Sistem za upravljanje saobraćajem

Članovi tima

- Vladislav Radović SV 27/2021
- Nikola Mitrović SV 18/2021

Motivacija

Saobraćaj u modernim gradovima je postao složen i teško upravljiv zbog velikog broja vozila, gužvi, nesreća i nepredviđenih situacija. Trenutni sistemi za upravljanje saobraćajem uglavnom ne koriste dovoljno podatka kako bi regulisali saobraćaj optimalno.

Cilj projekta je razvoj inteligentnog sistema zasnovanog na znanju, koji automatski analizira podatke o saobraćaju, detektuje kritične situacije i predlaže optimalne odluke za regulaciju prometa.

Pregled problema

Gradovi se suočavaju sa problemima poput gužvi, nesreća i nepredviđenih situacija, dok postojeći sistemi za regulaciju saobraćaja uglavnom ne koriste dovoljno dostupnih podataka i nisu prilagodljivi promenama u realnom vremenu.

Neophodno je razviti sistem koji prikuplja i analizira podatke iz senzora, kamera i GPS uređaja, otkriva kritične situacije i donosi optimalne odluke. Zbog velike količine podataka, sistem mora omogućiti obradu u realnom vremenu i pregled trendova kroz vreme.

Postojeća rešenja su često ograničena i teško se integrišu sa drugim sistemima, pa je cilj razviti proširiv i konfigurabilan sistem zasnovan na znanju, koji povećava automatizaciju, sigurnost i efikasnost saobraćaja.

Metodologija rada

Ulaz u sistem:

Ulazi uključuju:

- `TrafficDensity(crossroad, value)`
- `Vehicle(crossroad, speed, type)`
- `Accident(crossroad, severity)`
- `Weather(type, intensity)`
- `EmergencyVehicle(location, direction)`
- `IllegalParking(crossroad)`
- `EventDay(eventType, expectedTraffic)`
- `PedestrianDetected(crossroad)`
- `PublicTransportDelay(line, minutes)`
- `Crossroad(id, connections)`

Svaki od ovih ulaza utiče na trenutnu procenu stanja saobraćaja, a odluke donete u jednom trenutku služe kao ulaz za narednu optimizaciju semaforskih režima i rute vozila.

Baza znanja:

Baza znanja je strukturisana po kontekstima i pravila su grupisana prema vremenskim periodima i tipovima događaja.

Primer konteksta i pravila:

- **Rush hour – produži zeleno**

rule "Rush hour - extend green"

when

TrafficDensity(crossroad == $c, value > 70) // gužva veća od 70%

TimeOfDay(hour >= 7 && hour <= 9) // jutarnji špic

```
then
   insert(new TrafficAction($c, "extendGreen", 30)); // produži zeleno za 30 sekundi
end
```

- **Detekcija nesreće – blokada raskrsnice**

```
rule "Accident detected - block crossroad"
when
   Accident(crossroad == $c, severity == "high")
then
   insert(new TrafficAction($c, "blockCrossroad"));
   insert(new Notification("Nesreća na " + $c + " - raskrsnica blokirana"));
end
```

- **Prioritet za hitna vozila**

```
rule "Emergency vehicle priority"
when
   EmergencyVehicle(location == $c, direction == $d)
then
   insert(new TrafficAction($c, "allowPass", "priority"));
end
```

- **Loše vreme – produži zeleno**

```
rule "Bad weather extend green"

when

  Weather(type == "rain" || type == "snow", intensity == "medium" || intensity == "high")

then

  // Produži zeleno za sve raskrsnice zbog lošeg vremena

  insert(new TrafficAction("All", "extendGreen", 15));

  insert(new Notification("Loše vreme - produženje zelenog svetla na svim raskrsnicama"));

end
```

- **Nepropisno parkiranje – obaveštenje**

```
rule "Illegal parking notification"

when

  IllegalParking(crossroad == $c)

then

  insert(new Notification("Nepropisno parkiranje detektovano na " + $c));

end
```

- **Pešaci na prelazu – crveno za vozila**

```
rule "Pedestrian crossing - red light"

when

  PedestrianDetected(crossroad == $c)

then

  insert(new TrafficAction($c, "setRedForVehicles"));
```

```
end
```

- **Zauzetost semafora – balansiranje traka**

```
rule "Semaphore congestion - balance lanes"
when
  TrafficDensity(crossroad == $c, value > 80)
then
  insert(new TrafficAction($c, "balanceLanes"));
  insert(new Notification("Zauzetost semafora visoka na " + $c + " - balansiranje traka"));
end
```

- **Kašnjenje javnog prevoza – prioritet autobusu**

```
rule "Bus delay - priority"
when
  PublicTransportDelay(line == $line, minutes > 10)
then
  insert(new TrafficAction("All", "prioritizeBus", $line));
  insert(new Notification("Linija " + $line + " kasni - prioritet na semaforima"));
end
```

