Who Wants to be a Professor?

Linguistic Gaming with Python

Winter Semester 2023/2024

Submitted by

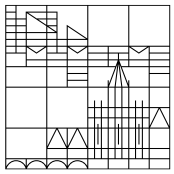
Daniel Wambach

Mary-Kate Murphy

Michelle Vuong

At the

Universität Konstanz



Faculty of Humanities

Department of Linguistics

Submitted to: Mark-Matthias Zymla

Submission Date: 15.03.2024

1. Game Idea and General Layout

Our game "Who Wants to Be a Professor?" was inspired by the television game show "Who Wants to Be a Millionaire?", where contestants try to answer as many trivia questions as they can to win the prize money. The further they get, the more money they win, with a maximum of one million dollars. Instead of general trivia questions, the questions in our game focus on the field of semantics. Hence, "Who Wants to Be a Professor?" challenges the player's semantic knowledge throughout the game with increasing degrees of difficulty, i.e., Bachelor, Master, and PhD difficulty. The incentive of the game is that players will obtain all three titles by the end of the game by passing all three levels of difficulty.

The game consists of a total of nine levels. Levels one to three display questions of Bachelor difficulty, levels four to six represent Master difficulty and seven to nine represent PhD difficulty. Each level consists of a question and four possible answers, where three of the four answers are false. To help the player with the various difficulty levels, a *Cheatsheet* is accessible from the Welcome page. This contains relevant information about propositional and predicate logic symbols, which are the primary focus of the Bachelor level questions, explanations of scope, binding and the Laws of Quantifiers for the Master and PhD level questions. With that, a joker card labelled "50/50" is included to additionally help the player along in the game. Use of the joker card eliminates two of the three false answers, leaving only one true and one false answer left to choose from, hence a 50/50 chance of winning the level.

To avoid repetitive questions, which would result in players losing interest and quickly learning off all the answers by heart, each degree of difficulty contains a set of 10 questions (see **Appendix** for the list of questions). Our program is constructed so that randomised question is chosen and displayed, and then removed to avoid it from being randomly chosen on the next level. These above steps were achieved by defining the function *dictionary_assignment()* in which three dictionaries Bachelor, Master and PhD were assigned the corresponding questions and answer. Each dictionary consists of 10 keys i.e., the questions, and each key has four values which represent the four possible answers to the question.

2. Division of Work

This section aims to provide as clear of an overview as possible as to who completed which parts. Not every piece of code will be accounted for as we also came together and collectively wrote and altered our code so that it would run smoothly with the code written by our other team members.

2.1 Joint work

In general, the game was a team effort with many parts that were created jointly in multiple sessions where the three of us came together to brainstorm, solve problems and write parts of the script together. After each session, each team member took home a task that they would work on until we met again to discuss our progress. In our first session, we brainstormed the game idea and decided to alter the game show "Who Wants to Become a Millionaire?" to create a quiz about semantics and decided to visualize it using a Graphical User Interface (GUI). We thought about which scenes would be necessary (welcome, play, info, cheat sheet, congrats, winning, losing scene) and how we should structure the main game loop.

Roughly, we divided the workload in the following way: Daniel would set up the primary game loop and functions, Michelle would set up the GUI and improve the playability of the game, and Mary-Kate would be responsible for creating and successfully implementing the Bachelor, Master and PhD questions. Although we divided the workload like this roughly, all of us were still involved in every part of the game as we continuously discussed and improved the game together each time we met.

In the final stages, we collectively edited comments in the code and adjusted the positions and colours of boxes, text and rendered images. Lastly, Michelle and Daniel created the recording to explain the code used and Mary-Kate wrote up the project description document.

2.2 Daniel

After brainstorming the necessary scenes for the game and its transitions, Daniel created the initial layouts for the scenes. For this, he created individual functions for a welcome scene, play scene, info scene, cheat scene, congrats scene, winning scene and losing scene. Daniel then defined the main game loop that continuously checks the value of the 'scene' variable and directs the flow of the game to different functions based on its value, see line 45 of code.

