Artificial intelligence - Project 2
- Constraints satisfaction and predicate propositional logic -

Dorofte Andrei, Iacob Liviu

8/12/2020

1 Problem description

Pentru a putea determina cine are prioritate in intersectie se urmaresc niste reguli de circulatie din codul rutier: Daca este politist in intersectie nu mai conteaza restul prioritatilor.

Daca pe o strada este semafor functional, atunci indicatorul aferent este irelevant si se respecta semnificatia semaforului.

Daca nu este nici semafor si nici politist, atunci se ia in considerare prioritatea de indicator. Daca 2 strazi au aceeasi prioritate de indicator(acelasi indicator), atunci se aplica prioritatea de dreapta (daca este o masina in dreapta ta cu acelasi indicator, atunci el are prioritate).

Pe de asupra, orice prioritate de indicator este anulata daca pe trecerea de pietoni se afla un pieton. Acest lucru nu este valabil si pentru strazile cu semafoare, deoarece daca semaforul este verde pentru masini, atunci semaforul pentru pietoni este rosu => nu ar trebui sa fie pietoni pe trecere.

Pe fiecare strada poate sa fie masina sau nu. Fiecare masina are o directie(x, y) ceea ce inseamna ca masina vine de pe strada x si vrea sa mearga pe strada y.

Pe fiecare trecere de pietoni se poate afla sau nu un pieton.

Rezolvarea problemei presupune demonstrarea faptului ca o masina poate sau nu merge pe directia (a, b). Ca masina sa poate merge, trebuie sa nu fie politist in intersectie si sa aiba prioritate fata de toate celelalte masini din intersectie, dar si sa nu fie pieton pe trecere nici pe strada a si nici pe strada b.

In imaginea de mai jos este reprezentat un caz de testare. Sunt masini pe strazile a (cu directie(a, b)), b (cu directie(b, c)) si c (cu directie(c, e)). In intersectie nu este politist. Pe strada a este un semafor care indica culoarea verde cat si un indicator de prioritate.Pe strada b este un indicator de stop si un pieton angajat in traversare. Pe strada c, d si e nu sunt nici indicatoare (none), si nici pietoni angajati in traversare.

Se testeaza pass(a, b), parametrii pentru pass() trebuie sa fie aceeasi cu cei din directie(). Se afiseaza o eroare, deoarece masina nu poate trece chiar daca are verde din cauza pietonului de pe b. Daca se schimba pieton(b) cu -pieton(b) se observa ca pass(a, b) va fi adevarat. *Problema incepe de la o intersectie din Cluj, mai exact intersectia strazii Marinescu cu Babes. Aceasta intresectie este una de 5 strazi, avand indicatoare, semafoare, si treceri de pietoni. Din acest fapt rezulta diferite prioritati. Se poate de asemenea ca in intersectie sa se afle un politist, fapt care anuleaza orice prioritate de semafor sau de indicator. Pentru "usurarea" problemei, am presupus ca toate strazile sunt cu sens dublu de circulatie.*

Pentru a putea determina cine are prioritate in intersectie se urmaresc niste reguli de circulatie din codul rutier: Daca este politist in intersectie nu mai conteaza restul prioritatilor.

Daca pe o strada este semafor functional, atunci indicatorul aferent este irelevant si se respecta semnificatia semaforului.

Daca nu este nici semafor si nici politist, atunci se ia in considerare prioritatea de indicator. Daca 2 strazi au aceeasi prioritate de indicator(acelasi indicator), atunci se aplica prioritatea de dreapta (daca este o masina in dreapta ta cu acelasi indicator, atunci el are prioritate).

Pe de asupra, orice prioritate de indicator este anulata daca pe trecerea de pietoni se afla un pieton. Acest lucru nu este valabil si pentru strazile cu semafoare, deoarece daca semaforul este verde pentru masini, atunci semaforul pentru pietoni este rosu => nu ar trebui sa fie pietoni pe trecere.

