMostOne(.)`和 `exactlyOne(.)`。

可以通过运行：

```
python autograder.py -q q2
```

来测试和调试你的代码。

任务 3：构建吃豆人世界（Pacphysics and Satisfiability）

在这个任务中，你将实现吃豆人世界里的基本逻辑表达式，并通过构建适当的逻辑表达式的知识库 (KB) 来证明吃豆人在哪里以及不在哪里。

请在 `logicPlan.py` 中实现以下函数：

- `pacmanSuccessorAxiomSingle`：生成一个表达式，定义吃豆人在时间 t 位于(x,y)的充分必要条件。可参考 `SLAMSuccessorAxiomSingle` 的实现。
- `pacphysicsAxioms`：生成一系列吃豆人世界的公理。 对于时间 t，返回一个逻辑语句包含
  - 对于 `all_coords` 中的所有(x,y)，如果(x,y)存在围墙，则吃豆人在时间 t 不位于(x,y)。
  - 吃豆人在时间 t 恰好位于 `non_outer_wall_coords` 中的一个。
  - 吃豆人在时间 t 恰好采取 `DIRECTIONS` 中 4 个动作中的一个。
  - 传感器：调用 `sensorModel` 的结果。可以是 `None`。

    - 转移模型：调用 `successorAxioms` 的结果，描述吃豆人在当前时间的不同位置结束时的状态。需考虑在边缘位置的情况。可以是 `None`。
- `checkLocationSatisfiability`：给定一个转移`(x0_y0, action0, x1_y1)`、`action1` 和 `problem`，该函数将返回两个模型的元组`(model1, model2)`：
    - `x0_y0`：在时间 t=0，吃豆人的位置
    - `action0`：在时间 t=0,吃豆人采取的动作，`DIRECTIONS` 中 4 个动作中的一个。
    - `x1_y1`：在时间 t=1，吃豆人可能的位置。
    - `action1`：占位符，对当前位置无影响，只是为了匹配 autograder 的格式。
    - `problem`：`logicAgents.LocMapProblem` 的一个实例。
    - 在 `model1` 中，吃豆人在时间 t=1 位于(x1,y1)。
    - 在 `model2` 中，吃豆人在时间 t=1 不位于(x1,y1)。
    - 思考如何构建知识库（KB）？
    - 使用 `findModel` 查询 SAT 求解器来返回模型。

可以通过运行：

```
python autograder.py -q q3
```

来测试和调试你的代码。

任务 4：使用逻辑进行路径规划（Path Planning with Logic）

使用命题逻辑实现以下函数来规划吃豆人到达目标位置的动作序列：

- `positionLogicPlan(problem)`：给定一个 `logicPlan.PlanningProblem` 的实例，返回一个动作字符串序列。在实现中可以考虑逐步地添加知识到知识库，并在每个时间步查询模型。

为了在较小的迷宫上测试你的代码，可以运行：

```
python pacman.py -l maze2x2 -p LogicAgent -a fn=plp
python pacman.py -l tinyMaze -p LogicAgent -a fn=plp
```

可以通过运行：

```
python autograder.py -q q4
```

来测试和调试你的代码。

任务 5：吃掉所有豆子（Eating All the Food）

使用命题逻辑实现以下函数来规划吃豆人吃掉所有豆子的动作序列：

- `foodLogicPlan(problem)`：给定一个 `logicPlan.PlanningProblem` 的实例，返回一个动作字符串序列。实现方式可以参考任务 4，需要增加 Food[x,y]_t 变量和关于 food 的后继状态公理。

为了测试你的代码，可以运行：

```
python pacman.py -l testSearch -p LogicAgent -a
fn=flp,prob=FoodPlanningProblem
```

可以通过运行：

```
python autograder.py -q q5
```

来测试和调试你的代码。

各个任务的更详细描述可以参考：
https://inst.eecs.berkeley.edu/~cs188/sp24/projects/proj3/

作业报告

本次作业**可以两人为一组来完成**，需要提交报告和代码。对于以上 5 个任务，报告需分别详细介绍代码的实现过程并分析实验结果。使用 QQ 群文件中的报告模版（课程作业\报告模版.doc）撰写实验报告。如果作业由两人完成，**需要在报告中体现两人的具体分工。**

作业提交

将作业报告存储为PDF文件，并与相应的源码打包为zip文件，命名为学号1_学号2.zip，例如 221900001_221900002.zip。

上传到百度网盘：
https://pan.baidu.com/disk/main#/transfer/send?surl=AD0AAAAAABK5xQ
注意：与课程作业 1 是不同的网址。

提交截止日期： 10 月 31 日 23:59:59

迟交作业的处理原则：迟交一周以内，折扣系数为 0.8；迟交一周以上，折扣系数为 0.6。

学术诚信

允许同学之间的相互讨论，但是**署你名字的工作必须由你自己独立完成**。

如果发现作业之间高度相似将被判定为互相抄袭行为，**抄袭和被抄袭双方的成绩都将被取消**。

应项目开发者的要求，**严禁将作业答案发布在网上**。