Daniel implemented the "50/50" joker card by creating the variable 'JOKER' outside of the loop and setting its value to 'True' so that it could be used within the game loop. Within the play scene function, 'JOKER' was set to a global variable so that if the player used the "50/50" option, the variable's boolean would change to 'False', disallowing its use for a second time. To further deter the player from using the "50/50" option for a second time, its box colour changed from medium blue to dark magenta and the "50/50" label was removed. When the

“50/50” option is clicked for the first time, the ‘JOKER’ boolean is checked and if ‘True’, two false answers are randomly chosen from variable *answers* and their corresponding boxes change colour. This visually indicates to the player that these answers are no longer available.

After Mary-Kate had defined *ba_selection()*, *ma_selection()* and *phd_selection()* functions (which randomly generate the questions and answers from a specific difficulty level) outside of the loop, Daniel created the code that would render the appropriate questions and answers of a particular difficulty level. For this, a global variable called ‘LEVEL’, which corresponds to the level the player is on, was created outside of the game loop and starts on level one. The variable ‘LEVEL’ was then included under *def play(screen)* in an if-elif-else statement and retrieved the appropriate questions and answers for the level that the player was on.

2.3 Mary-Kate

Mary-Kate was responsible for creating the Bachelor, Master and PhD difficulty level questions and answers and implementing them as dictionaries within the *dictionary_assignment()* function. The questions are keys, and the four possible answers are the values, which were written as a list consisting of four tuples. The tuples were used as we wanted the contents of the tuple to be immutable, that is, an answer and its subsequent Boolean reflecting whether the answer was ‘True’ or ‘False’. As the *dictionary_assignment()* function was defined outside of the game loop, the names of the dictionaries were turned into global variables so that they could be accessed from anywhere in the game.

Three other functions were defined outside of the game loop, namely *ba_selection()*, *ma_selection()* and *phd_selction()*. These functions were placed outside of the game loop because when a question and its answer, i.e., key and value, arose we did not want that same question to occur again. In order to delete this key and value from the dictionary we had to put the function outside of the game loop or else once the game loop started again the full original dictionary would be called upon, allowing the previously seen key and value to surface again. Within these functions, *ba_selection()*, *ma_selection()* and *phd_selction()*, a random key and that key’s corresponding value are chosen. The values are then shuffled as the first value is always the correct answer, as demonstrated by the Boolean ‘True’. Lastly, Mary-Kate created the Cheatsheet and its code to load and position the image when players clicked on ‘Cheatsheet’.

2.3 Michelle

Michelle was responsible for the GUI and visual component of the game. She created and positioned the rectangles containing the texts in the welcome, play, info, cheatsheet, congrats, losing and winning scenes. Due to the fact that some of the questions and answers were very long and would span a length greater than the display boxes provided for them, Michelle ensured that should the questions and answers exceed a certain length, they would continue on a new line under one another, bound to the size of the display box provided. For this, the *display_text()* function was defined.

To make it more explicit which level the player is currently completing on the play scene, Michelle adapted some code previously implemented in the "Memory Puzzle" game. The "Memory Puzzle" game was accessed from the session 8 folder from the Linguistic Gaming with Python class. She altered the coordinates so that a highlighted frame would appear around the level being played, for example BA-1, MA-2. If the player chooses the correct answer, the highlight subsequently moves up to the next level. For this, the function called *drawHighlightBox()* was defined.

Lastly, Michelle created the code to load images of Günther Jauch from the German version of the television game show "Wer wird Millionär?" and his delightful expressions on the welcome, losing, and winning screens. She also edited and loaded the image of the "Who Wants to Be a Millionaire?" logo to make it "Who Wants to Be a Professor?" in the welcome screen. Image references are included in the bibliography below.

3. Operating System, Python packages and Version

We ran into difficulty trying to integrate some propositional and predicate logic symbols into Pygame. We tried two Python packages, Kanren and MiniKanren, which are both packages governing logic programming, however, they are not responsible for printing any of the symbols we needed (Byrd, n.d.; Rocklin., n.d.). Instead, combinations of keyboard characters and individual letters were used as replacements, for example A became '∀', E became '∃' and v became '∨'. These replacements were also included on the Cheatsheet image that the users can check to help them progress through the levels of the game. We designed and ran our game on two operating systems, MacOS and Windows. The Python version used was Python 3.12.