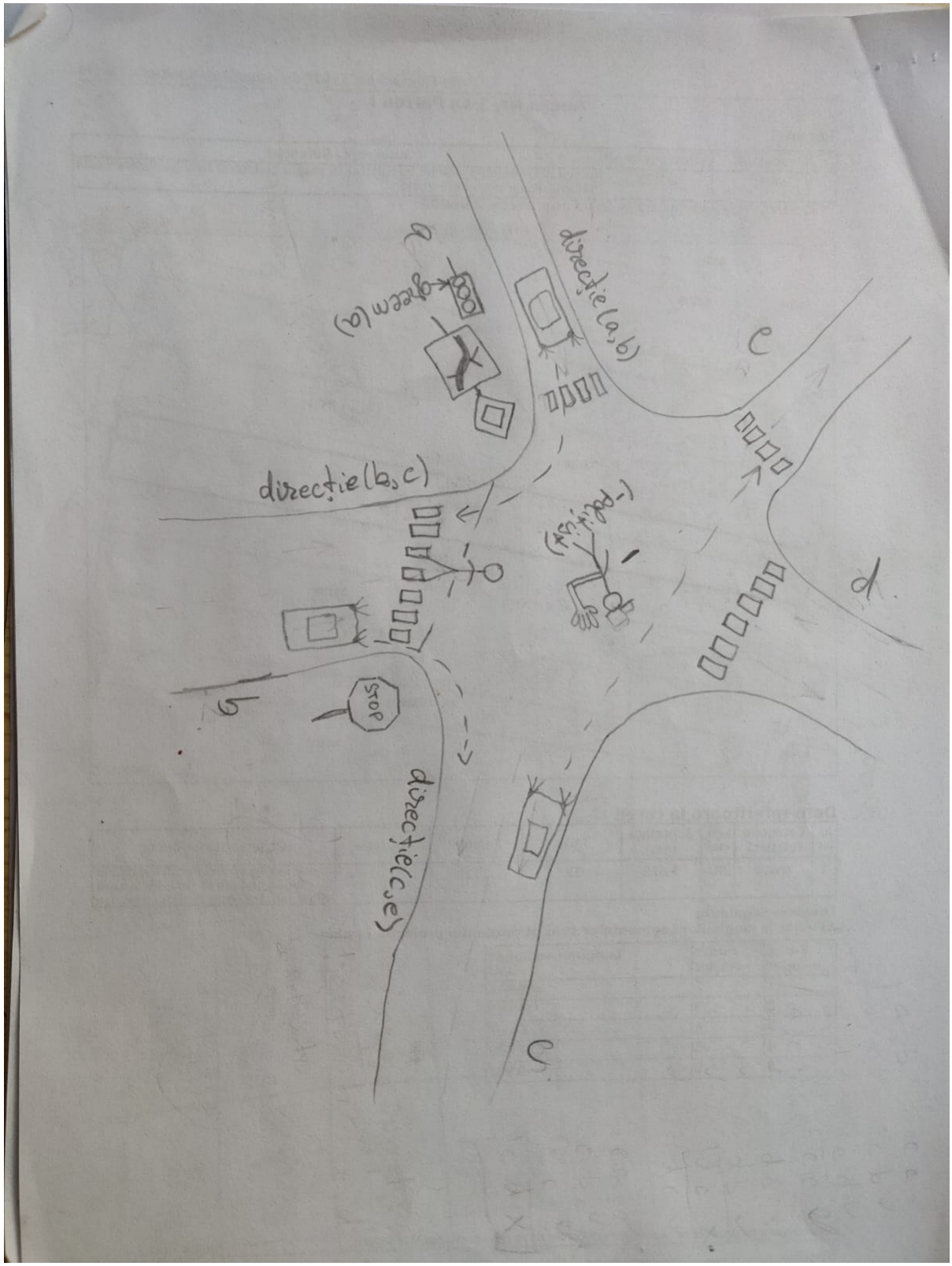
Pe fiecare strada poate sa fie masina sau nu. Fiecare masina are o directie(x, y) ceea ce inseamna ca masina vine de pe strada x si vrea sa mearga pe strada y.

