Formalne metode u softverskom inženjerstvu

Laboratorijska vježba br. 4

U ovoj laboratorijskoj vježbi upoznaćemo se sa konceptom redukcije problema i primijeniti ga za rješavanje jednog praktičnog problema raspoređivanja.

Ispod je data definicija funkcija za rješavanja SAT problema.

In [2]:
```python
def verify_sat(formula, assignment):
    return all(any((lit > 0) == assignment[abs(lit)] for lit in clause) for clause

def solve_sat(formula):
    """
    Solve a 3SAT formula given as a list of clauses.
    Each clause is a list of integers. The integer absolute value is the variable i
    If the integer is positive, the variable is unnegated; if negative, it is negat
    Example clause: [1, -2, 3] represents (x1 OR ¬x2 OR x3)
    """
    # Extract all variable indices from the formula
    variables = sorted({abs(lit) for clause in formula for lit in clause})
    n = len(variables)

    # Brute-force over all possible assignments (2^n possibilities)
    # Algorithmic complexity: O(2^n * m * k), where n is the number of variables, m
    for bits in range(2**n):
        # Build assignment: dictionary mapping variable -> Boolean value
        assignment = {var: bool(bits >> i & 1) for i, var in enumerate(variables)}
        if verify_sat(formula, assignment):
            return assignment
    return None
```

In [3]:
```python
# Example usage:
formula = [
    [-1,  2,  3], # (¬x1 OR  x2 OR  x3) AND
    [-1, -2, -3], # (¬x1 OR ¬x2 OR ¬x3) AND
    [ 2, -3,  4], # ( x2 OR ¬x3 OR  x4) AND
    [ 3,  4,  5], # ( x3 OR  x4 OR  x5)
]
solution = solve_sat(formula)
print(solution)
```

```
{1: False, 2: True, 3: True, 4: False, 5: False}
```

Zadatak

Neophodno je izvršiti raspoređivanje različitih vrsta časova za fitnes studio u N dostupnih termina. Fitnes studio ima dovoljan broj sala, tako da količina sala ne predstavlja problem pri raspoređivanju. Međutim, neke vrste časova zahtijevaju specifične sale, tako da postoje

mogući konflikti u raspoređivanju zbog kojih se dvije različite vrste časova ne mogu održavati istovremeno. Takođe, moguće je da različte vrste časova drži isti instruktor, tako da se onda te vrste časova ne mogu održavati istovremeno. Pored toga, postoje i restrikcije po pitanju termina u kojima se pojedinačne vrste časove mogu održavati, npr. zbog preferencija instruktora.

Ispod je dat programski kod koji predstavlja sve ulazne podatke za određivanje rasporeda.

In [32]:
```
# ------------------------------
# Fitness Studio Class Scheduling Input
# ------------------------------

# Available time slots.
time_slots = [1, 2, 3]

# Classes offered at the fitness studio.
classes = ['Yoga', 'Spinning', 'Pilates', 'Zumba', 'Cardio', 'Strength']

# Classes that cannot be scheduled at the same time.
conflicts = [
    ('Yoga', 'Pilates'),
    ('Yoga', 'Cardio'),
    ('Spinning', 'Strength'),
    ('Zumba', 'Strength')
]

# Which time slots each class may use.
restrictions = {
    'Yoga':      [1, 2],
    'Pilates':   [1],
    'Spinning':  [2, 3],
    'Zumba':     [1],
    'Cardio':    [3],
    'Strength':  [2]
}
```

Ovi podaci treba da se enkoduju u SAT formulu. To je djelimično urađeno ispod, a kao zadatak neophodno je ispuniti TODO stavke.

In [33]:
```
# ------------------------------
# Mapping Variables and Generating Clauses
# ------------------------------

# Map each (class, time_slot) pair to a unique integer variable.
vars = {(cls, t) : i for i, (cls, t) in enumerate(((cls, t) for cls in classes for

clauses = []

# Constraint 1: Each class must be scheduled in at least one time slot.
clauses += [[vars[(cls, t)] for t in time_slots] for cls in classes]

# Constraint 2: Each class is scheduled in at most one time slot.
clauses += [[-vars[(cls, t1)], -vars[(cls, t2)]] for cls in classes for i, t1 in en
```

```
# TODO Constraint 3: Conflicting classes cannot share the same time slot.
clauses += [[-vars[cls1, t], -vars[cls2,t]] for cls1, cls2 in conflicts for t in ti

# Constraint 4: Imposed scheduling restrictions.
clauses += [[-vars[(cls, t)]] for cls in classes for t in time_slots if t not in se
```

Nakon što je to prethodna stavka završena, možemo pokušati riješiti SAT instancu.

In [34]:
```
# ------------------------------
# Solve the 3SAT Instance
# ------------------------------
solution = solve_sat(clauses)
if solution:
    # Decode the solution into a schedule.
    schedule = {}
    for (cls, t), var in vars.items():
        if solution.get(var, False):
            schedule[cls] = t
    print("\nFitness Class Schedule:")
    print(schedule)
else:
    print("\nNo valid fitness class schedule found.")
```

```
Fitness Class Schedule:
{'Yoga': 2, 'Spinning': 3, 'Pilates': 1, 'Zumba': 1, 'Cardio': 3, 'Strength': 2}
```

Ako nije moguće pronaći raspored, pokušati minimalno izmijeniti ograničenja, tako da bude moguće dobiti rješenje. Zatim ponovo izvršiti redukciju i proceduru rješavanja.