Solve Dinner Planning with CSP

HTWG - MSI - DIMA - Task 1B

Johannes Brandenburger, Tobias Tögel

github.com/johannesbrandenburger/htwg-msi-dima

Approach

- formulate dependencies
- build CNF (clauses)
- create own simple SAT algorithm
  - search for unit clauses ("A")
  - add them to result
  - eliminate them and their negations in the clauses
  - recursively call the function with additional unit clauses

Implementation

- formulate dependencies
- build CNF (clauses)
- create own simple SAT algorithm
  - search for unit clauses ("A")
  - add them to result
  - eliminate them and their negations in the clauses
  - recursively call the function with additional unit clauses

```
## Abbreviations

Mr H
Mrs W
Emma E
Georg G
Ivana I

## Constrains in CNF

(-H or W)          -> ["-H", "W"]
(I or G)           -> ["I", "G"]
(W or E)           -> ["W", "E"]
(not W or not E)   -> ["-W", "-
(-E or G)          -> ["-E", "G"]
(-G or E)          -> ["-G", "E"]
(-I or G)          -> ["-I", "G"]
(-I or H)          -> ["-I", "H"]
```

Implementation

- formulate dependencies
- build CNF (clauses)
- create own simple SAT algorithm
  - search for unit clauses ("A")
  - add them to result
  - eliminate them and their negations in the clauses
  - recursively call the function with additional unit clauses

```python
def remove_unit_clauses(result, clauses):
    while True:
        unitClauses = []
        for clause in clauses:
            if len(clause) == 1 and clause[0] not in unitClauses:
                unitClauses.append(clause[0])
        if len(unitClauses) == 0:
            break

        # remove all clauses that contain unit clauses (literals)
        for (i, clause) in enumerate(clauses):
            for unitClause in unitClauses:
                if unitClause in clause:
                    clauses[i] = None
        clauses = [clause for clause in clauses if clause is not None]

        # remove all negated unit clauses (literals) from all clauses
        for (i, clause) in enumerate(clauses):
            for unitClause in unitClauses:
                if neg(unitClause) in clause:
                    clauses[i].remove(neg(unitClause))

        # add the unitClauses to the result
        for unitClause in unitClauses:

            # if unitClause is already negated in the result -> no solution
            if neg(unitClause) in result:
                return [], clauses

            if unitClause not in result:
                result.append(unitClause)

    return unique(result), clauses
```

Implementation

- formulate dependencies
- build CNF (clauses)
- create own simple SAT algorithm
  - search for unit clauses ("A")
  - add them to result
  - eliminate them and their negations in the clauses
  - recursively call the function with additional unit clauses

```python
def simple_csp_solver(result: list[str], clauses: list[list[str]]):

    # first remove all unit clauses
    result, clauses = remove_unit_clauses(result, clauses)

    # if null clause is present, return an empty list
    for clause in clauses:
        if len(clause) == 0:
            return []

    # if no clauses are left, return the result
    if len(clauses) == 0:
        return result

    # call recursively with two new clauses
    literal = clauses[0][0]
    clauses_1 = clauses + [[literal]]
    clauses_2 = clauses + [[neg(literal)]]
    result_1 = simple_csp_solver(result, clauses_1)
    if len(result_1) > 0:
        return result_1
    return simple_csp_solver(result, clauses_2)

# solve the dinner planning problem
constrains = [...] # see markdown
should_solution = ["-H", "-W", "-I", "G", "E"] # found by brute force
is_solution = simple_csp_solver([], constrains)
print("should_solution: ", should_solution) # ['-H', '-W', '-I', 'G',
print("is_solution: ", is_solution) # ['-H', '-I', '-W', 'E', 'G']
assert(sort_array(should_solution) == sort_array(is_solution))
```

Solve Dinner Planning with CSP

HTWG - MSI - DIMA - Task 1B

Johannes Brandenburger, Tobias Tögel

github.com/johannesbrandenburger/htwg-msi-dima