

Softver za proveru modela zasnovan na SMT-u: Eksperimentalno poređenje četiri algoritma

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Isidora Đurđević
isidoradjurdjevic.100@gmail.com

8. maj 2015.

Sažetak

Postoji veliki broj algoritama koji se koriste za proveru modela i verifikaciju softvera. U ovom radu će biti grubo prikazana četiri algoritma koja se zasnivaju na SMT rešavaču: ograničena provera modela (eng. *bounded model checking*), k-indukcija (eng. *k-induction*), apstrakcija predikata (eng. *predicate abstraction*) i lenja apstrakcija sa interpolantima (eng. *abstraction with interpolants*)[1].

ključne reči: Verifikacija softvera, Provera modela, Analiza programa, K-indukcija, Lenja apstrakcija, SMT rešavač, Algoritam impakta

Sadržaj

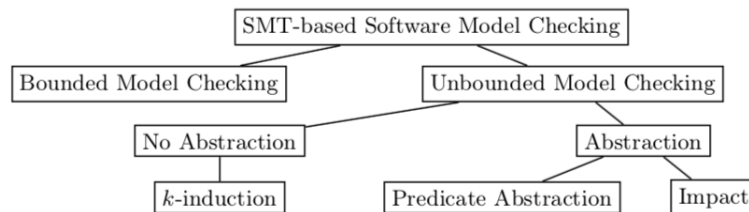
1	Uvod	2
2	Osnovni pristupi za proveru modela	2
2.1	Ograničena provera modela	2
2.2	Neograničena provera modela - bez apstrakcije	2
3	Algoritmi	3
3.1	Ograničena provera modela	3
3.2	k-Indukcija	3
3.3	Apstrakcija predikata	4
3.4	Algoritam impakta	4
4	Zaključak	4
	Literatura	5

1 Uvod

U nastavku teksta će najpre biti objašnjeni algoritmi za proveru modela i na koji način oni pretražuju stanja programa, nakon čega ćemo objasniti i rad samih algoritama koji se zasnivaju na SMT-u, i zašto neki algoritmi mogu da reše određene verifikacione zadatke, dok drugi algoritmi to ne mogu. Prikazaćemo koje su prednosti i mane svakog pristupa, kao i gde se mogu primeniti ovi algoritmi.

2 Osnovni pristupi za proveru modela

Ograničena provera modela, k-indukcija, apstrakcija predikata i algoritam impakta su algoritmi koji su zasnovani na SMT-u, što znači da su putanje programa predstavljene kao SMT formule. Na slici 1 se može videti kategorizacija osnovnih pristupa provere modela softvera koji su zasnovani na SMT-u.



Slika 1: Osnovni pristupi provere modela

2.1 Ograničena provera modela

U ograničenoj proveri modela, program se otpetljava do neke granice. Otpetljavanjem programa dobijamo sve putanje programa, predstavljene kao SMT formule, gde svaka putanja prolazi kroz određena stanja u kojima se program može naći. Takva formula će biti zadovoljiva u slučaju da ta formula zadovoljava specifikacije softvera od njenog početka do kraja. Ograničena provera modela je odličan način provere zbog toga što je moguće ispitati veliki broj stanja, bez izvršavanja apstrakcije. Nekoliko implementacija su se učinile uspešnim: Cbmc, ESBMC, Llbmc i Smack [1].

2.2 Neograničena provera modela - bez apstrakcije

Moguće je izvršiti i neograničenu proveru modela, tj. svesti ograničenu proveru modela na neograničenu tako što se koristi induktivna invarijanta petlje. Dakle, neophodno je da postoji invarijanta petlje, koja će važiti pre ulaska u petlju, tokom izvršavanja petlje i nakon njenog završetka.

Posmatrajmo naredni fragment pseudo koda:

```
x = 2
while (true): x = 2*x - 1
```

U ovom slučaju, sigurna invarijanta petlje je $x \geq 0$, jer uslov važi pre ulaska u petlju, tokom izvršavanja petlje, i nakon njenog završetka. Dakle, u našem slučaju treba postojati sigurna invarijanta petlje koja važi za sve putanje od početka programa, do petlje i nakon njenog završetka.

Ujedno je i najveći izazov odrediti sigurnu invarijantu petlje. Postoji nekoliko implementacija: Cbmc, CPAchecker, Esbmc, Pkind i 2ls [1].

3 Algoritmi

3.1 Ograničena provera modela

Neophodno je prvo uvesti pojam ABE, naime ABE (eng. *adjustable-block encoding*) se koristi za otpetljavanje CFA-e (eng. *control-flow automaton*) u ARG (eng. *abstract reachability graph*). U ograničenoj proveru modela (eng. *bounded model checking*) prostor stanja analiziranog programa je pretražen bez bilo kakve apstrakcije tako što se otpetljaju petlje konačan broj puta k . Na ovaj način postoji samo jedan blok ograničene veličine na ulazu programa. Limit za otpetljavanje ovom metodom je definisan konačnim brojem k . S obzirom na to da samo jedan blok čini ceo program, formula putanje stanja je definisana skupom svih putanja programa kojim se može doći od početnog stanja do tekućeg stanja. Kada se otpetlja petlja do granice k , pretraga stanja se zaustavlja. Onda se proverava zadovoljivost disjunkcija formula putanja svih stanja u pretraženom prostoru SMT rešavačem. Ukoliko je formula zadovoljiva to znači da program zadovoljava sve specifikacije softvera. Inače, formula je nezadovoljiva u okviru prvih k otpetljavanja petlje.

