

# Problem sinteze mrežne konfiguracije

Seminarski rad u okviru kursa  
Verifikacija softvera  
Matematički fakultet

Marija Mijailović  
mijailovicmarija@hotmail.com

maj 2018.

## Sažetak

U ovom radu je predstavljena osnovna ideja rada pod nazivom „Sinteza mrežne konfiguracije” (eng. *Network-Wide Configuration Synthesis*) izloženog na konferenciji CAV 2017. godine. [1]

## Sadržaj

<b>1</b>	<b>Uvod</b>	<b>2</b>
1.1	Motivacija . . . . .	2
<b>2</b>	<b>Sinteza ulaza za Datalog</b>	<b>4</b>
2.1	Sinteza ulaza za polu-pozitivan Datalog . . . . .	4
2.2	Iterativna sinteza ulaza za Datalog . . . . .	5
<b>3</b>	<b>Implementacija i eksperimenti</b>	<b>5</b>
3.1	Eksperimenti . . . . .	5
<b>4</b>	<b>Zaključak</b>	<b>6</b>
	<b>Literatura</b>	<b>6</b>

# 1 Uvod

Računarskim mrežama se teško upravlja, u slučaju velikog broja zahteva mrežni operateri moraju da shvate svaku pojedinačnu konfiguraciju od stotine uređaja koji pokreću složene protokole. Samim tim nije iznenađujuće što operateri često prave greške koje dovode do pada mreže. Da bi se rešio ovaj problem, u ovom radu je predstavljen novi pristup koji automatski izračunava mrežnu konfiguraciju koja je u skladu sa zahtevima operatera. Ključni zadatak  *smanjenje problema pronalaženje tačnih ulaznih konfiguracija*, se svodi na zadatak  *koji sintetizuje ulazne podatke za Datalog program*. U ovom radu opisan je algoritam koji vrši sintezu ulaznih podataka za Datalog programe. Ovaj algoritam se primenjuje izvan domena mreže, ali je podržan u okviru prvog sistema sinteze konfiguracije na mreži, nazvan *SyNET*, koji podržava višestruke protokole rutiranja (OSPF i BGP) i statičke putanje.

## 1.1 Motivacija

**Problem mrežne konfiguracije** možemo predstaviti na sledeći način: Ako nam je data specifikacija mreže  $N$ , koja definiše ponašanje svih protokola rutiranja (eng. *Routing protocol*), i  $R$  skup zahteva na mreži za sledeće stanje, treba otkriti konfiguraciju  $C$ , tako da ruteri konvergiraju ka kompatibilnom sledećem stanju sa  $R$ . To jest, operater jednostavno pruža zahteve visokog nivoa  $R$  za sledeće stanje, i konfiguracija  $C$  se dobija automatski. Međutim, dolaženje do rešenja za problem sinteze konfiguracije u mreži je izazov iz tri razloga:

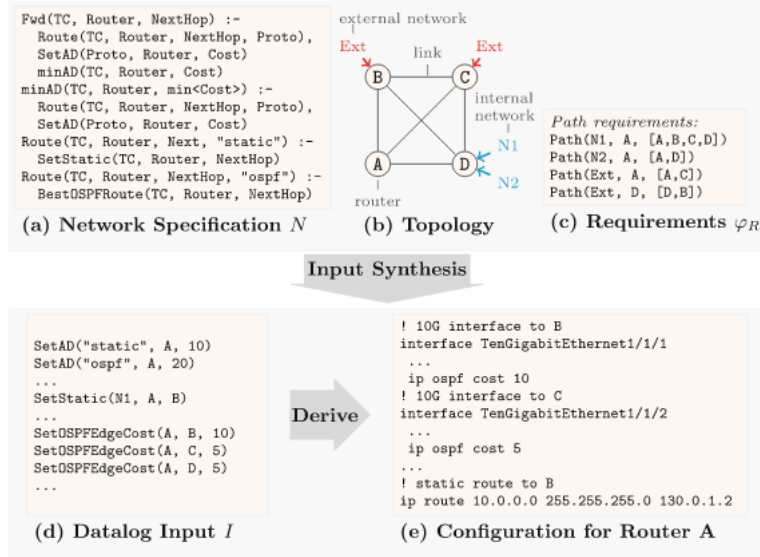
**Raznolikost** - protokoli se razlikuju među sobom. Na primer, protokol Otvoriti najkraći put prvo (eng. *Open Shortest Path First (OSPF)*) direktno upravlja saobraćajem duž najkraće putanje, dok Granični prolaz protokol (eng. *Border Gateway Protocol (BGP)*) direktno upravlja saobraćajem duž ne-najkraće putanje. Takođe, BGP ne može proslediti saobraćaj duž višestrukih putanja, dok OSPF podržava rutiranje višestrukih putanja i time je pogodniji za balansiranje opterećenja mreže.

