

CTL i primeri svojstava koji se mogu izraziti u CTL-u

Seminarski rad u okviru kursa
Verifikacija softvera
Matematički fakultet

Nikola Grulović, 1112/2018
n.grulovic@outlook.com

15. januar 2020

Sažetak

Logika stabla izračunavanja (eng. Computation Tree Logic, ili CTL) je vrsta logike razgraničavanja (eng. Branching-time logic), i predstavlja neku vrstu dodatka na linearnu temporlanu logiku (LTL). U ovom seminarskom radu prvo će biti definisan jezik CTL-a, tj. njegova sintaksa i semantika. Nakon toga biće prikazani neki primeri svojstva koji mogu da se izraze u njemu.

Sadržaj

1	Uvod	2
2	Jezik logike stabla izračunavanja	2
2.1	Sintaksa CTL-a	2
2.2	Semantika CTL-a	3
2.2.1	Tranzicioni sistem	3
2.2.2	Tačnost CTL formule	3
2.2.3	Ekvivalencija CTL formula	4
2.2.4	Praktični primeri CTL formula	4
3	Svojstva CTL-a	5
3.1	Svojstvo životosti	5
3.2	Sigurnosno svojstvo	5
3.3	Svojstvo pravednosti	5
	Literatura	6

1 Uvod

Linearna temporalna logika (LTL) vreme modeluje linearno, kao niz vremenskih trenutaka izomorfni skupu prirodnih brojeva. Definirano je da stanje sistema zadovoljava LTL formulu ukoliko zadovoljava sve putanje od zadanog stanja.[3]

Svojstva koja potvrđuju postojanje puta ne mogu biti izražena u LTL-u. Logika razgraničavanja rešava ovaj problem omogućavajući eksplicitno kvantifikaciju puta. Jedna od tih logika jeste logika stabla izračunavanja. U CTL, pored temporalnih operatora U, F, G i X iz LTL, postoje kvantifikatori A i E, koji imaju značenje 'sve putanje' i 'postoji putanja'. Na primer:

- Postoji dostižna putanja koja zadovoljava q : $EF\ q$.
- Za sva dostižna stanja koja zadovoljavaju p , moguće je održavati p dok se ne dođe do stanja q : $AG\ (p \rightarrow E[p\ U\ q])$.
- Postoji dostižno stanje od kojih sva dostižna stanja zadovoljavaju p : $EF\ AG\ p$.

2 Jezik logike stabla izračunavanja

Jezik logike stabla izračunavanja je sličan jeziku linearne temporalne logike, s dodatkom kvantifikatora A i E. U ovom poglavlju prvo opisujemo sintaksu, a potom proučavamo semantiku CTL-a.

2.1 Sintaksa CTL-a

Logika stabla izračunavanja (CTL) je logika razgraničavanja, što znači da je njen model vremena struktura poput stabla u kojoj budućnost nije određena; U budućnosti postoje različiti putevi, od kojih bi svaki mogao biti 'stvarni' put koji se realizuje. Kao i kod LTL radimo sa fiksnim skupom atomskih formula (kao što su p, q, r, \dots , ili p_1, p_2, p_3, \dots).

Definicija 2.1[4] *Formule CTL definišemo induktivno preko Backus-Naurov forme (kratko BNF), kao što je urađeno kod LTL:*

$$\begin{aligned} \varphi ::= & \top \mid \perp \mid p \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid AX\ \varphi \mid EX\ \varphi \\ & \mid AF\ \varphi \mid EF\ \varphi \mid AG\ \varphi \mid EG\ \varphi \mid A[\varphi\ U\ \varphi] \mid E[\varphi\ U\ \varphi] \end{aligned}$$

gde se p uzima iz skupu atomskih formula.

Možemo da primetimo da je svaki od temporalnih veznika CTL-a par simbola. Prvi par je jedan od A i E. A znači 'duž svih puteva', a E znači 'duž najmanje jednog puta'. Drugi par je jedan od X, F, G ili U simbola, što znači 'neXt state', 'some Future state', 'all future states (Globally)' and 'Until', u datom poretku. U CTL-u su parovi simbola poput EU nedeljivi. Primećujemo da su AU i EU binarni simboli. X, F, G i U simboli se ne mogu naći a da im prvo ne prethodi ili A ili E; slično svakom A ili E mora da sledi jedan od X, F, G ili U simbola. Obično 'weak-until' (W) i 'release' (R) nisu uključeni u CTL ali se oni mogu izvesti.

