Translator de probleme cripto-aritmetice

Vlad Ursache

Cuprins

Rezumat

Aceasta este documentația pentru mini-proiectul de logică în cadrul disciplinei Inteligență Artificială. Acesta constă într-un translator din expresii aritmetice în cod pentru mace4 în vederea rezolvării a 18 probleme cripto-aritmetice: 2 de la laborator, 15 de pe Braingle și una formulată de mine.

1 Introducere

Motivația alegerii temei 32 este afinitatea personală pentru puzzle-uri și criptografie. Am vrut de asemenea să scriu un program care să rezolve o întreagă clasă de probleme. Acest utilitar scris în Python rezolvă clasa de probleme "Mathology". Sintaxa este următoarea: "python3 main.py problem.txt problem.in [base] [> logs.out]" .

2 Mod de funcționare

De dragul exemplificării, vom folosi problema "True Love". Fișierul asociat conține expresia "Vlad + Bia = love". Translatorul, scris în Python, preia acest fișier ca argument din linia de comandă și generează fișierul .in care poate fi pasat utilitarului mace4. Acest fișier arată astfel:

```
set(arithmetic).
assign(domain_size, 10).
assign(max_models, -1).

list(distinct).
        [V, L, A, D, B, I, O, E].
end_of_list.

formulas(assumptions).
        V != 0.
        B != 0.
        L != 0.
        V * 1000 + L * 100 + A * 10 + D + B * 100 + I * 10 + A =
            L * 1000 + O * 100 + V * 10 + E.
end_of_list.
```

Modelele sunt generate de mace4. Mai multe detalii putem găsi în folder-ul output.

```
============================== STATISTICS ==============================

For domain size 10.

Current CPU time: 0.00 seconds (total CPU time: 3.87 seconds).
Ground clauses: seen=32, kept=32.
Selections=554482, assignments=5544820, propagations=0, current_models=103.
Rewrite_terms=29824128, rewrite_bools=27759925, indexes=0.
Rules_from_neg_clauses=0, cross_offs=716084.

============================== end of statistics =======================

User_CPU=3.87, System_CPU=0.22, Wall_clock=4.

Exiting with 103 models.

Process 30795 exit (all_models) Wed Dec 14 20:21:54 2022
The process finished Wed Dec 14 20:21:54 2022
```

3 Detalii de implementare

Script-ul citește și analizează conținutul. Acesta construiește mulțimea de litere folosită în puzzle. Acesta este domeniul problemei. Scriptul atenționează utilizatorul în cazul în care folosește fișiere invalide. Mai departe, generează conținutul fișierului care urmează a fi pasat. Un parametru opțional la nivelul liniei de comandă este și baza de numerație utilizată pentru generarea soluțiilor (implicit 10). În cazul în care baza este prea mică sau numărul de variabile este prea mare, utilizatorul este atenționat. O alternativă mult mai elegantă ar fi fost utilizarea modulului "ortools.sat.python", dar la laborator folosim mace4. În cele din urmă, conținutul este scris, iar o comandă la nivelul sistemului de operare apelează mace4 cu parametrii necesari.

4 Concluzii

Translatorul preia expresii cripto-aritmetice sub formă de fișiere .txt. O comandă la nivelul sistemului de operare trimite fișierul generat către mace4 în vederea rezolvării propriu-zise a puzzle-ului. Script-ul rezolvă așadar toate problemele din clasa "Mathology", inclusiv pe cele 18 atașate în proiect.

A Completări

Această secțiune conține codul sursă și alte completări.

```
import os
import sys
import re


def main():
    try:
        with open(sys.argv[1], "r") as file:
            contents = file.read()
    except FileNotFoundError:
        print("Input file not found!")

    try:
        base = int(sys.argv[3])
    except IndexError:
        base = 10

    contents = re.sub(r'[+=\n]', ' ', contents)
```

```
    contents = contents.upper()
    letters = []
    for character in contents:
        if character not in letters and character != ' ':
            letters += character
    operators = contents.split()

    assert len(letters) <= base

    contents = "set(arithmetic).\nassign(domain_size, " + str(base) + \
                            ").\nassign(max_models, -1).\n\nlist(" \
                            "distinct).\n\t["

    length = len(letters)
    for i in range(length):
        contents += letters[i]
        if i != length - 1:
            contents += ", "
    contents += "].\nend_of_list.\n\nformulas(assumptions).\n"

    for operator in operators:
        contents += "\t" + operator[0] + " != 0.\n"
    contents += "\t"

    length = len(operators)
    for i in range(length):
        j = len(operators[i])
        for character in operators[i]:
            factor = pow(base, j - 1)
            if factor == 1:
                contents += character
            else:
                contents += character + " * " + str(factor)
            j -= 1
            if (i == length - 2) and j == 0:
                contents += " = "
            elif not ((i == length - 1) and j == 0):
                contents += " + "
    contents += ".\nend_of_list.\n"

    try:
        with open(sys.argv[2], "w") as file:
            file.write(contents)
    except FileNotFoundError:
        print("Output file not found!")

    command = "mace4 -f " + sys.argv[2]
    os.system(command)


if __name__ == '__main__':
    main()
```

În total, avem 70 de linii de cod.