Pe fiecare trecere de pietoni se poate afla sau nu un pieton.

Rezolvarea problemei presupune demonstrarea faptului ca o masina poate sau nu merge pe directia (a, b). Ca masina sa poate merge, trebuie sa nu fie politist in intersectie si sa aiba prioritate fata de toate celelalte masini din intersectie, dar si sa nu fie pieton pe trecere nici pe strada a si nici pe strada b.

In imaginea de mai jos este reprezentat un caz de testare. Sunt masini pe strazile a (cu directie(a, b)), b (cu directie(b, c)) si c (cu directie(c, e)). In intersectie nu este politist. Pe strada a este un semafor care indica culoarea verde cat si un indicator de prioritate.Pe strada b este un indicator de stop si un pieton angajat in traversare. Pe strada c, d si e nu sunt nici indicatoare (none), si nici pietoni angajati in traversare.

Se testeaza pass(a, b), parametrii pentru pass() trebuie sa fie aceeasi cu cei din directie(). Se afiseaza o eroare, deoarece masina nu poate trece chiar daca are verde din cauza pietonului de pe b. Daca se schimba pieton(b) cu -pieton(b) se observa ca pass(a, b) va fi adevarat. .


a
semn(a)
directie(a,b)
directie(b,c)
STOP
directie(c,e)

2 Code description

differentFrom(x, y): se refera la faptul ca strada x e diferita de strada y. Au fost scrise constante pentru toate combinatiile de cate 2 strazi diferite. Este un predicat simetric.

rightPriority(x, y): se refera la faptul ca x si y sunt doua strazi sunt alaturate, si strada x are prioritate de dreapta fata de strada y. Se aplica doar in cazul in care cele doua strazi au aceeasi prioritate dupa celelalte criterii.

semaphore(x): se refera la faptul ca pe strada x exista un semafor. Acest predicat implica culoarea semaforului care poate fi verde sau galben sau rosu, dar doar o singura culoare pe rulare. Pentru modelarea acestui fapt au fost scrise relatii intre predicatele green(x), yellow(x) si red(x).

indicator(x): se refera la faptul ca pe strada x exista un indicator. Acesta poate sa fie unul de piroritate(x), de stop(x), sau none(x) (lipseste indicatorul). Modelarea predicatului none(x) ca un indicator a fost pentru a generaliza problema si a putea schimba ulterior mai usor datele de intrare. La fel ca la semaphore(x), indicator(x) implica doar un singur indicator pe rulare.

samePriority(x, y): se refera la faptul ca doua strazi au aceeasi prioritate de indicator sau semafor. Acest predicat este simetric si daca este adevarat pentru 2 strazi, atunci se aplica prioritatea de dreapta intre cele 2 strazi.

priority(x, y): se refera la faptul ca strada x are prioritate fata de strada y. Aici au fost folosite toate predicatele anterioare si regulile de circulatie pentru a determina prioritatile dintre strazi. Au fost luate in calcul toate variantele posibile, dar in cazul in care doua strazi au aceeasi prioritate si nu sunt alaturate(nu se poate aplica prioritate de dreapta), acest predicat nu poate determina care dintre ele are prioritate. Drept urmare va rezulta ca niciunul nu are prioritate fata de celalalt.

intra2(x): se refera la posibilitatea masinii (care vine de pe strada x) de a intra in intersectie. Tine cont daca are pieton pe trecere si daca are prioritate fata de toate celelalte masini din intersectie.

pass(x, y): se refera la posibilitatea masinii (care vine de pe strada x si vrea sa mearga pe strada y) de a traversa intersectia. Acest predicat tine cont de predicatul intra2(x), de directia pe care masina doreste sa mearga, daca este politist sau nu in intersectie, si daca este pieton pe strada y. Verificarea pietonului de pe strada y este facuta pentru a nu bloca intersectia, deoarece daca poate intra in intersectie nu inseamna si ca poate iesi din intersectie. Acesta este predicatul scop, care va fi demonstrat ca goal pentru testarea proiectului.

Datele de intrare pentru modelarea problemei sunt:

politist | -politist. Nu se poate sa fie si politist si -politist.

semaphore(x), urmat de culoarea unica semafolului de pe strada x (green(x) | yellow(x) | red(x)). Daca se specifica -semaphore(x) atunci nu trebuie specificata culoare, deoarece nu este semafor.

indicator(x), urmat de semnificatia unica indicatorului de pe strada x (prioritate(x) | stop(x) | none(x)). Trebuie specificat acest atribut pentru toate strazile, indiferent daca au sau nu indicator(pe strazile care nu au indicator, se va specifica indicator(x) si none(x)).

pieton(x), daca este pieton pe trecerea de pe strada x sau -pieton(x) daca nu este pieton.

masina(x), daca este masina pe strada x in intersectie sau -masina(x) daca strada x nu are masina.

directie(x, y), masina x doreste sa mearga pe strada y.

Ca scop (goal) se va testa pass(x, y). Pentru acest fapt, datele de intrare trebuie specificate integral. Este necesara specificarea tuturor predicatelor datelor de intrare. Daca masina x poate traversa intersectia catre strada y, atunci Proover9 va afisa o fereastra cu demonstratia, altfel va afisa o fereastra de eroare. Pentru rezolvarea problemei, am folosit urmatoarele predicate:

differentFrom(x, y): se refera la faptul ca strada x e diferita de strada y. Au fost scrise constante pentru toate combinatiile de cate 2 strazi diferite. Este un predicat simetric.

rightPriority(x, y): se refera la faptul ca x si y sunt doua strazi sunt alaturate, si strada x are prioritate de dreapta fata de strada y. Se aplica doar in cazul in care cele doua strazi au aceeasi prioritate dupa celelalte criterii.

semaphore(x): se refera la faptul ca pe strada x exista un semafor. Acest predicat implica culoarea semaforului care poate fi verde sau galben sau rosu, dar doar o singura culoare pe rulare. Pentru modelarea acestui fapt au fost scrise relatii intre predicatele green(x), yellow(x) si red(x).

indicator(x): se refera la faptul ca pe strada x exista un indicator. Acesta poate sa fie unul de piroritate(x), de stop(x), sau none(x) (lipseste indicatorul). Modelarea predicatului none(x) ca un indicator a fost pentru

a generaliza problema si a putea schimba ulterior mai usor datele de intrare. La fel ca la semaphore(x), indicator(x) implica doar un singur indicator pe rulare.

samePriority(x, y): se refera la faptul ca doua strazi au aceeasi prioritate de indicator sau semafor. Acest predicat este simetric si daca este adevarat pentru 2 strazi, atunci se aplica prioritatea de dreapta intre cele 2 strazi.

priority(x, y): se refera la faptul ca strada x are prioritate fata de strada y. Aici au fost folosite toate predicatele anterioare si regulile de circulatie pentru a determina prioritatile dintre strazi. Au fost luate in calcul toate variantele posibile, dar in cazul in care doua strazi au aceeasi prioritate si nu sunt alaturate(nu se poate aplica prioritate de dreapta), acest predicat nu poate determina care dintre ele are prioritate. Drept urmare va rezulta ca niciunul nu are prioritate fata de celalalt.

intra2(x): se refera la posibilitatea masinii (care vine de pe strada x) de a intra in intersectie. Tine cont daca are pieton pe trecere si daca are prioritate fata de toate celelalte masini din intersectie.

pass(x, y): se refera la posibilitatea masinii (care vine de pe strada x si vrea sa mearga pe strada y) de a traversa intersectia. Acest predicat tine cont de predicatul intra2(x), de directia pe care masina doreste sa mearga, daca este politist sau nu in intersectie, si daca este pieton pe strada y. Verificarea pietonului de pe strada y este facuta pentru a nu bloca intersectia, deoarece daca poate intra in intersectie nu inseamna si ca poate iesi din intersectie. Acesta este predicatul scop, care va fi demonstrat ca goal pentru testarea proiectului.

Datele de intrare pentru modelarea problemei sunt:

politist | -politist. Nu se poate sa fie si politist si -politist.

semaphore(x), urmat de culoarea unica semafolului de pe strada x (green(x) | yellow(x) | red(x)). Daca se specifica -semaphore(x) atunci nu trebuie specificata culoare, deoarece nu este semafor.

indicator(x), urmat de semnificatia unica indicatorului de pe strada x (prioritate(x) | stop(x) | none(x)). Trebuie specificat acest atribut pentru toate strazile, indiferent daca au sau nu indicator(pe strazile care nu au indicator, se va specifica indicator(x) si none(x)).

pieton(x), daca este pieton pe trecerea de pe strada x sau -pieton(x) daca nu este pieton.

masina(x), daca este masina pe strada x in intersectie sau -masina(x) daca strada x nu are masina.

directie(x, y), masina x doreste sa mearga pe strada y.

Ca scop (goal) se va testa pass(x, y). Pentru acest fapt, datele de intrare trebuie specificate integral. Este necesara specificarea tuturor predicatelor datelor de intrare. Daca masina x poate traversa intersectia catre strada y, atunci Proover9 va afisa o fereastra cu demonstratia, altfel va afisa o fereastra de eroare. .

Code:

```
% Saved by Prover9-Mace4 Version 0.5, December 2007.

set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4).   % Options for Mace4
  assign(max_seconds, 60).
end_if.

formulas(assumptions).

%differentFrom(x,y)
%streets: a, b ,c ,d, e
differentFrom(x, y)->differentFrom(y, x).
differentFrom(a, b).
differentFrom(a, c).
differentFrom(a, d).
```

```
differentFrom(a, e).
differentFrom(b, c).
differentFrom(b, d).
differentFrom(b, e).
differentFrom(c, d).
differentFrom(c, e).
differentFrom(d, e).

%rightPriority(x, y): prioritate de dreapta.
rightPriority(x, y) -> -rightPriority(y, x).
rightPriority(b, a).
rightPriority(c, b).
rightPriority(d, c).
rightPriority(e, d).
rightPriority(a, e).
-rightPriority(a, c).
-rightPriority(a, d).
-rightPriority(b, d).
-rightPriority(b, e).
-rightPriority(c, e).
-rightPriority(c, a).
-rightPriority(d, a).
-rightPriority(d, b).
-rightPriority(e, b).
-rightPriority(e, c).

%semaphore(x): daca este semafor pe strada.
semaphore(x)-> green(x) | yellow(x) | red(x).
green(x)-> -yellow(x) & -red(x).
yellow(x)-> -green(x) & -red(x).
red(x)-> -green(x) & -yellow(x).

%indicator(x): daca este indicator pe strada
indicator(x)-> prioritate(x) | stop(x) | none(x).
prioritate(x)-> -stop(x) & -none(x).
stop(x)-> -prioritate(x) & -none(x).
none(x)-> -prioritate(x) & -stop(x).

%samePriority(x, y): au aceaasi prioritate -> prioritate de dreapta.
samePriority(x, y) -> samePriority(y, x).
prioritate(x) & prioritate(y) & differentFrom(x, y)-> samePriority(x, y).
stop(x) & stop(y) & differentFrom(x, y)-> samePriority(x, y).
none(x) & none(y) & differentFrom(x, y)-> samePriority(x, y).

%priority(a, b): a are prioritate fata de b
priority(x, y) -> -priority(y, x).
semaphore(x) & red(x) & differentFrom(x, y) -> priority(y, x).
semaphore(x) & green(x) & differentFrom(x, y)-> priority(x, y).

-semaphore(x) & -semaphore(y) & samePriority(x, y) & rightPriority(x, y)-> priority(x, y).
-semaphore(x) & -semaphore(y) & prioritate(x) & (stop(y) | none(y)) & differentFrom(x, y)-> priority(x,
-semaphore(x) & -semaphore(y) & prioritate(y) & (stop(x) | none(x)) & differentFrom(x, y)-> priority(y,
-semaphore(x) & -semaphore(y) & none(x) & stop(y) & differentFrom(x, y)-> priority(x, y).
-semaphore(x) & -semaphore(y) & none(y) & stop(x) & differentFrom(x, y)-> priority(y, x).
```

```
75
76  %intra(x): are prioritate fata de toti, poate sa intre
77
78  masina(x) & -pieton(x) & (masina(y) & priority(x, y) | -masina(y)) -> intra(x).
79
80  masina(a) & -pieton(a)
81  & (masina(b) & priority(a, b) | -masina(b))
82  & (masina(c) & priority(a, c) | -masina(c))
83  & (masina(d) & priority(a, d) | -masina(d))
84  & (masina(e) & priority(a, e) | -masina(e))->intra2(a).
85
86  masina(b) & -pieton(b)
87  & (masina(a) & priority(b, a) | -masina(a))
88  & (masina(c) & priority(b, c) | -masina(c))
89  & (masina(d) & priority(b, d) | -masina(d))
90  & (masina(e) & priority(b, e) | -masina(e))->intra2(b).
91
92  masina(c) & -pieton(c)
93  & (masina(b) & priority(c, b) | -masina(b))
94  & (masina(a) & priority(c, a) | -masina(a))
95  & (masina(d) & priority(c, d) | -masina(d))
96  & (masina(e) & priority(c, e) | -masina(e))->intra2(c).
97
98  masina(d) & -pieton(d)
99  & (masina(b) & priority(d, b) | -masina(b))
100 & (masina(c) & priority(d, c) | -masina(c))
101 & (masina(d) & priority(d, a) | -masina(a))
102 & (masina(e) & priority(d, e) | -masina(e))->intra2(d).
103
104 masina(e) & -pieton(e)
105 & (masina(b) & priority(e, b) | -masina(b))
106 & (masina(c) & priority(e, c) | -masina(c))
107 & (masina(d) & priority(e, d) | -masina(d))
108 & (masina(a) & priority(e, a) | -masina(a))->intra2(e).
109
110 %pass(a, b): daca masina poate traversa intersectia din a in b.
111 %pass(a, b) <- a are prioritate fata de toate restul masinilor + nu e pieton pe a si nici pe b + nu e p
112
113 intra2(x) & directie(x, y) & -pieton(y) & -politist -> pass(x, y).
114
115 %masina(b) & -pieton(b) & (priority(b, c) & masina(c) | -masina(c)).
116 %practic intra(x) nu verifica daca are prioritate fata de toate masinile din intersectie.
117 %daca e masina pe x si are prioritate fata de toate celelalte strazi care au masini- > intra(x) (+ piet
118 %intra <- masina(x) & prioritate(x, y) oricare ar fi y cu prop. masina(y).
119
120 %%interssectia:
121 -politist.
122
123 %semaphore(x): este semafor pe strada x.
124 semaphore(a).
125 green(a).
126 -semaphore(b).
127 -semaphore(c).
128 -semaphore(d).
```

```
-semaphore(e).

%indicator(x): este indicator pe strada x (poate fi niciunul (none)).
indicator(a).
prioritate(a).
indicator(b).
stop(b).
indicator(c).
none(c).
indicator(d).
none(d).
indicator(e).
none(e).

%pieton(x): este pieton pe trecere => masina nu poate trece.
-pieton(a).
-pieton(b).
-pieton(c).
-pieton(d).
-pieton(e).

%masina(x): este o masina pe strada x.
masina(a).
masina(b).
masina(c).
-masina(d).
-masina(e).

%directie(x, y): masina pe strada x care vrea sa mearga pe strada y.
directie(a, b).
directie(b, c).
directie(c, e).

end_of_list.

formulas(goals).

pass(a, b).

end_of_list.
```