3.2 k-Indukcija

K-indukcija takođe spada u algoritme koje se ne zasnivaju na apstrakciji. Algoritam za k-indukciju se sastoji iz dve faze: Prva faza je ekvivalentna ograničenoj proveru modela sa ograničenjem k i zove se *bazni slučaj indukcionog dokaza*. U slučaju da se nešto ne slaže sa specifikacijom softvera u baznom slučaju, algoritam se zaustavlja i prijavljuje se greška, inače se ulazi u drugu fazu. U drugoj fazi, ABE se koristi da ponovo pretraži prostor stanja analiziranog programa, gde analiza i ABE blok počinju od početka petlje tako da svaka formula putanje od svakog stanja uvek predstavlja skup konkretnih putanja programa od početka petlje do tog stanja u programu. Otpetljavanje CFA-e se zaustavlja posle $k+1$ ponavljanja. Posle toga se proverava SMT rešavačem zadovoljivost implikacije, odnosno proverava se da li iz negacije disjunkcija svih formula putanja za sva stanja koja su dostignuta u okviru prvih k otpetljavanja sledi negacija disjunkcija svih formula putanja koja su dostignuta u okviru prvih $k+1$ otpetljavanja. Ovaj faza se naziva induktivnim korakom. U slučaju da je implikacija tačna, program je ispravan, inače se prijavljuje greška.

3.3 Apstrakcija predikata

Apstrakcija predikata koja se zasniva na CEGAR-u direktno primenjuje ABE. Formula apstraktnog stanja aproksimira dostižna konkretna stanja korišćenjem kombinacije predikata preko programskih varijabli iz datog seta predikata. Ovu apstrakciju izračunava SMT rešavač. Koristeći CEGAR moguće je primeniti lenju apstrakciju počevši sa praznom inicijalnom preciznošću. Kad se analiza susretne sa apstraktnim stanjem na lokaciji greške, putanja programa koja vodi do ovog stanja se rekonstruiše i proverava se izvodljivost koristeći SMT rešavač. Ako je izvodljivo izvršavanje putanje sa greškom, onda algoritam prijavljuje grešku i prekida se. Inače, preciznost se poboljšava i analiza se ponovo pokreće.

3.4 Algoritam impakta

Lenja apstrakcija sa interpolantima, poznatiji kao algoritam impakta zahvaljujući svojoj prvoj implementaciji u alatu Impakt (engl Impact), takođe koristi ABE za otpetljavanje slično kao u apstrakciji predikata. Lenja apstrakcija sve vreme generiše i sređuje jedan apstraktni model, tako da različiti delovi modela mogu razlikovati u preciznosti. Međutim, ovaj algoritam ne zasniva svoje apstrakcije na preciznosti. Inicijalizacijom novih formula apstraktnih stanja na tačno, algoritam neprekidno primenjuje sledeća tri koraka dokle god se neke promene mogu napraviti [1]:

(1) *Proširenje(s)*: Ukoliko stanje nema naslednika i nije obeležen kao pokriven, naslednik stanja od s se pravi sa inicijalnim stanjem apstraktne formule postavljenim na vrednost tačno.

(2) *Poboljšavanje(s)*: Ukoliko je s apstraktno stanje na lokaciji greške sa apstraktnim stanjem formule postavljenim na tačno, induktivni interpolanti za putanju koja vodi od korena ARG-a do ovog stanja s se rešavaju koristeći SMT-rešavač.

(3) *Pokrij(s_1, s_2)*: Stanje s_1 se obeležava kao da je pokriveno drugim stanjem s_2 u slučaju da ni s_2 niti bilo koji njegov predak nije pokriven, da oba stanja pripadaju istoj programskoj lokaciji, da apstraktna formula od s_2 sledi iz formule s_1 , da s_1 nije predak od s_2 , kao i da je s_2 napravljen pre s_1 . U suštini, lenja apstrakcija funkcioniše tako što se najpre primeni apstrakcija, (odabere se konačan broj predikata i model apstrakcije se automatski generiše kao konačan automat stanja), zatim se vrši verifikacija (proverava se da li apstraktni model zadovoljava specifikaciju programa. U slučaju da je model bez greške, onda je i originalan program bez greške, inače je pronađen je kontraprimer koji ne zadovoljava specifikaciju.), nakon čega se proverava da li se kontraprimer može primeniti u originalnom programu. U slučaju da je to moguće, pronađena je greška u originalnom programu, inače izabran skup predikata ne može da dokaže ispravnost programa i neophodno je dodati nove predikate postojećem skupu.

4 Zaključak

Prikazali smo četiri najpopularnija algoritma za verifikaciju softvera i proveru modela, iz čega možemo zaključiti da iako je ograničena provera modela jako ograničena u dokazivanju ispravnosti, ona je i najjasnija od ova četiri algoritma, da k-indukcija zahteva generator pomoćne invarijante da bi bila primenljiva u praksi, a da predikatska apstrakcija i algoritam

impakta zahtevaju interpolacione tehnike. Dakle, ne postoji pravi ili najefikasniji algoritam za proveru modela, već treba uzeti u obzir prednosti i mane svakog pristupa i prema tome primeniti odgovarajući algoritam.

Literatura

- [1] Matthias Dangl Dirk Beyer. SMT-based Software Model Checking: An Experimental Comparison of Four Algorithms.