4. Bibliography

Byrd, W. (n.d.). miniKanren.org. Minikanren.org. Retrieved February 21, 2024, from http://minikanren.org/

Clipart, P. (n.d.). Who Wants To Be A Millionaire? 2014 Millionaire 2017 Game show Quiz, android transparent background PNG clipart | HiClipart. Www.hiclipart.com. Retrieved March 14, 2024, from https://www.hiclipart.com/free-transparent-background-png-clipart-ogbam

Jerzy, N. (2016, May 30). Nicht jedem Sieger winkt die Million. N-Tv.de. https://www.n-tv.de/leute/Nicht-jedem-Sieger-winkt-die-Million-article17809381.html

Mundhenk, M. (2023, November 17). „Wer wird Millionär?“: Richard David Precht enthüllt, wie Günther Jauch seinen Kandidaten hilft. TVMovie. https://www.tvmovie.de/news/wer-wird-millionaer-richard-david-precht-guenther-jauch-137318

Rocklin, M. (n.d.). kanren: Logic Programming in python. PyPI. Retrieved February 21, 2024, from https://pypi.org/project/kanren/

Appendix

The items in bold and/ or with ‘X’ written beside them represent the correct answer.

Bachelor Level

1. Which of the following strings is not a formula in propositional logic?
   a. $p \rightarrow (p \wedge q)$
   b. **$(p \wedge q)(\rightarrow p)$ X**
   c. $((p \wedge r) \rightarrow q)$
   d. $\neg r \rightarrow q$

2. Which formula is a tautology?
   a. $p \rightarrow \neg r$
   b. $r \wedge q$
   c. **$r \vee \neg r$ X**
   d. $r \rightarrow q$

3. What is the main operator of $(p \wedge \neg q) \leftrightarrow (r \rightarrow (\neg(p \wedge \neg p)))$
   a. $\wedge$
   b. $\rightarrow$
   c. $\leftrightarrow$ **X**
   d. $\neg$

4. Which symbol represents disjunction?
   a. $\neg$
   b. $\rightarrow$
   c. $\vee$ **X**
   d. $\wedge$

5. Which symbol represents negation?
   a. $\neg$ **X**
   b. $\rightarrow$
   c. $\leftrightarrow$
   d. $\wedge$

6. Which symbol represents conjunction?
   a. $\neg$
   b. $\rightarrow$
   c. $\leftrightarrow$
   d. $\wedge$ **X**

7. Which symbol represents consequent?
   a. $\neg$
   b. $\rightarrow$ **X**
   c. $\leftrightarrow$
   d. $\wedge$

8. Which symbol represents equivalence?
   a. $\neg$

b. →
c. ↔ **X**
d. ∧

9. What is the symbol for a universal quantifier?
   a. ∃
   b. **∀ X**
   c. *e*
   d. a

10. What is the symbol for an existential quantifier?
   a. ∃ **X**
   b. ∀
   c. *e*
   d. a

Master Level

1. Semantics is the study of _____ meaning in language. It explores the relationships between words and their meanings, as well as how these meanings combine to form coherent sentences and convey information.
   a. syntactic
   b. **lexical X**
   c. phonological
   d. emotional

2. Which of the following allows for a more nuanced representation of relationships and quantification within statements by using variables, predicates, and quantifiers?
   a. lambda calculus
   b. propositional logic
   c. predicate logic X
   d. algebra

3. ∧, ∨, → , <-> in propsitional logical are known as:
   a. logical connectings
   b. logical operators X
   c. operating logicals
   d. connecting logicals

4. Which formula contains two bound variables?
   a. ∃x(Fy)
   b. Fx ∧ Qy
   c. ∀y(Fy → Qax)
   d. **∃x(Fx ∧ x(Fxy)) ∧ Ly X**

Explanation: If a variable is in the scope of a quantifier it is bound, e.g. ∃x(Fx). A free variable would be ∃y(Fx).

5. Which formula represents “No pilot is friendly” ?
   a. **∀x(Px→ ¬Fx) X**
   b. ¬∀x(Px→ ¬Fx)
   c. ∃x(Px ∧ Fx)
   d. ∀x(¬Fx)

6. Which formula represents “Every pilot has a friendly sibling” ?
   a. ∃x(Px ∧ ∃y(Fy ∧ Sx,y))
   b. **∀x(Px→ (∃yFy ∧ Sx,y)) X**
   c. ∀x(Px ∧ (∃y(Fy ∧ Sx,y)))
   d. ∃x(Px → ∃y(Fy ∧ Sx,y))

7. Which formula contains two free variables?
   a. ∃x(Fy ∨ Qx)
   b. **Fx ∧ Qy X**
   c. ∀y(Fy → Qa,x) → ∃x(Fx ∨ Sx)
   d. ∃x(Fx ∧ ∀x(Fx,y)) ∧ Ly

8. ((p ∧ q) → ¬r) is a…
   a. **contingency X**
   b. tautology
   c. contradiction
   d. none of the above

9. ((p ∧ q) → (r ∧ ¬s)) is a…
   a. tautology
   b. contradiction
   c. **contingency X**
   d. none of the above

10. What propositional logic formula is said to be ‘necessarily false’ or ‘false by logical necessity’?
    a. tautology
    b. contradiction X
    c. contingency
    d. none of the above

PhD Level

1. What formula does not represent “Nobody saw nobody”?
   a. ¬∀x¬∀y(Sx,y)
   b. **¬∃x¬∃y(Sx,y) X**
   c. ∀x∃y(Sx,y)
   d. ¬∃x∀y(Sx,y)

This means “Not everyone saw everyone”, entailing that there does not exist any person x who saw any person y.

3. Which formula reads "Every candidate told Mary a lie" but the lie was always different?
   a. **$\forall x \exists y(Cx \rightarrow Ly \wedge Tx,y,m)$ X**
   b. $\exists y \forall x(Cx \rightarrow Ly \wedge Tx,y,m)$
   c. $\neg \forall x(Cx \rightarrow \exists yLy \wedge Tx,y,m)$
   d. $\forall x \exists y(Cx \wedge LyTx,y,m)$

3. Which formula reads "I won't be upset if you don't come"?
   a. **$\neg q \rightarrow \neg p$ X**
   a. $\neg(q \rightarrow p)$
   b. $\neg q \rightarrow p$
   c. $q \rightarrow \neg p$

4. Which is the stronger reading if p = Sue visited grandma, q = Sue visited grandpa?
   a. **$\neg p \wedge \neg q$ X**
   b. $\neg(p \wedge q)$
   c. $\neg\neg(p \rightarrow \neg q)$
   d. none of the above

5. Given Q is a formula, $Q \rightarrow \forall x(Wx) \leftrightarrow \forall x(Q \rightarrow Wx)$ follows the law of quantifier movement, provided that:
   a. **x is not free in Q X**
   b. x is free in Q
   c. Q does not contain quantifiers
   d. Wx is not existentially quantified

6. The following, $\forall x(\neg Wx) \leftrightarrow \neg \exists x(Wx)$ is an example of:
   a. **Laws of Quantifier Negation X**
   b. Laws of Quantifier Distribution
   c. Laws of Quantifier (In)dependence
   d. Laws of Quantifier Movement

7. The following, $\exists x \forall y(Wx,y) \rightarrow \forall y \exists x(Wx,y)$ is an example of:
   a. Laws of Quantifier (In)dependence X
   b. Laws of Quantifier Distribution
   c. Laws of Quantifier Negation
   d. Laws of Quantifier Movement

8. The following, $\forall x(Wx) \vee \forall x(Px) \rightarrow \forall x(Wx \vee Px)$ is an example of:
   a. **Laws of Quantifier Distribution X**
   b. Laws of Quantifier Negation
   c. Laws of Quantifier (In)dependence
   d. Laws of Quantifier Movement

9. Which formula reads "Every candidate was interviewed by Penny or by Bernadette" in its strong reading that only one did all the interviews?
   a. $\forall x\ (Cx \rightarrow I(p,x)) \vee (\forall x\ (Cx \rightarrow I(b,x)$ X
   b. $\forall x\ (Cx \rightarrow I(p,x) \vee I(b,x))$
   c. $\forall x\ (Cx \rightarrow I(x,p)) \vee (\forall x\ (Cx \rightarrow I(x,b)$
   d. $\exists x\ (Cx \rightarrow I(p,x) \vee I(b,x))$

10. Which formula reads “Lucia fed a cat that every child liked” ?
    a. **∃x(Cx ∧ ∀y(CHy → Ly,x) ∧ Fl,x)) X**
    b. ∀x((CHx ∧ ∃y(Cy ∧ Lx,y)) → Fl,x))
    c. ∃x((CHx ∧ ∃y(Cy ∧ Lx,y)) → Fl,x))
    d. ∃x(Cx ∧ ∀y(CHy → Ly,x) → Fl,x))