**Zavisnost** - različiti protokoli često zavise jedan od drugog, kako bi se osiguralo da oni zajedno izračunavaju sledeće stanje. Na primer, BGP zavisi od konfiguracije unutar domena mreže.

**Izvodljivost** - prostor za pretragu konfiguracija je ogroman i stoga je teško da se pronade ona konfiguracija koja daje sledeće stanje koje zadovoljava zahteve.

Na Slici 1(b) prikazana je jednostavna mrežna topologija, sastavljena od 4 rutera označenih sa A, B, C i D. Ruteri B i C su povezani sa spoljašnjom mrežom Ext, a ruter D je povezan sa dve unutrašnje mreže N1 i N2. U nastavku, pojam klase saobraćaja se odnosi na skup paketa (npr. paketi namenjeni N1).

**Računanje sledećeg stanja** - svaki ruter pokreće i OSPF i BGP protokole, a može biti konfigurisan i statičkim putanjama. Računanje OSPF-a se svodi na pronalaženju putanje sa najmanjom cenom do unutrašnje destinacije (N1 ili N2), kao i do svih rutera u mreži. Računanje BGP-a se svodi na pronalaženje putanje za prelazak na spoljašnje destinacije (Ext). Kada BGP i OSPF završe računanje njihovih sledećih unosa, svaki ruter uzima te unose. Unija svih prosleđenih unosa označava se kao sledeće stanje mreže.



Slika 1: Sinteza konfiguracije na mreži. Ulaz je: (a) mrežna specifikacija  $N$  u Datalogu (b) topologija mreže, i (c) uslovi za rutiranje  $\phi_R$ . Izlaz je: (d) Datalog ulaz  $I$  koji rezultira u sledećem stanju koje zadovoljava zahteve. Konfiguracije (e) su izvedene iz  $I$ .

**Specifikacija mreže** - Na slici 1(a) možemo videti nekoliko Datalog pravila. Prvo pravilo  $Fwd(TC, Router, NextHop)$  predstavlja sledeće stanje ako Router prosledi pakete za TC klasu preko rutera  $NextHop$ . Predikat  $Route(TC, Router, NextHop, Proto)$  prikuplja sledeće ulaze definisane preko OSPF-a i statičkih ruta. Sledeće pravilo  $minAD(TC, Router, min < Cost >)$  računa minimalne administrativne troškove po svim protokolima. Takođe se uslovi rutiranja mogu se i direktno izraziti u odnosu na predikat  $Fwd$ , kao što je prikazano na slici 1(c).

**Uslovi rutiranja  $\phi_R$**  - Posmatrajmo četiri različita uslova data na slici 1(c). Prva dva navode da A mora prosledi pakete za klase N1 i N2 i to duž putanje  $A \rightarrow B \rightarrow C \rightarrow D$  i  $A \rightarrow D$ , respektivno.

- generiranjem statičke putanje - na osnovu sledećeg unosa na A paketi se šalju za N1 do B
- određivanjem težina puta tako da putanje imaju najmanju OSPF cenu
- ruteru A dodelimo viši prioritet za sledeće unose na osnovu statičkih putanja nego sledeći unos OSPF-a. Pošto je statička putanja za sledeći ulaz generisana samo za odredište N1, a ne i N2, to znači da će unos za N1 proslediti saobraćaj na ruter B dok će unos za N2 biti OSPF generisan.

Poslednja dva zahteva navode da A i D moraju proslediti pakete za klase saobraćaja Ext i to duž putanje  $A \rightarrow C$  i  $D \rightarrow B$ , respektivno. Ovo je moguće zadovoljiti:

- postavljanjem identičnih BGP rutera na lokalne konfiguracije B i C

- određivanjem težine puta tako da putanje imaju najmanju OSPF cenu, tako da BGP koristiti ove rezultate da bi izračunao svoje sledeće unose na Ext.

Na slici 1(e) je prikazano lokalno podešavanje rutera A.

Konačno, problem sinteze mrežne konfiguracije formulišemo kao:

**Definicija 1.1** *Problem sinteze mrežne konfiguracije:*

**Ulaz** Deklarativna mrežna specifikacija  $N$  i uslovi rutiranja  $\phi_R$ .

**Izlaz** Datalog ulaz  $I$  takav da  $[[N]]I \models \phi_R$ , ako takav ulaz postoji; inače, vrati UNSAT.

Gde je  $[[N]]I$  fiksna tačka Datalog programa  $N$  za ulaz  $I$ . Međutim ovaj problem je neodlučiv i da bi se rešio predstavljen je novi iterativni algoritam.

## 2 Sinteza ulaza za Datalog

U ovom delu opisan je iterativni algoritam sinteze ulaza koji deli Datalog program  $P$  u slojeve  $P_1, \dots, P_n$ , i pronalazi ulaz  $I_i$  za svaki sloj  $P_i$  i zatim konstruiše ulaz  $I$  za Datalog program  $P$ . Svaki sloj  $P_i$  je polu-pozitivan Datalog program. U nastavku, prvo predstavljamo algoritam  $S_{SemiPos}$  koji se koristi da se izvrši sinteza ulaza za jedan sloj (što je polu-pozitivan program). Zatim predstavljamo opšti algoritam, nazvan  $S_{Strat}$ , koji iterativno primenjuje  $S_{SemiPos}$  za svaki sloj koji sintetise ulaze za Datalog programe.

### 2.1 Sinteza ulaza za polu-pozitivan Datalog

Ključna ideja je da se problem sinteze ulaza svede na zadovoljavanje SMT ograničenja: Dat je polu-pozitivan Datalog program  $P$  i ograničenje  $\phi$ , kodiramo  $\exists I. [[P]]I \models \phi$  u SMT ograničenju  $\psi$ . Ako je  $\psi$  zadovoljavajuće, onda iz modela  $\psi$  možemo izvući ulaz  $I$  takav da  $[[P]]I \models \phi$ .

---

#### Algorithm 1. Algorithm $S_{SemiPos}$ for semi-positive Datalog

---

**Input:** Semi-positive Datalog program  $P$  and a constraint  $\varphi$

**Output:** An input  $I$  such that  $[[P]]I \models \varphi$  or  $\perp$

---

```

1 begin
2    $\varphi' \leftarrow \text{SIMPLIFY}(\varphi)$ 
3   for  $k \in [1..bound_k]$  do
4      $\varphi_k \leftarrow \text{REWRITE}(\varphi', k)$ 
5      $\psi \leftarrow [P]_k \wedge \varphi_k$ 
6     if  $\exists J. J \models \psi$  then
7        $I \leftarrow \{p(\vec{t}) \in J \mid p \in \text{edb}(P)\}$ , where  $J \models \psi$ 
8       return  $I$ 
9   return  $\perp$ 

```

---

Slika 2: Algoritam  $S_{SemiPos}$

Algoritam  $S_{SemiPos}(P, \phi)$ , dat je na slici 2, prvo se poziva funkcija  $\text{Simplify}(\phi)$  koja:

- instancira bilo koji kvantifikator u  $\phi$

- transformiše rezultat u konjukciju klauzula, gde je svaka klauzula disjunkcija literala

Zatim, algoritam iterativno prolazi kroz Datalog pravila, do unapred zadate granice, nazvane  $bound_k$ . U svakom koraku for-petlje, algoritam generiše SMT ograničenje koje obuhvata:

- koji atomi se izvedu nakon  $k$  primena  $P$  pravila
- koji atomi nikad nisu izvedeni od strane  $P$ .

SMT ograničenje označeno je sa  $[P]_k$ . Algoritam takođe prepisuje pojednostavljeno ograničenje  $\phi'$  koristeći funkciju  $Rewrite(\phi', k)$  koja rekurzivno prolazi konjukcije i disjunkcije u pojednostavljenom ograničenju  $\phi'$  i mapira pozitivne literalne na predikat  $p_k(t)$  i negativne literalne na  $\neg p(t)$ :

$$REWRITE(\varphi, k) = \begin{cases} p_k(\bar{t}) & \text{if } \varphi = p(\bar{t}) \\ \neg p(\bar{t}) & \text{if } \varphi = \neg p(\bar{t}) \\ REWRITE(\varphi_1, k) \vee \dots \vee REWRITE(\varphi_n, k) & \text{if } \varphi = \varphi_1 \vee \dots \vee \varphi_n \\ REWRITE(\varphi_1, k) \wedge \dots \wedge REWRITE(\varphi_n, k) & \text{if } \varphi = \varphi_1 \wedge \dots \wedge \varphi_n \end{cases}$$

Ako je ograničenje  $[P]_k \bigwedge \phi_k$  zadovoljivo, onda je ulaz  $I$  izveden.

## 2.2 Iterativna sinteza ulaza za Datalog

Prikaz algoritma dat je na Slici 3. Pretpostavlja se bez gubitka opštosti da je ograničenje fiksne tačke  $\phi$  definisano preko predikata koje se pojavljuju u najvišem sloju  $P_n$ . Počevši od najvišeg sloja  $P_n$ ,  $S_{Strat}$  generiše ulaz  $I_n$  za  $P_n$  tako da  $[[P_n]]I_n \models \phi$ . Zatim se iterativno sintetiše ulaz za donje slojeve  $P_{n-1}, \dots, P_1$  koristeći algoritam  $S_{SemiPos}$ . Na kraju, da bi se konstruisao ulaz za  $P$ , algoritam kombinuje ulaze sintetizovane za sve slojeve i vraća ulaz  $I$ .

## 3 Implementacija i eksperimenti

SyNET je implementiran u Python-u i automatski kodira Datalog programe. Koristi Python API da bi proverio da li su generisana SMT ograničenja zadovoljavajuća i da dobije model. SyNET koristi prirodno razdvajanje protokola: spoljašnje putanje obavlja BGP, dok unutrašnjim rukuju IGP protokoli (OSPF sa statičkim rutama).

### 3.1 Eksperimenti

Kako bi se istražile performanse i skalabilnost SyNET-a, eksperimentisalo se sa različitim:

- topologijama
- zahtevima
- kombinacijom protokola.

Za ispravnost testova, pokreću se sve sintetizovane konfiguracije u simuliranom okruženju Cisco rutera i proverava se da sledeći putevi odgovaraju.

Pokazuje se da je SyNET sistem praktičan i da daje tačne ulazne podatke, u razumnom vremenskom roku, za mreže realne veličine ( $> 50$  rutera) koje prosleđuju pakete za više klasa saobraćaja.

---

**Algorithm 2.** Input synthesis algorithm  $\mathcal{S}_{Strat}$  for stratified Datalog

---

**Input:** Stratified Datalog program  $P = P_1 \cup \dots \cup P_n$ , constraint  $\varphi$  over  $P_n$   
**Output:** An input  $I$  such that  $\llbracket P \rrbracket_I \models \varphi$  or  $\perp$

```

1 begin
2    $\mathcal{F}_1 \leftarrow \emptyset, \dots, \mathcal{F}_n \leftarrow \emptyset; I_1 \leftarrow \perp, \dots, I_n \leftarrow \perp; i \leftarrow n$ 
3   while  $i > 0$  do
4     if  $|\mathcal{F}_i| > bound_{\mathcal{F}}$  then
5        $\mathcal{F}_i \leftarrow \emptyset; \mathcal{F}_{i+1} \leftarrow \mathcal{F}_{i+1} \cup \{I_{i+1}\}$ 
6        $i \leftarrow i + 1; \quad // \text{ backtrack to higher stratum}$ 
7       continue
8      $\psi_{\mathcal{F}} \leftarrow \bigwedge_{I' \in \mathcal{F}_i} (\neg \bigwedge_{p \in edb(P_i)} \text{ENCODEPRED}(I', p))$ 
9     if  $i = n$  then
10       $\psi_i \leftarrow \varphi$ 
11    else
12       $\psi_i \leftarrow \bigwedge_{p \in \Delta_i} \text{ENCODEPRED}(I_{i+1} \cup \dots \cup I_n, p)$ 
13      where  $\Delta_i = (edb(P_i) \cup idb(P_i)) \cap (edb(P_{i+1}) \cup \dots \cup edb(P_n))$ 
14       $I_i = \mathcal{S}_{SemiPos}(P_i, \psi_i \wedge \psi_{\mathcal{F}})$ 
15      if  $I_i \neq \perp$  then
16         $i \leftarrow i - 1$ 
17      else
18        if  $i < n$  then
19           $\mathcal{F}_i \leftarrow \emptyset; \mathcal{F}_{i+1} \leftarrow \mathcal{F}_{i+1} \cup \{I_{i+1}\}$ 
20           $i \leftarrow i + 1 \quad // \text{ backtrack to higher stratum}$ 
21        else
22          return  $\perp$ 
23  return  $I = \{p(\bar{t}) \in I_1 \cup \dots \cup I_n \mid p \in edb(P)\}$ 

```

---

Slika 3: Algoritam  $\mathcal{S}_{Strat}$

## 4 Zaključak

U radu je formulisan problem sinteze mrežne konfiguracije kao problem pronalaženja ulaza Datalog programa i predstavljen je nov algoritam za rešavanje problema. Algoritam se zasniva na podeli Datalog pravila u slojeve i iterativnoj sintezi za svaki pojedinačni sloj korišćenjem SMT rešavača. Algoritam je primenjen u sistemu nazvanom SyNET i pokazalo se da je skalabilan realnoj veličini mreže koristeći bilo koju kombinaciju OSPF, BGP i statičkih ruta. S obzirom da je SyNET-u potrebno više od 24h da sintetiše konfiguracije za najveće mreže danas, a mreže su hijerarhijski organizovane u regije kako bi se mogle konfigurisati nezavisno. Autori rada u budućem radu planiraju da istraže sintezu takvih hijerarhijskih konfiguracija.

## Literatura

- [1] Vanbever L. Vechev M. El-Hassany A., Tsankov P. *Network-Wide Configuration Synthesis*. Springer, Cham, 2017. on-line at [https://doi.org/10.1007/978-3-319-63390-9\\_14](https://doi.org/10.1007/978-3-319-63390-9_14).