- **Masovni događaj – preusmeravanje rute**

```
rule "Mass event - reroute traffic"
```

when

EventDay(expectedTraffic > 1000, eventType == "concert" || eventType == "sports")

then

insert(new TrafficAction("All", "rerouteTraffic"));

insert(new Notification("Masovni događaj danas - preusmeravanje saobraćaja"));

end

- **Noćni režim – skraćivanje zelenog svetla**

rule "Night mode - shorten green"

when

TimeOfDay(hour >= 23 || hour <= 5)

then

insert(new TrafficAction("All", "shortenGreen", 15)); // smanji zeleno za 15 sekundi

End

- Sva pravila u sistemu koriste **forward chaining**, gde se factovi i akcije propagiraju kroz lanac pravila.
- **Nivoi ulančavanja**:
  - **Prvi nivo** – detekcija osnovnog događaja (npr. gužva, nesreća, kašnjenje autobusa).
  - **Drugi nivo** – aktivacija pravila koja kombinuju više ulaza i analiziraju uzroke (npr. blokada raskrsnice, loše vreme, masovni događaji).
  - **Treći nivo** – generisanje optimizovanih odluka i akcija (produžavanje zelenog svetla, preusmeravanje saobraćaja, prioritet hitnim vozilima i javnom prevozu).

Primer ulančavanja pravila:

1. `TrafficDensity > 80%` → insert `HeavyTraffic` fact
2. `HeavyTraffic + Accident` → insert `CriticalCongestion` fact
3. `CriticalCongestion` → insert `TrafficAction(extendGreen)` i `Notification`

`Akumulaciona funkcija:`

rule "Accident frequency in 15 min"

when

accumulate($a : Accident() over window:time(15m), $count : count($a))

then

if ($count > 3) insert(new Notification("Previše nesreća u poslednjih 15 min"));

end

Slično se može koristiti za: prosečnu brzinu vozila, broj pešaka, maksimalnu zauzetost raskrsnica.

Baza znanja se popunjava kombinacijom:

- istorijskih obrazaca gužvi;
- pravila eksperata (saobraćajnih inženjera);
- real-time senzorskih podataka;
- automatski generisanih pravila prema trendovima saobraćaja.

Backward chaining:

Ako postoji sumnja na **kritičnu gužvu** ili incident, sistem koristi istorijske podatke i trenutne senzorske ulaze da postavi dijagnozu problema:

- **Rekurzivni query – upstreamCause** (glavni za uzrok gužve)

query upstreamCause(String crossroad, String cause)

// Ako postoji nesreća na raskrsnici, to je uzrok

Accident(crossroad == crossroad, $c := "accident")

```
    ; return(crossroad, $c)

or
    // Ako je tu hitno vozilo, to je uzrok
    EmergencyVehicle(location == crossroad, $c := "emergencyVehicle")
    ; return(crossroad, $c)

or
    // Ako je gužva ovde, proveri da li dolazi sa povezane raskrsnice
    TrafficDensity(crossroad == crossroad, value > 80)
    Crossroad(id == crossroad, $next : connections)
    // rekursivni poziv za susednu raskrsnicu
    upstreamCause($next, $c)
    ; return($next, $c)

or
    // Loše vreme kao uzrok
    Weather(intensity == "high", $c := "badWeather")
    ; return(crossroad, $c)

or
    // Masovni događaj kao uzrok
```

```
    EventDay(expectedTraffic > 1000, $c := "massEvent")

    ; return(crossroad, $c)

end
```

Sistem ne reaguje samo na trenutne događaje, već koristi **backward chaining** i **rekurzivne querije** da postavi dijagnozu problema i identifikuje primarni uzrok kritične gužve ili incidenta. Kada se detektuje problem, kao što je gužva, kašnjenje autobusa ili hitno vozilo na raskrsnici, sistem pokreće query `upstreamCause`, koji istražuje sve moguće uzroke: nesreće, gužvu, loše vreme, masovne događaje ili blokade od strane vozila.

Ovaj query se rekurzivno poziva za sve povezane raskrsnice i ulaze dok ne pronađe **glavni uzrok**. Na taj način se formira **stablo dijagnoze**, gde svaki čvor predstavlja potencijalni uzrok, a grane povezuju posledice sa stvarnim događajima u mreži saobraćajnica.