Napomena 2.1[4] *Prioriteti veznika su slični kao kod predikatske i propozicione logike. Unarni veznici (koji se sastoje od \neg i temporalnih veznika AG, EG, AF, EF, AX i EX) su najvećeg prioriteta. Sledeći po prioritetu su \wedge i \vee , a najnižeg prioriteta \rightarrow, AU i EU .*

Prirodno, možemo da koristimo zagrade da izmenimo prioritete operatorima. U daljem tekstu biće prikazani neki primeri dobro definisanih i loše definisanih CTL formula. Predpostavimo da su p , q i r atomične formule. Naredne formule su dobro definisane[4]:

- $AG (q \rightarrow EG r)$, primetimo da ovo nije isto kao $AG q \rightarrow EG r$, prema napomeni 2.1 formula je ekvivalenta $(AG q) \rightarrow (EG r)$
- $EF E[r \cup q]$
- $A[p_1 \cup A[p_2 \cup p_3]]$
- $AG (p \rightarrow A[p \cup (\neg p \wedge A[\neg p \cup q])])$.

Primeri loše definisanih formula:

- $EF G r$
- $A \neg G \neg p$
- $EF (r \cup q)$
- $A[(r \cup q) \wedge (p \cup r)]$.

Razlog zašto je formula $A[(r \cup q) \wedge (p \cup r)]$ loše definisana, je što sinaksa ne dozvoljava da stavljamo logičke veznike (kao \wedge) direktno između $A[]$ ili $E[]$. Pojavljivanje A ili E mora da prati sa jednim od simbola G , F , X ili U ; ako su praćeni sa U , moraju da budu u formi $A[\psi \cup \phi]$. Sada, ψ i ϕ mogu da sadrže \wedge , jer su proizvoljne formule; pa $A[(p \wedge q) \cup (\neg r \rightarrow \neg q)]$ je dobro definisana formula.

Može se primetiti da su AU i EU binarni veznici koji mešaju infiksnu i prefiksnu notaciju. U čisto infiksnoj, pisali bi $\psi_1 AU \psi_2$, a u čisto prefiksnoj $AU (\psi_1, \psi_2)$ [4].

Definicija 2.2[4] *Podformula CTL formule ψ je bilo koja formula ϕ čije stablo parsiranja je podstablo ψ ' stabla parsiranja.*

2.2 Semantika CTL-a

2.2.1 Tranzicioni sistem

Definicija 2.3[2] *Tranzicioni sistem je uređena šestorka $(S, Act, \rightarrow, I, v, L)$ gde je:*

- S skup stanja
- $\rightarrow \subseteq S \times Act \times S$ relacija prelaska $(s_i \rightarrow s_j)$
- $I \subseteq S$ skup početnih stanja
- v skup iskaznih promenljivih (skup predikata)
- $L : S \rightarrow v$ funkcija mapiranja (svakom stanju pridružuje skup predikata koji važe u tom stanju)

2.2.2 Tačnost CTL formule

CTL formule su interpretirane preko tranzicionog sistema (Definicija iz 2.2.1). Neka $M = (S, \rightarrow, L)$ bude takav sistem za CTL, $s \in S$ i ϕ je CTL formula. Formula ϕ se posmatra kao rekurzija (preko stabla parsiranja) i može da se intepretira kao:

- ϕ je atomična formula, zadovoljivost zavisi od L .

- Ako se na najvišem nivou stabla parsiranja nalaze logički veznici (\wedge , \vee , \neg itd.), zadovoljivost formule se utvrđuje preko istinitnosnih tablica i daljom rekurzijom kroz formulu ϕ .
- Ako se na najvišem nivou stabla parsiranja nalazi operator koji počinje sa A, onda se zadovoljivost formule utvrđuje kad sve putanje iz s zadovoljavaju 'LTL formulu' koja se dobija uklanjanjem A operatora iz početne formule.
- Slično kao malopre, ako se na vrhu stabla parsiranja nalazi operator koji počinje sa E, onda se zadovoljivost formule utvrđuje kad neke putanje iz s zadovoljavaju 'LTL formulu' koja se dobija uklanjanjem E operatora iz početne formule.

U poslednja dva slučaja, rezultat uklanjanja operatora A ili E ne garantuje da će dobijena formula biti 'LTL formula' jer je moguće da će sadržati A ili E dalje u formuli. Ovo neće smetati jer se to razrešava rekurzijom.

2.2.3 Ekvivalencija CTL formula

Definicija 2.4[2] Dve CTL formule ϕ i ψ su semantički ekvivalentne akko proizvoljno stanje u proizvoljnom modelu zadovoljava ϕ , onda mora da zadovoljava i ψ . Označava se sa $\phi \equiv \psi$.

Već smo primetili da je A univerzalni kvantifikator na putanjama i E je odgovarajući egzistencijalni kvantifikator. Štaviše, G i F su takođe univerzalni i egzistencijalni kvantifikatori, koji imaju opseg na određenom putu. Uzimajući u obzir ove činjenice, ne iznenađuje nas da postoje de Morganova pravila:

- $\neg AF \phi \equiv EG \neg \phi$
- $\neg EF \phi \equiv AG \neg \phi$
- $\neg AX \phi \equiv EX \neg \phi$

Takođe postoje i ekvivalencije:

- $AF \phi \equiv A[\top \cup \phi]$
- $EF \phi \equiv E[\top \cup \phi]$

koje su slične odgovarajućim ekvivalencijama u LTL-u.

2.2.4 Praktični primeri CTL formula

Korisno je pogledati neke tipične primere formula i njihovo poređenje sa LTL-om. Predpostavićemo da su neki opisi atomični, kao na primer 'zauzet (busy)' i 'zahtevao (requested)':[4]

1. Moguće je da se dobije stanje otpočeo(*started*) koje čeka, a stanje spreman(*ready*) ne: $EF(started \wedge \neg ready)$.
2. Za bilo koje stanje, ako se pojavi neko zahtevanje (*requested*) resursa, onda će ono eventualno biti obrađeno (*acknowledged*): $AG(requested \rightarrow AF acknowledged)$.
3. Odgovarajući proces je uključen (*enabled*) beskonačno često na svakom putu računanja: $AG(AF enabled)$.
4. Šta god da se desi, odgovarajući proces će eventualno da postane trajno zaključan (*deadlock*): $AF(AG deadlock)$.
5. Iz bilo kog stanja moguće je da se dobije stanje ponovnog pokretanja (*restart*), tačnije da se vrati na početno stanje: $AG(EF restart)$.

6. Proces može uvek da zahteva da uđe u kritičnu sekciju. Ovo je nemoguće izraziti pomoću LTL-a. $AG (n_1 \rightarrow t_1)$.
7. Procesi ne moraju da uđu u kritičnu sekciju u striktnom redosledu. Ovo takođe nije moguće izraziti direktno u LTL-u, ali je moguće izraziti negaciju. CTL nam dozvoljava da to direktno izrazimo: $EF (c_1 \wedge E[c_1 U (\neg c_1 \wedge E[\neg c_2 U c_1])])$.

3 Svojstva CTL-a

Svojstva se formulišu kao CTL formule tako da sistem koji se verifikuje zadovoljava dato svojstvo akko je odgovarajuća formula valjana u tranzicionom sistemu M koji je model početnog sistema.

Primer svojstva koje se mogu izraziti na CTL jeziku a ne mogu na LTL, je svako svojstvo koje uključuje vremensko rezonovanje o svim putanjama. Na primer iz svakog dostižnog stanja se može stići u neko početno stanje (Primer 5 iz poglavlja 2.2.4)[2].

3.1 Svojstvo životosti

Definicija 3.1 *Svojstvo životosti je svojstvo koje izražava da će se neka pozitivna pojava sigurno desiti u budućnosti [1].*

Primer 3.1 Nekada u budućnosti ću biti bogat:

$$AF \text{ rich}$$

Primer 3.2 Negde u budućnosti će x biti veće od 5:

$$AF(x > 5)$$

Primer 3.3 Sigurno će važiti kada program počne da radi, da će se u nekom trenutku završiti:

$$AG(\text{start} \rightarrow AF \text{ terminate})$$

Takva svojstva obično počinju sa $AF(\phi)$.

3.2 Sigurnosno svojstvo

Definicija 3.2 *Sigurnosno svojstvo je svojstvo koje izražava da se neka negativna pojava nikada neće desiti [1].*

Primer 3.4 Neće se nikada desiti da temperatura reaktora pređe 1000C:

$$AG \neg(\text{reactor_temp} > 1000)$$

Primer 3.5 Neće se nikada desiti da $x = 0$ i za svako sledeće pojavljivanje promenljive x, y, z dođe do $y = z/x$:

$$AG \neg((x = 0) \wedge AX AX AX (y = z/x))$$

Takva svojstva obično počinju sa $AG \neg(\phi)$.

3.3 Svojstvo pravednosti

Definicija 3.3 *Svojstvo pravednosti je svojstvo koje izražava da će se neka pojava dešavati beskonačno puta tokom izvršavanja. Obično je korisna za zakazivanje procesa, slanje poruka, itd [1].*

Primer 3.6 Sigurno će se desiti u budućnosti da će se proces uključiti:

$AG (AF \textit{ enabled})$

Takva svojstva obično počinju sa $AG \ AF \ (\phi)$.

Literatura

- [1] Alessandro Artale. Formal Methods - Computation Tree Logic (CTL). 2019.
- [2] Milena Vujošević Janičić. Verifikacija Softvera - Proveravanje modela. 2019.
- [3] Ivona Jurošević. LTL i primeri svojstava koji se mogu izraziti u LTL-u. 2018.
- [4] Mark Ryan Michael Huth. *Logic in Computer Science*. Cambridge University Press, The Edinburgh Building, Cambridge CB2 8RU, UK, 2004.