PP 2017-2018

TEMA 2

PROLOG

Termen predare: 20 mai (soft)
23 mai (hard)

Responsabili tema: Alin Popa
Mihai Masala

Publicare: 7 mai
Ultimul update: **13 mai 21:20**

1. Introducere

Tema constă în rezolvarea (folosind Prolog) a unei probleme de tip “Shortest path” [0], în care path-ul găsit trebuie sa respecte o constrângere dată printr-o formulă LTL (linear temporal logic) [1].

Concret: fie un graf orientat în care fiecare nod are o culoare.
Graful este reprezentat în Prolog printr-o listă ce conține două elemente [**C**, **E**]:
C: o listă de liste [n_i, c_i] – cu semnificația că nodul n_i din graf are culoarea c_i
E: o listă de liste [n_i, n_j] – cu semnificația că există o muchie de la nodul n_i la nodul n_j (atenție, nu e obligatoriu să existe și muchia în sens invers)

Exemplu:

```
G = [
     [[1,galben], [2,rosu], [3,albastru], [4,verde], [5,rosu],
     [6,albastru], [7,rosu], [8,galben], [9,albastru],
     [10,mov]],
     [[1,2],[1,3],[2,6],[3,2],[3,4],[4,5],[5,6],[5,7],[5,8],
     [6,4],[6,5],[6,7],[7,8],[8,5],[8,9],[8,10],[9,10]]
].
```

O formulă LTL din această temă va arăta în felul următor:

Formula ::= Formula ∧ Formula (și logic) |

Formula ∨ Formula (sau logic) |

~Formula (negație) |

<culoare> (adică: primul nod are culoarea <culoare>) |

F <culoare> (future) |

G <culoare> (global) |

<culoare> U <culoare> (until) |

X Formula (next) |

valid.

F, G, U și X **și valid** au următoarele definiții:

Future verde = path-ul trebuie să conțină la un moment dat un nod verde

Global verde = path-ul trebuie să aibă DOAR noduri verzi

verde Until roșu = până la primul nod roșu, path-ul trebuie să aibă DOAR noduri verzi

Next F = începând cu nodul următor, formula F are valoarea true (adică pentru subpath-ul care începe cu al doilea nod din path și merge până la final)

Formula care unifică cu "valid" este considerată îndeplinită.

În Prolog, o formulă LTL va unifica fie cu o culoare (un atom), **fie cu atomul valid**, fie cu unul dintre următoarele predicate (unde cu C sunt notate obiecte ce unifică cu culori, iar cu F cele care unifică cu formule):

- and(F1, F2)
- or(F1, F2)
- not(F)
- future(C)
- global(C)

- until(C1, C2)
- next(F)

Exemplu:

```
F = and(
      or(rosu, verde),
      or(
            next(until(mov, roz)),
            next(next(global(alb)))
      )
).
```

Adică: path-ul trebuie să aibă primul nod fie roșu, fie verde; și fie, începând cu al doilea nod, toate nodurile sunt mov până la întâlnirea primului nod roz, fie, incepând de la al treilea nod, toate nodurile sunt albe.

[0] https://en.wikipedia.org/wiki/Shortest_path_problem
[1] https://en.wikipedia.org/wiki/Linear_temporal_logic

2. Cerință

Să se scrie un predicat în Prolog numit getPath/5:
`getPath(From, To, Graph, Formula, Path).`

From – nodul de start
To – nodul de final
Graph – un graf reprezentat conform definiției de mai sus
Formula – o formulă LTL reprezentată conform definiției de mai sus
Path – o listă de noduri ce respectă următoarele proprietăți:

1. începe cu nodul From și se termină cu nodul To
2. este un path în graf (i.e. există muchii între nodurile consecutive din listă) care nu trece de două ori prin același nod
3. respectă formula Formula
4. este **cea mai scurtă cale** (ca număr de noduri) dintre căile ce respectă proprietățile 1-3

Exemplu de query:

```
getPath(1, 2, G, F, P),
G = [[[1,alb], [3, rosu], [2, alb], [4, rosu]],
     [[1,2], [1,3], [3,2], [3,4], [4,2]]],
F = future(rosu).
```

Pentru acest query, predicatul vostru ar trebui să unfice:

```
P = [1,3,2]
```

Alt exemplu:

```
getPath(1, 2, G, F, P),
G = [[[1,alb], [3, rosu], [2, alb], [4, rosu]],
     [[1,2], [1,3], [3,2], [3,4], [4,2]]],
F = future(negru).
```

Rezultatul trebuie să fie `false` (nu există niciun path de la 1 la 2 care să respecte formula pentru ca niciun nod din graf nu e negru).

3. Punctaj

Tema va fi punctată automat. Testele vor fi publicate soon™

Tema valorează **1.75 puncte** din nota finală la PP. Punctajul este distribuit astfel:

- 1.00 – teste automate

- 0.50 – implementare eficienta
- 0.25 – aspectul codului și fișier README

Depunctare pentru trimiterea temei după deadline-ul soft: -0.10 pe zi

4. FAQ

1. **Ce înseamnă "implementare eficientă"?**

Algoritmul vostru poate să:

- nu genereze în mod inutil toate căile posibile daca unele din ele evident nu sunt bune
- nu testeze degeaba o cale de lungime n+1 dacă există o cale validă de lungime n (hint: folosiți BFS)
- nu recalculeze cu totul rezultatul unei formule daca o parte din formulă s-a îndeplinit deja la nodurile anterioare (exemplu: dacă formula este and(or(roșu,negru), F) și primul nod este unul roșu, de la al doilea nod pe care îl adăugați in path nu mai are sens re-evaluarea or(roșu,negru) pentru primul nod – este necesara doar evaluarea F pentru restul path-ului)

Implementarea folosind BFS asigură obținerea punctajului pentru eficiență. Menționați în README orice considerați că aduce un plus de eficiență programului vostru față de o abordare naivă (puteți primi bonus).

2. **Graful poate conține cicluri?**

Da. Dar path-ul nu trebuie să treacă de două ori prin același nod / aceeași muchie.

3. **Dacă formula este until(X, Y), un path cu toate nodurile de culoarea X care nu conține niciun nod de culoarea Y este valid?**

Da, folosim convenția numită "weak until".

4. **Dacă formula este next(next(verde)) și există muchie de la From la To, un path care conține doar [From, To] este valid?**

Nu, toate formulele next(F) trebuie satisfăcute.

5. **Ce trebuie sa scriu în README?**

Câteva indicații:

- fii scurt și la obiect
- scrie cum funcționeaza predicatul tău (1-2 fraze, max. 500 caractere)
- scrie ce abordare ai folosit pentru eficiență, dacă e cazul
- scrie ce ai încercat și nu ți-a ieșit dar crezi ca erai pe aproape
- numele și grupa

6. **Există punctaje parțiale?**

Da, testele vor fi făcute astfel încât să testeze pe rând funcționalitățile temei (exemplu: puteți primi punctaj pentru implemetarea future chiar dacă nu ați implementat global).

7. **Graful este orientat?**

Da.

8. **Formula se aplică și la nodul de start?**

Da, spre exemplu:

```
and(rosu,
    and(next(negru),
        and(next(next(alb)), next(next(next(verde))) )
        ))
```

Formula de mai sus arată că path-ul căutat trebuie să aibă:

- primul nod (nodul de start) roșu
- al doilea nod negru
- al treilea nod alb

- și al patrulea nod verde

Aceeași formulă putea fi exprimată și altfel:

```
and( rosu,
     next(and( negru,
               and(next(alb), next(next(verde))
          ))
```

9. **Există bonus?**

Da, puteți primi punctaj bonus pentru lucruri ca:

- structura codului
- alte artificii care aduc eficiență (în afară de BFS)
- cazuri speciale acoperite
- și altele

Bonusul se reportează în afara celor 1.75 puncte ale temei în limita a 0.25 puncte.

10. **Update 13 mai 21:20**

Am introdus atomul „valid” cu care o formulă poate unifica. Acest atom nu va apărea în sub-structuri ale formulei (exemplu: and(valid, ..)). Va fi folosit numai pentru testele simple, în care se verifică parcurgerea corectă a grafului fără a avea restricții pentru culorile nodurilor.