Primer toka:

1. Sistem detektuje kašnjenje autobusa (fact `PublicTransportDelay`).
2. Upit `whyBusDelay` aktivira `upstreamCause` za raskrsnicu gde je problem primećen.
3. Query proverava svaki potencijalni uzrok: gužvu, nesreću, hitno vozilo, loše vreme ili masovni događaj.
4. Ako neki uzrok vodi ka drugoj raskrsnici, query se rekurzivno aktivira i proširuje stablo dijagnoze.
5. Kada se primarni uzrok identifikuje, insertuju se **TrafficAction** i **Notification** objekti koji optimizuju saobraćaj: produžavanje zelenog svetla, preusmeravanje rute, prioritet hitnim vozilima ili javnom prevozu.

- ***Zašto gužva?***

```
query whyTrafficJam($crossroad: String, $cause: String)
    upstreamCause($crossroad, $cause)
end
```

- ***Zašto hitna vozila kasne?***

- ```
  query whyEmergencyDelayed($crossroad: String)
      EmergencyVehicle(location == $crossroad, direction != "free")
      upstreamCause($crossroad, $cause)
  end
  ```

- ***Zašto autobus kasni?***

- ```
  query whyBusDelay($line: String, $crossroad: String, $cause:
  String)
      PublicTransportDelay(line == $line, minutes > 10)
      upstreamCause($crossroad, $cause)
  end
  ```

- ***Da li je gužva posledica lošeg vremena?***

- ```
  query isWeatherCause($crossroad: String)
      TrafficDensity(crossroad == $crossroad, value > 80)
      Weather(intensity == "high")
  end
  ```

- ***Da li je gužva posledica masovnog događaja?***

- ```
  query isEventCause($crossroad: String)
      TrafficDensity(crossroad == $crossroad, value > 80)
      EventDay(expectedTraffic > 1000)
  end
  ```

- ***Ko blokira raskrsnicu? (vozilo ili nesreća)***

- ```
  query whoBlocksCrossroad($crossroad: String, $cause: String)
      upstreamCause($crossroad, $cause)
  end
  ```
  ```

- ***Koja raskrsnica je izvor problema?***

- ```
  query sourceCrossroad($cause: String, $crossroad: String)
      upstreamCause($crossroad, $cause)
  end
  ```

- ***Da li postoji alternativna ruta slobodna?***

- ```
  query isAlternativeRouteFree($crossroad: String, $alternative:
  String)
      Crossroad(id == $crossroad, connections contains
  $alternative)
      TrafficDensity(crossroad == $alternative, value < 50)
  end
  ```

- ***Da li je pešački prelaz uzrok usporenja?***

- ```
  query isPedestrianCause($crossroad: String)
      PedestrianDetected(crossroad == $crossroad)
      TrafficDensity(crossroad == $crossroad, value > 70)
  end
  ```

-
- **Da li postoji alternativna ruta slobodna?**
- **Da li je pešački prelaz uzrok usporenja?**

Complex Event Processing (CEP):

- **Prosečna brzina u 5 minuta**
- **Najduži period gužve (30 min)**
- **Broj nesreća u 15 minuta**
- **Broj pešaka u 10 minuta**
- **Detekcija naglog rasta gužve**
- **Prosečno kašnjenje autobusa u 1h**
  ```

- **Maksimalna zauzetost raskrsnice u 20 min**
- **Broj zaustavljenih vozila u 10 min**
- **Ukupan broj obaveštenja vozačima u 1h**
- **Trend pogoršanja vremena u 30 min**

Ako se u roku od 10 minuta pojavi nagli rast gužve na 3 raskrsnice koje se nalaze u blizini lokacije događaja → aktiviraj rerutiranje saobraćaja.

```
rule "Mass event congestion spread"
when
  EventDay(expectedTraffic > 1000)
  accumulate(
    TrafficDensity($c : crossroad, value > 80) over window:time(10m),
    $count : countDistinct($c)
  )
  $count >= 3
then
  insert(new Notification("Masovni događaj utiče na više raskrsnica – potrebno preusmeravanje"));
end
```

Ako autobus kasni više od 15 minuta, a u poslednjih 10 minuta postoji velika gužva **i/ili** nesreća na njegovoj ruti, i dodatno loše vreme → razlog kašnjenja je kritičan.

```
rule "Critical bus delay cause"
when
  PublicTransportDelay(line == $line, minutes > 15)
  ( TrafficDensity(value > 80) or Accident(severity == "high") )
  Weather(intensity == "high")
then
  insert(new Notification("Autobus " + $line + " kasni zbog kombinacije gužve/nesreće i lošeg vremena"));
end
```

Izveštaji i monitoring:

Sistem podržava tri vrste izveštaja:

1. **Normal izveštaj** – trenutna statistika gužvi, brzine i broja nesreća po raskrsnici i vremenu.

Raskrsnica	Gustina saobraćaja (%)	Prosečna brzina (km/h)	Broj nesreća (u poslednjih 15 min)	Broj pešaka (10 min)	Status
Crossroad 1	85	18	2	35	GUŽVA
Crossroad 2	40	45	0	12	NORMALNO
Crossroad 3	95	10	1	50	KRITIČNO

```
public class NormalReport {

    private long generatedAt;

    private String crossroad;

    private double avgSpeed5min;

    private int trafficDensity;

    private int accidents15min;

    private int pedestrians10min;


    public NormalReport(String crossroad, double avgSpeed5min, int trafficDensity, int accidents15min, int pedestrians10min) {
```

```
        this.generatedAt = System.currentTimeMillis();
        this.crossroad = crossroad;
        this.avgSpeed5min = avgSpeed5min;
        this.trafficDensity = trafficDensity;
        this.accidents15min = accidents15min;
        this.pedestrians10min = pedestrians10min;
    }
}
```

```
rule "Generate NormalReport"
when
    $td : TrafficDensity($c : crossroad, $density : value)
    accumulate( $v : Vehicle(crossroad == $c) over window:time(5m), $avgSpeed :
average($v.speed) )
    accumulate( $a : Accident(crossroad == $c) over window:time(15m), $accCount :
count($a) )
    accumulate( $p : PedestrianDetected(crossroad == $c) over window:time(10m),
$pedCount : count($p) )
then
    NormalReport r = new NormalReport($c, $avgSpeed, $density, (int)$accCount,
(int)$pedCount);
    insert(r);
end
```

2. **Trend izveštaj (AtSomeTime)** – analiza saobraćaja u određenim periodima, npr. jutarnja gužva u poslednjih mesec dana.

Datum	Period (07:00–09:00)	Prosečna gustina (%)	Prosečna brzina (km/h)	Broj nesreća
01.08.2025	73	22	1	
05.08.2025	82	18	2	
10.08.2025	65	28	0	
...	...	...	...	...

```
declare DailyTrafficSummary

   crossroad : String

   date : java.util.Date

   period : String

   avgDensity : double

   avgSpeed : double

   accidents : int

end



rule "Generate TrendReport"

when

   $req : ReportRequest(type == "Trend", $cr : crossroadId, $from : fromDate, $to : toDate, $period : period)
```

```
  accumulate(
    $d : DailyTrafficSummary(crossroad == $cr, date >= $from, date <= $to, period == $period),
    $avgD : average($d.avgDensity),
    $avgS : average($d.avgSpeed),
    $sumAcc : sum($d.accidents)
  )
then
  TrendReport tr = new TrendReport($cr, $period, $from, $to, $avgD, $avgS, (int)$sumAcc);
  insert(tr);
end
```

3. **MaxPeriod izveštaj** – pronalaženje najdužih perioda kritične gužve ili nesreća za planiranje saobraćajnih intervencija.

Raskrsnica	Period gužve	Trajanje (min)	Prosečna gustina (%)	Broj nesreća
Crossroad5	08:10 – 09:05	55	92	1
Crossroad2	16:20 – 17:15	55	88	3
Crossroad1	14:00 – 14:40	40	85	0

```
rule "Start congestion"
when
  TrafficDensity(crossroad == $c, value > 80)
```

```
    not( CongestionActive(crossroad == $c) )
then
    insert(new CongestionActive($c, System.currentTimeMillis()));
end
--------------------------------------------------------------------------------
rule "End congestion"
when
    $ca : CongestionActive($c : crossroad, $start : startTime)
    TrafficDensity(crossroad == $c, value < 70)
then
    long end = System.currentTimeMillis();
    insert(new CongestionInterval($c, $start, end));
    delete($ca);
end
------------------------------------------------------------------------------------------------------
rule "Generate MaxPeriodReport"
when
    $req : ReportRequest(type == "MaxPeriod", $cr : crossroadId)
    accumulate(
        $ci : CongestionInterval(crossroad == $cr),
        $maxDur : max($ci.duration)
    )
```

```
then
   MaxPeriodReport rpt = new MaxPeriodReport($cr, $maxDur);
   insert(rpt);
end
```

Template-i:

Template (`rush_hour_rule.drt`):

```
rule "Rush hour - @crossroad"
when
    TrafficDensity(crossroad == "@crossroad", value > @threshold)
then
    insert(new TrafficAction("@crossroad", "extendGreen",
@extendSeconds));
end
```

Primer CSV fajla koji generiše pravila:

crossroad	threshold	extendSeconds
Crossroad 1	70	30
Crossroad 2	75	25
Crossroad 3	80	20

Primer template-a za fact

Ako imamo puno raskrsnica sa različitim gustinama saobraćaja:

Fact template (`traffic_density_template.drt`):

```
declare TrafficDensity
    crossroad: String
    value: int
end
```

CSV primer:

crossroad	value
Crossroad 1	85
Crossroad 2	60
Crossroad 3	90

Klasni dijagram: