UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
DEPARTMAN ZA RAČUNARSTVO I AUTOMATIKU
KATEDRA ZA PRIMENJENE RAČUNARSKE NAUKE

Doktorska disertacija

Analiza osobina dinamičkih postuslova u Horovim tripletima

Mentor:
Dr Dušan Malbaški

Kandidat:
Mr Aleksandar Kupusinac

Novi Sad
2010

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Докторска дисертација
Аутор, **АУ**:		Мр Александар Купусинац
Ментор, **МН**:		Др Душан Малбашки
Наслов рада, **НР**:		АНАЛИЗА ОСОБИНА ДИНАМИЧКИХ ПОСТУСЛОВА У ХОРОВИМ ТРИПЛЕТИМА
Језик публикације, **ЈП**:		Српски
Језик извода, **ЈИ**:		Српски/Енглески
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		АП Војводина, Нови Сад
Година, **ГО**:		2010.
Издавач, **ИЗ**:		ауторски репринт
Место и адреса, **МА**:		Факултет техничких наука, 21000 Нови Сад, Трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		4/145/103/1/1/0/1
Научна област, **НО**:		Примењене рачунарске науке и информатика
Научна дисциплина, **НД**:		Теорија програмирања
Предметна одредница/Кључне речи, **ПО**:		Предикатска логика првог реда, S-програмски рачун, Динамички постуслови, Анализа инваријанта у класи
УДК		
Чува се, **ЧУ**:		Библиотека Факултета техничких наука у Новом Саду
Важна напомена, **ВН**:		---
Извод, **ИЗ**:		Докторска дисертација презентује нов и општији начин анализирања семантике структурираних и објектно оријентисаних програма и то искључиво у оквирима предикатске логике првог реда. Докторска дисертација разматра следеће теме: 1.) S-програмски рачун, 2.) Дефиниција и особине динамичких постуслова у S-рачуну, 3.) Концептуалне дефиниције објекта, класе и инваријанте, 4.) Анализа инваријаната у класи (SP-анализа и DP-анализа).
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	Др Мирослав Хајдуковић
	Члан:	Др Раде Дорословачки
	Члан:	Др Силвиа Гилезан
	Члан:	Др Ивана Берковић
	Члан, ментор:	Др Душан Малбашки

Потпис ментора

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:	
Identification number, **INO**:	
Document type, **DT**:	Monographic publication
Type of record, **TR**:	Textually printed document
Contents code, **CC**:	Ph.D. Thesis
Author, **AU**:	Aleksandar Kupusinac, M.Sc.
Mentor, **MN**:	Dušan Malbaški, Ph.D.
Title, **TI**:	ANALYSES OF CHARACTERISTICS OF DYNAMIC POSTCONDITIONS IN HOARE TRIPLETS
Language of text, **LT**:	Serbian
Language of abstract, **LA**:	Serbian/English
Country of publication, **CP**:	Republic of Serbia
Locality of publication, **LP**:	AP Vojvodina, Novi Sad
Publication year, **PY**:	2010.
Publisher, **PB**:	Author's reprint
Publication place, **PP**:	Faculty of Technical Sciences, 21000 Novi Sad, Trg Dositeja Obradovića 6
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)	4/145/103/1/1/0/1
Scientific field, **SF**:	Applied computer science and informatics
Scientific discipline, **SD**:	Theory of programming
Subject/Key words, **S**/**KW**:	First-order predicate logic, S-program calculus, Dynamic postconditions, Analyses of invariants in class
UC	
Holding data, **HD**:	Library of the Faculty of Technical Sciences in Novi Sad
Note, **N**:	---
Abstract, **AB**:	Doctoral thesis presents a new and more general method for analizing of structured and object-oriented program semantics, based on the first-order predicate logic. Doctoral thesis consideres next topics: 1.) S-program calculus, 2.) Definition and characteristics of dynamic postconditions in S-calculus, 3.) Conceptual definitions of object, class and invariant, 4.) Analyses of invariants in class (SP-analyses and DP-analyses).
Accepted by the Scientific Board on, **ASB**:	
Defended on, **DE**:	

Defended Board, **DB**:	President:	Miroslav Hajduković, Ph.D.	
	Member:	Rade Doroslovački, Ph.D.	
	Member:	Silvia Gilezan, Ph.D.	Menthor's sign
	Member:	Ivana Berković, Ph.D.	
	Member, Mentor:	Dušan Malbaški, Ph.D.	

Sadržaj

Uvod

Softverski inženjering podrazumeva proizvodnju kvalitetnog softverskog sistema, pa je od velikog značaja razvoj teorijskih i praktičnih metoda koje mogu proveriti i poboljšati kvalitet softverskog sistema. Kriterijumi kvaliteta softverskog sistema su mnogobrojni i mogu se podeliti na unutrašnje i spoljašnje kriterijume [77]. Spoljašnji kriterijumi (brzina, jednostavnost korišćenja softvera i sl.) su spolja vidljivi i obično predstavljaju predmet interesovanja krajnjih korisnika, koji nisu profesionalni programeri. Međutim, profesionalnog programera više će interesovati unutrašnji kriterijumi kvaliteta (višekratna upotreba, proširivost, robustnost, korektnost i sl.), jer ključ za postizanje kvaliteta po spoljašnjim kriterijumima, leži u postizanju kvaliteta po unutrašnjim kriterijumima. Drugim rečima, da bi krajnji korisnici mogli da uživaju u vidljivim kvalitetima jednog softverskog sistema, potrebno je da projektanti i programeri prethodno obezbede skrivene kvalitete tog softverskog sistema, što podrazumeva postizanje kvaliteta po unutrašnjim kriterijumima.

Kriterijumi kvaliteta su raznovrsni i nalaze se u stalnom sukobu, stoga je neophodno da se profesionalni programer odluči za optimalnu varijantu i napravi kompromis između sukobljenih kriterijuma kvaliteta u zavisnosti od namene softverskog sistema koji projektuje. Međutim, ma koliko bilo neophodno da se takvi kompromisi prave, postoji jedan kriterijum kvaliteta koji predstavlja izuzetak, a to je *korektnost* [53]. Ne postoji opravdanje za profesionalnog programera koji zbog postizanja boljeg kvaliteta na osnovu drugih kriterijuma dovede u pitanje korektnost softverskog sistema, pa se zbog toga može zaključiti da korektnost predstavlja jedan od primarnih kriterijuma kvaliteta softverskog sistema.

- Korektnost neke programske jedinice (program, klasa, potprogram, metoda) jeste njena sposobnost da funkcioniše prema svojoj specifikaciji, pri čemu se pod specifikacijom podrazumeva precizan opis šta data programska jedinica treba da radi.

Primetimo da je korektnost relativan pojam, jer jedna programska jedinica sama za sebe nije ni korektna ni nekorektna. Programska jedinica je korektna, odnosno nekorektna samo u odnosu na neku specifikaciju. Postupak utvrđivanja korektnosti programske jedinice u odnosu na zadatu specifikaciju naziva se *verifikacija*. Drugim rečima, verifikacija obuhvata opisivanje semantike date programske jedinice, odnosno dobijanje semantičkog opisa, a zatim proveru da li je dobijeni semantički opis ekvivalentan zadatoj specifikaciji.

Nasuprot slobodnoj formi i velikoj izražajnoj moći, ali i nepreciznosti i dvosmislenosti prirodnih jezika [25] [58] [92] [59], formalni jezik čini skup reči na nekom konačnom

alfabetu, izgrađenih pomoću strogih pravila koja čine formalnu gramatiku [45]. U ovom izlaganju pažnju ćemo usmeriti na dva formalna jezika, a to su: predikatski račun i programski jezik. Predikatski račun predstavlja formalni jezik koji predstavlja uopštenje matematičkog jezika i čije izražajne moći su veće. Programski jezici jesu formalni jezici koji obezbeđuju vezu čoveka sa mašinom. Glavni motiv ovog izlaganja jeste činjenica da se program, i to kako strukturirani, tako i objektni, može posmatrati kao predikatski transformator, odnosno da se dokazivanje korektnosti programa svodi na dokazivanje valjanosti odgovarajuće predikatske formule.

Najzad, treba napomenuti da verifikacija softvera predstavlja aktuelnu oblast i to ne samo u teoriji, već i u programerskoj praksi. Nekorektan softver u raznim naučnim i stručnim oblastima može napraviti ogromne, a nekad i fatalne štete [83]. Tako na primer, nekorektan softver u biomedicinskom inženjeringu [60] [89] može ugroziti ljudski život, u industrijskoj proizvodnji [40] i optimizaciji upravljanja u industriji [50] [52] može izazvati ogromne gubitke, u algoritamskoj trgovini može napraviti pogrešnu procenu rizika ulaganja, što je naročito opasno kod ulaganja na berzama u razvoju, koje odlikuje velika nestabilnost i nepredvidivost [28], u kriptografiji može ugroziti tajnost podataka [75] itd.

Poglavlje 1 prikazuje razvoj S-programskog računa (kraće S-računa) koji predstavlja specijalnu podvrstu računa predikatske logike prvog reda [57]. U osnovi S-računa nalaze se aksiome i teoreme predikatske logike prvog reda. S-račun jeste račun S-formula koje su zadate nad apstraktnim skupom stanja neke virtuelne mašine. S-formule predstavljaju opštiji mehanizam za analizu programske semantike. Horove formule totalne i parcijalne korektnosti jesu dve specijalne S-formule i sva pravila Horove logike mogu se izvesti samo uz korišćenje aksioma i teorema predikatske logike prvog reda. U ovom poglavlju ćemo razmotriti vezu između formula totalne i parcijalne korektnosti. Horova aksioma dodele i pravila o specijalnim sintaksnim jedinicama nisu potrebni u S-računu, već dodela i ostale specijalne sintaksne jedinice se razmatraju kroz definisanje odgovarajućih S-formula. S-račun jeste moćan mehanizam kako za dokazivanje korektnosti programa tako i za dokazivanje novih teorema S-računa, koji koristi samo aksiome i teoreme predikatske logike prvog reda. U oba slučaja, dokazivanje se svodi na dokazivanje valjanosti odgovarajuće S-formule, pa se za izvođenje svih dokaza mogu koristiti i automatski dokazivači (u ovom izlaganju ćemo koristiti `Coq`), što daje poseban značaj S-računu. Kao primer primene S-računa, na kraju ovog poglavlja biće izvedeni dokazi Dajkstrinih teorema o najširem preduslovu wp [26]. Dajktra je formulisao četiri teoreme o najširem postuslovu i dokazao ih na način, za koji se ne može reći da je sa matematičkog gledišta strogo formalan [27] [33]. U ovom poglavlju, primenom S-računa dokazaćemo na strogo formalan način sve četiri Dajstrine teoreme, a zatim ćemo formulisati i dokazati još i novu, petu teoremu, a to je zakon negacije najšireg postuslova wp.

Postuslove koji su logičke funkcije početnog i završnog stanja zovemo dinamičkim postuslovima. U poglavlju 2 razmotrićemo osnovne osobine dinamičkih postuslova u okviru S-računa, sa ciljem da razvijemo matematički alat koji će nam kasnije koristiti za analizu semantike objektno orijentisanog programa. Posebnu pažnju ćemo posvetiti analizi osobina najužeg postuslova sp i najužeg dinamičkog postuslova $\hat{sdp}$.

Usled velike ekspanzije i brzog razvoja objektno orijentisanog programiranja, kao bočni efekat nastali su problemi u sâmom značenju pojmova i termina, pa čak i onih fundamentalnih, kao što su objekat, klasa i invarijanta. Terminologija objektne metodologije često je u koliziji sa već ustaljenom terminologijom u drugim naučnim oblastima (posebno u filozofiji). Zbog toga se nameće potreba da se značenje pojmova i upotreba termina usaglase i na taj način stvori jedan konzistentan konceptualno–terminološki sistem, koji je temelj svake naučne oblasti, pa tako i objektno orijentisanog programiranja. U poglavlju 3 ćemo izložiti konceptualne definicije klase, objekta i invarijante, bazirane na pojmu (konceptu).

Invarijante u klasi zauzimaju centralno mesto kada se govori o verifikaciji objektno orijentisanog programa, jer postoji korepodencija između semantike objektno orijentisanog programa i semantike klase [67]. U poglavlju 4 ćemo se baviti statičkom modularnom analizom klase, odnosno prikazaćemo analizu invarijanata u klasi primenom dinamičkih postuslova ili kraće DP-analizu. DP-analiza predstavlja moćan mehanizam za izračunavanje invarijanata, koji se bazira isključivo na predikatskom računu prvog reda. DP-analiza koristi dinamičke postuslove metoda u klasi, koji predstavljaju logičke funkcije početnog i završnog stanja. Centralno mesto u ovom poglavlju zauzima operacija generalizovane supstitucije pomoću koje izvodimo rekurzivne algoritme za izračunavanje stroge invarijante, odnosno ostalih invarijanata objekta i klase. Da bi primena DP-analize bila sveobuhvatna, razmotrićemo izračunavanje invarijante vlasnika i komponente u različitim situacijama, zatim nadklase i potklase, i najzad objekata sa varijabilnom strukturom, čija polja su pokazivači/reference koji pokazuju na memorijski prostor izvan objekta. Na taj način, pokazaćemo da se DP-analiza može koristiti za analizu kako nasleđivanja, tako i različitih klijentskih veza. Cilj ovoga izlaganja jeste da prikaže i demonstrira nov način izračunavanja invarijanata u klasi za koji je dovoljno poznavati predikatsku logiku prvog reda.

Glava 1

S-programski račun

Stvari treba pojednostavljivati koliko god je to moguće, ali ne i više od toga.
Albert Ajnštajn (1879–1955)

1.1 Uvod

Jedan od glavnih motiva za razvijanje S-programskog računa (kraće S-računa) jeste ideja da se program može razmatrati kao predikat i/ili logički izraz [87] [44] [36] [103] [43] [37] [38]. Ideja o razmatranju veze između Flojd–Horove logike [32] [41] i predikatske logike prvog reda postoji u radovima Kuka [18], Blesa i Gureviča [12], sa ciljem da se analizira kompletnosti Horove logike [5]. Blass i Gurevich govore o potrebi da se u Horovu logiku uvede predikatska logika prvog reda, ali oni ujedno smatraju da bi to značajno povećalo složenost Horove logike. Nasuprot tome, mi smatramo da upravo ta veza može uprostiti dokazivanje i istovremeno uopštiti ideje Horove logike na apstraktnom skupu stanja, ali je pre svega potrebno jasno razdvojiti domen interpretacije od domena apstraktnog skupa stanja. U svojim radovima Bek, Akademi i fon Vrajt su razvili ideju o posebnom programskom računu (*refinement calculus* [7]) koji bi spojio Horove ideje sa predikatskom logikom prvog reda. Međutim, pitanje determinizma programa u formulama totalne i parcijalne korektnosti Bek i fon Vrajt su rešili uvođenjem dodatnih formula anđeoske i demonske korektnosti (*angelical* i *demonical correctness* [6]), ali su time istovrememo povećali složenost njihovog računa. Povećana složenost i nejasno razdvajanje domena interpretacije od domena apstraktnog skupa stanja smanjile su dokazivačku moć njihog računa. Naša ideja je bila da razvijemo takav programski račun koji će spojiti Horove ideje sa predikatskom logikom prvog reda, jasno razdvojiti domen interpretacije od domena apstraktnog skupa stanja (slično kao [82]), kome neće predstavljati problem pitanje determinizma programa i koji će moći direktno da razmatra totalnu i parcijalnu korektnosti programa, odnosno bez novih pojmova i uvođenja dodatnih formula.

Ovde ćemo prikazati razvoj S-računa, koji predstavlja matematički alat za opisivanje i analizu programske semantike [73] [57]. Opštost S-računa pri analizi programske

semantike je očigledna, s obzirom na činjenicu se on bazira na S-formulama definisanim nad apstraktnim skupom stanja, a ne na nekoj interpretaciji tog skupa. Zbog toga je ovaj račun i dobio naziv S (početno slovo engleske reči *state*). Cilj ovog izlaganja je da prikaže razvoj S-računa i istovremeno da argumentuje i sledećih šest stavova:

1.) S-račun koristi apstraktni skup stanja i predstavlja opštiji alat za opisivanje programske semantike.

2.) Horove formule totalne i parcijalne korektnosti jesu dve specijalne S-formule.

3.) S-račun se zasniva na aksiomama i teoremama predikatske logike prvog reda, a naredbu dodele i specijalne sintaksne jedinice (*if-then*, *if-then-else*, *while* itd.) razmatra kroz definisanje odgovarajućih S-formula, pa S-računu nisu potrebni ni aksioma dodele ni pravila o specijalnim sintaksnim jedinicama iz Horove logike.

4.) S-račun otvara mogućnost da se deklaracija programske promenljive razmatra kao specijalna sintaksna jedinica, odnosno kroz definisanje odgovarajuće S-formule.

5.) Opšta pravila Horove logike jesu teoreme S-računa i mogu se izvesti samo uz korišćenje aksioma i teorema predikatske logike prvog reda.

6.) Dokazivanje u S-računu je jednostavno (potrebno i dovoljno je znati predikatsku logiku prvog reda), a ujedno, S-račun omogućava i to da se korektnost programa i nove teoreme S-računa mogu dokazivati primenom nekog od automatskih dokazivača.

Aksiomatski sistem S-računa čine aksiome predikatske logike prvog reda [45] [47], a svaka teorema predikatske logike prvog reda jeste teorema S-računa i obrnuto. U poglavlju 1.2 ćemo prikazati osnovne komponente, aksiome i neke od osnovnih teorema S-računa.

Horova logika obuhvata formule totalne i parcijalne korektnosti, aksiomu dodele i brojna pravila [3] [35]. Horove formule totalne i parcijalne korektnosti se obično označavaju oznakama $\{P\}S\{Q\}$ i $P\{S\}Q$, pri čemu je njihovo značenje definisano u opisnoj formi. Nasuprot tome, S-račun za definisanje značenja formula totalne i parcijalne korektnosti koristi strogu matematičku formu. U poglavlju 1.2 ćemo pokazati da su Horove formule totalne i parcijalne korektnosti dve specijalne S-formule.

U poglavlju 1.3 ćemo pokazati da opšta pravila Horove logike jesu teoreme S-računa i da se mogu izvesti korišćenjem samo aksioma i teorema predikatske logike prvog reda. Na taj način doći ćemo do zaključka da je Horova logika specijalan slučaj, odnosno uprošćenje S-računa, a sâmim tim i specijalan slučaj, odnosno uprošćenje predikatske logike prvog reda. Poglavlje 1.4 razmatra vezu između formula totalne i parcijalne korektnosti.

Horova aksioma dodele i pravila o specijalnim sintaksnim jedinicama nisu potrebni u S-računu. U S-računu dodela i ostale specijalne sintaksne jedinice se razmatraju kroz definisanje odgovarajućih S-formula, tačnije definisanjem odgovarajućih S-relacija, gde termin „S-relacija" označava binarnu relaciju na apstraktnom skupu stanja. Na primer, naredba dodele $a := e;$ jeste interpretacija odgovarajuće S-relacije $S_{a:=e;}$ koju uvodimo

definicijom. U poglavlju 1.5 ovog rada biće prikazane definicije S-relacija čije interpretacije su naredbe: *no-operation*, dodela, *if-then-else*, *if-then*, *while*, sekvenca, *abort* and *random*. Determinizam se u S-računu razmatra kao poseban slučaj indeterminizma, pa će se prilikom analize *while* petlje posmatrati potencijalno i garantovano terminiranje. Terminiranje, determinizam i indeterminizam u S-računu detaljnije ćemo analizirati kroz nekoliko primera. Pored toga, u poglavlju 1.5 ćemo prikazati definiciju S-relacije čija interpretacija jeste deklaracija programske promenljive. Zbog svoje opštosti, S-račun otvara mogućnost da se deklaracija programske promenljive može razmatrati kao specijalna sintaksna jedinica, što predstavlja problem teorijama koje se bave analizom semantike programa, a koriste interpretirani skup stanja [15] [23] [34] [35] [95]. Mogućnost razmatranja deklaracije programske promenljive je, pre svega, značajno za analizu semantike programa napisanih u programskim jezicima kod kojih deklaracija programske promenljive jeste naredba (npr. u Javi [30] [98]), jer otvara mogućnost za automatsko dokazivanje korektnosti programa [82].

Dajkstra je formulisao četiri teoreme o najširem preduslovu wp, a to su redom zakon isključenja čuda, zakon monotonosti, zakon konjunkcije i zakon disjunkcije wp. Sve četiri teoreme Dajkstra je dokazao na način, za koji se sa matematičkog gledišta, ne može reći da je strogo formalan [27] [33]. U poglavlju 1.6, kao primer primene S-računa, dokazaćemo na strogo formalan način sve četiri Dajkstrine teoreme o najširem postuslovu wp. Pored toga, uz pomenute četiri teoreme formulisaćemo i dokazaćemo novu, petu teoremu, a to je zakon negacije wp. Na kraju ćemo prikazati još i strogo formalan dokaz Dajkstrine teoreme o totalnoj korektnosti. Dajkstra je 1975. godine uveo predikatski operator wp [26] i definisao ga na interpretiranom skupu stanja, čime je otvoren problem egzistencije wp. Pošto S-račun koristi apstraktni skup stanja i strogo razlikuje potencijalno i garantovano terminiranje, time omogućava da wp, kao jedan S-predikat, uvek postoji.

Cilj ovog izlaganja nije da umanji značaj Horove logike, već naprotiv, pravi cilj ovog izlaganja jeste da uopšti i pokaže univerzalnost i primenjivost Horovih ideja na apstraktnom skupu stanja. S-račun treba shvatiti kao matematički most koji na strogo formalan način spaja Horove ideje sa klasičnom predikatskom logikom prvog reda. Spajanje Horovih ideja sa predikatskom logikom prvog reda ima veliki značaj. U takvom spoju Horova logika predstavlja pogodan mehanizam za opisivanje semantike programa, dok se u njenoj pozadini nalazi predikatska logika prvog reda sa moćnim matematičkim aparatom za dokazivanje. Shodno tome, dokazivanje korektnosti programa [24] [8], kao i dokazivanje nove teoreme u S-računu se svodi na dokazivanje valjanosti odgovarajuće S-formule. Na osnovu toga, možemo zaključiti da nam je za dokazivanje korektnosti programa ili nove teoreme S-računa dovoljan skroman matematički aparat, odnosno, da će za dokazivanje biti potrebno i dovoljno poznavanje aksioma, teorema i postupaka dokazivanja u predikatskoj logici prvog reda [16] [21] [48] [94] [73]. S-račun dobija poseban značaj, ako se uzme u obzir i to da se za dokazivanje valjanosti S-formula mogu koristiti razni automatski dokazivači, pa ćemo neke od teorema dokazati pomoću automatskog dokazivača `Coq` [11] [101].

1.2 Osnovne komponente S-programskog računa

Osnovne komponente S-računa su:

1.) Skup apstraktnih stanja (apstraktni skup stanja) A,

2.) Promenljive stanja (S-promenljive) $x, y, z, \ldots,$

3.) Konstante stanja (S-konstante) $s_1, s_2, s_3, \ldots,$

4.) n-arne S-formule $F_1, F_2, F_3, \ldots,$

5.) 1-arne S-formule ili S-predikati $P, Q, R, \ldots,$

6.) 2-arne S-formule ili S-relacije $S_1, S_2, S_3, \ldots,$

7.) Programske promenljive $a, b, c, \ldots,$

8.) Programske konstante $c_1, c_2, c_3, \ldots,$

9.) Skup logičkih operacija $\{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$, gde je $\neg$ negacija, $\wedge$ konjunkcija, $\vee$ disjunkcija, $\Rightarrow$ imlikacija i $\Leftrightarrow$ ekvivalencija,

10.) Skup kvantifikatora $\{\forall, \exists\}$, gde je $\forall$ univerzalni, a $\exists$ egzistencijalni kvantifikator,

11.) Skup logičkih konstanti $\{\top, \bot\}$, gde $\top$ predstavlja *tačno*, a $\bot$ predstavlja *netačno*,

12.) Zagrade ( ) i [ ] koristimo za promenu prioriteta operacija.

Svaka S-konstanta opisuje neko apstraktno stanje virtuelne mašine. Apstraktni skup stanja A je skup svih S-konstanti. S-predikati su logičke funkcije na apstraktnom skupu stanja, tj. $P : A \to \{\top, \bot\}$. Uvodimo dva posebna S-predikata τ and ϕ na ovaj način

$(TAU) \qquad \forall x \in A,\ \tau(x) = \top,$
$(PHI) \qquad \forall x \in A,\ \phi(x) = \bot.$

S-relacije su binarne relacije na apstraktnom skupu stanja, tj. $S \subseteq A \times A$. Drugim rečima, S-relacije su logičke funkcije na skupu $A \times A$, tj. $S : A \times A \to \{\top, \bot\}$. Logičke konstante $\{\top, \bot\}$ ili n-arne S-formule (tj. logičke funkcije $F : \underbrace{A \times A \times \cdots \times A}_{n} \to \{\top, \bot\}$) jesu atomičke S-formule.

Definicija 1.2.1 *S-formule se dobijaju na sledeći način:*

a.) Svaka atomička S-formula je S-formula.

b.) Ako su F_1 i F_2 S-formule, tada su $\neg F_1, F_1 \wedge F_2, F_1 \vee F_2, F_1 \Rightarrow F_2, F_1 \Leftrightarrow F_2$ takođe S-formule.

c.) Svaka formula dobijena konačnom primenom koraka pod a.) i b.) jeste S-formula.

Neka je $\{v_1, v_2, \ldots, v_n\}$ skup programskih promenljivih, koje redom uzimaju vrednost iz skupova $D_1, D_2, \ldots, D_n$. Neka je A' podskup skupa A, koji ima kardinalitet $Card(A') = Card(D_1 \times D_2 \times \cdots \times D_n)$. Interpretacija skupa A u odnosu na skup $\{v_1, v_2, \ldots, v_n\}$ je bijekcija koja svakoj S-konstanti iz skupa A' dodeljuje odgovarajući vektor programskih konstanti iz skupova $D_1, D_2, \ldots, D_n$ (tzv. *vektor stanja programa*). S-relacija $S(x, y)$ sadrži uređene parove (x, y), gde je $x \in A$ početno stanje, a $y \in A$ konačno stanje. Interpretirana restrikcija S-relacije na skupu A' se naziva sintaksna jedinica u kojoj figuriraju programske promenljive $\{v_1, v_2, \ldots, v_n\}$. Sintaksna jedinica može biti zapisana na razne načine, a jedan od najpoznatijih načina je svakako programski kôd. Sintaksna jedinica može biti naredba, blok naredbi, potprogram ili program. Termin „predikat" ćemo koristiti za interpretiranu restrikciju S-predikata na skupu A', koji zapravo jeste logički izraz u kojem figuriraju programske promenljive $\{v_1, v_2, \ldots, v_n\}$. To znači da ćemo razmatrati dva odvojena domena: domen apstraktnih stanja, u kojem postoje S-konstane, S-promenljive, S-predikati i S-relacije i domen interpretacije, u kojem postoje vektori programskih konstanti, programske promenljive, predikati i sintaksne jedinice. Jednostavnije rečeno, S-konstantu interpretiramo odgovarajućim vektorom programskih konstanti iz skupa $D_1, D_2, \ldots, D_n$, S-predikate interpretiramo kao logičke izraze, a S-relacije interpretiramo kao sintaksne jedinice u kojima figuriraju programske promenljive $\{v_1, v_2, \ldots, v_n\}$. Interpretaciju ćemo označiti sa „:". Na primer, $x : a > 0 \wedge b = 5$ znači da S-promenljiva x predstavlja sva stanja u kojima programske promenljive a i b zadovoljavaju uslove $a > 0$ i $b = 5$.

Simbol $\leftrightarrow$ ćemo koristiti kao $\alpha \leftrightarrow F$, gde je α neka oznaka, a F neka S-formula, pa će $\alpha \leftrightarrow F$ značiti da je oznaka α skraćeni zapis za S-formulu F. Ako su F_1 i F_2 dve S-formule identične kao nizovi simbola, tada ćemo reći da su one sintaksno jednake i pisati $F_1 = F_2$. Ako pak F_1 i F_2 imaju isto značenje, a nisu identične kao nizovi simbola, tada ćemo reći da su one semantički ekvivalentne i pisati $F_1 \equiv F_2$.

S-račun koristi S-formule, a zasniva se jedino na aksiomama i teoremama predikatske logike prvog reda. Horove formule $\{P\}S\{Q\}$ i $P\{S\}Q$ posmatramo kao dve specijalne S-formule, koje uvodimo na ovaj način:

- Formula totalne korektnosti (FTK):

$$\forall x[P(x) \Rightarrow (\exists y S(x, y) \wedge \forall z(S(x, z) \Rightarrow Q(z)))] \ .$$

- Formula parcijalne korektnosti (FPK):

$$\forall x[(P(x) \wedge \exists y S(x, y)) \Rightarrow \forall z(S(x, z) \Rightarrow Q(z))] \ .$$

Prilikom pisanja S-formula pridržavaćemo se konvencije iz predikatske logike prvog reda, po kojoj prioriteti logičkih operacija od najvećeg do najmanjeg su redom: negacija $\neg$, konjunkcija $\wedge$, disjunkcija $\vee$, implikacija $\Rightarrow$ i ekvivalencija $\Leftrightarrow$, a zagradama ( ) ili [ ] se taj prioritet može promeniti.

Koristeći S-formule (FTK) i (FPK) možemo precizno definisati totalnu i parcijalnu korektnost S-relacije S u odnosu na S-predikate:

Definicija 1.2.2 *S-relacija S je totalno korektna u odnosu na preduslov P i postuslov Q akko je S-formula $\forall x[P(x) \Rightarrow (\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow Q(z)))]$ valjana.*

Definicija 1.2.3 *S-relacija S je parcijalno korektna u odnosu na preduslov P i postuslov Q akko je S-formula $\forall x[(P(x) \wedge \exists y S(x,y)) \Rightarrow \forall z(S(x,z) \Rightarrow Q(z))]$ valjana.*

Horova formula totalne korektnosti, u oznaci $\{P\}S\{Q\}$, definiše se rečenicom:

- ako sintaksna jedinica S započinje izvršavanje u stanju koje zadovoljava predikat P, tada ona terminira u stanju koje zadovoljava predikat Q [35].

Veza između ove rečenice i S-formule (FTK) je očigledna – ako za svako stanje x važi S-predikat P, tada važi S-formula $\forall x \exists y S(x,y) \wedge \forall x \forall z(S(x,z) \Rightarrow Q(z))$. Za stanje x tada kažemo da je to početno stanje. Smisao S-formule $\forall x \exists y S(x,y)$ je u tome da za svako početno stanje x postoji stanje y takvo da za uređen par stanja (x,y) važi $(x,y) \in S$ i tada za stanje y kažemo da je to završno stanje. Smisao S-formule $\forall x \forall z(S(x,z) \Rightarrow Q(z))$ je u tome da ako za svako početno stanje x i svako završno stanje z važi $(x,z) \in S$, tada u završnom stanju z važi S-predikat Q.

Horova formula parcijalne korektnosti, u oznaci $P\{S\}Q$, definiše se rečenicom:

- ako sintaksna jedinica S započinje izvršavanje u stanju koje zadovoljava predikat P i ako ona terminira, tada završno stanje zadovoljava predikat Q [42][35].

U S-računu ćemo reći na ovaj način – ako je u stanju x tačan predikat P i ako postoji završno stanje y, takvo da važi $(x,y) \in S$, tada je formula $\forall x \forall z(S(x,z) \Rightarrow Q(z))$ valjana.

Kada je reč o pitanju indeterminizma, u S-računu nisu potrebne nikakve dodatne formule, kao što su formule anđeoske i demonske korektnosti [7], jer formule (FTK) i (FPK) sadrže $\forall x \forall z(S(x,z) \Rightarrow Q(z))$. Na taj način, S-račun u sebi sadrži formalno implementiranu Dajkstrinu ideju: „Eventually I came to regard nondeterminacy as the normal situation, determinacy being reduced to a - not even very interesting - special case" [27].

S-račun je posebna vrsta predikatske logike prvog reda, odnosno možemo reći da je to račun predikatske logike prvog reda nad S-formulama. Aksiomatski sistem S-računa čine aksiome predikatske logike prvog reda, s tom napomenom da u aksiomama S-računa

sada figuriraju S-formule F, G i H:

(A_1) $F \Rightarrow (G \Rightarrow F)$
(A_2) $(F \Rightarrow (G \Rightarrow H)) \Rightarrow ((F \Rightarrow G) \Rightarrow (F \Rightarrow H))$
(A_3) $(\neg F \Rightarrow \neg G) \Rightarrow (G \Rightarrow F)$
(A_4) $\forall x F(x) \Rightarrow F(t)$ (term t je slobodan za x u $F(t)$)
(A_5) $\forall x(F \Rightarrow G) \Rightarrow (F \Rightarrow \forall x G)$ (promenljiva x nije slobodna u F)

S-račun koristi pravila izvođenja iz predikatske logike prvog reda, u kojima sada figuriraju S-formule F i G:

a.) Modus ponens (MPN):

$$\frac{F, F \Rightarrow G}{G}$$

b.) Generalizacija (GEN) :

$$\frac{F}{\forall x F}$$

Sve teoreme, odnosno valjane formule predikatske logike prvog reda su istovremeno teoreme S-računa i obrnuto. Ovde ćemo ukratko navesti samo neke od već dokazanih teorema predikatske logike koje će nam trebati za dalja izvođenja dokaza u ovom radu, s tim da u njima sada figuriraju S-formule F, G, H i K i već pomenuti S-predikati τ i ϕ (koji su redom definisani u (TAU) i (FI)):

(T_1) $\forall x \forall y F \Leftrightarrow \forall y \forall x F$
(T_2) $\exists x \forall y F \Rightarrow \forall y \exists x F$
(T_3) $\forall x F \Leftrightarrow F$
(T_4) $\forall x(F \wedge G) \Leftrightarrow \forall x F \wedge \forall x G$
(T_5) $\forall x F \vee \forall x G \Rightarrow \forall x(F \vee G)$
(T_6) $\neg \forall x F \Leftrightarrow \exists x \neg F$
(T_7) $\forall x F \Leftrightarrow \forall x(F \Leftrightarrow \tau)$
(T_8) $\forall x(\tau \Rightarrow F) \Leftrightarrow \forall x F$
(T_9) $\forall x \neg F \Leftrightarrow \forall x(F \Leftrightarrow \phi)$
(T_{10}) $\forall x(F \Leftrightarrow F \wedge F)$
(T_{11}) $\forall x(F \Leftrightarrow F \vee G)$
(T_{12}) $\forall x(\neg F \vee \neg G) \Leftrightarrow \forall x \neg(F \wedge G)$
(T_{13}) $\forall x(\neg F \wedge \neg G) \Leftrightarrow \forall x \neg(F \vee G)$
(T_{14}) $\forall x(F \Rightarrow G) \Rightarrow (\forall x F \Rightarrow \forall x G)$
(T_{15}) $\forall x(F \Rightarrow G) \Leftrightarrow \forall x(\neg F \vee G)$
(T_{16}) $\forall x[(F \Rightarrow H) \wedge (H \Rightarrow G)] \Rightarrow \forall x(F \Rightarrow G)$
(T_{17}) $\forall x[(F \Rightarrow G) \wedge (H \Rightarrow K)] \Rightarrow \forall x[(F \vee H) \Rightarrow (G \vee K)]$
(T_{18}) $\forall x[(F \Rightarrow G) \wedge (H \Rightarrow K)] \Rightarrow \forall x[(F \wedge H) \Rightarrow (G \wedge K)]$
(T_{19}) $\forall x[(F \Rightarrow G) \wedge (F \Rightarrow H)] \Leftrightarrow \forall x(F \Rightarrow G \wedge H)$

(T_{20}) $\forall x[(F \Rightarrow G) \wedge (F \Rightarrow H)] \Leftrightarrow \forall x(F \Rightarrow G \vee H)$
(T_{21}) $\forall x[(F \Rightarrow H) \wedge (G \Rightarrow H)] \Leftrightarrow \forall x(F \vee G \Rightarrow H)$
(T_{22}) $\forall x[(F \Rightarrow G) \vee (H \Rightarrow K)] \Rightarrow \forall x[(F \wedge H) \Rightarrow (G \vee K)]$

Dokazivanje korektnosti programa ili nove teoreme S-računa svodi se na dokazivanje valjanosti odgovarajuće S-formule za šta nam je dovoljan skroman matematički aparat, odnosno, za dokazivanje u S-računu dovoljno je poznavati aksiome, teoreme i postupke dokazivanja u predikatskoj logici prvog reda. Sada ćemo dokazati jednu teoremu, koja predstavlja alternativni oblik formule totalne korektnosti (FTK) i koja će nam biti od velike koristi za pojednostavljenje pojedinih dokaza koji slede u dalje izlaganju.

Teorema 1.2.4 (Alternativni oblik (FTK)**)** *Sledeća S-formula je validna:*

$$\{P\}S\{Q\} \Leftrightarrow \forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow Q(z)].$$

Dokaz.

Pošto je:
$\{P\}S\{Q\} \leftrightarrow \forall x[P(x) \Rightarrow (\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow Q(z)))]$,
na osnovu teoreme (T_{19})*, desna strana ekvivalencije postaje:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \Rightarrow (S(x,z) \Rightarrow Q(z))]$,
a na osnovu teoreme (T_{15})*, postaje:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \Rightarrow (\neg S(x,z) \vee Q(z))]$
$\equiv \forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[\neg P(x) \vee (\neg S(x,z) \vee Q(z))]$.
Zatim, na osnovu teoreme (T_{12})*, dobijamo:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[\neg(P(x) \wedge S(x,z)) \vee Q(z)]$
i konačno, na osnovu teoreme (T_{19})*, dobijamo:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow Q(z)]$.

□

Vezivanje S-računa za apstraktni skup stanja A predstavlja važan detalj u ovom izlaganju, jer dolazimo do zaključka da za upotrebu Horove logike nije nužno raspolagati jednoznačnim opisom svakog apstraktnog stanja ponaosob. Odatle možemo zaključiti da nema potrebe za uvođenjem vektora stanja programa (vektora vrednosti svih programskih promenljivih u programu). Poznato je da vektor stanja programa unosi poteškoće u modelovanje, jer je nejasno kako ga tumačiti kada neke od programskih promenljivih nisu definisane [27]. Pored toga, vektor stanja je vezan isključivo za dati program i ni na koji način ne može se vezati za skup stanja virtuelne mašine kada program nije aktivan. Dodatne probleme unose potprogrami sa sopstvenim vektorima stanja koji postoje samo dok je potprogram aktivan. S druge strane, apstraktni skup stanja A se odnosi na neku virtuelnu mašinu, tako da je aktuelan u svakom trenutku, bez obzira na to da li je posmatrani program aktivan ili nije. U S-računu apstraktni skup stanja nekog programa A' jeste

podskup apstraktnog skupa stanja A, odnosno $A' \subseteq A$, a sâm program jeste restrikcija na skupu A' odgovarajuće S-relacije za koju važi $S \subseteq A \times A$.

1.3 Opšti zakoni Horove logike

U ovom poglavlju ćemo razmotriti opšte zakone Horove logike [3] [35], kao što su zakoni konsekvencije, konjunkcije, disjunkcije i negacije. Opšti zakoni su u Horovoj logici dati u vidu pravila, a u ovom izlaganju mi ih razmatramo kao teoreme S-računa. Pojedine teoreme ćemo dokazati i pomoću automatskog dokazivača `Coq`.

Teorema 1.3.1 (Zakoni konsekvencije) *Sledeće S-formule su valjane:*

a.) $\forall x(P(x) \Rightarrow R(x)) \wedge \{R\}S\{Q\} \;\Rightarrow\; \{P\}S\{Q\}$,

b.) $\{P\}S\{R\} \wedge \forall z(R(z) \Rightarrow Q(z)) \;\Rightarrow\; \{P\}S\{Q\}$,

c.) $\forall x(U(x) \Rightarrow P(x)) \wedge \forall z(Q(z) \Rightarrow V(z)) \wedge \{P\}S\{Q\} \;\Rightarrow\; \{U\}S\{V\}$.

Dokaz.

a.) Pošto je:
$\{R\}S\{Q\} \;\leftrightarrow\; \forall x[R(x) \Rightarrow (\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow Q(z)))]$,
leva strana implikacije se može napisati ovako:
$\forall x(P(x) \Rightarrow R(x)) \wedge \forall x[R(x) \Rightarrow (\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow Q(z)))]$
i na osnovu teoreme (T_{16})*, dobijamo:*
$\forall x[P(x) \Rightarrow (\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow Q(z)))]$,
tj.
$\{P\}S\{Q\}$.

b.) Na osnovu teoreme 1.2.4, leva strana implikacije može se napisati ovako:
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow R(z)] \;\wedge\; \forall z(R(z) \Rightarrow Q(z))$
i na osnovu teoreme (T_{16})*, dobijamo:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow Q(z)]$,
tj.
$\{P\}S\{Q\}$.

c.) Na osnovu teoreme 1.3.1.a.), leva strana implikacije se može napisati ovako:
$\forall z(Q(z) \Rightarrow V(z)) \;\wedge\; \{U\}S\{Q\}$
i na osnovu teoreme 1.3.1.b.), dobijamo:
$\{U\}S\{V\}$.

□

Teoreme 1.3.1 se mogu dokazati automatskim dokazivačem `Coq`, što je dato u prilogu A. Najzad, na osnovu Teoreme (T_3), iz teorema 1.3.1 dobijamo poznata Horova pravila konsekvencije [3] [35]:

$$\frac{(P \Rightarrow R), \{R\}S\{Q\}}{\{P\}S\{Q\}}, \qquad \frac{\{P\}S\{R\}, (R \Rightarrow Q)}{\{P\}S\{Q\}} \quad \text{i} \quad \frac{(U \Rightarrow P), (Q \Rightarrow V), \{P\}S\{Q\}}{\{U\}S\{V\}}.$$

Teorema 1.3.2 (Zakoni konjunkcije) *Sledeće S-formule su valjane:*

a.) $\{P\}S\{Q\} \wedge \{R\}S\{W\} \;\Rightarrow\; \{P \vee R\}S\{Q \vee W\}$,

b.) $\{P\}S\{Q\} \wedge \{R\}S\{W\} \;\Rightarrow\; \{P \wedge R\}S\{Q \wedge W\}$.

Dokaz.

a.) Na osnovu teoreme 1.2.4, leva strana implikacije se može napisati ovako:
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow Q(z)] \;\wedge\; \forall x[R(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[R(x) \wedge S(x,z) \Rightarrow W(z)]$.
Na osnovu teoreme (T_{17})*, dobijamo:*
$\forall x[(P(x) \vee R(x)) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[((P(x) \wedge S(x,z)) \vee (R(x) \wedge S(x,z))) \Rightarrow (Q(z) \vee W(z))]$
$\equiv \forall x[(P(x) \vee R(x)) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[((P(x) \vee R(x)) \wedge (S(x,z) \vee S(x,z))) \Rightarrow (Q(z) \vee W(z))]$
$\equiv \forall x[(P(x) \vee R(x)) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[(P(x) \vee R(x)) \wedge S(x,z) \Rightarrow (Q(z) \vee W(z))]$,
tj.
$\{P \vee R\}S\{Q \vee W\}$.

b.) Na osnovu teoreme 1.2.4, leva strana implikacije se može napisati ovako:
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow Q(z)] \;\wedge\; \forall x[R(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[R(x) \wedge S(x,z) \Rightarrow W(z)]$.
Na osnovu teoreme (T_{18})*, dobijamo:*
$\forall x[(P(x) \wedge R(x)) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[((P(x) \wedge S(x,z)) \wedge (R(x) \wedge S(x,z))) \Rightarrow (Q(z) \wedge W(z))]$
$\equiv \forall x[(P(x) \wedge R(x)) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[(P(x) \wedge R(x)) \wedge S(x,z) \Rightarrow (Q(z) \wedge W(z))]$,
tj.
$\{P \wedge R\}S\{Q \wedge W\}$.

□

Posledica 1.3.3 (Zakon rezolucije) *Sledeća S-formula je valjana:*

$$\{P\}S\{Q\} \wedge \{\neg P\}S\{W\} \;\Rightarrow\; \{\tau\}S\{Q \vee W\}.$$

Dokaz.

Ako u teoremi 1.3.2.a.) zamenimo R sa $\neg P$ dobijamo:
$\{P\}S\{Q\} \wedge \{\neg P\}S\{W\} \;\Rightarrow\; \{P \vee \neg P\}S\{Q \vee W\}$,
tj.
$\{P\}S\{Q\} \wedge \{\neg P\}S\{W\} \;\Rightarrow\; \{\tau\}S\{Q \vee W\}$.

□

Teorema 1.3.4 (Zakoni disjunkcije) *Sledeća S-formula je valjana:*

$$\{P\}S\{Q\} \vee \{R\}S\{W\} \;\Rightarrow\; \{P \wedge R\}S\{Q \vee W\}.$$

Dokaz.

Na osnovu teoreme 1.2.4, leva strana implikacije se može napisati ovako:
$(\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow Q(z)]) \;\vee\; (\forall x[R(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[R(x) \wedge S(x,z) \Rightarrow W(z)])$
$\equiv\; (\forall x[P(x) \;\Rightarrow\; \exists y S(x,y)] \vee \forall x[R(x) \;\Rightarrow\; \exists y S(x,y)]) \;\wedge\; (\forall x \forall z[P(x) \wedge S(x,z) \;\Rightarrow\; Q(z)] \vee \forall x \forall z[R(x) \wedge S(x,z) \Rightarrow W(z)])$.
Na osnovu teoreme (T_{22}), dobijamo:
$\forall x[(P(x) \wedge R(x)) \Rightarrow \exists y S(x,y)] \;\wedge\; \forall x \forall z[(P(x) \wedge S(x,z)) \wedge (R(x) \wedge S(x,z)) \Rightarrow (Q(z) \vee W(z))]$
$\equiv \forall x[(P(x) \wedge R(x)) \Rightarrow \exists y S(x,y)] \;\wedge\; \forall x \forall z[(P(x) \wedge R(x)) \wedge S(x,z) \Rightarrow (Q(z) \vee W(z))]$,
tj.
$\{P \wedge R\}S\{Q \vee W\}$.

□

Teoreme 1.3.2.a), 1.3.2.b) i 1.3.4 se mogu dokazati automatskim dokazivačem `Coq`, što je dato u prilogu A.

Teorema 1.3.5 (Zakoni konjunkcije i disjunkcije) *Sledeće S-formule su valjane:*

a.) $\{P \vee R\}S\{Q\} \;\Leftrightarrow\; \{P\}S\{Q\} \wedge \{R\}S\{Q\}$,

b.) $\{P\}S\{Q \wedge R\} \;\Leftrightarrow\; \{P\}S\{Q\} \wedge \{P\}S\{R\}$,

c.) $\{P \vee U\}S\{Q \wedge W\} \;\Leftrightarrow\; \{P\}S\{Q\} \wedge \{U\}S\{W\} \wedge \{P\}S\{W\} \wedge \{U\}S\{Q\}$,

d.) $\{P\}S\{Q\} \vee \{P\}S\{W\} \;\Rightarrow\; \{P\}S\{Q \vee W\}$.

Dokaz.

a.) Leva strana ekvivalencije se može napisati ovako:
$\forall x[(P(x) \vee R(x)) \Rightarrow \exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow Q(z))]$

i na osnovu teoreme (T_{21})*, dobijamo:*
$\forall x[(P(x) \Rightarrow \exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow Q(z))) \wedge (R(x) \Rightarrow \exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow Q(z)))]$,
tj.
$\{P\}S\{Q\} \wedge \{R\}S\{Q\}$.

b.) Na osnovu teoreme 1.2.4, leva strana ekvivalencije se može napisati ovako:
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow Q(z) \wedge R(z)]$
i na osnovu teoreme (T_{18})*, dobijamo:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow Q(z)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow R(z)]$
$\equiv \forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow Q(z)] \wedge \forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge$
$\forall x \forall z[P(x) \wedge S(x,z) \Rightarrow R(z)]$,
tj.
$\{P\}S\{Q\} \wedge \{P\}S\{R\}$.

c.) Na osnovu teoreme 1.3.5.a.) i leve strane ekvivalencije dobijamo:
$\{P \vee U\}S\{Q \wedge W\} \Leftrightarrow \{P\}S\{Q \wedge W\} \wedge \{U\}S\{Q \wedge W\}$
i na osnovu teoreme 1.3.5.b.), dobijamo:
$\{P\}S\{Q \wedge W\} \wedge \{U\}S\{Q \wedge W\} \Leftrightarrow \{P\}S\{Q\} \wedge \{U\}S\{W\} \wedge \{P\}S\{W\} \wedge \{U\}S\{Q\}$.

d.) Ako u teoremi 1.3.4 zamenimo R sa P dobijamo:
$\{P\}S\{Q\} \vee \{P\}S\{W\} \Rightarrow \{P \wedge P\}S\{Q \vee W\}$
i na osnovu teoreme (T_{10})*, dobijamo:*
$\{P\}S\{Q\} \vee \{P\}S\{W\} \Rightarrow \{P\}S\{Q \vee W\}$.

□

Teorema 1.3.6 (Opšti zakon isključenja čuda) *Sledeća S-formula je valjana:*

$\{P\}S\{\phi\} \Leftrightarrow (P \Leftrightarrow \phi)$, *i.e.* $\{P\}S\{\phi\} \Leftrightarrow \neg P$.

Dokaz.

Leva strana ekvivalencije se može napisati ovako:
$\forall x[P(x) \Rightarrow (\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \phi(z)))]$.
Pošto je S-formula $\forall x \forall z(S(x,z) \Rightarrow \phi(z))$ *valjana akko je* $\forall x \forall z \neg S(x,z)$ *valjana, dobijamo:*
$\forall x[P(x) \Rightarrow (\exists y S(x,y) \wedge \forall z \neg S(x,z))] \equiv \ \forall x[P(x) \Rightarrow \phi(x)]$
i na osnovu teoreme (T_9)*, dobijamo:*
$\forall x \neg P(x)$,
tj.
$\neg P$.

□

Teorema 1.3.7 (Zakoni negacije) *Sledeće S-formule su valjane:*

a.) $\{P\}S\{Q\} \wedge \{R\}S\{\neg Q\} \;\Rightarrow\; \neg(P \wedge R)$,

b.) $\{P\}S\{Q\} \wedge \{P\}S\{\neg Q\} \;\Leftrightarrow\; \forall x \neg P(x)$,

c.) $[\{P\}S\{\neg Q\} \Rightarrow \neg\{P\}S\{Q\}] \;\Leftrightarrow\; \exists x P(x)$,

d.) $\{P\}S\{Q\} \wedge \{\neg P\}S\{Q\} \;\Leftrightarrow\; \forall x \exists y S(x,y) \wedge \forall x \forall z(S(x,z) \Rightarrow Q(z))$,

e.) $\exists x \exists z S(x,z) \wedge \neg Q(z) \;\Rightarrow\; [\{\neg P\}S\{Q\} \Rightarrow \neg\{P\}S\{Q\}]$.

Dokaz.

a.) *Ako u teoremi 1.3.2.b.) zamenimo* W *sa* $\neg Q$*, dobijamo:*
$\{P\}S\{Q\} \wedge \{R\}S\{\neg Q\} \Rightarrow \{P \wedge R\}S\{Q \wedge \neg Q\}$
$\equiv \; \{P\}S\{Q\} \wedge \{R\}S\{\neg Q\} \Rightarrow \{P \wedge R\}S\{\phi\}$
i na osnovu teoreme 1.3.6, dobijamo:
$\{P\}S\{Q\} \wedge \{R\}S\{\neg Q\} \Rightarrow \neg(P \wedge R)$.

b.) *Ako u teoremi 1.3.5.b.) zamenimo* R *sa* $\neg Q$*, dobijamo:*
$\{P\}S\{Q\} \wedge \{P\}S\{\neg Q\} \Leftrightarrow \{P\}S\{Q \wedge \neg Q\}$
$\equiv \; \{P\}S\{Q\} \wedge \{P\}S\{\neg Q\} \Leftrightarrow \{P\}S\{\phi\}$
i na osnovu teoreme 1.3.6, dobijamo:
$\{P\}S\{Q\} \wedge \{P\}S\{\neg Q\} \Leftrightarrow \neg P$,
tj.
$\{P\}S\{Q\} \wedge \{P\}S\{\neg Q\} \Leftrightarrow \forall x \neg P(x)$.

c.) *Na osnovu teoreme* (T_{15})*, leva strana ekvivalencije postaje:*
$\neg\{P\}S\{\neg Q\} \vee \neg\{P\}S\{Q\}$
a zatim, na osnovu teoreme (T_{12})*, dobijamo:*
$\neg[\{P\}S\{\neg Q\} \wedge \{P\}S\{Q\}]$.
Nakon toga, na osnovu teoreme 1.3.7.b.), dobijamo:
$\neg[\forall x \neg P(x)]$
i najzad, na osnovu teoreme (T_6)*, dobijamo:*
$\exists x P(x)$.

d.) *Ako u teoremi 1.3.5.a.) zamenimo* R *sa* $\neg P$ *dobijamo:*
$\{P \vee \neg P\}S\{Q\}$
$\equiv \; \{\tau\}S\{Q\}$.
Pošto je:
$\{\tau\}S\{Q\} \;\leftrightarrow\; \forall x[\tau(x) \Rightarrow \exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow Q(z))]$,
na osnovu teoreme (T_8)*, dobijamo:*
$\forall x[\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow Q(z))]$.

e.) Na osnovu teoreme (T_{15})*, desna strana ekvivalencije se može napisati ovako:*
$\neg\{\neg P\}S\{Q\} \vee \neg\{P\}S\{Q\}$
i na osnovu teoreme (T_{12})*, dobijamo:*
$\neg[\{\neg P\}S\{Q\} \wedge \{P\}S\{Q\}]$.
Zatim, na osnovu teoreme 1.3.7.d.), dobijamo:
$\neg\forall x[\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow Q(z))]$
$\equiv \exists x\neg[\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow Q(z))]$.
Na osnovu teoreme (T_{13})*, dobijamo:*
$\exists x[\neg\exists y S(x,y) \vee \neg\forall z(S(x,z) \Rightarrow Q(z))]$
$\equiv \exists x[\forall y\neg S(x,y) \vee \exists z\neg(S(x,z) \Rightarrow Q(z))]$
i na osnovu teoreme (T_{15})*, dobijamo:*
$\exists x[\forall y\neg S(x,y) \vee \exists z\neg(\neg S(x,z) \vee Q(z))]$.
Nakon toga, na osnovu teoreme (T_{12})*, dobijamo:*
$\exists x[\forall y\neg S(x,y) \vee \exists z(S(x,z) \wedge \neg Q(z))]$
$\equiv \exists x\forall y\neg S(x,y) \vee \exists x\exists z(S(x,z) \wedge \neg Q(z))$
i najzad, na osnovu teoreme (T_{11})*, dobijamo:*
$\exists x\exists z(S(x,z) \wedge \neg Q(z)) \Rightarrow \exists x\exists z(S(x,z) \wedge \neg Q(z)) \vee \exists x\forall y\neg S(x,y)$.

□

Teoreme 1.3.6, 1.3.7.*a*) i 1.3.7.*b*) se mogu dokazati automatskim dokazivačem `Coq`, što je dato u prilogu A.

Posledica 1.3.8 *Sledeća S-formula je valjana:*

$$\{P\}S\{Q\} \wedge \{P\}S\{\neg Q\} \;\Leftrightarrow\; (P \Leftrightarrow \phi).$$

Dokaz.

Na osnovu teoreme 1.3.7.b.), dobijamo:
$\forall x\neg P(x)$,
tj.
$P \Leftrightarrow \phi$.

□

Posledica 1.3.9 *Sledeća S-formula je valjana:*

$$[\{P\}S\{\neg Q\} \Rightarrow \neg\{P\}S\{Q\}] \;\Leftrightarrow\; \neg(P \Leftrightarrow \phi).$$

Dokaz.

Na osnovu teoreme 1.3.7.c.), dobijamo:
$\exists x P(x)$

$\equiv \neg(\forall x \neg P(x))$,
tj.
$\neg(P \Leftrightarrow \phi)$.

□

1.4 Veza totalne i parcijalne korektnosti

Neka su:
$A \equiv \forall x P(x)$,
$B \equiv \forall x \exists y S(x, y)$,
$C \equiv \forall x \forall z (S(x, z) \Rightarrow Q(z))$.

Sada S-formule (FTK) i (FPK) možemo napisati u ovom obliku:
$\{P\}S\{Q\} \leftrightarrow A \Rightarrow B \wedge C$,
$P\{S\}Q \leftrightarrow A \wedge B \Rightarrow C$.

S-formule $A \Rightarrow B \wedge C$ i $A \wedge B \Rightarrow C$ ćemo napisati u savršenoj konjunktivnoj i savršenoj disjunktivnoj normalnoj formi. Savršena konjunktivna normalna forma izgleda ovako:

$$\begin{aligned}\{P\}S\{Q\} \leftrightarrow\ & (\neg A \vee \neg B \vee C) \wedge (\neg A \vee B \vee \neg C) \wedge (\neg A \vee B \vee C) \\ & \equiv (\neg A \vee \neg B \vee C) \wedge ((\neg A \vee B) \vee \neg C) \wedge ((\neg A \vee B) \vee C) \\ & \equiv (\neg A \vee \neg B \vee C) \wedge ((\neg A \vee B) \vee (\neg C \wedge C)) \\ & \equiv (\neg A \vee \neg B \vee C) \wedge ((\neg A \vee B) \vee \bot) \\ & \equiv (\neg A \vee \neg B \vee C) \wedge (\neg A \vee B) \\ & \equiv (\neg A \vee \neg B \vee C) \wedge (A \Rightarrow B),\end{aligned}$$

$P\{S\}Q \leftrightarrow (\neg A \vee \neg B \vee C)$,

odakle očigledno važi:

$$\{P\}S\{Q\} \Leftrightarrow P\{S\}Q \wedge (A \Rightarrow B),$$

odnosno

$$\boxed{\{P\}S\{Q\} \Leftrightarrow P\{S\}Q \wedge \forall x (P(x) \Rightarrow \exists y S(x, y)).}$$

Savršena disjunktivna normalna forma izgleda ovako:

$$\{P\}S\{Q\} \leftrightarrow (A\wedge B\wedge C)\vee(\neg A\wedge B\wedge C)\vee(\neg A\wedge B\wedge\neg C)\vee(\neg A\wedge\neg B\wedge C)\vee(\neg A\wedge\neg B\wedge\neg C),$$

$$\begin{aligned} P\{S\}Q \leftrightarrow\ & \{P\}S\{Q\}\vee(A\wedge\neg B\wedge C)\vee(A\wedge\neg B\wedge\neg C)\\ & \equiv\{P\}S\{Q\}\vee((A\wedge\neg B)\wedge(C\vee\neg C))\\ & \equiv\{P\}S\{Q\}\vee((A\wedge\neg B)\wedge\top)\\ & \equiv\{P\}S\{Q\}\vee(A\wedge\neg B)\\ & \equiv\{P\}S\{Q\}\vee\neg(\neg A\vee B)\\ & \equiv\{P\}S\{Q\}\vee\neg(A\Rightarrow B), \end{aligned}$$

odakle očigledno važi:

$$P\{S\}Q \Leftrightarrow \{P\}S\{Q\}\vee\neg(A\Rightarrow B),$$

odnosno

$$\boxed{P\{S\}Q \Leftrightarrow \{P\}S\{Q\}\vee\neg\forall x(P(x)\Rightarrow\exists y S(x,y)).}$$

1.5 Specijalne S-relacije

U Horovoj logici, tzv. specijalne sintaksne jedinice, kao što su *if-then*, *if-then-else*, *while* itd. se uvode pravilima, dok se dodela uvodi aksiomom [3] [35]. S-račun sve specijalne sintaksne jedinice razmatra kroz definisanje odgovarajućih S-formula. Drugim rečima, u S-računu za svaku specijalnu sintaksnu jedinicu (npr. dodelu) definišemo odgovarajuću S-relaciju, koja jeste neki podskup skupa $A\times A$, gde je A apstraksni skup stanja. Razmotrimo sada smisao ove ideje za sintaksnu jedinicu dodela. Neka je data programska promenljiva a tipa *integer*. Sintaksna jedinica $a := 5;$ jeste interpretacija S-relacije $S_{a:=5;}$ koja virtuelnu mašinu prebacuje iz stanja x u stanje y, gde stanje x interpretiramo kao stanje u kojem programska promenljiva a ima neku vrednost iz njenog domena $D_{integer}$, tj. $x : a \in D_{integer}$, a stanje y interpretiramo kao stanje u kojem programska promenljiva a ima vrednost 5, tj. $y : a = 5$. Dakle, S-relaciju $S_{a:=5;}$ definišemo kao skup uređenih parova (x, y), $x, y \in A$ za koje važi $x : a \in D_{integer}$ i $y : a = 5$ ili pomoću S-formule $\forall x\forall y S_{a:=5;}(x, y) \Leftrightarrow x : a \in D_{integer} \wedge y : a = 5$.

Neka $x, y, y_1, y_2, \ldots, y_n, z \in A$, gde je A apstraktni skup stanja i neka je a programska promenljiva tipa $Type$. Sada ćemo prikazati definicije specijalnih S-relacija *abort*,

no-operation, dodela, *if-then-else*, *if-then*:

Definicija 1.5.1 (Abort) *S-relacija* S_{abort} *se definiše:*

a.) kao skup $S_{abort} = \emptyset$, *ili*

b.) S-formulom $\forall x \forall y S_{abort}(x, y) \Leftrightarrow \phi(x)$.

Definicija 1.5.2 (No-operation) *S-relacija* S_{nop} *se definiše:*

a.) kao skup $S_{nop} = \{(x, y) | x = y\}$, *ili*

b.) S-formulom $\forall x \forall y S_{nop}(x, y) \Leftrightarrow x = y$.

Definicija 1.5.3 (Dodela) *S-relacija* $S_{a:=e;}$ *se definiše:*

a.) kao skup $S_{a:=e;} = \{(x, y) | x : a \in D_{Type} \wedge y : a = e\}$, *ili*

b.) S-formulom $\forall x \forall y S_{a:=e;}(x, y) \Leftrightarrow x : a \in D_{Type} \wedge y : a = e$.

Definicija 1.5.4 (If-then-else) *S-relacija* $S_{if-then-else}$ *se definiše:*

a.) kao skup $S_{if-then-else} = \{(x, y) | (B(x) \wedge S_1(x, y)) \vee (\neg B(x) \wedge S_2(x, y))\}$, *ili*

b.) S-formulom $\forall x \forall y S_{if-then-else}(x, y) \Leftrightarrow (B(x) \wedge S_1(x, y)) \vee (\neg B(x) \wedge S_2(x, y))$.

Definicija 1.5.5 (If-then) *S-relacija* $S_{if-then}$ *se definiše:*

a.) kao skup $S_{if-then} = \{(x, y) | (B(x) \wedge S(x, y)) \vee (\neg B(x) \wedge S_{nop}(x, y))\}$, *ili*

b.) S-formulom $\forall x \forall y S_{if-then}(x, y) \Leftrightarrow (B(x) \wedge S(x, y)) \vee (\neg B(x) \wedge S_{nop}(x, y))$.

Definicija 1.5.6 (Sekvenca) *S-relacija* $S_{[S_1;S_2]}$ *se definiše:*

a.) kao skup $S_{[S_1;S_2]} = \{(x, y) | \exists z (S_1(x, z) \wedge S_2(z, y))\}$, *ili*

b.) S-formulom $\forall x \forall y S_{[S_1;S_2]}(x, y) \Leftrightarrow \exists z (S_1(x, z) \wedge S_2(z, y))$.

Determinizam u S-računu se posmatra kao specijalan slučaj indeterminizma. Determinističku sintaksnu jedinicu opisuje S-relacija u kojoj za svaka dva uređena para stanja $(x_1, y_1) \in S$ i $(x_2, y_2) \in S$ važi $(x_1 = x_2) \Rightarrow (y_1 = y_2)$. Drugim rečima, S-relacija opisuje indeterminističku sintaksnu jedinicu akko S nije funkcija. Posmatrajmo sada funkciju $random()$ koja predstavlja jednostavan generator slučajnih brojeva iz skupa $\{0, 1\}$. Povratnu vrednost funkcije ćemo označiti sa $retValue$, a stanja y_0 i y_1 ćemo interpretirati sa $y_0 : retValue = 0$ i $y_1 : retValue = 1$.

Definicija 1.5.7 (Random) *S-relacija S_{random} se definiše:*

a.) kao skup $S_{random} = \{(x, y_0), (x, y_1)\}$, *ili*

b.) S-formulom $\forall x \forall y S_{random}(x, y) \Leftrightarrow y : retValue = 0 \vee retValue = 1$.

Razmatranje petlji je nešto komplikovanije, a uzrok tome je činjenica da postoje tri vrste ponašanja *while* petlje:

1.) Petlja ne terminira počev od inicijalnog stanja x.

2.) Petlja potencijalno terminira počev od inicijalnog stanja x (npr. kada terminiranje petlje zavisi od neke indeterminističke sintaksne jedinice koja se nalazi u njenom telu).

3.) Petlja garantovano terminira počev od inicijalnog stanja x.

Definicija 1.5.8 (While petlja – potencijalno terminirajuća)
S-relacija S^P_{while} se definiše:

a.) kao skup $S^P_{while} = \{(x, y)|[\neg B(x) \wedge S_{nop}(x, y)] \vee [B(x) \wedge S(x, y) \wedge \neg B(y)] \vee [\exists y_1 \exists y_2 \ldots \exists y_n B(x) \wedge S(x, y_1) \wedge B(y_1) \wedge S(y_1, y_2) \wedge B(y_2) \wedge S(y_2, y_3) \wedge \cdots \wedge B(y_n) \wedge S(y_n, y) \wedge \neg B(y)]\}$, *ili*

b.) S-formulom $\forall x \forall y S^P_{while}(x, y) \Leftrightarrow [\neg B(x) \wedge S_{nop}(x, y)] \vee [B(x) \wedge S(x, y) \wedge \neg B(y)] \vee [\exists y_1 \exists y_2 \ldots \exists y_n B(x) \wedge S(x, y_1) \wedge B(y_1) \wedge S(y_1, y_2) \wedge B(y_2) \wedge S(y_2, y_3) \wedge \cdots \wedge B(y_n) \wedge S(y_n, y) \wedge \neg B(y)]$.

Definicija 1.5.9 (While petlja – garantovano terminirajuća)
S-relacija S^G_{while} se definiše:

a.) kao skup $S^G_{while} = \{(x, y)|[\neg B(x) \wedge S_{nop}(x, y)] \vee [B(x) \wedge S(x, y) \wedge \neg B(y)] \vee [\forall y_1 \forall y_2 \ldots \forall y_n B(x) \wedge S(x, y_1) \wedge B(y_1) \wedge S(y_1, y_2) \wedge B(y_2) \wedge S(y_2, y_3) \wedge \cdots \wedge B(y_n) \wedge S(y_n, y) \wedge \neg B(y)]\}$, *ili*

b.) S-formulom $\forall x \forall y S^G_{while}(x, y) \Leftrightarrow [\neg B(x) \wedge S_{nop}(x, y)] \vee [B(x) \wedge S(x, y) \wedge \neg B(y)] \vee [\forall y_1 \forall y_2 \ldots \forall y_n B(x) \wedge S(x, y_1) \wedge B(y_1) \wedge S(y_1, y_2) \wedge B(y_2) \wedge S(y_2, y_3) \wedge \cdots \wedge B(y_n) \wedge S(y_n, y) \wedge \neg B(y)]$.

S-formule koje definišu S-relacije S^P_{while} i S^G_{while} u sebi sadrže tri disjunkta. Prvi dusjunkt $\forall x \forall y \neg B(x) \wedge S_{nop}(x, y)$ pokriva slučaj kada se telo petlje uopšte neće izvršiti. Drugi disjunkt $\forall x \forall y B(x) \wedge S(x, y) \wedge \neg B(y)$ pokriva slučaj kada će se telo petlje izvršiti tačno jedanput. Treći disjunkt je $n + 2$-arna S-formula gde je n broj međustanja $y_1, y_2, \ldots, y_n$ kroz koja petlja prolazi. S-formule S^P_{while} i S^G_{while} se razlikuju u trećem disjunktu, u kojem

S^P_{while} sadrži egzistencijalne kvantifikatore (tj. $\exists y_1 \exists y_2 \ldots \exists y_n$) dok S^G_{while} sadrži univerzalne kvantifikatore (tj. $\forall y_1 \forall y_2 \ldots \forall y_n$).

Očigledno, S-relacija S^G_{while} je strožija od S^P_{while}, tj. važi $S^G_{while} \subseteq S^P_{while}$, odnosno $\forall x \forall y S^G_{while}(x, y) \Rightarrow S^P_{while}(x, y)$. Drugim rečima, ako $(x, y) \in S^G_{while}$, tada $(x, y) \in S^P_{while}$, a obrnuto ne važi. Ako sada analiziramo jednu *while* petlju, koristeći obe S-relacije, dobijamo kompletnu sliku o njenom ponašanju:

1.) Ako inicijalo stanje x zadovoljava $\forall z, (x, z) \notin S^P_{while}$ tada petlja ne terminira počev od stanja x.

2.) Ako inicijalo stanje x zadovoljava $\exists z, (x, z) \in S^P_{while} \wedge (x, z) \notin S^G_{while}$ tada petlja potencijalno terminira počev od stanja x.

3.) Ako inicijalo stanje x zadovoljava $\exists z, (x, z) \in S^G_{while}$ tada petlja garantovano terminira počev od stanja x.

Razmatranja petlji koje potencijalno terminiraju i koje ne terminiraju imaju više teorijski značaj, dok za programersku praksu najznačajnije je razmatranje petlji koje garantovano terminiraju. Pored garantovanog terminiranja, važno je da i vremenski interval terminiranja bude prihvatljiv [37]. U Primeru 1.5.3 ćemo pokazati da beskonačnoj petlji odgovara S-relacija koja je prazan skup, što po definiciji 1.5.1 odgovara naredbi *abort*. U Primeru 1.5.6 ćemo razmatrati potencijalno terminirajuću petlju, gde postoji očigledna razlika između S-formula S^P_{while} i S^G_{while}. S-formula S^G_{while} eliminiše sva inicijalna stanja počev od kojih petlja potencijalno terminira i tako obezbeđuje da terminiranje bude garantovano. Takođe, S-formula S^G_{while} eliminiše problem postojanja Dajkstrinog operatora wp i obezbeđuje da u S-računu on uvek postoji, što ćemo ilustrovati u Primeru 1.6.1.

U prethodnom poglavlju dokazane su teoreme koje predstavljaju opšte zakone Horove logike. U ovom poglavlju, specijalne sintaksne jedinice razmatramo pomoću odgovarajućih specijalnih S-relacija koje smo uveli definicijama. Na ovaj način razvili smo mehanizam pomoću kojeg se može dokazati korektnost sintaksne jedinice u odnosu na zadatu specifikaciju. Svaka sintaksna jedinica se posmatra kao interpretacija odgovarajuće S-relacije, a programska specifikacija kao uređen par S-predikata (P, Q), gde je P preduslov, a Q postuslov. Očigledno, dokazivanje kako totalne tako i parcijalne korektnosti sintaksne jedinice u odnosu na zadatu specifikaciju svodi se na dokazivanje valjanosti odgovarajuće S-formule u kojoj figuriraju S-relacija S i S-predikati P i Q. Glavne prednosti ovakvog pristupa dokazivanja korektnosti su opštost i jednostavnost. Opštost se postiže iz same činjenice da je S-račun baziran na apstraktnom skupu stanja neke virtuelne mašine. Jednostavnost S-računa je u tome što za dokazivanje nije potrebno poznavati naročito složen matematički aparat. Da bi se dokazala korektnosti programa ili nova teorema S-računa dovoljno je poznavati predikatsku logiku prvog reda.

Razne teorije, koje se bave opisivanjem i analizom programske semantike, a koriste interpretirani skup stanja ne razmatraju deklaracije programskih promenljivih. Razlog

je u tome što kod interpretiranog skupa stanja svako stanje je opisano vektorom stanja, pa bi pokušaj razmatranja deklaracije programske promenljive izazvao probleme, jer je nejasno kako tumačiti vektor stanja u trenutku kada neke od programskih promenljivih još nisu definisane. Činjenica da S-račun koristi apstraktni skup stanja se može iskoristiti za razmatranje semantike deklaracije programske promenljive. U S-računu za deklaraciju programske promenljive posmatramo odgovarajuću S-relaciju, koju uvodimo definicijom:

Definicija 1.5.10 (Deklaracija) *S-relacija $S_{a:Type;}$ se definiše:*

a.) kao skup $S_{a:Type;} = \{(x, y)|y : a \in D_{Type}\}$, ili

b.) S-formulom $\forall x \forall y S_{a:Type;}(x, y) \Leftrightarrow y : a \in D_{Type}$.

Mogućnost razmatranja deklaracije programske promenljive, takođe, predstavlja prednost S-računa i ima poseban značaj za opisivanje i analizu semantike programskih jezika kod kojih se deklaracija programske promenljive tretira kao naredba (npr. C/C++ [99] [29] [70] [71]) ili kod kojih se čak može pojaviti bilo gde unutar kôda (npr. Java [30] [98]). Na taj način, S-račun otvara mogućnost za automatsku verifikaciju takvih programa.

1.5.1 Primer

Dokazaćemo da je data S-relacija S:

$S: \ a : integer; \ a := 5; \ if \ a > 0 \ then \ a := 10 \ else \ a := 100;$

korektna u odnosu na specifikaciju, koja je data kao uređen par S-predikata (P, Q), gde je preduslov $P : \top$, a postulsov $Q : a = 10$.

Dakle, treba dokazati valjanost S-formule $\{P\}S\{Q\}$. Pored P, S i Q, uvodimo i sledeće oznake:

$S_1: \ a : integer;$
$S_2: \ a := 5; \ if \ a > 0 \ then \ a := 10 \ else \ a := 100;$
$S_3: \ a := 5;$
$S_4: \ if \ a > 0 \ then \ a := 10 \ else \ a := 100;$
$S_5: \ a := 10;$
$S_6: \ a := 100;$
$R: \ a \in D_{integer}$
$T: \ a = 5$
$B: \ a > 0$
$W: \ a = 10$

$U: \ a = 100$

Dokazaćemo totalnu korektnost S-relacije S u odnosu na specifikaciju (τ, W), tj. dokazaćemo valjanost S-formule $\{\tau\}S\{W\}$. S-relacija S je sekvenca $[S_1; S_2]$, pa na osnovu definicije 1.5.6 sledi da je sledeća S-formula valjana:

$$\forall x \forall y S(x, y) \Leftrightarrow \exists z S_1(x, z) \wedge S_2(z, y). \tag{1}$$

Sada ćemo dokazati da je S-relacija S_1 totalno korektna u odnosu na specifikaciju (τ, R), tj. dokazaćemo valjanost S-formule $\{\tau\}S_1\{R\}$. S_1 je deklaracija programske promenljive $a : integer$, pa prema definiciji 1.5.10 sledi da su sledeće S-formule valjane:

$$\forall x \forall y S_1(x, y) \Leftrightarrow y : a \in D_{integer}, \tag{2}$$
$$\{\tau\}S_1\{R\}. \tag{3}$$

Zatim ćemo dokazati da je S-relacija S_2 totalno korektna u odnosu na specifikaciju (R, W), tj. dokazaćemo valjanost S-formule $\{R\}S_2\{W\}$. S-relacija S_2 je sekvenca $[S_3; S_4]$. Na osnovu definicije 1.5.6 sledi da je sledeća S-formula valjana:

$$\forall x \forall y S_2(x, y) \Leftrightarrow \exists z S_3(x, z) \wedge S_4(z, y) \tag{4}$$

S-relacija S_3 je dodela. Na osnovu definicije 1.5.3 sledi da su sledeće S-formule valjane:

$$\forall x \forall z S_3(x, z) \Leftrightarrow x : a \in D_{integer} \wedge z : a = 5, \tag{5}$$
$$\{R\}S_3\{T\}, \tag{6}$$
$$B(z). \tag{7}$$

S-relacije S_5 i S_6 su dodele. Na osnovu definicije 1.5.3 S-formule (8) - (11) su valjane:

$$\forall z \forall y S_5(z, y) \Leftrightarrow z : a \in D_{integer} \wedge y : a = 10, \tag{8}$$
$$\{R\}S_5\{W\}, \tag{9}$$
$$\forall z \forall y S_6(z, y) \Leftrightarrow z : a \in D_{integer} \wedge y : a = 100, \tag{10}$$
$$\{R\}S_6\{U\}. \tag{11}$$

Pošto $T \Rightarrow R$, na osnovu teoreme 1.3.1.*a*.) sledeće S-formule su valjane:

$$\{T\}S_5\{W\}, \tag{12}$$
$$\{T\}S_6\{U\}. \tag{13}$$

S-relacija S_4 je *if-then-else*, pa na osnovu definicije 1.5.4 sledeće S-formule su valjane:

$$\forall z \forall y S_4(z,y) \Leftrightarrow (B(z) \wedge S_5(z,y)) \vee (\neg B(z) \wedge S_6(z,y)). \tag{14}$$

Na osnovu (7), (12) i (14) dobijamo:

$$\{T\}S_4\{W\}. \tag{15}$$

Na osnovu (4), (6) i (15) dobijamo:

$$\{R\}S_2\{W\}. \tag{16}$$

Na osnovu (1), (3) i (16) dobijamo:

$$\{\tau\}S\{W\}.$$

Pošto $P \Rightarrow \tau$ i $W \Rightarrow Q$, na osnovu teoreme 1.3.1.*c.*) zaključujemo da je S-formula

$$\{P\}S\{Q\}$$

valjana, što je i trebalo dokazati.

1.5.2 Primer

Dokazaćemo da je data S-relacija S:

$S:$ *while* $i <= n$ *do begin* $f := f * i;\ i := i + 1$ *end*;

korektna u odnosu na specifikaciju (P, Q), gde je preduslov $P : i = 2 \wedge n = 4 \wedge f = 1$ i postuslov $Q : f = 24$.

Drugim rečima, treba dokazati valjanost S-formule $\{P\}S\{Q\}$. Pored S, P i Q, uvodimo i sledeće oznake:

$R:\ i = 5 \wedge n = 4 \wedge f = 24$
$B:\ i <= n$
$S_1:\ f := f * i;\ i := i + 1;$

$S_2: \ f := f * i;$
$S_3: \ i := i + 1;$

Na osnovu definicije 1.5.3, sledeće S-formule su valjane:

$$\forall x \forall z_1 S_2(x, z_1) \ \Leftrightarrow \ x : i = 2 \wedge n = 4 \wedge f = 1 \ \wedge \ z_1 : i = 2 \wedge n = 4 \wedge f = 2,$$
$$\forall z_1 \forall y_1 S_3(z_1, y_1) \ \Leftrightarrow \ z_1 : i = 2 \wedge n = 4 \wedge f = 2 \ \wedge \ y_1 : i = 3 \wedge n = 4 \wedge f = 2,$$

pa na osnovu definicije 1.5.6, dobijamo:

$$\forall x \forall y_1 S_1(x, y_1) \ \Leftrightarrow \ x : i = 2 \wedge n = 4 \wedge f = 1 \ \wedge \ y_1 : i = 3 \wedge n = 4 \wedge f = 2. \tag{17}$$

Iz S-formule (17) zaključujemo da su sledeće S-formule valjane:

$$\forall x B(x), \tag{18}$$
$$\forall y_1 B(y_1). \tag{19}$$

Na osnovu definicije 1.5.3, sledeće S-formule su valjane:

$$\forall y_1 \forall z_2 S_2(y_1, z_2) \ \Leftrightarrow \ y_1 : i = 3 \wedge n = 4 \wedge f = 2 \ \wedge \ z_2 : i = 3 \wedge n = 4 \wedge f = 6,$$
$$\forall z_2 \forall y_2 S_3(z_2, y_2) \ \Leftrightarrow \ z_2 : i = 3 \wedge n = 4 \wedge f = 6 \ \wedge \ y_2 : i = 4 \wedge n = 4 \wedge f = 6,$$

pa na osnovu definicije 1.5.6, dobijamo:

$$\forall y_1 \forall y_2 S_1(y_1, y_2) \ \Leftrightarrow \ y_1 : i = 3 \wedge n = 4 \wedge f = 2 \ \wedge \ y_2 : i = 4 \wedge n = 4 \wedge f = 6. \tag{20}$$

Iz S-formule (20) zaključujemo da je sledeća S-formula valjana:

$$\forall y_2 B(y_2). \tag{21}$$

Na osnovu definicije 1.5.3, sledeće S-formule su valjane:

$$\forall y_2 \forall z_3 S_2(y_2, z_3) \ \Leftrightarrow \ y_2 : i = 4 \wedge n = 4 \wedge f = 6 \ \wedge \ z_3 : i = 4 \wedge n = 4 \wedge f = 24,$$
$$\forall z_3 \forall y S_3(z_3, y) \ \Leftrightarrow \ z_3 : i = 4 \wedge n = 4 \wedge f = 24 \ \wedge \ y : i = 5 \wedge n = 4 \wedge f = 24,$$

pa na osnovu definicije 1.5.6, dobijamo:

$$\forall y_2 \forall y S_1(y_2, y) \;\Leftrightarrow\; y_2 : i = 4 \wedge n = 4 \wedge f = 6 \;\wedge\; y : i = 5 \wedge n = 4 \wedge f = 24. \tag{22}$$

Iz S-formule (22) zaključujemo da je sledeća S-formula valjana:

$$\forall y \neg B(y). \tag{23}$$

Iz S-formula (17) - (23) zaključujemo da je sledeća S-formula valjana:

$$\forall x \forall y S(x, y) \;\Leftrightarrow\; \forall y_1 \forall y_2 B(x) \wedge S_1(x, y_1) \wedge B(y_1) \wedge S_1(y_1, y_2) \wedge B(y_2) \wedge S_1(y_2, y) \wedge \neg B(y)$$

i da terminiranje S počev od inicijalnog stanja x je garantovano (definicija 1.5.9). Zaključujemo da je sledeća S-formula valjana:

$$\{P\}S\{R\}.$$

Pošto $R \Rightarrow Q$, na osnovu teoreme 1.3.1.*b.*) zaključujemo da je S-formula:

$$\{P\}S\{Q\}$$

valjana, što je i trebalo dokazati.

1.5.3 Primer *(Beskonačna petlja)*

Dokažimo $\forall x \forall y S(x, y) \Leftrightarrow \phi(x)$, gde je:

$S :$ *while* $\top$ *do* $S1$;

Neka je:
$B :\ \top$

Očigledno
$\forall x B(x)$.

Neka je:
$\forall x \forall y_1 S_1(x, y_1)$
$\forall y_1 \forall y_2 S_1(y_1, y_2)$
$\ldots$
$\forall y_{k-1} \forall y_k S_1(y_{k-1}, y_k)$

$\forall y_k \forall y_{k+1} S_1(y_k, y_{k+1})$
$\ldots$

Matematičkom indukcijom ćemo dokazati da program ne terminira počev od inicijalnog stanja x. Nakon prvog prolaza virtuelna mašina se nalazi u stanju y_1 i važi $B(y_1)$, pa očigledno program ne terminira nakon prvog prolaza. Pretpostavimo da program nije terminirao nakon k-tog prolaza i da se virtuelna mašina nalazi u stanju y_k i važi $B(y_k)$, tada nakon $k+1$-og prolaza mašina će biti u stanju y_{k+1} i važiće $B(y_{k+1})$. Dakle, program neće terminirati ni nakon $k+1$-og prolaza. Na osnovu definicije 1.5.8 zaključujemo da naša S-relacija ne sadrži nijedan uređen par (x, y), gde je x početno, a y završno stanje. Dakle, dokazali smo da za bilo koje početno stanje x završno stanje ne postoji, odnosno zaključujemo da je S prazan skup. Naš dokaz smo izveli tako što smo S razmatrali kao potencijalno terminirajuću petlju, ali s obzirom da važi $S^G_{while} \subseteq S^P_{while}$, a to znači kada bi istu petlju razmatrali kao garantovano terminirajuću (definicija 1.5.9) opet bi dobili isti rezultat, tj. $S = \emptyset$. Najzad, zaključujemo

$$\forall x \forall y S(x, y) \;\Leftrightarrow\; \phi(x),$$

što je i trebalo dokazati.

1.5.4 Primer

Dokažimo $\forall x \forall y S(x, y) \Leftrightarrow [x : a \leq 0 \wedge y : a \leq 0]$, gde je:

$S :\ while\ a > 0\ do\ a := a + 1;$

Neka je:
$S_1 :\ a := a + 1;$
$B :\ a > 0$

Očigledno
$\forall x : a > 0,\ B(x),$
$\forall x : a \leq 0,\ \neg B(x).$

Na osnovu definicije 1.5.9, dobijamo:
$\forall x : a \leq 0, \forall y S(x, y) \Leftrightarrow \neg B(x) \wedge S_{nop}(x, y) \Leftrightarrow \top \wedge S_{nop}(x, y) \Leftrightarrow S_{nop}(x, y),$
odnosno
$\forall x \forall y S(x, y) \Leftrightarrow x : a \leq 0 \wedge y = x,$
tj.
$\forall x \forall y S(x, y) \Leftrightarrow x : a \leq 0 \wedge y : a \leq 0.$

Oznakama $a', a'', \ldots, a^{(k)}$ ćemo označiti vrednosti programske promenljive a nakon prvog, drugog,..., k-tog prolaza. S-relacija S_1 je dodela (definicija 1.5.3), pa dobijamo sledeće S-formule:

$\forall x \forall y_1 S_1(x, y_1) \Leftrightarrow x : a > 0 \wedge y_1 : a' = a + 1$
$\forall y_1 \forall y_2 S_1(y_1, y_2) \Leftrightarrow y_1 : a' > 0 \wedge y_2 : a'' = a' + 1$
$\ldots$
$\forall y_{k-1} \forall y_k S_1(y_{k-1}, y_k) \Leftrightarrow y_{k-1} : a^{(k-1)} > 0 \wedge y_k : a^{(k)} = a^{(k-1)} + 1$
$\forall y_k \forall y_{k+1} S_1(y_k, y_{k+1}) \Leftrightarrow y_k : a^{(k)} > 0 \wedge y_{k+1} : a^{(k+1)} = a^{(k)} + 1$
$\ldots$

Dokaz da program ne terminira počev od početnog stanja $x : a > 0$ izvešćemo indukcijom. Nakon prvog prolaza virtuelna mašina se nalazi u stanju $y_1 : a' = a + 1$ i važi $B(y_1)$, pa očigledno program ne terminira nakon prvog prolaza. Pretpostavimo da program nije terminirao nakon k-tog prolaza i da se virtuelna mašina nalazi u stanju $y_k : a^{(k)} = a^{(k-1)} + 1$ i važi $B(y_k)$, tada nakon $k + 1$-og prolaza mašina će biti u stanju $y_{k+1} : a^{(k+1)} = a^{(k)} + 1$ i važiće $B(y_{k+1})$, pa zaključujemo da ni nakon $k + 1$-og prolaza program neće terminirati. Na osnovu definicije 1.5.8 (slično kao u Primeru 1.5.3), dobijamo:

$$\forall x : a > 0, \forall y S(x, y) \Leftrightarrow \phi(x).$$

Dakle, dokazali smo da počev od stanja $x : a \leq 0$ petlja garantovano terminira, a da počev od stanja $x : a > 0$ ne terminira. Najzad, dobijamo:

$\forall x \forall y S(x, y) \Leftrightarrow [x : a \leq 0 \wedge y : a \leq 0] \vee \phi(x)$,
tj.
$\forall x \forall y S(x, y) \Leftrightarrow [x : a \leq 0 \wedge y : a \leq 0]$,

što je i trebalo dokazati.

1.5.5 Primer

Uovom primeru ćemo dokazati $\forall x \forall y S(x, y) \Leftrightarrow [x : a = 0 \wedge y : a = 1] \vee [x : a \neq 0 \wedge y = x]$, gde je:

$S :$ *while* $a = 0$ *do* $a := a + 1$;

Neka je:

$S_1 :\ a := a + 1;$
$B :\ a = 0$

Očigledno
$\forall x : a = 0,\ B(x),$
$\forall x : a \neq 0,\ \neg B(x).$

S-relacija S_1 je dodela (definicija 1.5.3), pa dobijamo sledeće S-formule:
$\forall x : a = 0, \forall y S_1(x, y) \Leftrightarrow x : a = 0 \wedge y : a = 1.$

Na osnovu definicije 1.5.9, dobijamo:
$\forall x : a = 0, \forall y S(x, y) \Leftrightarrow \neg B(x) \wedge \neg B(y) \wedge S_1(x, y) \Leftrightarrow \top \wedge \top \wedge [x : a = 0 \wedge y : a = 1],$
tj.
$\forall x \forall y S(x, y) \Leftrightarrow x : a = 0 \wedge y : a = 1.$

S druge strane, dobijamo i ovo:
$\forall x : a \neq 0, \forall y S(x, y) \Leftrightarrow \neg B(x) \wedge S_{nop}(x, y) \Leftrightarrow \top \wedge [x : a \neq 0 \wedge y = x],$
tj.
$\forall x \forall y S(x, y) \Leftrightarrow x : a \neq 0 \wedge y = x.$

Dokazali smo da počev od stanja $x : a \in D_{Type}$ petlja garantovano terminira, pa dobijamo:
$\forall x \forall y S(x, y) \ \Leftrightarrow\ [x : a = 0 \wedge y : a = 1] \vee [x : a \neq 0 \wedge y = x],$

što je i trebalo dokazati.

1.5.6 Primer

Dokazaćemo $\forall x \forall y S(x, y) \Leftrightarrow [x : a \leq 0 \wedge y : a \leq 0]$, gde je:

$S :\ while\ a > 0\ do\ begin\ r := random();\ if\ r = 1\ then\ a := a + 1\ else\ a := a - 1\ end;$ [1]

Neka je:
$S_1 :\ r := random();\ if\ r = 1\ then\ a := a + 1\ else\ a := a - 1;$
$S_2 :\ r := random();$
$S_3 :\ if\ r = 1\ then\ a := a + 1\ else\ a := a - 1;$
$S_4 :\ a := a + 1;$
$S_5 :\ a := a - 1;$

[1] U literaturi se ova sintaksna jedinica kraće zapisuje: $while\ a > 0\ do\ a := (a + 1\ or\ a - 1);$.

$B: \ a > 0$
$B_1: \ r = 1$

Očigledno
$\forall x : a > 0, \ B(x)$,
$\forall x : a \leq 0, \ \neg B(x)$.

Na osnovu definicije 1.5.9, dobijamo:
$\forall x : a \leq 0, \forall y S(x, y) \Leftrightarrow \neg B(x) \wedge S_{nop}(x, y) \Leftrightarrow \top \wedge S_{nop}(x, y) \Leftrightarrow S_{nop}(x, y)$,
odnosno
$\forall x \forall y S(x, y) \Leftrightarrow x : a \leq 0 \wedge y = x$,
tj.
$\forall x \forall y S(x, y) \Leftrightarrow x : a \leq 0 \wedge y : a \leq 0$.

S-relacija S_2 je dodela (definicija 1.5.3), pa na osnovu definicije S-relacije S_{random} (definicija 1.5.7), dobijamo sledeće S-formule:
$\forall x : a > 0, \forall z S_2(x, z) \Leftrightarrow (x : a > 0 \wedge r \in \{0, 1\}) \wedge (z : a > 0 \wedge (r = 0 \vee r = 1))$.

S-relacija S_3 je *if-then-else* (definicija 1.5.4), pa dobijamo
$\forall z \forall y S_3(z, y) \Leftrightarrow (B_1(z) \wedge S_4(z, y)) \wedge (\neg B_1(z) \wedge S_5(z, y))$.

S-relacija S_1 je sekvenca $[S_2; S_3]$ (definicija 1.5.6), pa dobijamo
$\forall x : a > 0, \forall y S_1(x, y) \Leftrightarrow (x : a > 0 \wedge r \in \{0, 1\}) \wedge (y : (r = 0 \vee r = 1) \wedge (a' = a + 1 \vee a' = a - 1))$.

Na osnovu toga, dobijamo sledeće S-formule:
$\forall x \forall y_1 S_1(x, y_1) \ \Leftrightarrow \ (x : a > 0 \wedge r \in \{0, 1\}) \wedge (y_1 : (r = 0 \vee r = 1) \wedge (a' = a + 1 \vee a' = a - 1))$
$\forall y_1 \forall y_2 S_1(y_1, y_2) \ \Leftrightarrow \ (y_1 : a' > 0 \wedge r \in \{0, 1\}) \wedge (y_2 : (r = 0 \vee r = 1) \wedge (a'' = a' + 1 \vee a'' = a' - 1))$
$\ldots$
$\forall y_{k-1} \forall y_k S_1(y_{k-1}, y_k) \ \Leftrightarrow \ (y_{k-1} : a^{(k-1)} > 0 \wedge r \in \{0, 1\}) \wedge (y_k : (r = 0 \vee r = 1) \wedge (a^{(k)} = a^{(k-1)} + 1 \vee a^{(k)} = a^{(k-1)} - 1))$
$\forall y_k \forall y_{k+1} S_1(y_k, y_{k+1}) \ \Leftrightarrow \ (y_k : a^{(k)} > 0 \wedge r \in \{0, 1\}) \wedge (y_{k+1} : (r = 0 \vee r = 1) \wedge (a^{(k+1)} = a^{(k)} + 1 \vee a^{(k+1)} = a^{(k)} - 1))$
$\ldots$

Indukcijom ćemo dokazati da program potencijalno terminira počev od početnog stanja $x : a > 0$. Nakon prvog prolaza virtuelna mašina se nalazi u stanju $y_1 : (r = 0 \vee r = 1) \wedge (a' = a + 1 \vee a' = a - 1)$ i može i ne mora da važi $B(y_1)$, pa program može i ne mora da terminira. Ako pretpostavimo da program nije terminirao nakon k-tog prolaza i da virtuelna mašina se nalazi u stanju $y_k : (r = 0 \vee r = 1) \wedge (a^{(k)} =$

$a^{(k-1)} + 1 \vee a^{(k)} = a^{(k-1)} - 1$), tada nakon $k+1$-og prolaza mašina će biti u stanju $y_{k+1} : (r = 0 \vee r = 1) \wedge (a^{(k+1)} = a^{(k)} + 1 \vee a^{(k+1)} = a^{(k)} - 1)$, gde opet $B(y_{k+1})$ može i ne mora da važi. Očigledno, program može i ne mora da terminira, pa zaključujemo da počev od početnog stanja $x : a > 0$ program potencijalno terminira. Drugim rečima, na osnovu definicije 1.5.8, dobijamo:
$\forall x : a > 0, \forall y S(x, y) \Leftrightarrow \tau(x)$.

Međutim, istovremeno na osnovu definicije 1.5.9, dobijamo:
$\forall x : a > 0, \forall y S(x, y) \Leftrightarrow \phi(x)$.

Dokazali smo da počev od početnog stanja $x : a \leq 0$ petlja garantovano terminira. Najzad, dobijamo:
$\forall x \forall y S(x, y) \Leftrightarrow [x : a \leq 0 \wedge y : a \leq 0] \vee \phi(x)$,
tj.
$\forall x \forall y S(x, y) \Leftrightarrow [x : a \leq 0 \wedge y : a \leq 0]$,

što je i trebalo dokazati.

1.5.7 Primer

Dokažimo $\forall x \forall y S(x, y) \Leftrightarrow [(x : a \in D_{Type_1} \wedge b \in D_{Type_2}) \wedge (y : a \leq 0 \wedge b \in D_{Type_2})]$, gde je:

$S :$ *while* $a > 0$ *do begin* $a := a - 1$; $r := random()$; *if* $r = 1$ *then* $b := b + 1$ *else* $b := b - 1$ *end*; [2]

Neka je:
$S_1 :$ $a := a - 1$; $r := random()$; *if* $r = 1$ *then* $b := b + 1$ *else* $b := b - 1$;
$B :$ $a > 0$

Kao u Primeru 1.5.6, na osnovu definicije 1.5.9, dobijamo:
$\forall x : a \leq 0 \wedge b \in D_{Type_2}, \forall y S(x, y) \Leftrightarrow \neg B(x) \wedge S_{nop}(x, y) \Leftrightarrow \top \wedge S_{nop}(x, y) \Leftrightarrow S_{nop}(x, y)$,
odnosno
$\forall x \forall y S(x, y) \Leftrightarrow (x : a \leq 0 \wedge b \in D_{Type_2}) \wedge (y = x)$,
tj.
$\forall x \forall y S(x, y) \Leftrightarrow (x : a \leq 0 \wedge b \in D_{Type_2}) \wedge (y : a \leq 0 \wedge b \in D_{Type_2})$.

Zatim, dobijamo sledeće S-formule:
$\forall x \forall y_1 S_1(x, y_1) \Leftrightarrow (x : a > 0 \wedge r \in \{0, 1\} \wedge b \in D_{Type_2}) \wedge (y_1 : a' = a - 1 \wedge (r = 0 \vee r = 1) \wedge (b' = b + 1 \vee b' = b - 1))$

[2] U literaturi se ova sintaksna jedinica kraće zapisuje: *while* $a > 0$ *do* $a := a - 1$; $b := (b + 1$ *or* $b - 1)$ *end*;.

$\forall y_1 \forall y_2 S_1(y_1, y_2) \Leftrightarrow (y_1 : a' > 0 \wedge r \in \{0,1\} \wedge b' \in D_{Type_2}) \wedge (y_2 : a'' = a' - 1 \wedge (r = 0 \vee r = 1) \wedge (b'' = b' + 1 \vee b'' = b' - 1))$
...
$\forall y_{k-1} \forall y_k S_1(y_{k-1}, y_k) \Leftrightarrow (y_{k-1} : a^{(k-1)} > 0 \wedge r \in \{0,1\} \wedge b^{(k-1)} \in D_{Type_2}) \wedge (y_k : a^{(k)} = a^{(k-1)} - 1 \wedge (r = 0 \vee r = 1) \wedge (b^{(k)} = b^{(k-1)} + 1 \vee b^{(k)} = b^{(k-1)} - 1))$
$\forall y_k \forall y_{k+1} S_1(y_k, y_{k+1}) \Leftrightarrow (y_k : a^{(k)} > 0 \wedge r \in \{0,1\} \wedge b^{(k)} \in D_{Type_2}) \wedge (y_{k+1} : a^{(k+1)} = a^{(k)} - 1 \wedge (r = 0 \vee r = 1) \wedge (b^{(k+1)} = b^{(k)} + 1 \vee b^{(k+1)} = b^{(k)} - 1))$
...

Na osnovu definicije 1.5.9 sledi:
$\forall x : a > 0 \wedge b \in D_{Type_2}, \forall y S(x, y)$,
tj.
$\forall x \forall y S(x, y) \Leftrightarrow [(x : a > 0 \wedge b \in D_{Type_2}) \wedge (y : a = 0 \wedge b \in D_{Type_2})]$.

Dokazali smo da počev od stanja $x : a \in D_{Type_1} \wedge b \in D_{Type_2}$ petlja mora da terminira. Najzad, dobijamo:
$\forall x \forall y S(x, y) \Leftrightarrow [(x : a \leq 0 \wedge b \in D_{Type_2}) \wedge (y : a \leq 0 \wedge b \in D_{Type_2})] \vee [(x : a > 0 \wedge b \in D_{Type_2}) \wedge (y : a = 0 \wedge b \in D_{Type_2})]$,
a zatim
$\forall x \forall y S(x, y) \Leftrightarrow [(x : a \in D_{Type_1} \wedge b \in D_{Type_2}) \wedge (y : a \leq 0 \wedge b \in D_{Type_2})]$,

što je i trebalo dokazati.

1.6 Teorema o najširem preduslovu *wp*

U prethodnom izlaganju pokazano je kako se S-račun može koristiti za dokazivenje korektnosti programa. U ovom poglavlju, pokazaćemo kako se u S-računu na strogo formalan način mogu dokazivati nove teoreme. Prvo ćemo dati definiciju najšireg preduslova, a odmah zatim dokazati njegovu jedinstvenost do nivoa ekvivalencije. Posle toga, razmotrićemo i dokazaćemo Dajkstrine teoreme o najširem preduslovu *wp*, a to su redom zakon isključenja čuda, monotonosti, konjunkcije i disjunkcije. Iako korektni, originalni dokazi ovih teorema nisu strogo formalni [27] [33], pa će naš zadatak biti da ih izvedemo na strogo formalan način. Pored pomenutih teorema, uvešćemo i dokazaćemo jednu novu teoremu, a to je zakon negacije (teorema 1.6.7), da bi tako dobili kompletnu sliku o ponašanju operatora *wp*. Na kraju ovog poglavlja dokazaćemo Dajkstrinu teoremu o totalnoj korektnosti. Pojedine teoreme ćemo dokazati automatskim dokazivačem `Coq`.

U poglavlju 1.5, definisali smo potencijalno i garantovano terminiranje. Takođe, pokazali smo da petlji koja ne terminira odgovara prazan skup. U S-računu, operator *wp* je zapravo jedan S-predikat koji opisuje sva početna stanja počev od kojih data S-relacija garantovano terminira i daje očekivani rezultat. Na taj način postižemo da u S-računu *wp* uvek postoji (Primer 1.6.1). Problem postojanja *wp* u Dajkstrinoj analizi nastao je

kao posledica korišćenja interpretiranog skupa stanja, pa je Dajkstra deklarativno svoja razmatranja ograničavao na stanja u kojima *wp* postoji [26].

Definicija 1.6.1 (Najširi preduslov) *Neka je S garantovano terminirajuća S-relacija. Najširi preduslov S-relacije S u odnosu na postuslov Q jeste S-predikat $wp(S,Q)$ ako važi:*
(WP_1) $\{wp(S,Q)\}S\{Q\}$,
(WP_2) $\{P\}S\{Q\} \Rightarrow \forall x(P(x) \Rightarrow wp(S,Q)(x))$.

Teorema 1.6.2 (Teorema o jedinstvenosti najšireg preduslova) *Najširi preduslov S-relacije S u odnosu na postuslov Q, u oznaci $wp(S,Q)$, je jedinstven do nivoa ekvivalencije.*

Dokaz.

Pretpostavimo suprotno, odnosno pretpostavimo da za S-relaciju S postoje dva najšira preduslova u odnosu na postuslov Q za koja važi:
$wp_1(S,Q) \Leftrightarrow wp_2(S,Q) \equiv \bot$.
Na osnovu (WP_1) iz definicije 1.6.1, dobijamo:
$\{wp_1(S,Q)\}S\{Q\} \equiv \top$,
$\{wp_2(S,Q)\}S\{Q\} \equiv \top$.
Na osnovu (WP_2), dobijamo:
$wp_1(S,Q) \Rightarrow wp_2(S,Q)$,
$wp_2(S,Q) \Rightarrow wp_1(S,Q)$,
odakle zaključujemo da važi:
$wp_1(S,Q) \Leftrightarrow wp_2(S,Q) \equiv \top$,
što je suprotno polaznoj pretpostavci, pa osnovu toga zaključujemo da polazna teorema važi.

□

1.6.1 Primer

U ovom primeru ćemo odrediti *wp* za sledeće S-relacije:

a.) S_{abort},

b.) S iz Primera 1.5.3 (*beskonačna petlja*),

c.) S iz Primera 1.5.6 za dati postuslov $Q : a = 0$,

d.) S iz Primera 1.5.7 za date postuslove $Q : a = 0 \wedge b = 6$ i $R : a = 0 \wedge b \in \{0, 2, 4, 6, 8, 10\}$,

e.) S : *while* $a < 10$ *do begin* $b := 100/a$; $a := a + 1$ *end*; za dati postuslov $Q : a = 10$.

Rešenja:

a.) Na osnovu definicije 1.5.1, dobijamo $\forall x \forall y S_{abort}(x,y) \Leftrightarrow \phi(x)$, pa na osnovu (WP_1), dobijamo:
$\{wp(S_{abort}, Q)\}S_{abort}\{Q\} \leftrightarrow \forall x wp(S_{abort}, Q)(x) \Rightarrow \exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow Q(z))$, pa je
$wp(S_{abort}, Q) \equiv \bot$ (što je u saglasnosti sa [27]).

b.) U Primeru 1.5.3 pokazali smo da za beskonačnu petlju važi $\forall x \forall y S(x,y) \Leftrightarrow \phi(x)$, pa je $wp(S,Q) \equiv \bot$.

c.) U Primeru 1.5.6 smo dokazali da petlja potencijalno terminira za svako stanje $x : a \in D_{Type}$, a da garantovano terminira samo za početna stanja $x : a \leq 0$. Ako razmatramo petlju kao garantovano terminirajuću (definicija 1.6.1), dobijamo $wp(S,Q) : a = 0$, što je i prihvatljivo rešenje. Međutim, kada bi istu petlju razmatrali kao potencijalno terminirajuću (definicija 1.6.1) dobili bi $wp(S,Q) : a \geq 0$, što je svakako neprihvatljivo rešenje.

d.) Rešenja su $wp(S,Q) \equiv \bot$ i $wp(S,R) : a = 5 \wedge b = 5$, pošto je:

$$\begin{aligned} x &: a = 5 \wedge b = 5 \\ y_1 &: a = 4 \wedge b \in \{4,6\} \\ y_2 &: a = 3 \wedge b \in \{3,5,7\} \\ y_3 &: a = 2 \wedge b \in \{2,4,6,8\} \\ y_4 &: a = 1 \wedge b \in \{1,3,5,7,9\} \\ y &: a = 0 \wedge b \in \{0,2,4,6,8,10\} \end{aligned}$$

e.) Pošto je:
$\forall x \forall y S(x,y) \Leftrightarrow [(x : a \geq 10 \wedge b \in D_{Type}) \wedge (y = x)] \vee [(x : 0 < a < 10 \wedge b \in D_{Type}) \wedge (y : a = 10 \wedge b \in D_{Type})]$,
dobijamo rešenje:
$wp(S,Q) : 0 < a \leq 10$.

Teorema 1.6.3 (Dajkstrin zakon isključenja čuda) *Sledeća S-formula je valjana:*

$wp(S, \phi) \Leftrightarrow \phi$.

Dokaz.

Na osnovu (WP_1) iz definicije 1.6.1, dobijamo:
$\{wp(S,\phi)\}S\{\phi\}$,
pa na osnovu teoreme 1.3.6, zaključujemo da važi:
$wp(S,\phi) \Leftrightarrow \phi$.

□

Teorema 1.6.4 (Dajkstrin zakon monotonosti) *Sledeća S-formula je valjana:*

$$(Q \Rightarrow R) \;\Rightarrow\; (wp(S,Q) \Rightarrow wp(S,R)).$$

Dokaz.

Na osnovu (WP_1) *iz definicije 1.6.1, dobijamo:*
$\{wp(S,Q)\}S\{Q\}$,
pa na osnovu teoreme 1.3.1.b.), zaključujemo da važi:
$\{wp(S,Q)\}S\{Q\} \wedge \forall x(Q(x) \Rightarrow R(x)) \;\Rightarrow\; \{wp(S,Q)\}S\{R\}$.
Na osnovu (WP_2) *iz definicije 1.6.1, dobijamo:*
$wp(S,Q) \Rightarrow wp(S,R)$.

□

Teorema 1.6.5 (Dajkstrin zakon konjunkcije) *Sledeća S-formula je valjana:*

$$wp(S,Q) \wedge wp(S,R) \;\Leftrightarrow\; wp(S,Q \wedge R).$$

Dokaz.

Prvo, dokažimo implikaciju s leva u desno:
$wp(S,Q) \wedge wp(S,R) \;\Rightarrow\; wp(S,Q \wedge R)$.
Na osnovu (WP_1) *iz definicije 1.6.1, dobijamo sledeće S-formule:*
$\{wp(S,Q)\}S\{Q\}$,
$\{wp(S,R)\}S\{R\}$.
Na osnovu teoreme 1.3.2.b.), dobijamo:
$\{wp(S,Q)\}S\{Q\} \wedge \{wp(S,R)\}S\{R\} \;\Rightarrow\; \{wp(S,Q) \wedge wp(S,R)\}S\{Q \wedge R\}$,
pa na osnovu (WP_2) *iz definicije 1.6.1, dobijamo:*
$wp(S,Q) \wedge wp(S,R) \;\Rightarrow\; wp(S,Q \wedge R)$
i zaključujemo da implikacija s leva u desno važi.
Zatim, dokažimo implikaciju s desna u levo:
$wp(S,Q \wedge R) \;\Rightarrow\; wp(S,Q) \wedge wp(S,R)$.
Na osnovu (WP_1) *iz definicije 1.6.1, dobijamo sledeću S-formulu:*
$\{wp(S,Q \wedge R)\}S\{Q \wedge R\}$.
Na osnovu teoreme 1.3.5.b.), dobijamo:
$\{wp(S,Q \wedge R)\}S\{Q\} \wedge \{wp(S,Q \wedge R)\}S\{R\}$,
pa na osnovu (WP_2) *iz definicije 1.6.1, dobijamo:*
$(wp(S,Q \wedge R) \Rightarrow wp(S,Q)) \;\wedge\; (wp(S,Q \wedge R) \Rightarrow wp(S,R))$.
Nakon toga, na osnovu teoreme (T_{19})*, dobijamo:*
$wp(S,Q \wedge R) \;\Rightarrow\; wp(S,Q) \wedge wp(S,R)$
i zaključujemo da implikacija s desna u levo važi.

□

Teorema 1.6.6 (Dajkstrin zakon disjunkcije) *Sledeća S-formula je valjana:*

$$wp(S,Q) \vee wp(S,R) \;\Rightarrow\; wp(S,Q \vee R).$$

Dokaz.

Na osnovu teoreme 1.3.2.a.), dobijamo:
$\{wp(S,Q)\}S\{Q\} \wedge \{wp(S,R)\}S\{R\} \;\Rightarrow\; \{wp(S,Q) \vee wp(S,R)\}S\{Q \vee R\}$,
pa na osnovu (WP_2) *iz definicije 1.6.1, dobijamo:*
$wp(S,Q) \vee wp(S,R) \;\Rightarrow\; wp(S,Q \vee R)$.

□

Teorema 1.6.7 (Zakon negacije) *Sledeća S-formula je valjana:*

$$\neg(wp(S,Q) \wedge wp(S,\neg Q)).$$

Dokaz.

Na osnovu (WP_1) *iz definicije 1.6.1, zamenom* P *sa* $wp(S,Q)$ *i* R *sa* $wp(S,\neg Q)$ *u teoremi 1.3.7.a.), dobijamo:*
$\neg(wp(S,Q) \wedge wp(S,\neg Q)$.

□

Teorema 1.6.8 (Dajkstrina teorema o totalnoj korektnosti) *Sledeća S-formula je valjana:*

$$\{P\}S\{Q\} \;\Leftrightarrow\; (P \Rightarrow wp(S,Q)).$$

Dokaz.

Prvo, dokažimo implikaciju s leva u desno:
$\{P\}S\{Q\} \;\Rightarrow\; (P \Rightarrow wp(S,Q))$.
Na osnovu (WP_2) *iz definicije 1.6.1, zaključujemo da implikacija s leva u desno važi.*
Zatim, dokažimo implikaciju s desna u levo:
$(P \Rightarrow wp(S,Q)) \;\Rightarrow\; \{P\}S\{Q\}$.
Na osnovu (WP_1) *iz definicije 1.6.1, dobijamo sledeću S-formulu:*
$\{wp(S,Q)\}S\{Q\}$.
Zamenom R *sa* $wp(S,Q)$ *u teoremi 1.3.1.a.), dobijamo:*

$\forall x(P(x) \Rightarrow wp(S,Q)(x)) \wedge \{wp(S,Q)\}S\{Q\} \;\Rightarrow\; \{P\}S\{Q\}$
i zaključujemo da implikacija s desna u levo važi.
Pošto obe implikacije važe, zaključujemo da i početna ekvivalencija važi.

□

Teoreme 1.6.3, 1.6.7 i 1.6.8 se mogu dokazati automatskim dokazivačem `Coq`, što je dato u prilogu A.

Glava 2

Dinamički postuslovi u S-programskom računu

Ne može se dva puta ući u istu reku, jer uvek pritiče sveža voda.
Heraklit (oko 535-475 p.n.e.)

2.1 Uvod

Za početak razmotrimo jedan primer. Posmatrajmo S-relaciju $S : a := a + 1$; i preduslov $P : a \in D_{Type}$. Postavlja se pitanje kako u ovakvoj situaciji odrediti postuslov? Na prvi pogled deluje nemoguće, ali jedno rešenje jeste da postuslov bude logička funkcija ne samo završnog stanja, već i početnog stanja. Uvedimo oznaku a' kojom ćemo označiti vrednost programske promenljive a u završnom stanju. Dakle, moguće rešenje za traženi postuslov jeste $\hat{Q} : a' = a + 1$ i tada očigledno važi $\{P\}S\{\hat{Q}\}$. Postuslov $\hat{Q}(x, z)$ koji je logička funkcija početnog stanja $x \in A$ i završnog stanja $z \in A$ zovemo *dinamički postuslov*. Da bismo u daljem tekstu jasno razlikovali dinamički postuslov koristimo oznaku $\hat{}$, pa ćemo pisati npr. $\hat{Q}(x, z)$ ili kraće $\hat{Q}$. Običan postuslov $R(z)$ koji je logička funkcija završnog stanja $z \in A$, ćemo ovde nazvati *statičkim postuslovom*.

Sada formula totalne korektnosti $\{P\}S\{\hat{Q}\}$ dobija drugačiji oblik:

- Dinamička formula totalne korektnosti $(DFTK)$:

$$\forall x[P(x) \Rightarrow (\exists y S(x, y) \wedge \forall z(S(x, z) \Rightarrow \hat{Q}(x, z)))] \ .$$

Razlika između (FTK) i $(DFTK)$ je u tome što se druga ne odnosi na završna stanja već na *prelaze*. Interpretacija dinamičke formule totalne korektnosti, u oznaci $\{P\}S\{\hat{Q}\}$, jeste drugačija:

- ako sintaksna jedinica S započinje izvršavanje u stanju koje zadovoljava preduslov P, tada ona terminira i biće izvršen prelaz koji zadovoljava dinamički postuslov $\hat{Q}$.

U daljem izlaganju, razmotrićemo osnovne osobine dinamičkih postuslova, sa ciljem da razvijemo matematički alat koji će nam kasnije biti od velikog značaja za analizu semantike objektno orijentisanog programa.

2.2 Osobine dinamičkih postuslova

U ovom poglavlju ćemo dokazati zakone dinamičke konsekvencije, kontingencije, konjunkcije, disjunkcije i negacije. Na taj način ćemo pokazati da opšti Horovi zakoni važe ne samo u statičkom, već i u dinamičkom ambijentu.

Teorema 2.2.1 (Alternativni oblik $(DFTK)$**)** *Sledeća S-formula je validna:*

$$\{P\}S\{\hat{Q}\} \leftrightarrow \forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow \hat{Q}(x,z)].$$

Dokaz.

Pošto je:
$\{P\}S\{\hat{Q}\} \leftrightarrow \forall x[P(x) \Rightarrow (\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z)))]$,
na osnovu teoreme (T_{19})*, desna strana ekvivalencije postaje:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \Rightarrow (S(x,z) \Rightarrow \hat{Q}(x,z))]$,
a na osnovu teoreme (T_{15})*, postaje:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \Rightarrow (\neg S(x,z) \vee \hat{Q}(x,z))]$
$\equiv \forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[\neg P(x) \vee (\neg S(x,z) \vee \hat{Q}(x,z))]$.
Zatim, na osnovu teoreme (T_{12})*, dobijamo:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[\neg(P(x) \wedge S(x,z)) \vee \hat{Q}(x,z)]$
i konačno, na osnovu teoreme (T_{19})*, dobijamo:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow \hat{Q}(x,z)]$.

□

Teorema 2.2.2 (Zakoni dinamičke konsekvencije) *Sledeće S-formule su valjane:*

a.) $\forall x(P(x) \Rightarrow R(x)) \wedge \{R\}S\{\hat{Q}\} \;\Rightarrow\; \{P\}S\{\hat{Q}\}$,

b.) $\{P\}S\{\hat{R}\} \wedge \forall x \forall z(\hat{R}(x,z) \Rightarrow \hat{Q}(x,z)) \;\Rightarrow\; \{P\}S\{\hat{Q}\}$,

c.) $\{P\}S\{\hat{R}\} \wedge \forall x \forall z(\hat{R}(x,z) \Rightarrow Q(z)) \;\Rightarrow\; \{P\}S\{Q\}$,

d.) $\{P\}S\{R\} \wedge \forall x \forall z(R(z) \Rightarrow \hat{Q}(x,z)) \;\Rightarrow\; \{P\}S\{\hat{Q}\}$,

e.) $\forall x(U(x) \Rightarrow P(x)) \wedge \forall x \forall z(\hat{Q}(x,z) \Rightarrow \hat{V}(x,z)) \wedge \{P\}S\{\hat{Q}\} \;\Rightarrow\; \{U\}S\{\hat{V}\}$.

Dokaz.

a.) Pošto je:
$\{R\}S\{\hat{Q}\} \leftrightarrow \forall x[R(x) \Rightarrow (\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z)))]$,
leva strana implikacije se može napisati ovako:
$\forall x(P(x) \Rightarrow R(x)) \wedge \forall x[R(x) \Rightarrow (\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z)))]$
i na osnovu teoreme (T_{16})*, dobijamo:*
$\forall x[P(x) \Rightarrow (\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z)))]$,
tj.
$\{P\}S\{\hat{Q}\}$.

b.) Na osnovu teoreme 2.2.1, leva strana implikacije može se napisati ovako:
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow \hat{R}(x,z)] \ \wedge \ \forall x \forall z(\hat{R}(x,z) \Rightarrow \hat{Q}(x,z))$
i na osnovu teoreme (T_{16})*, dobijamo:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow \hat{Q}(x,z)]$,
tj.
$\{P\}S\{\hat{Q}\}$.

c.) Na osnovu teoreme 2.2.1, leva strana implikacije može se napisati ovako:
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow \hat{R}(x,z)] \ \wedge \ \forall x \forall z(\hat{R}(x,z) \Rightarrow Q(z))$
i na osnovu teoreme (T_{16})*, dobijamo:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow Q(z)]$,
tj.
$\{P\}S\{Q\}$.

d.) Na osnovu teoreme 1.2.4, leva strana implikacije može se napisati ovako:
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow R(z)] \ \wedge \ \forall x \forall z(R(z) \Rightarrow \hat{Q}(x,z))$
i na osnovu teoreme (T_{16})*, dobijamo:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow \hat{Q}(x,z)]$,
tj.
$\{P\}S\{\hat{Q}\}$.

e.) Na osnovu teoreme 2.2.2.a.), leva strana implikacije se može napisati ovako:
$\forall x \forall z(\hat{Q}(x,z) \Rightarrow \hat{V}(x,z)) \ \wedge \ \{U\}S\{\hat{Q}\}$
i na osnovu teoreme 2.2.2.b.), dobijamo:
$\{U\}S\{\hat{V}\}$.

□

Teorema 2.2.3 (Zakoni dinamičke kontingencije) *Sledeće S-formule su valjane:*

a.) $[\{R\}S\{\hat{Q}\} \wedge \forall x \forall z((P(x) \wedge \hat{Q}(x,z)) \Rightarrow R(z))] \ \Rightarrow \ \{P\}S\{R\}$,

b.) $[\{P\}S\{R\} \wedge \forall x \forall z(R(z) \Rightarrow (P(x) \wedge \hat{Q}(x,z)))] \ \Rightarrow \ \{P\}S\{\hat{Q}\}$.

Dokaz.

a.) Pretpostavimo suprotno, tj. pretpostavimo da važi:
$[\{R\}S\{\hat{Q}\} \wedge \forall x \forall z((P(x) \wedge \hat{Q}(x,z)) \Rightarrow R(z))] \;\Rightarrow\; \{P\}S\{R\} \equiv \bot.$ (1)
S-formula (1) *je valjana akko:*
$[\{R\}S\{\hat{Q}\} \wedge \forall x \forall z((P(x) \wedge \hat{Q}(x,z)) \Rightarrow R(z))] \equiv \top,$ (2)
$\{P\}S\{R\} \leftrightarrow \forall x[P(x) \Rightarrow (\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow R(z)))] \equiv \bot.$ (3)
S-formula (2) *je valjana akko:*
$\{P\}S\{\hat{Q}\} \leftrightarrow \forall x[P(x) \Rightarrow (\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z)))] \equiv \top,$ (4)
$\forall x \forall z((P(x) \wedge \hat{Q}(x,z)) \Rightarrow R(z)) \equiv \top.$ (5)
Iz S-formule (3) *sledi:*
$\forall x P(x) \equiv \top,$ (6)
$\forall x[\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow R(z))] \equiv \bot.$ (7)
S obzirom na (5)*, iz S-formule* (4) *dobijamo:*
$\forall x[\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z))] \equiv \top,$ (8)
S-formula (8) *je valjana akko:*
$\forall x \exists y S(x,y) \equiv \top,$ (9)
$\forall x \forall z(S(x,z) \Rightarrow \hat{Q}(x,z)) \equiv \top.$ (10)
S obzirom na (9)*, iz S-formule* (7) *dobijamo:*
$\forall x \forall z(S(x,z) \Rightarrow R(z)) \equiv \bot.$ (11)
S-formula (11) *je valjana akko:*
$\forall x \forall z S(x,z) \equiv \top,$ (12)
$\forall z R(z) \equiv \bot.$ (13)
S obzirom na (12)*, iz S-formule* (10) *dobijamo:*
$\forall x \forall z \hat{Q}(x,z) \equiv \top.$ (14)
S obzirom na (6) *i* (14)*, iz S-formule* (5) *dobijamo:*
$\forall z R(z) \equiv \top,$ (15)
gde dolazimo do kontradikcije, jer S-formula (13) *je u suprotnosti sa* (15)*. Time smo dokazali da pretpostavka* (1) *ne važi, pa zaključujemo da polazna teorema važi.*

b.) Pretpostavimo suprotno, tj. pretpostavimo da važi:
$[\{P\}S\{R\} \wedge \forall x \forall z(R(z) \Rightarrow (P(x) \wedge \hat{Q}(x,z)))] \;\Rightarrow\; \{P\}S\{\hat{Q}\} \equiv \bot.$ (1)
S-formula (1) *je valjana akko:*
$[\{P\}S\{R\} \wedge \forall x \forall z(R(z) \Rightarrow (P(x) \wedge \hat{Q}(x,z)))] \equiv \top,$ (2)
$\{P\}S\{\hat{Q}\} \leftrightarrow \forall x[P(x) \Rightarrow (\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z)))] \equiv \bot,$ (3)
S-formula (3) *je valjana akko:*
$\forall x P(x) \equiv \top,$ (4)
$\forall x[\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z))] \equiv \bot.$ (5)
S-formula (2) *je valjana akko:*
$\{P\}S\{R\} \leftrightarrow \forall x[P(x) \Rightarrow (\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow R(z)))] \equiv \top,$ (6)
$\forall x \forall z(R(z) \Rightarrow (P(x) \wedge \hat{Q}(x,z))) \equiv \top.$ (7)
S obzirom na (4)*, iz S-formule* (6) *dobijamo:*
$\forall x[\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow R(z))] \equiv \top.$ (8)
S-formula (8) *je valjana akko:*
$\forall x \exists y S(x,y) \equiv \top,$ (9)
$\forall x \forall z(S(x,z) \Rightarrow R(z)) \equiv \top.$ (10)

S obzirom na (10)*, iz S-formule* (5) *dobijamo:*
$\forall x \forall z(S(x,z) \Rightarrow \hat{Q}(x,z)) \equiv \bot.$ (11)
S-formula (11) *je valjana akko:*
$\forall x \forall z S(x,z) \equiv \top,$ (12)
$\forall x \forall z \hat{Q}(x,z) \equiv \bot.$ (13)
S obzirom na (12)*, iz S-formule* (10) *dobijamo:*
$\forall z R(z) \equiv \top.$ (14)
S obzirom na (4) *i* (13)*, iz S-formule* (7) *dobijamo:*
$\forall z R(z) \equiv \bot,$ (15)
gde dolazimo do kontradikcije, jer S-formula (14) *je u suprotnosti sa* (15)*. Time smo dokazali da pretpostavka* (1) *ne važi, pa zaključujemo da polazna teorema važi.*

□

Teoreme 2.2.3.*a.*) i 2.2.3.*b.*) imaju veliki značaj, jer ih možemo koristiti za prelazak iz režima rada sa statičkim postuslovom u režim rada sa dinamičkim postuslovom, a pri tom da se ne naruši totalna korektnost. Sada ćemo dokazati zakone koji objedinjuju dinamičku konsekvenciju i kontigenciju i te ćemo nazvati mešovitim zakonima.

Teorema 2.2.4 (Mešoviti zakoni) *Sledeće S-formule su valjane:*

a.) $[\forall x(U(x) \Rightarrow P(x)) \wedge \forall x \forall z((P(x) \wedge \hat{Q}(x,z)) \Rightarrow V(z)) \wedge \{P\}S\{\hat{Q}\}] \;\Rightarrow\; \{U\}S\{V\},$

b.) $[\forall x(U(x) \Rightarrow P(x)) \wedge \forall x \forall z(Q(z) \Rightarrow (U(x) \wedge \hat{V}(x,z))) \wedge \{P\}S\{Q\}] \;\Rightarrow\; \{U\}S\{\hat{V}\}.$

Dokaz.

a.) Na osnovu teoreme 2.2.3.a.), leva strana implikacije se može napisati ovako:
$\forall x(U(x) \Rightarrow P(x)) \wedge \{P\}S\{V\},$
odakle na osnovu teoreme 1.3.1.a.), dobijamo:
$\{U\}S\{V\}.$

b.) Na osnovu teoreme 1.3.1.a.), leva strana implikacije se može napisati ovako:
$\forall x \forall z(Q(z) \Rightarrow (U(x) \wedge \hat{V}(x,z))) \wedge \{U\}S\{Q\},$
odakle na osnovu teoreme 2.2.3.b.), dobijamo:
$\{U\}S\{\hat{V}\}.$

□

Primenom teoreme (T_3) iz mešovitih zakona dobijamo sledeća pravila:

$$\frac{U \Rightarrow P,\ (P \wedge \hat{Q}) \Rightarrow V,\ \{P\}S\{\hat{Q}\}}{\{U\}S\{V\}} \qquad \frac{U \Rightarrow P,\ Q \Rightarrow (U \wedge \hat{V}),\ \{P\}S\{Q\}}{\{U\}S\{\hat{V}\}}$$

Prvo pravilo nam omogućava da napravimo lanac dokaza koji nam može koristiti za analizu sekvence $[S_1; S_2; \ldots; S_n]$:

$$\frac{P \Rightarrow U_1, (U_1 \wedge \hat{V_1}) \Rightarrow R_1, \{U_1\}S_1\{\hat{V_1}\}}{\{P\}S_1\{R_1\}}$$

$$\frac{R_1 \Rightarrow U_2, (U_2 \wedge \hat{V_2}) \Rightarrow R_2, \{U_2\}S_2\{\hat{V_2}\}}{\{R_1\}S_2\{R_2\}}$$

$$\ldots$$

$$\frac{R_{n-1} \Rightarrow U_n, (U_n \wedge \hat{V_n}) \Rightarrow Q, \{U_n\}S_n\{\hat{V_n}\}}{\{R_{n-1}\}S_n\{Q\}}$$

odakle zaključujemo da važi:

$$\{P\}S\{Q\}$$

gde je S sekvenca $[S_1; S_2; \ldots; S_n]$.

U daljem tekstu ćemo razmotriti zakone dinamičke konjunkcije, disjunkcije i negacije.

Teorema 2.2.5 (Zakoni dinamičke konjunkcije) *Sledeće S-formule su valjane:*

a.) $\{P\}S\{\hat{Q}\} \wedge \{R\}S\{\hat{W}\} \;\Rightarrow\; \{P \vee R\}S\{\hat{Q} \vee \hat{W}\}$,

b.) $\{P\}S\{\hat{Q}\} \wedge \{R\}S\{\hat{W}\} \;\Rightarrow\; \{P \wedge R\}S\{\hat{Q} \wedge \hat{W}\}$.

Dokaz.

a.) *Na osnovu teoreme 2.2.1, leva strana implikacije se može napisati ovako:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow \hat{Q}(x,z)] \;\wedge\; \forall x[R(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[R(x) \wedge S(x,z) \Rightarrow \hat{W}(x,z)]$.
Na osnovu teoreme (T_{17})*, dobijamo:*
$\forall x[(P(x) \vee R(x)) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[((P(x) \wedge S(x,z)) \vee (R(x) \wedge S(x,z))) \Rightarrow (\hat{Q}(x,z) \vee \hat{W}(x,z))]$
$\equiv \forall x[(P(x) \vee R(x)) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[((P(x) \vee R(x)) \wedge (S(x,z) \vee S(x,z))) \Rightarrow (\hat{Q}(x,z) \vee \hat{W}(x,z))]$
$\equiv \forall x[(P(x) \vee R(x)) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[(P(x) \vee R(x)) \wedge S(x,z) \Rightarrow (\hat{Q}(x,z) \vee \hat{W}(x,z))]$,
tj.
$\{P \vee R\}S\{\hat{Q} \vee \hat{W}\}$.

b.) Na osnovu teoreme 2.2.1, leva strana implikacije se može napisati ovako:
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow \hat{Q}(x,z)] \ \wedge \ \forall x[R(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[R(x) \wedge S(x,z) \Rightarrow \hat{W}(x,z)]$.
Na osnovu teoreme (T_{18})*, dobijamo:*
$\forall x[(P(x) \wedge R(x)) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[((P(x) \wedge S(x,z)) \wedge (R(x) \wedge S(x,z))) \Rightarrow (\hat{Q}(x,z) \wedge \hat{W}(x,z))]$
$\equiv \ \forall x[(P(x) \wedge R(x)) \ \Rightarrow \ \exists y S(x,y)] \wedge \forall x \forall z[(P(x) \wedge R(x)) \wedge S(x,z) \ \Rightarrow \ (\hat{Q}(x,z) \wedge \hat{W}(x,z))]$,
tj.
$\{P \wedge R\} S \{\hat{Q} \wedge \hat{W}\}$.

□

Posledica 2.2.6 (Zakon dinamičke rezolucije) *Sledeća S-formula je valjana:*

$$\{P\} S \{\hat{Q}\} \wedge \{\neg P\} S \{\hat{W}\} \ \Rightarrow \ \{\tau\} S \{\hat{Q} \vee \hat{W}\}.$$

Dokaz.

Ako u teoremi 2.2.5.a.) zamenimo R sa $\neg P$ *dobijamo:*
$\{P\} S \{\hat{Q}\} \wedge \{\neg P\} S \{\hat{W}\} \ \Rightarrow \ \{P \vee \neg P\} S \{\hat{Q} \vee \hat{W}\}$,
tj.
$\{P\} S \{\hat{Q}\} \wedge \{\neg P\} S \{\hat{W}\} \ \Rightarrow \ \{\tau\} S \{\hat{Q} \vee \hat{W}\}$.

□

Teorema 2.2.7 (Zakoni dinamičke disjunkcije) *Sledeća S-formula je valjana:*

$$\{P\} S \{\hat{Q}\} \vee \{R\} S \{\hat{W}\} \ \Rightarrow \ \{P \wedge R\} S \{\hat{Q} \vee \hat{W}\}.$$

Dokaz.

Na osnovu teoreme 2.2.1, leva strana implikacije se može napisati ovako:
$(\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow \hat{Q}(x,z)]) \ \vee \ (\forall x[R(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[R(x) \wedge S(x,z) \Rightarrow \hat{W}(x,z)])$
$\equiv \ (\forall x[P(x) \ \Rightarrow \ \exists y S(x,y)] \vee \forall x[R(x) \ \Rightarrow \ \exists y S(x,y)]) \ \wedge \ (\forall x \forall z[P(x) \wedge S(x,z) \ \Rightarrow \ \hat{Q}(x,z)] \vee \forall x \forall z[R(x) \wedge S(x,z) \Rightarrow \hat{W}(x,z)])$.
Na osnovu teoreme (T_{22})*, dobijamo:*
$\forall x[(P(x) \wedge R(x)) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[(P(x) \wedge S(x,z)) \wedge (R(x) \wedge S(x,z)) \Rightarrow (\hat{Q}(x,z) \vee \hat{W}(x,z))]$
$\equiv \ \forall x[(P(x) \wedge R(x)) \ \Rightarrow \ \exists y S(x,y)] \ \wedge \ \forall x \forall z[(P(x) \wedge R(x)) \wedge S(x,z) \ \Rightarrow \ (\hat{Q}(x,z) \vee$

$\hat{W}(x,z))]$,
tj.
$\{P \wedge R\}S\{\hat{Q} \vee \hat{W}\}$.

□

Teorema 2.2.8 (Zakoni dinamičke konjunkcije i disjunkcije) *Sledeće S-formule su valjane:*

a.) $\{P \vee R\}S\{\hat{Q}\} \Leftrightarrow \{P\}S\{\hat{Q}\} \wedge \{R\}S\{\hat{Q}\}$,

b.) $\{P\}S\{\hat{Q} \wedge \hat{R}\} \Leftrightarrow \{P\}S\{\hat{Q}\} \wedge \{P\}S\{\hat{R}\}$,

c.) $\{P \vee U\}S\{\hat{Q} \wedge \hat{W}\} \Leftrightarrow \{P\}S\{\hat{Q}\} \wedge \{U\}S\{\hat{W}\} \wedge \{P\}S\{\hat{W}\} \wedge \{U\}S\{\hat{Q}\}$,

d.) $\{P\}S\{\hat{Q}\} \vee \{P\}S\{\hat{W}\} \Rightarrow \{P\}S\{\hat{Q} \vee \hat{W}\}$.

Dokaz.

a.) *Leva strana ekvivalencije se može napisati ovako:*
$\forall x[(P(x) \vee R(x)) \Rightarrow \exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z))]$
i na osnovu teoreme (T_{21})*, dobijamo:*
$\forall x[(P(x) \Rightarrow \exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z))) \wedge (R(x) \Rightarrow \exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z)))]$,
tj.
$\{P\}S\{\hat{Q}\} \wedge \{R\}S\{\hat{Q}\}$.

b.) *Na osnovu teoreme 2.2.1, leva strana ekvivalencije se može napisati ovako:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow \hat{Q}(x,z) \wedge \hat{R}(x,z)]$
i na osnovu teoreme (T_{18})*, dobijamo:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow \hat{Q}(x,z)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow \hat{R}(x,z)]$
$\equiv \forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow \hat{Q}(x,z)] \wedge \forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow \hat{R}(x,z)]$,
tj.
$\{P\}S\{\hat{Q}\} \wedge \{P\}S\{\hat{R}\}$.

c.) *Na osnovu teoreme 2.2.8.a.) i leve strane ekvivalencije dobijamo:*
$\{P \vee U\}S\{\hat{Q} \wedge \hat{W}\} \Leftrightarrow \{P\}S\{\hat{Q} \wedge \hat{W}\} \wedge \{U\}S\{\hat{Q} \wedge \hat{W}\}$
i na osnovu teoreme 2.2.8.b.), dobijamo:
$\{P\}S\{\hat{Q} \wedge \hat{W}\} \wedge \{U\}S\{\hat{Q} \wedge \hat{W}\} \Leftrightarrow \{P\}S\{\hat{Q}\} \wedge \{U\}S\{\hat{W}\} \wedge \{P\}S\{\hat{W}\} \wedge \{U\}S\{\hat{Q}\}$.

d.) Ako u teoremi 2.2.7 zamenimo R sa P dobijamo:
$\{P\}S\{\hat{Q}\} \vee \{P\}S\{\hat{W}\} \Rightarrow \{P \wedge P\}S\{\hat{Q} \vee \hat{W}\}$
i na osnovu teoreme (T_{10}), dobijamo:
$\{P\}S\{\hat{Q}\} \vee \{P\}S\{\hat{W}\} \Rightarrow \{P\}S\{\hat{Q} \vee \hat{W}\}$.

□

Teorema 2.2.9 (Zakoni dinamičke negacije) *Sledeće S-formule su valjane:*

a.) $\{P\}S\{\hat{Q}\} \wedge \{R\}S\{\neg\hat{Q}\} \;\Rightarrow\; \neg(P \wedge R)$,

b.) $\{P\}S\{\hat{Q}\} \wedge \{P\}S\{\neg\hat{Q}\} \;\Leftrightarrow\; \forall x \neg P(x)$,

c.) $[\{P\}S\{\neg\hat{Q}\} \Rightarrow \neg\{P\}S\{\hat{Q}\}] \;\Leftrightarrow\; \exists x P(x)$,

d.) $\{P\}S\{\hat{Q}\} \wedge \{\neg P\}S\{\hat{Q}\} \;\Leftrightarrow\; \forall x \exists y S(x,y) \wedge \forall x \forall z(S(x,z) \Rightarrow \hat{Q}(x,z))$,

e.) $\exists x \exists z S(x,z) \wedge \neg\hat{Q}(x,z) \;\Rightarrow\; [\{\neg P\}S\{\hat{Q}\} \Rightarrow \neg\{P\}S\{\hat{Q}\}]$.

Dokaz.

a.) Ako u teoremi 2.2.5.b.) zamenimo $\hat{W}$ sa $\neg\hat{Q}$, dobijamo:
$\{P\}S\{\hat{Q}\} \wedge \{R\}S\{\neg\hat{Q}\} \Rightarrow \{P \wedge R\}S\{\hat{Q} \wedge \neg\hat{Q}\}$
$\equiv \; \{P\}S\{\hat{Q}\} \wedge \{R\}S\{\neg\hat{Q}\} \Rightarrow \{P \wedge R\}S\{\phi\}$
i na osnovu teoreme 1.3.6, dobijamo:
$\{P\}S\{\hat{Q}\} \wedge \{R\}S\{\neg\hat{Q}\} \Rightarrow \neg(P \wedge R)$.

b.) Ako u teoremi 2.2.8.b.) zamenimo R sa $\neg\hat{Q}$, dobijamo:
$\{P\}S\{\hat{Q}\} \wedge \{P\}S\{\neg\hat{Q}\} \Leftrightarrow \{P\}S\{\hat{Q} \wedge \neg\hat{Q}\}$
$\equiv \; \{P\}S\{\hat{Q}\} \wedge \{P\}S\{\neg\hat{Q}\} \Leftrightarrow \{P\}S\{\phi\}$
i na osnovu teoreme 1.3.6, dobijamo:
$\{P\}S\{\hat{Q}\} \wedge \{P\}S\{\neg\hat{Q}\} \Leftrightarrow \neg P$,
tj.
$\{P\}S\{\hat{Q}\} \wedge \{P\}S\{\neg\hat{Q}\} \Leftrightarrow \forall x \neg P(x)$.

c.) Na osnovu teoreme (T_{15}), leva strana ekvivalencije postaje:
$\neg\{P\}S\{\neg\hat{Q}\} \vee \neg\{P\}S\{\hat{Q}\}$
a zatim, na osnovu teoreme (T_{12}), dobijamo:
$\neg[\{P\}S\{\neg\hat{Q}\} \wedge \{P\}S\{\hat{Q}\}]$.
Nakon toga, na osnovu teoreme 2.2.9.b.), dobijamo:
$\neg[\forall x \neg P(x)]$
i najzad, na osnovu teoreme (T_6), dobijamo:
$\exists x P(x)$.

d.) Ako u teoremi 2.2.8.a.) zamenimo R sa $\neg P$ dobijamo:
$\{P \vee \neg P\}S\{\hat{Q}\}$
$\equiv \ \{\tau\}S\{\hat{Q}\}$.
Pošto je:
$\{\tau\}S\{\hat{Q}\} \ \leftrightarrow \ \forall x[\tau(x) \Rightarrow \exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z))]$,
na osnovu teoreme (T_8)*, dobijamo:*
$\forall x[\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z))]$.

e.) Na osnovu teoreme (T_{15})*, desna strana ekvivalencije se može napisati ovako:*
$\neg\{\neg P\}S\{\hat{Q}\} \vee \neg\{P\}S\{\hat{Q}\}$
i na osnovu teoreme (T_{12})*, dobijamo:*
$\neg[\{\neg P\}S\{\hat{Q}\} \wedge \{P\}S\{\hat{Q}\}]$.
Zatim, na osnovu teoreme 2.2.9.d.), dobijamo:
$\neg\forall x[\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z))]$
$\equiv \ \exists x\neg[\exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z))]$.
Na osnovu teoreme (T_{13})*, dobijamo:*
$\exists x[\neg\exists y S(x,y) \vee \neg\forall z(S(x,z) \Rightarrow \hat{Q}(x,z))]$
$\equiv \ \exists x[\forall y\neg S(x,y) \vee \exists z\neg(S(x,z) \Rightarrow \hat{Q}(x,z))]$
i na osnovu teoreme (T_{15})*, dobijamo:*
$\exists x[\forall y\neg S(x,y) \vee \exists z\neg(\neg S(x,z) \vee \hat{Q}(x,z))]$.
Nakon toga, na osnovu teoreme (T_{12})*, dobijamo:*
$\exists x[\forall y\neg S(x,y) \vee \exists z(S(x,z) \wedge \neg\hat{Q}(x,z))]$
$\equiv \exists x\forall y\neg S(x,y) \vee \exists x\exists z(S(x,z) \wedge \neg\hat{Q}(x,z))$
i najzad, na osnovu teoreme (T_{11})*, dobijamo:*
$\exists x\exists z(S(x,z) \wedge \neg\hat{Q}(x,z)) \Rightarrow \exists x\exists z(S(x,z) \wedge \neg\hat{Q}(x,z)) \vee \exists x\forall y\neg S(x,y)$.

□

Posledica 2.2.10 *Sledeća S-formula je valjana:*

$$\{P\}S\{\hat{Q}\} \wedge \{P\}S\{\neg\hat{Q}\} \ \Leftrightarrow \ (P \Leftrightarrow \phi).$$

Dokaz.

Na osnovu teoreme 2.2.9.b.), dobijamo:
$\forall x\neg P(x)$,
tj.
$P \Leftrightarrow \phi$.

□

Posledica 2.2.11 *Sledeća S-formula je valjana:*

$$[\{P\}S\{\neg\hat{Q}\} \Rightarrow \neg\{P\}S\{\hat{Q}\}] \;\Leftrightarrow\; \neg(P \Leftrightarrow \phi).$$

Dokaz.

Na osnovu teoreme 2.2.9.c.), dobijamo:
$\exists x P(x)$
$\equiv\; \neg(\forall x \neg P(x))$,
tj.
$\neg(P \Leftrightarrow \phi)$.

□

2.3 Najuži postuslovi *sp* i $\hat{sdp}$

Sada ćemo definisati S-predikate *sp* i $\hat{sdp}$ koje redom nazivamo najužim statičkim i najužim dinamičkim postuslovima.

Definicija 2.3.1 (Najuži statički postuslov) *Neka je S garantovano terminirajuća S-relacija. Najuži statički postuslov S-relacije S u odnosu na preduslov P jeste S-predikat $sp(S,P)$ ako važi:*
(SP_1) $\quad \{P\}S\{sp(S,P)\}$,
(SP_2) $\quad \{P\}S\{Q\} \;\Rightarrow\; \forall z(sp(S,P)(z) \Rightarrow Q(z))$.

Definicija 2.3.2 (Najuži dinamički postuslov) *Neka je S garantovano terminirajuća S-relacija. Najuži dinamički postuslov S-relacije S u odnosu na preduslov P jeste S-predikat $\hat{sdp}(S,P)$ ako važi:*
(SDP_1) $\quad \{P\}S\{\hat{sdp}(S,P)\}$,
(SDP_2) $\quad \{P\}S\{\hat{Q}\} \;\Rightarrow\; \forall x \forall z(\hat{sdp}(S,P)(x,z) \Rightarrow \hat{Q}(x,z))$.

Teorema 2.3.3 (Veza između *sp* i $\hat{sdp}$) *Neka je $\hat{sdp}(S,P)$ najuži dinamički, a $sp(S,P)$ najuži statički postuslov S-relacije S u odnosu na preduslov P. Tada važi:*

$$\hat{sdp}(S,P) \Leftrightarrow sp(S,P).$$

Dokaz.

Na osnovu (SP_1) iz definicije 2.3.1, važi:
$\{P\}S\{sp(S,P)\}$.
Na osnovu (SDP_1) iz definicije 2.3.2, važi:

$\{P\}S\{\hat{sdp}(S,P)\}$.
Na osnovu osobine (SP_2) *iz definicije 2.3.1 i osobine* (SDP_2) *iz definicije 2.3.2 dobijemo:*
$sp(S,P) \Rightarrow \hat{sdp}(S,P)$,
$\hat{sdp}(S,P) \Rightarrow sp(S,P)$.
Pošto smo dokazali implikacije u oba smera, zaključujemo da važi ekvivalencija, čime je polazna teorema dokazana.

□

Teorema 2.3.4 (Teoreme o najužim postuslovima) *Neka je data S-relacija S i preduslov P i neka S garantovano terminira počev od svakog početnog stanja x u kojem važi P(x). Tada važi:*

a.) $P \wedge S \equiv \hat{sdp}(S,P)$, *gde je* $\hat{sdp}(S,P)$ *najuži dinamički postuslov S-relacije S u odnosu na preduslov P.*

b.) $P \wedge S \equiv sp(S,P)$, *gde je* $sp(S,P)$ *najuži statički postuslov S-relacije S u odnosu na preduslov P.*

Dokaz.

a.) Dokazaćemo da $P \wedge S$ *zadovoljava osobine* (SDP_1) *i* (SDP_2) *iz definicije 2.3.2:*

(SDP_1) *Pošto S garantovano terminira počev od stanja x u kojem važi P(x), tada važi:*
$\forall x[P(x) \Rightarrow \exists y S(x,y)] \equiv \top$.
Na osnovu teoreme 2.2.1, dobijamo:
$\{P\}S\{P \wedge S\} \leftrightarrow \forall x[P(x) \Rightarrow \exists y S(x,y)] \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow P(x) \wedge S(x,z)]$
$\equiv \top \wedge \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow P(x) \wedge S(x,z)]$
$\equiv \forall x \forall z[P(x) \wedge S(x,z) \Rightarrow P(x) \wedge S(x,z)]$
$\equiv \top$,
čime je dokazana osobina (SDP_1).

(SDP_2) *Pretpostavimo suprotno, odnosno pretpostavimo da važi:*
$\{P\}S\{\hat{Q}\} \;\Rightarrow\; \forall x \forall z(S(x,z) \Rightarrow \hat{Q}(x,z)) \;\equiv\; \bot$, (1)
odakle sledi:
$\{P\}S\{\hat{Q}\} \;\equiv\; \top$, (2)
$\forall x \forall z(S(x,z) \Rightarrow \hat{Q}(x,z)) \;\equiv\; \bot$. (3)
Na osnovu S-formule (2)*, dobijamo:*
$\{P\}S\{\hat{Q}\} \leftrightarrow \forall x[P(x) \Rightarrow \exists y S(x,y) \wedge \forall z(S(x,z) \Rightarrow \hat{Q}(x,z))] \;\equiv\; \top$. (4)
Iz S-formule (4) *dobijamo:*
$\forall x P(x) \;\equiv\; \top$, (5)
$\forall x \exists y S(x,y) \wedge \forall x \forall z(S(x,z) \Rightarrow \hat{Q}(x,z)) \;\equiv\; \top$, (6)

Iz S-formule (6) *dobijamo:*
$\forall x \exists y S(x,y) \ \equiv \ \top,$ (7)
$\forall x \forall z (S(x,z) \Rightarrow \hat{Q}(x,z)) \ \equiv \ \top,$ (8)
gde dolazimo do kontradikcije, jer S-formula (3) *je u suprotnosti sa* (8)*. Time smo dokazali da pretpostavka* (1) *ne važi, pa zaključujemo da osobina* (SDP_2) *važi.*

b.) Na osnovu teoreme 2.3.3 važi:
$\hat{sdp}(S,P) \Leftrightarrow sp(S,P)$.
Na osnovu teoreme 2.3.4.a.) dobijamo:
$(P \wedge S) \Leftrightarrow sp(S,P)$.

□

Teorema 2.3.5 (Zakon jedinstvene konsekvencije) *Sledeća S-formula je valjana:*

$$\forall x[(P(x) \Rightarrow \exists y S(s,y)) \wedge \forall z((P(x) \wedge S(x,z)) \Rightarrow Q(z))] \ \Rightarrow \ \{P\}S\{Q\}.$$

Dokaz.

Na osnovu teoreme 2.3.4.a.) sledi:
$\forall x(P(x) \Rightarrow \exists y S(s,y)) \ \Rightarrow \ \{P\}S\{P \wedge S\}$.
Nakon toga, iz S-formule:
$\{P\}S\{P \wedge S\} \wedge \forall x \forall z((P(x) \wedge S(x,z)) \Rightarrow Q(z))$,
zamenom $P \wedge S$ *sa* $\hat{R}$ *u teoremi 2.2.2.c.), dobijamo:*
$\{P\}S\{Q\}$.

□

Teorema 2.3.6 (Teoreme o *S*-relacijama) *Sledeće S-formule su valjane:*

a.) $\{P\}S_1\{P \wedge S_1\} \wedge \forall x[P(x) \Rightarrow \exists y(S_1(x,y) \Rightarrow S_2(x,y))] \ \Rightarrow \ \{P\}S_2\{P \wedge S_2\}$,

b.) $\{P_1\}S_1\{P_1 \wedge S_1\} \wedge \forall x \exists y[(P_1(x) \Rightarrow S_1(x,y)) \Rightarrow (P_2(x) \Rightarrow S_2(x,y))] \ \Rightarrow$
$\{P_2\}S_2\{P_2 \wedge S_2\}$,

c.) $\{P_1\}S_1\{P_1 \wedge S_1\} \wedge \forall x_1 \exists y_1 \forall x_2 \exists y_2[(P_1(x_1) \Rightarrow S_1(x_1,y_1)) \Rightarrow (P_2(x_2) \Rightarrow S_2(x_2,y_2))] \ \Rightarrow$
$\{P_2\}S_2\{P_2 \wedge S_2\}$.

Dokaz.

a.) Na osnovu aksiome (A_2) *leva strana implikacije se može napisati ovako:*
$\{P\}S_1\{P \wedge S_1\} \wedge \forall x \exists y[(P(x) \Rightarrow S_1(x,y)) \Rightarrow (P(x) \Rightarrow S_2(x,y))]$,

odakle sledi:
$\forall x \exists y(P(x) \Rightarrow S_1(x,y)) \wedge [\forall x \exists y(P(x) \Rightarrow S_1(x,y)) \Rightarrow \forall x \exists y(P(x) \Rightarrow S_2(x,y))]$.
Na osnovu pravila modus ponens (MPN) *zaključujemo da važi:*
$\forall x \exists y(P(x) \Rightarrow S_2(x,y))$,
odakle, na osnovu teoreme 2.3.4.a.), važi:
$\{P\}S_2\{P \wedge S_2\}$.

b.) Leva strana implikacije se može napisati ovako:
$\forall x \exists y(P_1(x) \Rightarrow S_1(x,y)) \wedge \forall x \exists y[(P_1(x) \Rightarrow S_1(x,y)) \Rightarrow (P_2(x) \Rightarrow S_2(x,y))]$.
Na osnovu pravila modus ponens (MPN) *zaključujemo da važi:*
$\forall x \exists y(P_2(x) \Rightarrow S_2(x,y))$,
odakle, na osnovu teoreme 2.3.4.a.), važi:
$\{P_2\}S_2\{P \wedge S_2\}$.

c.) Leva strana implikacije se može napisati ovako:
$\forall x_1 \exists y_1(P_1(x_1) \Rightarrow S_1(x_1,y_1)) \wedge \forall x_1 \exists y_1 \forall x_2 \exists y_2[(P_1(x_1) \Rightarrow S_1(x_1,y_1)) \Rightarrow (P_2(x_2) \Rightarrow S_2(x_2,y_2))]$.
Na osnovu pravila modus ponens (MPN) *zaključujemo da važi:*
$\forall x_2 \exists y_2(P_2(x_2) \Rightarrow S_2(x_2,y_2))$,
odakle, na osnovu teoreme 2.3.4.a.), važi:
$\{P_2\}S_2\{P \wedge S_2\}$.

□

2.4 Teoreme o najužem statičkom postuslovu *sp*

Analogno sa izlaganjem o *wp* u poglavlju 1.6, ovde ćemo dokazati osnovne osobine najužeg statičkog postuslova *sp*. Pre svega, dokazaćemo jedinstvenost *sp* do nivoa ekvivalencije, a zatim, teoreme o najužem postuslovu *sp*, a to su redom zakon isključenja čuda, monotonosti, konjunkcije, disjunkcije i negacije. Pomoću *sp* dokazaćemo teoremu o totalnoj korektnosti i najzad, teoremu koja opisuje vezu između *wp* i *sp*.

Teorema 2.4.1 (Teorema o jedinstvenosti najužeg postuslova) *Najuži postuslov S-relacije S u odnosu na preduslov P, u oznaci* $sp(S,P)$*, je jedinstven do nivoa ekvivalencije.*

Dokaz.

Pretpostavimo suprotno, odnosno pretpostavimo da za S-relaciju S postoje dva najuža postuslova u odnosu na preduslov P za koja važi:
$sp_1(S,P) \Leftrightarrow sp_2(S,P) \equiv \bot$.
Na osnovu (SP_1) *iz definicije 2.3.1, dobijamo:*
$\{P\}S\{sp_1(S,P)\} \equiv \top$,
$\{P\}S\{sp_2(S,P)\} \equiv \top$.

Na osnovu (SP_2)*, dobijamo:*
$sp_2(S,P) \Rightarrow sp_1(S,P)$,
$sp_1(S,P) \Rightarrow sp_2(S,P)$,
odakle zaključujemo da važi:
$sp_1(S,P) \Leftrightarrow sp_2(S,P) \equiv \top$,
što je suprotno polaznoj pretpostavci, pa osnovu toga zaključujemo da polazna teorema važi.

□

Teorema 2.4.2 (Zakon isključenja čuda) *Sledeća S-formula je valjana:*

$sp(S,\phi) \equiv \phi$.

Dokaz.

Na osnovu teoreme 2.3.4.b.), dobijamo:
$sp(S,\phi) \equiv \phi \wedge S$,
odnosno
$sp(S,\phi) \equiv \phi$.

□

Teorema 2.4.3 (Zakon monotonosti) *Sledeća S-formula je valjana:*

$(P \Rightarrow R) \;\Rightarrow\; (sp(S,P) \Rightarrow sp(S,R))$.

Dokaz.

Na osnovu teoreme 2.3.4.b.), dobijamo:
$(P \Rightarrow R) \;\Rightarrow\; ((P \wedge S) \Rightarrow (R \wedge S))$.
Na osnovu teoreme (T_{18})*, dobijamo:*
$(P \Rightarrow R) \;\Rightarrow\; ((P \Rightarrow R) \wedge (S \Rightarrow S))$
$\equiv (P \Rightarrow R) \;\Rightarrow\; ((P \Rightarrow R) \wedge \top)$
$\equiv (P \Rightarrow R) \;\Rightarrow\; (P \Rightarrow R)$
$\equiv (P \Rightarrow R) \;\Rightarrow\; (P \Rightarrow R)$
$\equiv \top$.

□

Teorema 2.4.4 (Zakon konjunkcije) *Sledeća S-formula je valjana:*

$sp(S,P) \wedge sp(S,R) \Leftrightarrow sp(S, P \wedge R)$.

Dokaz.

Na osnovu teoreme 2.3.4.b.), dobijamo:
$sp(S,P) \wedge sp(S,R) \Leftrightarrow (P \wedge S) \wedge (R \wedge S)$
$\equiv sp(S,P) \wedge sp(S,R) \Leftrightarrow (P \wedge S \wedge R \wedge S)$
$\equiv sp(S,P) \wedge sp(S,R) \Leftrightarrow ((P \wedge R) \wedge S)$,
odakle, opet na osnovu teoreme 2.3.4.b.), dobijamo:
$sp(S,P) \wedge sp(S,R) \Leftrightarrow sp(S, P \wedge R)$.

□

Teorema 2.4.5 (Zakon disjunkcije) *Sledeća S-formula je valjana:*

$sp(S,P) \vee sp(S,R) \Leftrightarrow sp(S, P \vee R)$.

Dokaz.

Na osnovu teoreme 2.3.4.b.), dobijamo:
$sp(S,P) \vee sp(S,R) \Leftrightarrow (P \wedge S) \vee (R \wedge S)$
$\equiv sp(S,P) \wedge sp(S,R) \Leftrightarrow ((P \vee R) \wedge (S \vee S))$,
$\equiv sp(S,P) \wedge sp(S,R) \Leftrightarrow ((P \vee R) \wedge S)$,
odakle, opet na osnovu teoreme 2.3.4.b.), dobijamo:
$sp(S,P) \wedge sp(S,R) \Leftrightarrow sp(S, P \vee R)$.

□

Teorema 2.4.6 (Zakon negacije) *Sledeća S-formula je valjana:*

$\neg(sp(S,P) \wedge sp(S,\neg P))$.

Dokaz.

Na osnovu teoreme 2.3.4.b.), dobijamo:
$\neg((P \wedge S) \wedge (\neg P \wedge S))$
$\equiv \neg(P \wedge S \wedge \neg P \wedge S)$,
$\equiv \neg(P \wedge \neg P \wedge S)$,
$\equiv \neg(\bot \wedge S)$,
$\equiv \neg(\bot)$,
$\equiv \top$.

□

Teorema 2.4.7 (Teorema o totalnoj korektnosti) *Sledeća S-formula je valjana:*

$\{P\}S\{Q\} \Leftrightarrow (sp(S,P) \Rightarrow Q)$.

Dokaz.

Implikacija s leva u desno direktno sledi iz osobine (SP_2) *iz definicije 2.3.1. Potrebno je još dokazati implikaciju s desna u levo. S obzirom da na osnovu* (SP_1) *važi:*
$\{P\}S\{sp(S,P)\} \equiv \top$,
iz S-formule:
$\{P\}S\{sp(S,P)\} \wedge (sp(P,S) \Rightarrow Q)$,
sledi
$\{P\}S\{Q\}$,
čime je dokazana implikacija s desna u levo. Pošto smo dokazali implikacije u oba smera, zaključujemo da važi ekvivalencija, čime je polazna teorema dokazana.

□

Teorema 2.4.8 (Veza između wp **i** sp**)** *Sledeća S-formula je valjana:*

$[P \Leftrightarrow wp(S,Q)] \Leftrightarrow [Q \Leftrightarrow sp(S,P)]$.

Dokaz.

Na osnovu osobina (WP_1) *iz definicije 1.6.1 i* (SP_1) *iz definicije 2.3.1 važi:*
$[P \Leftrightarrow wp(S,Q)] \;\Rightarrow\; \{P\}S\{Q\}$,
$[Q \Leftrightarrow sp(S,P)] \;\Rightarrow\; \{P\}S\{Q\}$,
odakle dobijamo:
$\{P\}S\{Q\} \Leftrightarrow \{P\}S\{Q\}$
$\equiv \top$.

□

Glava 3

Klasa, objekat i invarijanta

Misao čini čovekovu veličinu.
Blez Paskal (1623–1662)

3.1 Uvod

Proučavanje neke nove naučne i stručne oblasti počinje sa upoznavanjem osnovnih pojmova i termina, odnosno sa proučavanjem njihovih definicija. Nažalost, objektno orijentisano programiranje je nastajalo evolutivnim putem, kao odgovor sa jedne strane na sve veće zahteve softverskog tržišta, a sa druge strane zbog sâmog progresa u nauci i načinu razmišljanja. Prema tome, u toku izgrađivanja teorije i prakse uspostavljeno je faktičko stanje u kojem postoje različiti pogledi na fundamentalne koncepte objektne metodologije – objekat i klasu. Takođe, ni terminologija u objektnoj metodologiji nije zadovoljavajuća, jer često za isti pojam postoji nekoliko termina (npr. objekat se često naziva i instancom klase). Pored sinonimije, u objektnoj metodologiji postoji još veći problem, a to je neoprezna upotreba termina koji su preuzeti iz drugih naučnih oblasti (npr. najčešće iz filozofije), što može da izazove pogrešnu asocijaciju [71]. Tipičan primer jeste termin *objekat* koji obično označava stvar, predmet ili korelat subjektu [74]. U objektnoj metodologiji za termin objekat postoji više različitih definicija [68], kao što su:

- Objekat predstavlja individuu, pojedinost koja se može identifikovati, jedinicu ili entitet, realan ili apstraktan, sa dobro definisanom ulogom u domenu problema [96],
- Objekat je bilo šta što poseduje oštre i jasno definisane granice [20],
- Objekat je stvar koja se može identifikovati i koja igra određenu ulogu u odnosu na zahtev za izvršavanje operacija [71],
- Objekat je kolekcija operacija koje dele stanje [71],

- Objekat je inkapsulacija skupa operacija koje se mogu eksterno pobuđivati i stanja koje pamti efekte primene metoda [71],
- Objekat je instanca klase u vreme izvršavanja [76],
- Objekat je model entiteta koji ima identitet, stanje i ponašanje, gde se pod entitetom podrazumeva postojanje nečeg [71][70].

Slična situacija je kada je u pitanju termin *klasa* i mogu se uočiti tri osnovna aspekta klase [71]:

- Klasa po definiciji služi za grupisanje objekata,
- Klasa ima namenu da opredeli strukturu i ponašanje svih objekata koji joj pripadaju (tj. klasa ima prirodu šeme ili šablona),
- Klasa služi kao generator objekata.

Očigledno, navedene definicije imaju prilično „slobodnu" formu, a posebna mana im je što ne omogućavaju da se termini objekat i klasa definišu nezavisno jedan od drugog. Definicija koja definiše objekat kao model entiteta je još najprikladnija, ali otvara pitanje šta je to *entitet*, a šta je *model* i kako koristiti te termine, a ne doći u koliziju sa već usvojenom terminologijom u filozofiji. Međutim, čak i ta definicija ima jedan očigledan problem, a to je da softverski inženjer, prilikom pisanja programa, ne barata konkretnim jedinicama posmatranja (entitetima), već barata mislima o njima i njih modeluje.

U ovom izlaganju ćemo izložiti konceptualne definicije klase i objekta, koje omogućavaju da se termini objekat i klasa definišu nezavisno jedan od drugog. Naše izlaganje počinje i bazira se na definiciji *pojma* ili *koncepta*, koji predstavlja misao o bitnim karakteristikama jedinice posmatranja i na taj način dobijamo realniju sliku o procesu modelovanja. Takođe, zahvaljujući ovakvom pristupu možemo jasno definisati šta je to *softverski model pojma* koji projektant stvara u odnosu na domen problema.

3.2 Individualni i klasni pojam

Prilikom stvaranja objektno orijentisanog programa softverski inženjeri najviše vremena će posvetiti analizi i modelovanju. Međutim, ovde treba napomenuti činjenicu da se softverski inženjeri bave mislima o učesnicima u informacionom sistemu, odnosno jedinica posmatranja se tretira preko misli o njoj, tačnije preko misli o njenim bitnim karakteristikama. Na primer, projektanti informacionog sistema fakulteta ne barataju sa konkretnim studentima i nastavnicima, već koriste misli o njima, odnosno misli o njihovim bitnim karakteristikama. Na osnovu toga, zaključujemo da su klasa i objekat direktno vezani za misli i da nije potreban nikakav posrednik u vidu entiteta ili nečeg trećeg, pa će ovo izlaganje započeti sledećom definicijom:

- Misao o bitnim karakteristikama predmeta jeste **pojam** ili **koncept** [85],

pri čemu se reč „predmet" ovde koristi u najširem smislu i označava predmet posmatranja i/ili razmišljanja. Svaki predmet u najširem smislu poseduje određene karakteristike koje možemo podeliti na bitne i nebitne. Misao o bilo kojoj karakteristici se naziva **oznaka**, a specijalno, misao o bitnim karakteristikama se naziva **bitna oznaka**. Nebitne karakteristike se mogu izvesti iz bitnih karakteristika. Na primer, bitne karakteristike pojma TROUGAO jesu *biti konveksni mnogougao* i *imati tri stranice*, dok jednakost visina kod jednakostraničnog trougla nije bitna karakteristika, jer sledi iz osobine jednakosti stranica.

Pojam ima sadržaj i opseg [85]. Sadržaj pojma je skup njegovih bitnih karakteristika. Na primer, sadržaj pojma FUNKCIJA čine osobine *biti relacija* i *ako* (a, b) *i* (a, c) *pripadaju funkciji, tada važi* $b = c$. Neka su A i B dva pojma. Ako za pojam B možemo reći svako B je istovremeno i A, tada je A generički (viši) pojam u odnosu na B, a B je vrsni (niži) pojam u odnosu na A. Opseg pojma jeste skup njegovih vrsnih pojmova. Na primer, opseg pojma TROUGAO čine pojmovi JEDNAKOSTRANIČNI TROUGAO, JEDNAKOKRAKI TROUGAO, PRAVOUGLI TROUGAO itd.

Sadržaj i opseg pojma se nalaze u obrnutoj srazmeri, odnosno niži pojam ima manji opseg, ali mu je sadržaj veći od višeg pojma i obrnuto, viši pojam ima veći opseg, ali mu je sadržaj manji od nižeg pojma. Ovaj zaključak je logičan, jer se niži pojmovi dobijaju iz viših dodavanjem semantike u vidu oznaka, ali istovremeno se na taj način sužava opseg, jer je niži pojam specifičniji.

Pojmove možemo klasifikovati na razne načine, ali za naše dalje izlaganje važna je podela na **individualne** i **klasne pojmove**. Individualni pojmovi se odnose na individualne predmete. Individualni predmeti koji imaju zajedničke oznake čine klasu, a misao o datoj klasi jeste klasni pojam. Na primer, individualni pojmovi HAJDN, MOCART i BETOVEN jesu misli o poznatim kompozitorima, kojima je zajedničko to da pripadaju periodu klasicizma i da je njihov stvaralački rad više ili manje vezan za grad Beč. Na osnovu zajedničkih oznaka, oni čine klasu, a misao o toj klasi jeste klasni pojam BEČKI KLASIČAR.

Očigledno, individualni pojmovi mogu i ne moraju biti realni, dok klasni pojmovi nikada nisu realni. Na primer, klasni pojam BEČKI KLASIČAR nije realan pojam, dok individualni pojam MOCART je realan (kompozitor Mocart je postojao u periodu 1756–1791 godine). Međutim, i individualni i klasni pojam TROUGAO nisu realni.

Metodom apstrakcije od klasnih pojmova mogu se dobiti još apstraktniji klasni pojmovi. Na taj način se dobija hijerarhija klasnih pojmova, koja po obliku podseća na stablo. Na primer, klasni pojam BEČKI KLASIČAR pripada opsegu klasnog pojma KOMPOZITOR, a ovaj opet pripada opsegu klasnog pojma UMETNIK itd.

Ustaljena praksa je da učenje objektno orijentisanog programiranja počinje sa razmatranjem entiteta. Po ISO definiciji *entitet* je bilo koja konkretna ili apstraktna stvar koja postoji, koja je postojala ili je mogla postojati, uključujući i veze između ovih stvari. U ovom izlaganju entitet se tretira kao sinonim za jedinicu posmatranja i smatra se da

nema potrebe stavljati ga u prvi plan, jer kao što je na početku ovog poglavlja rečeno, projektant informacionog sistema barata pojmovima, tj. mislima. Zbog toga, u ovom izlaganju u prvom planu se nalazi *pojam*. Najzad, možemo reći da svet u najopštijem smislu, koji projektant razmatra, čine pojmovi, koji imaju oznake i koji međusobno stoje u nekom odnosu, tj. u nekoj vezi.

3.3 Modelovanje

U prethodnom poglavlju smo rekli šta je to sadržaj pojma, ali ćemo ovde reći da određivanje sadržaja pojma nije nimalo lak posao. Na primer, pojam GRAĐANIN ima bitne oznake *ime i prezime*, *matični broj*, *adresu* i *broj lične karte*, ali isto tako svaki građanin ima *visinu*, *težinu*, *boju kose*, *broj cipela* itd. Upravo ovde se pokazuje opravdanost uvođenja domena problema. Naime, za razliku od logičara, projektant softvera ne mora voditi računa o svim bitnim oznakama pojma, već samo o onima koje su relevantne, tj. one koje su od interesa za dati domen problema koji je predmet analize. Na primer, projektant informacionog sistema poreske uprave će za pojam GRAĐANIN izabrati *ime i prezime*, *matični broj*, *adresu*, *podatke o prihodima* i sl., ali sigurno neće izabrati *visinu* ili *težinu*, iako svaki građanin poseduje pomenute osobine. S druge strane, projektant informacionog sistema zdravstvene ustanove će pored *imena i prezimena*, *matičnog broja*, *adrese* i sl., sigurno izabrati i *visinu* i *težinu*, jer ove osobine su neophodne lekaru prilikom određivanja terapije, ali neće izabrati *podatke o prihodima*.

- Oznake pojma koje su od interesa u datom domenu problema zovu se **relevantne oznake pojma** [51][71].

Uvođenjem relevantnih oznaka, umesto bitnih oznaka, rešava se problem donošenja odluke da li je neka oznaka bitna ili ne. Na primer, *matični broj* građanina je oznaka koja je bitna u zemljama u kojima ona postoji, ali postoje zemlje u kojima ona ne postoji pa samim tim nije ni bitna. Dakle, uvođenje relevantnih oznaka koje su vezane za dati domen problema igra ključnu ulogu da naše izlaganje dobije i praktičan karakter, što je naročito važno za softverski inženjering, koji podrazumeva rešavanje praktičnih (inženjerskih) problema. Sada možemo definisati model, pri čemu treba istaći da sâm termin model, kao homomorfna slika nečeg, ima upotrebu u raznim situacijama (npr. maketa zgrade predstavlja model zgrade koja će biti sagrađena), međutim, softverski inženjer bavi se modelovanjem pojma, tj. modelovanjem misli. Prema tome, ovde ćemo definisati softversko modelovanje (kraće *modelovanje*) i softverski model (kraće *model*):

- Postupak izbora konačnog broja relevantnih oznaka pojma u odnosu na dati domen problema naziva se **softversko modelovanje**, a dobijeni konačni skup relevantnih oznaka naziva se **softverski model**.

Iz prethodne definicije možemo zapaziti da je modelovanje postupak kojim se dobija uprošćena slika pojma u datom domenu problema, pri čemu taj postupak nije jednoznačno

određen, tj. isti pojam se može modelovati na više različitih načina. Na primer, pojam GRAĐANIN se može modelovati konačnim skupom oznaka {*ime i prezime*, *matični broj*}, ali isto tako bi se mogao modelovati konačnim skupom {*ime i prezime*, *matični broj*, *adresa*, *broj pasoša*}. Drugim rečima, projektant je taj koji donosi odluku kako će modelovati neki pojam. Veoma je važno da dobijeni skup relevantnih oznaka bude potrebno i dovoljno deskriptivan, jer to je jedan od ključnih preduslova da softver, kao finalni proizvod, bude kvalitetan. Zbog toga projektant više vremena posvećuje modelovanju, a ne sâmom pisanju kôda.

3.4 Klasa i objekat

U ovom poglavlju ćemo navesti konceptualne definicije klase i objekta, ali ćemo prvo skrenuti pažnju na to da ove definicije imaju dve velike prednosti u odnosu na sve definicije koje smo spomenuli u uvodnom delu ovog izlaganja. Naime, konceptualne definicije su zasnovane na pojmovima (konceptima), tj. na dobro razrađenom i jasnom sistemu termina i njihovih značenja. Pored toga, konceptualne definicije klase i objekta su ravnopravne u semantičkom smislu, tj. klasa se ne definiše preko objekta niti obrnuto. Prvo ćemo navesti konceptualnu definiciju klase:

- Klasa objekata (kraće *klasa*) jeste softverski model klasnog pojma.

Oznake klasnog pojma ćemo podeliti u dve grupe: oznake u užem smislu i oznake tipa „sadrži klasni pojam". Oznake u užem smislu ćemo ovde zvati **odlikama**. Klasni pojam može u svom sastavu da sadrži i druge klasne pojmove, a njih ćemo ovde nazvati **fragmentima**. Na primer, klasni pojam DUŽ ima dva fragmenta (dva temena) koja jesu dva klasna pojma TAČKA. Na primer, klasni pojam AUTOMOBIL ima odliku *boja*. Odlike mogu biti **deskriptivne**, kao što su *boja*, *masa* i sl., ali i **operacione**, kao što su *mogućnost kretanja*, *mogućnost letenja* i sl. Sada ćemo navesti konceptualnu definiciju objekta:

- Objekat je softverski model individualnog pojma.

Očigledno, konceptualni pogled postavlja klasu i objekat u ravnopravni položaj. Razlika je samo u tome što klasa odgovara klasnom, a objekat individualnom pojmu. Klasa i objekat povezani su jednom pretpostavkom koja zapravo ima snagu postulata i glasi:

- Za svaki objekat postoji klasa koja poseduje sve njegove relevantne oznake i tada kažemo da objekat pripada datoj klasi.

Na primer, svaki pojedinačni objekat *jednakokraki trougao* (sa konkretnim vrednostima dužina stranica) pripada klasi *Jednakokraki trougao*. Fragmenti i odlike klasnog

pojma se pojavljuju u klasnoj varijanti, a fragmenti i odlike individualnog pojma u individualnoj varijanti. Na primer, ako je odlika klase *boja*, tada kod individue se ona pojavljuje kao *bela* ili *zelena*.

Pojmovi mogu biti složeni što znači da njihovi fragmenti mogu imati svoje fragmente, a ovi opet svoje itd. Na primer, udžbenik se sastoji od poglavlja, poglavlja sadrže pasuse, pasusi linije, a linije sadrže znake. Skup sastavljen od pojma, njegovih fragmenata, pa dalje njihovih fragmenata itd., uređen relacijom „biti fragment" čini **strukturu** takvog pojma. Ukoliko pojam ne sadrži fragmente, već samo odlike, tada kažemo da je takav pojam jednostavne strukture. Ukoliko pak pojam sadrži bar jedan fragment, tada kažemo da je takav pojam složene strukture. Na primer, pojam AUTOMOBIL sadrži fragment MOTOR, pa kažemo da ima složenu strukturu. Na primer, pojam TAČKA sadrži samo odlike, a to su vrednosti koordinata, pa kažemo da ima jednostavnu strukturu. S obzirom da je objekat model individualnog pojma, tada modelovanjem od strukture pojma dobijamo strukturu objekta. Analogno tome, pošto klasa predstavlja model klasnog pojma, modelovanjem od strukture klasnog pojma dobijamo strukturu klase. Prema tome, možemo govoriti o jednostavnoj i složenoj strukturi objekta, odnosno klase.

Esencijalne osobine objekta su da ima identitet, stanje i ponašanje. Konceptualna definicija nije u koliziji sa navedenim esencijalnim osobinama. Naime, pošto svaki individualni pojam ima identitet koji ga jednoznačno određuje, ta osobina se preslikava i na objekat, kao njegov model, odnosno svaki objekat ima identitet koji ga jednoznačno određuje. Stanje objekta sa jednostavnom strukturom određuju njegove deskriptivne odlike. Stanje objekta sa složenom strukturom određuju njegove deskriptivne odlike, ali i stanja njegovih fragmenata. Ako sada ovaj zaključak razmotrimo sa druge tačke gledišta, tj. ako za objekat posmatramo skup stanja, tada zaključujemo da se deskriptivne odlike izvode iz stanja, tj. odlike su funkcije stanja. Najzad, došli smo do faze realizacije u nekom od objektno orijentisanih programskih jezika. U fazi realizacije deskriptivne odlike predstavljaće **podatke-članove** objekta, dok će fragmenti predstavljati **objekte-članove** objekta. Operacione odlike determinišu ponašanje objekta, koje će biti opisano u njegovim **metodama**. Podatke-članove i objekte-članove obično zovemo jednim imenom **polja**.

Objekti, odnosno klase sa jednostavnom strukturom sadrže isključivo podatke-članove. Podaci-članovi imaju odgovarajući tip, pa pošto svi programski jezici barataju sa konačnim tipovima, zaključujemo da je i njihov apstraktni skup stanja konačan. Od objekata, odnosno klasa sa jednostavnom strukturom su izgrađeni objekti, odnosno klase sa složenom strukturom, a ovih još složeniji itd. Na osnovu toga, zaključujemo da je i njihov apstraktni skup stanja konačan. Drugim rečima, bez obzira na složenost strukture, apstraktni skup stanja objekta, odnosno klase je konačan.

Kada se iz klasnog pojma izuzmu oznake nivoa klase, dobija se skup oznaka (fragmenata i odlika) koje poseduje svaki individualni pojam vezan za tu klasu. Imajući u vidu definiciju objekta, logično sledi da objekti iste klase imaju istu strukturu i isto ponašanje. Klasa i objekat stoje u identičnom odnosu u kojem su tip podataka i promenljiva u standardnim programskim jezicima. Na primer, promenljive tipa *int* predstavljaju primerke (pojave) tog tipa, kao što objekti predstavljaju primerke (pojave, instance) svoje klase.

Odavde je očigledno da konceptualna definicija klase i ovo izlaganje nisu u koliziji sa ključnim aspektima klase koji su navedeni u uvodnom delu ovog rada.

Najzad, u literaturi iz objektno orijentisanog programiranja često se koristi termin *atribut*, nažalost, ne na sasvim ispravan način. Atribut po definiciji označava bitnu oznaku. U ovom izlaganju pokazali smo da postoje deskriptivne i operacione oznake. Pojedini autori atributima nazivaju samo deskriptivne odlike, iako se suština objektne metodologije sastoji upravo u izjednačavanju svih odlika, kako deskriptivnih tako i operacionih. Na primer, operaciona odlika *mogućnost letenja* je bitna oznaka pojma AVION. Naravno, ima autora koji eksplicitno navode da atributi obuhvataju sve bitne oznake, kako deskriptivne tako i operacione [2].

3.5 Invarijanta

Od svih oznaka pojma, i to kako deskriptivnih, tako i operacionih, uočavamo one invarijantne. Na primer, za pojam TROUGAO, pri čemu je ovde svejedno da li se radi o individualnom ili klasnom pojmu, oznaka $a+b>c$ je invarijantna, gde su a, b i c deskriptivne oznake koje predstavljaju dužine stranica trougla. Isto tako, oznaka $\alpha+\beta+\gamma=180^\circ$, gde su α, β i γ deskriptivne oznake koje predstavljaju unutrašnje uglove trougla. Očigledno, bez obzira na promene (transformacije) koje pojam pretrpi, invarijantna oznaka uvek je važeća. Očigledno postoji beskonačno mnogo invarijantnih oznaka, a sve one zajedno grade suštinu ili esenciju pojma.

Modelovanjem se bira konačan skup relevantnih oznaka, i to kako deskriptivnih, tako i operacionih. Ujedno, to povlači i izbor svih invarijantnih oznaka koje su relevantne za dati model. Takve oznake ćemo zvati **invarijantnim relevantnim oznakama**, a očigledno može ih biti beskonačno mnogo. Sve invarijantne relevantne oznake opisuju suštinu pojma u datom domenu problema. Na primer, rekli smo da su oznake $a+b>c$ i $\alpha+\beta+\gamma=180^\circ$ invarijantne oznake, međutim, ukoliko modelujemo trougao tako da model čine jedino dužine stranica a, b i c, tada oznaka $a+b>c$ je relevantna invarijantna oznaka, dok $\alpha+\beta+\gamma=180^\circ$ nije.

U fazi realizacijom dobijamo objekat, odnosno klasu, sa konačnim apstraktnim skupom stanja. Međutim, invarijantna relevantna oznaka pojma i dalje ostaje važeća, samo ovaj put u obliku restrikcije nad usvojenim konačnim skupom apstraktnih stanja.

- Invarijanta u objektu predstavlja restrikciju invarijantne relevatne oznake individualnog pojma na usvojenom skupu apstraktnih stanja.

- Invarijanta u klasi predstavlja restrikciju invarijantne relevatne oznake klasnog pojma na usvojenom skupu apstraktnih stanja.

Na osnovu činjenice da invarijantna oznaka može biti kako deskriptivna, tako i operaciona, nakon modelovanja, dobijamo nekoliko vrsta invarijanata [72]:

- Invarijante polja – potiču od invarijantnih deskriptivnih relevatnih oznaka, a nakon modelovanja predstavljaju relaciju definisanu nad poljima. Na primer, $a + b > c$ je invarijanta polja, gde polja a, b i c predstavljaju dužine stranica trougla.

- Funkcionalne invarijante – potiču od invarijantnih operacionih relevantnih oznaka, a nakon modelovanja povezuju metode u smislu primene. Na primer, uzastopna primena metoda *push* i *pop* ostavlja stek u stanju u kojem je bio pre primene.

- Invarijante odnosa – tipičan primer jeste kardinatilet veza između pojmova, pa tako na primer, kardinalitet veze između pojmova DUŽ i TAČKA jeste $1 : 2$.

- Mešoviti oblici.

Posmatrajmo sada objekat jednostavne strukture, koji sadrži samo podatke-članove nekog tipa. Na primer, ako podatak-član a predstavlja dužinu stranice trougla tipa *realan broj*, postavlja se pitanje – da li taj tip odgovara odgovarajućoj relevantnoj oznaci pojma? Iako sve deluje logično, nažalost, odgovor je odričan, jer dužina stranice trougla može biti jedino tipa *dužina*. Takav tip, nažalost, ne postoji ni u jednom programskom jeziku, već moramo improvizovati i reći da je podatak-član a tipa *pozitivan realan broj*. Međutim, tu nije kraj problemu, jer postavlja se pitanje – da li bilo koja tri pozitivna realna broja predstavljaju dužine stranica trougla? Opet je odgovor odričan, jer ta tri broja moraju zadovoljavati teoremu o nejednakosti dužina stranica trougla. Drugim rečima, invarijantna relevantna oznaka pojma TROUGAO mora se očuvati. Sada je potpuno jasan problem, sa jedne strane imamo zahteve koji potiču od sâme prirode oznaka pojma, a druge strane imamo realnost koju nameće deskriptivna moć programskih jezika. Zbog toga, potpuno opravdano je podeliti konačni skup apstraktnih stanja na dva disjunktna podskupa, a to su skupovi *validnih* i *invalidnih* stanja. Na primer, kada je u pitanju trougao, stanje $(3, 4, 5)$ je validno, dok stanja $(-3, -4, -5)$ ili $(5, 1, 1)$ su invalidna.

Invarijanta jeste svaki predikat koji je tačan u svakom validnom stanju, dok u invalidnom stanju može i ne mora biti tačan. Posebno, predikat koji jednoznačno razdvaja skup validnih od skupa invalidnih stanja, zvaćemo stroga invarijanta. Drugim rečima, stroga invarijanta jeste svaki predikat koji je tačan u svakom validnom stanju, a netačan u svakom invalidnom stanju. Invarijanata i strogih invarijanata ima beskonačno mnogo. Međutim, pošto sve stroge invarijante jednoznačno razdvajaju skupove validnih i invalidnih stanja, zaključujemo da su sve stroge invarijante jedinstvene do nivoa ekvivalencije.

Do sada smo govorili o invarijantama koje potiču od invarijantnih relevantnih oznaka pojma, tj. potiču iz domena problema. Međutim, postoje i invarijante koje nastaju u fazi realizacije, iako ih ne mora biti u domenu problema. Na primer, kod steka koji je realizovan sekvencijalno sa kapacitetom C postoji ograničenje da ne može imati više elemenata od C, dok kod steka koji je realizovan spregnuto takvo ograničenje ne postoji.

Centralno mesto u statičkoj analizi semantike objektno orijentisanog programa zauzima semantika klase i postoji korepodencija između njih [67]. Mi ćemo se fokusirati na static class-level modular analysis [14] [88] [65], koja se može se kretati u dva pravca:

- *as prescribed* – invarijanta je unapred zadata, a korektnost ponašanja objekta, odnosno klase se dokazuje u odnosu na tako zadatu invarijantu [77] [61].
- *as described* – smatra se da se objekat, odnosno klasa korektno ponašaju, pa se iz semantičkog opisa metoda izvodi invarijanta [1] [31] [90] [66] [67].

U daljem izlaganju fokusiraćemo se na *as described* razmatranje invarijanata polja u klasi i razviti dve analize:

- analizu primenom statičkih postuslova ili kraće SP-analizu, koja se bazira na najužim statičkim postuslovima metoda, odnosno na validnim završnim stanjima.
- analizu primenom dinamičkih postuslova ili kraće DP-analizu, koja se bazira na najužim dinamičkim postuslovima metoda, odnosno na validnim prelazima stanja.

Glava 4

Analiza invarijanata polja u klasi

Pojam invarijante, po mom mišljenju, jedan je od najsvetlijih koncepata koji mogu da se nauče iz objektno orijentisane metodologije. Jedino kada izvedemo invarijantu (klase koju pišemo) ili kada pročitamo i razumemo invarijantu (klase koju je neko drugi napisao), tada zaista osećamo da znamo koja je namena klase.

Bertrand Mejer (1950–)

4.1 Uvod

U ovom izlaganju bavićemo se *as decribed* analizom i koristeći jedinstven pristup baziran na promenama apstraktnih stanja, pokazaćemo kako se invarijante izračunavaju iz dinamičkih postuslova metoda, a sâmu analizu nazvaćemo *analiza invarijanata u klasi primenom dinamičkih postuslova* ili kraće *DP-analiza*. *DP*-analiza jeste specijalan slučaj *S*-programskog računa [57], odnosno specijalan slučaj predikatske logike prvog reda [16] [48] [94]. *S*-račun povezuje Horovu logiku [32] [41] [35] sa predikatskom logikom i pokazuje da se Horova logika može izvesti iz aksioma i teorema predikatske logike prvog reda. Odavde zaključujemo da *DP*-analiza, kao specijalan slučaj *S*-računa, prezentuje primenu Horove logike u objektno orijentisanom ambijentu.

Za razliku od običnog postuslova $Q(z)$, kojeg ćemo ovde nazvati statičkim postuslovom i koji predstavlja funkciju samo završnog stanja, dinamički postuslov $\hat{Q}(x, z)$ predstavlja funkciju i početnog i završnog stanja. Da bi se videla očigledna prednost ovakvog pristupa, prvo ćemo razmotriti *analizu invarijanata primenom statičkih postuslova* ili kraće *SP-analizu* i kroz primere objasniti njene mane. Naime, pokazaćemo da statički postuslov nije pogodan za opisivanje semantike međudejstva metoda u klasi. Nakon toga, razmotrićemo *DP*-analizu, u okviru koje centralno mesto zauzima operacija generalizovane supstitucije. Primenom generalizovane supstitucije i teoreme Tarskog [102] pokazaćemo da se invarijanta može računati kao nepokretna tačka na apstraktnom skupu stanja klase. Da bi izlaganje imalo i praktičan karakter formulisaćemo rekurzivne algo-

ritme za izračunavanje različitih vrsta invarijanata i demonstrirati njihov rad kroz nekoliko primera.

Invarijanta se može posmatrati kroz ponašanje pojedinačnog objekta ili kroz ponašanje cele klase i predstavlja neki predikat koji je tačan u svakom stanju koje se smatra validnim, nezavisno od toga šta smatramo za stanje. Dakle, najcelishodnije je invarijante razmatrati prvo na apstraktnom prostoru stanja, a kasnije interpretirati prostor stanja na vrednostima polja. Zbog toga, SP i DP analize ćemo razviti na apstraktnom prostoru stanja, gde ćemo invarijante posmatrati kao predikate na apstraktnom prostoru stanja [44] [43] [57], a u interpretaciji na poljima klase pretvorićemo ih u logičke izraze [37] [38].

Centralno mesto u *as described* analizi, odnosno u izračunavanju invarijante zauzima teorema Tarskog o nepokretnoj tački [102] [22], pa se ranije ideje o izračunavanju invarijante u programu [17] ili u *while* petlji [39] mogu dovesti u vezu sa izračunavanjem invarijante u klasi. Ideju o nepokretnoj tački na putanji stanja programa [19] koristi Francesko Logozzo i u svojim radovima [65] [66] [67] iznosi mogućnost da se invarijanta u klasi može računati kao nepokretna tačka na putanji dostupnih stanja. Gledano sa aspekta S-računa, Logozzovo izračunavanje invarijante bazira se na izračunavanju najužeg statičkog postuslova *sp*. Ako se za sve metode izračuna najuži postuslov počev od inicijalnih stanja S_0, dobijaju se dostupna stanja ΔS_0, odakle formiramo skup $S_1 = S_0 \cup \Delta S_0$. Slično, od skupa S_1 se može dobiti skup S_2 itd. Ako se ovaj postupak rekurzivno ponavlja dobijamo putanju skupova dostupnih stanja. Očigledno, nepokretna tačka na ovoj putanji opisuje sva dostupna stanja, odnosno to je stroga invarijanta.

Ovde treba primetiti jedan važan detalj - najuži statički postuslov *sp* opisuje samo završna stanja i ako se on koristi tada se javlja problem opisa semantike međudejstva metoda (Primer 4.2.2). Posmatrajmo klasu:

```
public class K {
     public int n;
     public K() { n=0; }
     public void incN() { n++; }
}
```

Metoda *incN* terminira za $n \in \{\dots, -2, -1, 0, 1, 2, \dots\}$. Metoda *incN* „ne zna" da konstruktor ostavlja objekat u stanju u kojem je $n = 0$. Ako se uzastopno aktivira metoda *incN*, tada sledbenik „ne zna" u kojem stanju je njegov prethodnik ostavio objekat. Pošto Logozzo razmatra putanje skupova dostupnih stanja na taj način indirektno problem međudejstva je rešen. Takvo rešenje jeste prihvatljivo, ali treba napomenuti da se ono nalazi u okviru operacione semantike [69].

Naša ideja jeste da problem opisivanja semantike međudejstva metoda rešimo uvođenjem dinamičkih postuslova. Naš doprinos jeste u tome što ćemo pokazati da se invarijante u klasi mogu izračunavati u okviru predikatske logike prvog reda. Korist od toga je velika, jer predikatska logika predstavlja klasičnu i širokoprihvaćenu oblast matematike i kao takva

poseduje moćan matematički aparat.

Polazna pretpostavka u Logozzo-vom radu jeste da apstraktni skup stanja klase može biti beskonačan [67], a odatle sledi da i putanje skupova dostupnih stanja mogu biti beskonačne, odnosno da se invarijanta, u opštem slučaju ne može izračunati, već je potrebno uvoditi određene aproksimacije.

Naša polazna pretpostavka jeste da je apstraktni skup stanja klase U_K konačan i particijom ga delimo na skup validnih V_K i skup invalidnih N_K stanja. Opravdanje za takvu pretpostavku jeste činjenica da se u ovom radu bavimo *as described* analizom. Kao što smo ranije rekli, *as described* analiza podrazumeva korektno ponašanje klase, a to znači da svaka metoda garantovano terminira. Počev od takve pretpostavke, uvođenjem i primenom generalizovane supstitucije razvićemo jedinstven matematički mehanizam iz kojeg se direktno izvode rekurzivni algoritmi za izračunavanje različitih invarijanata (čiste invarijante objekta i klase, kao i mešovite invarijante). Time ćemo pokazati da je naše rešenje opštije, odnosno da Logozzovo rešenje predstavlja specijalan slučaj naše analize.

U poglavlju 4.2 prikazaćemo SP-analizu i kroz primere objasniti problem opisivanja semantike međudejstva metoda, a zatim, u poglavlju 4.3 razvićemo DP-analizu i rekurzivne algoritme za automatsko izračunavanje invarijante. Poglavlje 4.4 razmatra primenu rekurzivnih algoritama za izračunavanje različitih invarijanata (čiste invarijante objekta i klase, kao i mešovite invarijante) na interpretiranom skupu apstraktnih stanja i kroz konkretne primere demonstrira njihov rad. Takođe, ovo Poglavlje razmatra slučaj kada metode imaju parametre. U poglavlju 4.5 razmatramo izračunavanje invarijante vlasnika i komponente u različitim situacijama (slučaj kada jedan vlasnik sadrži jednu komponentu, odnosno slučaj kada više vlasnika dele jednu komponentu), a u poglavlju 4.6 izračunavanje invarijante natklase i potklase. Ujedno, analiziraćemo veze invarijanata kada su klase povezane klijentskim vezama, odnosno nasleđivanjem. Najzad, u poglavlju 4.7 prikazaćemo izračunavanje invarijante objekata sa varijabilnom strukturom, tj. objekata koji sadrže pokazivače/reference kojima se ostvaruje veza sa memorijskim prostorom koji se nalazi izvan objekta, odnosno negde na hipu. Ovakvim izborom i redosledom tema, naša analiza dobija kompletnu formu i može se koristi za analizu klasa i sa jednostavnom i sa složenom strukturom.

4.2 Osnovni elementi SP-analize

Klasi K se pridružuje konačan apstraktni skup stanja U_K. Skup svih polja klase K označavamo sa Φ_K i podrazumevamo da u klasi K postoji bar jedno polje, tj. da važi $\Phi_K \neq \emptyset$, pa prema tome važi i $U_K \neq \emptyset$.

Definicija 4.2.1 (Apstraktni skup stanja) *Apstraktni skup stanja U_K je neprazan konačan skup apstraktnih stanja pridružen klasi K.*

Neka su $\phi_1, \phi_2, \ldots, \phi_{n_\phi}$ polja klase K. Neka je D_{ϕ_i} skup vrednosti ili domen polja ϕ_i, gde je $i = 1, 2, \ldots, n_\phi$. Ako je polje klasnog tipa, tada se pod skupom vrednosti po-

drazumevaju odgovarajući objekti, kao i vrednost *null*. Interpretacija apstraktnog skupa stanje jeste bijekcija koja svakom stanju iz skupa U_K dodeljuje odgovarajući vektor iz skupa $D_{\phi_1} \times D_{\phi_2} \times \cdots \times D_{\phi_n}$.

Skup svih metoda u klasi K označavamo sa M_K i podrazumevamo da važi $M_K \neq \emptyset$. U ovoj analizi podrazumevamo da svaka promena stanja nastaje isključivo aktiviranjem neke metode, odnosno da ne postoje operacije tipa $obj.field = expression$, koje su inače neobjektne [77] [78] [79]. Takođe, podrazumevamo da su promene stanja svakog objekta (ili klase) diskretne i da se odvijaju u vremenu. To znači da tokom prelaza iz stanja u stanje invarijanta može i da se naruši, da bi tek po završenom transferu bila ponovo uspostavljena, a to vodi ka problemima vezanim za tzv. *call-back* ili *reentrancy*, gde do narušavanja invarijante može doći zbog rekurzivnih poziva [77]. Podrazumevamo da su konstruktori u klasama potpuni, tj. da izvršavaju celu proceduru kreiranja i podrazumevamo da nema alijasa [78].

Apstraktni skup stanja U_K particijom delimo na *skup dostupnih* i *skup nedostupnih stanja*. Skup dostupnih stanja klase čine sva stanja u kojima se objekat ili klasa mogu naći u toku svog životnog veka [84] [61]. Stanja koja nisu dostupna zovemo nedostupnim. Skup U_K particijom delimo na još jedan način na *skup stacionarnih* i *skup nestacionarnih stanja*. Objekat se nalazi u stacionarnom stanju kada nijedna metoda nad tim objektom nije aktivna. Klasa se nalazi u stacionarnom stanju kada nijedna metoda klase nije aktivna. Stanja koja nisu stacionarna zovemo nestacionarna stanja. Za potrebe naše analize neophodna su stanja koja se nalaze u preseku skupa dostupnih i skupa stacionarnih stanja. Stanje je validno ako i samo ako je dostupno i stacionarno. Stanje koje nije validno nazivamo invalidnim. Ako sa V_K označimo *skup validnih stanja*, a sa N_K *skup invalidnih stanja*, tada važi $U_K = V_K \bigcup N_K \wedge V_K \bigcap N_K = \emptyset$. U našem razmatranju podrazumevamo da je skup validnih stanja V_K neprazan.

U S-računu virtuelna mašina radi sa apstraktnim skupom stanja A. Kao što smo svako stanje virtuelne mašine interpretirali preko vrednosti programskih promenljivih, ovde ćemo interpretirati preko stanja objekata (klase). Posmatrajmo objekat *obj* klase K. Neka objekat u sebi sadrži polje ϕ celobrojnog tipa. Na primer, tada $x : (s : \phi = 0)$ označava da se virtuelna mašina nalazi u stanju $x \in A$ u kojem se objekat *obj* nalazi u stanju $s \in U_K$, koje je takvo da za polje ϕ važi $\phi = 0$. Drugim rečima, ovde povlačimo jasnu razliku između skupa apstraktnih stanja virtuelne mašine A i skupa apstraktnih stanja klase U_K.

Definicija 4.2.2 (Nulto stanje objekta) *Nulto stanje, u oznaci o, je validno kvazistanje u kojem se objekat nalazi pre konstruisanja, odnosno posle destruisanja. Nulti predikat, u oznaci O, je predikat koji je tačan samo u nultom stanju.*

Gledano sa aspekta klase K, nulto stanje o ne pripada skupu U_K, zato smo nulto stanje proglasili za *validno kvazistanje* i prudružili ga skupu U_K.

Neka je $m(in, io, out)$ metoda u kojoj su *in* ulazni parametri, *io* ulazno-izlazni parametri i *out* izlazni parametri. Označimo sa u apstraktni ulaz kojeg formiraju *in* i

io, a sa i apstraktni izlaz kojeg formiraju *io* i *out*. Neka je s apstraktno stanje. Očigledno, preduslov metode zavisi u i s, a postuslov od i i s. Međutim, ne gubeći opštost u daljem izlaganju, prvo ćemo iz analize izostaviti apstraktni ulaz u i izlaz i, tj. razmatranje ćemo ograničiti isključivo na metode koje nemaju parametre i povratnu vrednost. Na taj način će naša analiza biti jednostavnija, a zatim ćemo priču proširiti na metode koje imaju parametre i povratnu vrednost (poglavlje 4.4.1). S obzirom da ćemo na početku analize razmatrati samo metode koje nemaju parametre i povratnu vrednost, stanje virtuelne mašine $x \in A$ možemo poistovetiti sa stanjem objekta (klase) $s \in U_K$. Shodno tome, Hoareova formula totalne korektnosti (FTK) će u objektnom ambijentu izgledati ovako:

$$(OFTK) \qquad \{P\}S\{Q\} \leftrightarrow \forall s[P(s) \Rightarrow \exists t S(s,t) \wedge \forall r(S(s,r) \Rightarrow Q(r))],$$

gde su $s, t, r \in U_K$, P i Q su predikati (u daljem tekstu *predikati*) na skupu U_K, tj. $P : U_K \rightarrow \{\top, \bot\}$, a m su relacije (u daljem tekstu *metode*) na skupu U_K, tj. $m : U_K \times U_K \rightarrow \{\top, \bot\}$. Predikat ćemo interpretirati logičkim izrazom u kojem figuriraju polja, na primer, $P : n > 0$. Nulti predikat interpretiramo ovako $O : (this = null)$. Dakle, nulto stanje je validno kvazistanje u kojem objekat-predstavnik *this* ima vrednost *null*. Metode ćemo interpretirati kôdom, a specijalno u ovom izlaganju koristićemo Java kôd [30] [98]. Interpretacija formule $\{P\}m\{Q\}$ jeste

- ako je pre izvršenja metode m objekat (klasa) bio u stanju u kojem je predikat P bio tačan, tada m terminira i objekat (klasa) će se naći u stanju u kojem je predikat Q tačan.

Uočimo da smo na ovaj način analizu dovoljno uprostili, a opet sa druge strane, nismo izgubili opštost u razmatranju.

Najzad, u objektnom ambijentu, terminiranju metode mora se dati poseban značaj, ali ne zbog eventualnih neizvodljivih operacija ili beskonačnih ciklusa, nego zbog poštovanja protokola klase. Uvodimo reinterpretaciju neterminiranja i kažemo da metoda ne terminira:

- Ako se u toku izvršenja zahteva neizvodljiva operacije (npr. deljenje nulom),
- Ako se u toku izvršenja pojavi beskonačni ciklus,
- Ako metoda generiše izuzetak ili bi trebalo da generiše, samo to ne čini zbog slabe zaštite.

Uočimo odmah da ovakva reinterpretacija neterminiranja ni na koji način ne ugrožava opštost razmatranja jer je, sa stanovišta invarijante, svejedno da li je izvršenje metode prekinuto ili je došlo do (nepoželjnog) završetka.

Kod *as prescribed* analize specifikacija klase se predstavlja kao skup uređenih parova (P_i, Q_i) gde je P_i preduslov, a Q_i postuslov metode m_i i onda se dokazuje korektnost

napisane klase u odnosu na zadatu specifikaciju. Međutim, s obzirom da ovde govorimo o *as described* analizi invarijante, koja polazi od klase koju smatramo korektnom, specifikacija će biti data kao skup tačnih predikata $\{P_i\}m_i\{Q_i\}$.

Definicija 4.2.3 (Specifikacija klase) *Specifikacija klase K, u oznaci $Spec_K$, jeste skup:*

$$Spec_K = \{\{P_i\}m_i\{Q_i\}|i = 1, ..., n\} \ ,$$

gde su $\{P_i\}m_i\{Q_i\}$ tačni predikati, a n je broj metoda.

Definicija 4.2.4 (Invarijanta) *Invarijanta u klasi K je svaki predikat I definisan nad njenim apstraktnim skupom stanja U_K, koji je tačan u svakom validnom stanju.*

Primetimo da se od invarijante zahteva da mora biti tačna u validnim stanjima, a može i ne mora biti tačna u invalidnim stanjima. Takva definicija je u skladu sa zaključkom: "invariant must hold every time control leaves a method of a class" [46]. S obzirom da je po definiciji skup validnih stanja V_K neprazan, invarijanta uvek postoji.

Definicija 4.2.5 (Stroga invarijanta) *Stroga invarijanta u klasi K je svaki predikat IS_K definisan nad skupom UK sa sledećim svojstvima:*

(SIK_1) *IS_K je invarijanta u klasi K,*

(SIK_1) *Ako je I invarijanta u klasi K tada važi $(\forall s \in U_K)IS_K(s) \Rightarrow I(s)$.*

Očigledno, skup svih invarijanata $\Im_K$ u klasi K je beskonačan, ali se iz njega mogu izdvojiti konačni podskupovi koje zovemo *baze invarijanata*.

Definicija 4.2.6 (Baza invarijanata) *Baza invarijanata u klasi K je svaki konačan podskup $B_K = \{I_1, I_2, \ldots, I_n\}$ skupa svih invarijanata $\Im_K$ u klasi K koji ima sledeće osobine:*

(B_1) *Za svaku strogu invarijantu IS_K važi $IS_K \Leftrightarrow I_1 \wedge I_2 \wedge \cdots \wedge I_n$,*

(B_1) *Ako iz skupa B_K izbacimo bilo koju invarijantu tada osobina (B_1) više ne važi.*

Teorema 4.2.7 *U klasi K stroga invarijanta IS_K uvek postoji i jedinstvena je do nivoa ekvivalencije, odnosno sve stroge invarijante su ekvivalentne.*

Dokaz.

Pošto je skup validnih stanja V_K neprazan, stroga invarijanta (različita od $\perp$) uvek postoji. Pretpostavimo da su I_1 i I_2 stroge invarijante, za koje važi $(I_1 \Leftrightarrow I_2) \equiv \perp$. Pošto su I_1 i I_2 stroge invarijante, na osnovu osobine (SIK_1) iz definicije 4.2.5 one su i invarijante, pa na osnovu (SIK_2) iz definicije 4.2.5 važi $I_1 \Rightarrow I_2 \equiv \top$ i $I_2 \Rightarrow I_1 \equiv \top$, odakle dobijamo $I_1 \Leftrightarrow I_2 \equiv \top$, što je suprotno polaznoj pretpostavci, pa je time teorema dokazana.

□

Teorema 4.2.8 *Važi $(\forall s \in U_K) IS_K(s) \Leftrightarrow s \in V_K$.*

Dokaz.

Na osnovu osobine (SIK_1) iz definicije 4.2.5 i na osnovu definicije 4.2.4 važi $(\forall s \in V_K) \Rightarrow IS_K(s)$, čime je dokazana implikacija s desna u levo. Pretpostavimo da je $t \in U_K$ takvo da važi $IS_K(t) \equiv \top$ i $t \notin V_K$. Posmatrajmo neki predikat W takav da važi $W(t) \equiv \top$ i $(\forall s \in V_K) W(s) \equiv \perp$. Kako je $(\forall s \in V_K) \neg W(s) \equiv \top$, sledi da je $\neg W$ invarijanta u klasi i da na osnovu (SIK_2) iz definicije 4.2.5 važi $(\forall s \in U_K) IS_K(s) \Rightarrow \neg W(s)$, a to znači da mora da važi i $IS_K(t) \Rightarrow \neg W(t)$. Međutim, s obzirom na naše pretpostavke da je $IS_K(t) \equiv \top$ i $W(t) \equiv \top$, dobijamo $IS_K(t) \Rightarrow \neg W(t) \equiv \perp$ i dolazimo do kontradikcije, pa je time dokazana i implikacija s leva u desno. Pošto smo dokazali implikacije u oba smera, zaključujemo da polazna ekvivalencija važi, pa je time teorema dokazana.

□

Posledica 4.2.9 *Stroga invarijanta u celosti definiše skup validnih stanja klase, tj.*

$$V_K = \{s | (s \in U_K) IS_K(s)\}.$$

Dokaz. *Direktno sledi iz teoreme 4.2.8.* □

U našoj analizi koristimo predikat $\Gamma(m)$, koji zovemo zaštitni preduslov ili gard (guard) i koji opisuje skup svih početnih stanja počev od kojih metoda m garantovano terminira [26].

Definicija 4.2.10 (Zaštitni preduslov) *Zaštitni preduslov ili gard (guard) metode m, u oznaci $\Gamma(m)$, je predikat za koji važi:*

(G_1) $\{\Gamma(m)\} m \{\top\}$, *tj. ako je zadovoljen preduslov $\Gamma(m)$ tada metoda m garantovano terminira,*

(G_2) $\{P\} m \{\top\} \Rightarrow (P \Rightarrow \Gamma(m))$.

Osobina (G_2) iz definicije 4.2.10 znači da je $\Gamma(m)$ najširi preduslov iz kojeg metoda m garantovano terminira, tj. važi $\Gamma(m) \Leftrightarrow wp(m, \top)$, gde je wp najširi preduslov (Definicija 1.6.1). Zaštitni preduslov i najširi preduslov su bliske, ali ne i identične kategorije. Naime, $\Gamma(m)$ opisuje sva početna stanja počev od kojih metoda m garantovano terminira, dok $wp(m, Q)$ opisuje sva početna stanja počev od kojih metoda m garantovano terminira u završno stanje u kojem važi predikat Q. U nastavku, dajemo nekoliko teorema vezanih za osnovne osobine zaštitnog preduslova.

Teorema 4.2.11 *Zaštitni preduslov uvek postoji.*

Dokaz.

Očigledno, ako postoji bar jedno početno stanje s za koje metoda m terminira tada postoji i $\Gamma(m)$. Ako takvog stanja nema, tada je $\Gamma(m) \equiv \bot$.

□

Teorema 4.2.12 *Ako za neki preduslov R važi $\{R\}m\{Q\}$ tada $R \Rightarrow \Gamma(m)$.*

Dokaz.

Na osnovu teoreme 1.3.1.b.), iz $\{R\}m\{Q\}$ i $Q \Rightarrow \top$, dobijamo $\{R\}m\{\top\}$, pa prema osobini (G_2) iz definicije 4.2.10 sledi $R \Rightarrow \Gamma(m)$.

□

Dokaz teoreme 4.2.12 može da se izvede i direktnim rezonovanjem. Naime, iz tačnosti $\{R\}m\{Q\}$ sledi da stanja opisana sa R obezbeđuju termininiranje metode m, dakle, formiraju neki podskup od skupa stanja kojeg opisuje predikat $\Gamma(m)$, pa prema tome važi $R \Rightarrow \Gamma(m)$.

Teorema 4.2.13 *Ako za neki predikat Q važi $\{\Gamma(m)\}m\{Q\}$ tada je $wp(m, Q) \Leftrightarrow \Gamma(m)$.*

Dokaz.

Ako važi $\{\Gamma(m)\}m\{Q\}$, tada po osobini (WP_2) iz definicije 1.6.1 važi $\Gamma(m) \Rightarrow wp(m, Q)$. Dalje, prema teoremi 4.2.12 za svaki predikat R za koji važi $\{R\}m\{Q\}$ dobijamo $R \Rightarrow \Gamma(m)$, pa na osnovu toga važi i $wp(m, Q) \Rightarrow \Gamma(m)$. Prema tome, važi $wp(m, Q) \Leftrightarrow \Gamma(m)$.

□

Sada možemo uraditi postavku SP-analize, tako što ćemo kroz teoreme razmotriti vezu između zaštitnog preduslova i invarijante. Na početku napominjemo da polazna tačka u SP-analizi jeste pretpostavka:

Pretpostavka 4.2.14 *Klasa K je zadata specifikacijom $Spec_K = \{\{\Gamma(m_i)\}m_i\{Q_i\}|i = 1, 2, ..., n\}$, gde su $\{\Gamma(m_i)\}m_i\{Q_i\}, i = 1, 2, ..., n$ tačni predikati.*

Teorema 4.2.15 *Data je klasa K u skladu sa pretpostavkom 4.2.14. Neka je predikat I takav da za svaku metodu $m \in M_K$ klase K važi $\{\Gamma(m)\}m\{I\}$, tada je I invarijanta u klasi.*

Dokaz.

Na osnovu pretpostavke 4.2.14, za svaku metodu $m \in M_K$ važi da počev od bilo kog stanja $s \in U_K, \Gamma(m)(s) \equiv \top$ metoda m terminira i završno stanje s' je validno, odnosno $s' \in V_K$. Pošto je završno stanje s' validno po definiciji 4.2.4 važi $I(s') \equiv \top$. Na osnovu toga zaključujemo da važi $\{\Gamma(m)\}m\{I\}$.

□

Teorema 4.2.16 *Data je klasa K u skladu sa pretpostavkom 4.2.14. Neka je $M = \{m_1, m_2, \ldots, m_k\}$ neki podskup skupa svih metoda, odnosno $M \subseteq M_K$. Tada za predikat*

$$Z \;\Leftrightarrow\; Q_1 \vee Q_2 \vee \cdots \vee Q_k$$

važi $\{\Gamma(m_i)\}m_i\{Z\}, i = 1, 2, \ldots, k$.

Dokaz.

Iz pretpostavke 4.2.14, zatim, iz činjenice da važi $Q_i \Rightarrow Z$ i najzad, iz teoreme 1.3.1.b.) neposredno sledi $\{\Gamma(m_i)\}m_i\{Z\}$.

□

Teorema 4.2.17 *Data je klasa K u skladu sa pretpostavkom 4.2.14. Tada važi*

$$I_\vee \;\Leftrightarrow\; O \vee Q_1 \vee Q_2 \vee \cdots \vee Q_n,$$

gde je $I_\vee$ jedna od invarijanata klase K.

Dokaz.

Iz pretpostavke 4.2.14, zatim, iz činjenice da važi $Q_i \Rightarrow I_\vee$, i najzad, iz teoreme 1.3.1.b.) sledi $\{\Gamma(m_i)\}m_i\{I_\vee\}$. Na osnovu teoreme 4.2.15 $I_\vee$ je invarijanta u klasi K.

□

Lema 4.2.18 *Data je klasa K u skladu sa pretpostavkom 4.2.14 koja sadrži metodu m. Neka je $sp(m, \Gamma(m))$ najuži postuslov za $\Gamma(m)$. Tada skup stanja određen sa sp sadrži sva validna stanja u koja se dolazi posle primene m i samo njih.*

Dokaz.

Neka je $\{s_1, s_2, \ldots, s_k\}$ skup svih stanja za koja važi $sp(m, \Gamma(m)) \equiv \top$. Pokazaćemo da ne može postojati stanje s' koje je invalidno, a za koje je $sp(m, \Gamma(m)) \equiv \top$. Naime, ako je s' invalidno stanje tada postoji predikat Z koji ovo stanje zadovoljava, dok ga ostala stanja iz skupa $\{s_1, s_2, \ldots, s_k\}$ ne zadovoljavaju. Tada bi, međutim, sva validna stanja zadovoljavala predikat $sp(m, \Gamma(m)) \wedge \neg Z$ pa stoga $sp(m, \Gamma(m))$ ne bi bio najuži postuslov.

□

Teorema 4.2.19 *Data je klasa K u skladu sa pretpostavkom 4.2.14 koja sadrži metode $m_i, i = 1, 2, \ldots, n$. Tada važi*

$$IS_K \;\Leftrightarrow\; O \vee sp(m_1, \Gamma(m_1)) \vee sp(m_2, \Gamma(m_2)) \vee \cdots \vee sp(m_n, \Gamma(m_n))$$

gde je IS_K stroga invarijanta u klasi K.

Dokaz.

Predikat $O \vee sp(m_1, \Gamma(m_1)) \vee sp(m_2, \Gamma(m_2)) \vee \cdots \vee sp(m_n, \Gamma(m_n))$ određuje skup svih stanja u kojima se objekat može naći posle primene bilo koje metode klase K. Po Lemi 4.2.18 ovaj skup ne može sadržati invalidno stanje.

□

Posledica 4.2.20 *Skup validnih stanja klase K je*

$$V_K = \{s | O(s) \vee sp(m_1, \Gamma(m_1))(s) \vee sp(m_2, \Gamma(m_2))(s) \vee \cdots \vee sp(m_n, \Gamma(m_n))(s)\}.$$

Dokaz. *Direktno sledi iz teoreme 4.2.19.* □

Ovim teoremama je koncipirana SP-analiza i sada nam preostaje još da elaboriramo njene karakteristike. Na prvi pogled, SP-analiza deluje zahvalno, jer na osnovu date klase, koju smatramo korektnom, određujemo njene invarijante. Međutim, ova analiza koristi najuže statičke postuslove $sp(m, \Gamma(m))$ i zasnovana je na pretpostavci 4.2.14. Drugim rečima, to znači da je svaka metoda napisana tako da ako terminira ona proizvodi validno stanje, inače ne terminira. Na sledećim primerima ćemo videti šta to zapravo znači u praksi.

4.2.1 Primer

Odredićemo stroge invarijante za klase K_1 i K_2, koje imaju isti skup validnih stanja $V_K = \{s_0\}$, gde je s_0 stanje u kojem važi $n = 0$.

```
public class K1 {
     public int n;
     public K1() { n=0; }
     public void saveN() {
          if(n!=0) throw RuntimeException();
          n=n;
     }
}

public class K2 {
     public int n;
     public K2() { n=0; }
     public void saveN() { n=n; }
}
```

Za klasu $K1$ važi:
$\Gamma(K_1) : (this = null)$,
$\Gamma(saveN) : (n = 0)$
$sp(K_1, \Gamma(K_1)) : (n = 0)$,
$sp(saveN, \Gamma(saveN)) : (n = 0)$,

odakle dobijamo:

$$IS_{K_1} : (this = null) \vee (n = 0) \vee (n = 0) \equiv (this = null) \vee (n = 0),$$

što je tačan rezultat.

Za klasu K_2 važi:
$\Gamma(K_2) : (this = null)$,
$\Gamma(saveN) : (n \in D_n)$, gde je D_n domen polja n,

$sp(K_2, \Gamma(K_2)) : (n = 0)$,
$sp(saveN, \Gamma(saveN)) : n \in D_n$,

odakle dobijamo:

$$IS_{K_2} : (this = null) \vee (n = 0) \vee (n \in D_n) \equiv (this = null) \vee (n \in D_n),$$

što je netačan rezultat.

Sada se postavlja pitanje zašto je za klasu K_1 dobijen tačan rezultat, a za klasu K_2 nije. Primetimo da u klasi K_1 važi $\Gamma(saveN) : (n = 0)$, odnosno metoda $saveN$ je napisana tako da terminira jedino u validno stanje, inače ne terminira, što je u skladu sa našom polaznom pretpostavkom 4.2.14, pa smo zato i dobili tačan rezultat. Dalje, primetimo da u klasi K_2 važi $\Gamma(saveN) : (n \in D_n)$, odnosno da metoda terminira kako u validna tako i u invalidna stanja, pa zbog toga je dobijen natačan rezultat.

Iz ovog primera zaključujemo da bi SP-analiza ispravno radila potrebno je da klasa bude napisana u skladu sa pretpostavkom 4.2.14. Međutim, ukoliko bi se klasa pisala u skladu sa pretpostavkom 4.2.14, tada bi takav stil programiranja izgubio objektno orijentisani karakter. Tačnije, metode bi dobile toliko veliku autonomiju, da bi mogle postojati kao slobodne funkcije koje koriste istu strukturu podataka (`record` u Pascal-u, odnosno `struct` u C-u) u kojoj bi se sada nalazila polja. To znači da bi klasa K_1 veoma jednostavno mogla da se pretvori u skup slobodnih funkcija, pa pisanje takve klase gubi smisao. Zbog toga SP-analiza ima više teorijski značaj u objektno orijentisanom programiranju.

4.2.2 Primer

U ovom primeru ćemo videti da najuži statički postuslov $sp(m, \Gamma(m))$ nije pogodan za opis semantike međudejstva metoda. Posmatrajmo klasu K:

```
public class K {
     public int n;
     public K() { n=0; }
     public void incN() { n++; }
}
```

Očigledno, SP-analiza bi ponovo dala netačan rezultat $IS_K : (this = null) \vee (n \in D_n)$. SP-analizom, u opštem slučaju, ne može se realno odrediti najuži postuslov $sp(m, \Gamma(m))$ metode m u ambijentu cele klase K, jer taj postuslov skoro uvek zavisi od međudejstva ostalih metoda. Na primer, za metodu $incN$ važi $sp(m, \Gamma(m)) : (n \in D_n)$, ali realna situacija u klasi K je drugačija, jer ako bi uzeli u obzir dejstvo konstruktora tada bi trebalo da je $sp(m, \Gamma(m)) : (n > 0)$.

Dakle, trebalo bi razviti matematički mehanizam kojim bi mogli opisati prelaze, tj. činjenicu da metoda $incN$ premešta objekat u stanje u kojem je n veće za 1 (npr.

$n' = n + 1$, gde n' označava vrednost promenljive n u završnom stanju) i svaki takav prelaz bi bio validan. Takođe, taj mehanizam bi uzimao u obzir činjenicu da konstruktor ostavlja objekat u validnom stanju s_0 u kojem je $n = 0$ i da jedino stanje s_0 može biti početno za metodu $incN$, koja zatim obavlja validne prelaze $n' = n + 1$. Tako bi se iz $s_0 : n = 0$ prešlo u validno stanje $s_1 : n = 1$, iz $s_1 : n = 1$ u validno stanje $s_2 : n = 2$ itd. Intuitivno, već sada je jasno da bi takav mehanizam opisao sva validna stanja i najzad, odredio da je $IS_K : (this = null) \vee (n \geq 0)$, što je tačan rezultat. Takav matematički mehanizam ćemo nazvati DP-analiza, što je i naša glavna tema u nastavku izlaganja.

4.3 Osnovi elementi DP-analize

U ovom poglavlju ćemo razviti DP-analizu koja razmatra validne prelaze. Ovde ćemo koristiti dinamičku formulu totalne korektnosti ($DFTK$) u objektnom ambijentu, koja sada izgleda ovako:

$$(ODFTK) \qquad \{P\}m\{Q\} \leftrightarrow \forall s[P(s) \Rightarrow (\exists t S(s,t) \wedge \forall r(S(s,r) \Rightarrow \hat{Q}(s,r)))],$$

gde su $s, t, r \in U_K$. Rekli smo da predikat $\hat{Q}$ nazivamo dinamički postuslov i on predstavlja funkciju početnog i završnog stanja metode, odnosno funkciju prelaza. Takođe, da ne bi dolazilo do zabune, i ovde ćemo dinamičke postuslove označavati oznakom " ˆ ". Interpretacija formule $\{P\}m\{\hat{Q}\}$ jeste sledeća:

- ako je u nekom stanju predikat P tačan, tada metoda m terminira i biće izvršen prelaz koji zadovoljava dinamički predikat $\hat{Q}$.

Na primer, ako za metodu $incN$ važi $\{\neg O\}incN\{\hat{Q}\} \equiv \top$, gde je $\hat{Q} : n' = n+1$, tada zaključujemo da metoda $incN$ bezuslovno povećava vrednost polja n za 1. Apostrofom ćemo označavati vrednost polja posle izvršenja metode, na primer, u prethodnom primeru oznaka n' označava vrednost programske promenljive n posle izvršenja metode $incN$, pa se dinamički predikat $\hat{Q} : n' = n + 1$ interpretira na sledeći način – nova vrednost polja n je za 1 veća nego što je bila. Prelaz posmatramo kao uređen par (s, s'), gde su $s, s' \in U_K$, a skup svih prelaza jeste $U_K \times U_K$. Dinamički postuslov $\hat{Q}$ predstavlja logičku funkciju na skupu $U_K \times U_K$, tj. $\hat{Q} : U_K \times U_K \to \{\top, \bot\}$ i time opisuje neki podskup skupa svih prelaza $U_K \times U_K$. Prelaze delimo na *validne* i *invalidne*, pri ćemu ovde napominjemo da validne prelaze ne treba vezivati za validna stanja. Na primer, posmatrajmo klasu K:

```
public class K {
     public int n;
     public K() { n=0; }
     public void incN() { n++; }
}
```

Očigledno, validni prelaz jeste (s_0, s_1), gde je $s_0 : n = 0$, a $s_1 : n = 1$, ali je isto tako validan prelaz i (s_{-5}, s_{-4}), gde je $s_{-5} : n = -5$, a $s_{-4} : n = -4$, iako su s_{-5} i s_{-4} invalidna stanja. Međutim, prelaz (s_0, s_2), gde je $s_2 : n = 2$ je invalidan, iako su s_0 i s_2 validna stanja. Zaključujemo da je prelaz validan akko ga metode mogu ostvariti, tj. ako metode posmatramo kao relacije na skupu $U_K \times U_K$, tada je prelaz (s, s') validan akko $(s, s') \in m, m \in M_K$.

Polazna tačka u DP-analizi jeste pretpostavka da svako terminiranje svake metode ostvaruje validan prelaz:

Pretpostavka 4.3.1 *Klasa K je zadata specifikacijom $Spec_K = \{\{\Gamma(m_i)\}m_i\{\hat{Q}_i\}|i = 1, 2, ..., n\}$, gde su $\{\Gamma(m_i)\}m_i\{\hat{Q}_i\}, i = 1, 2, ..., n$ tačni predikati.*

Posmatrajmo opet metodu $incN$ koja bezuslovno povećava vrednost polja n za 1. Očigledno, važi:

$$\{\neg O\}incN\{\hat{Q}\},$$
$$\{\neg O\}incN\{\hat{R}\},$$

gde su:

$$\hat{Q} : n' = n + 1,$$
$$\hat{R} : n' > n.$$

Primetimo da dinamički postuslov $\hat{Q}$ opisuje samo validne prelaze metode $incN$, dok postuslov $\hat{R}$ opisuje i validne i invalidne, jer očigledno važi $(n' = n + 1) \Rightarrow (n' > n)$, ali i $(n' = n + 2) \Rightarrow (n' > n)$. Drugim rečima, dinamički postuslov $\hat{R}$ opisuje širi skup prelaza, pa zato obuhvata i prelaze koji nisu validni. Zato ćemo se sada pozabaviti najužim dinamičkim postuslovom metode m u odnosu na preduslov $\Gamma(m)$, u oznaci $\hat{sdp}(m, \Gamma(m))$ (definicija 2.3.2).

Teorema 4.3.2 *Data je klasa K u skladu sa pretpostavkom 4.3.1 koja sadrži metodu m. Tada dinamički predikat $\hat{sdp}(m, \Gamma(m))$ određuje skup svih validnih prelaza koje može ostvariti metoda m i samo njih.*

Dokaz.

Neka je $\{(s_1, s'_1), (s_2, s'_2), ..., (s_k, s'_k)\}$ skup svih prelaza za koje važi $\hat{sdp}(m, \Gamma(m)) \equiv \top$. Pokazaćemo da ne može postojati invalidan prelaz (s, s'), a za koji je $\hat{sdp}(m, \Gamma(m)) \equiv \top$. Naime, ako je (s, s') invalidan prelaz tada postoji dinamički predikat $\hat{Z}$ koji ovaj prelaz zadovoljava, dok ga ostali prelazi iz skupa $\{(s_1, s'_1), (s_2, s'_2), ..., (s_k, s'_k)\}$ ne zadovoljavaju. Tada bi, međutim, svi validni prelazi zadovoljavali dinamički predikat $\hat{sdp}(m, \Gamma(m)) \wedge \neg\hat{Z}$ pa stoga $\hat{sdp}(m, \Gamma(m))$ ne bi bio najuži dinamički postuslov.

□

Na primer, neka je data metoda *incN* koja povećava vrednost polja n za 1, ali pod uslovom da je $n > 0$:

```
public void incN() {
     if(n<=0) throw RuntimeException();
     n++;
}
```

Tada je $\Gamma(m) : n > 0$, a $\hat{sdp}(m, P) : (n' = n + 1) \wedge (n > 0)$. Iz ovog primera zaključujemo da najuži dinamički postuslov $\hat{sdp}$ će uvek ima takvu formu da važi:

$$\hat{sdp}(m, \Gamma(m)) \equiv \Gamma(m) \wedge \hat{\alpha}(m, \Gamma(m)),$$

gde $\hat{\alpha}$ predstavlja *minimalni predikat prelaza*, a konkretno u prethodnom primeru važi $\hat{\alpha} : n' = n + 1$.

Definicija 4.3.3 (Dinamička stroga invarijanta) *Data je klasa K u skladu sa pretpostavkom 4.3.1. Dinamički predikat*

$$\hat{JS}_K \equiv \bigwedge_{m \in M_K} \hat{sdp}(m, \Gamma(m))$$

se naziva stroga dinamička invarijanta.

Teorema 4.3.4 *Data je klasa K u skladu sa pretpostavkom 4.3.1. Dinamički predikat*

$$\hat{JS}_K \equiv \bigwedge_{m \in M_K} \hat{sdp}(m, \Gamma(m))$$

opisuje sve validne prelaze ostvarljive metodama klase K, i samo njih.

Dokaz.

Na osnovu teoreme 4.3.2 za svaku metodu $m \in M_K$ važi da $\hat{sdp}(m, \Gamma(m))$ opisuje skup svih mogućih prelaza koje može ostvariti metoda m. Unijom svih tih skupova dobija se skup koji sadrži sve validne prelaze ostvarljive metodama klase K, a taj skup opisuje predikat $\bigwedge_{m \in M_K} \hat{sdp}(m, \Gamma(m))$.

□

Definicija 4.3.5 (Dinamička invarijanta) *Data je klasa K u skladu sa pretpostavkom 4.3.1. Svaki dinamički predikat $\hat{J}_K$ za koji važi $\hat{JS}_K \Rightarrow \hat{J}_K$ naziva se dinamička invarijanta.*

Teorema 4.3.6 *Data je klasa K u skladu sa pretpostavkom 4.3.1. Neka je dinamički predikat $\hat{J}$ takav da za svaku metodu $m \in M_K$ klase K važi*

$$\{\Gamma(m)\}m\{\hat{J}\}.$$

Tada je $\hat{J}$ dinamička invarijanta u klasi K.

Dokaz.

Na osnovu pretpostavke 4.3.1, za svaku metodu $m \in M_K$ važi da počev od bilo kog stanja $s \in U_K, \Gamma(m)(s) \equiv \top$ metoda m terminira i ostvaruje validan prelaz (s, s'), pa po teoremi 4.3.4 važi $\hat{J}S_K(s, s') \equiv \top$, odakle zaključujemo da važi $\hat{J}S_K \Rightarrow \hat{J}$, pa je $\hat{J}$ dinamička invarijanta klase K.

□

Teorema 4.3.7 *Data je klasa K u skladu sa pretpostavkom 4.3.1. Tada važi*

$$\hat{J}_\vee \;\Leftrightarrow\; \hat{Q}_1 \vee \hat{Q}_2 \vee \cdots \vee \hat{Q}_n,$$

gde je $\hat{J}_\vee$ jedna od dinamičkih invarijanata klase K.

Dokaz.

Iz pretpostavke 4.3.1, zatim iz činjenice da važi $\hat{Q}_i \Rightarrow \hat{J}_\vee$, $i = 1, 2, , n$, i najzad, iz Teoreme 2.2.2.b.) sledi $\{\Gamma(m_i)\}m_i\{\hat{J}_\vee\}$. Na osnovu Teoreme 4.3.6 $\hat{J}_\vee$ je dinamička invarijanta u klasi.

□

4.3.1 Generalisana supstitucija

Definicija 4.3.8 (Generalisana supstitucija) *Neka su $A(s)$ i $\hat{B}(s, s')$ predikati definisani respektivno nad U_K i $U_K \times U_K$. Pod generalisanom supstitucijom (kraće supstitucijom), u oznaci $A \otimes \hat{B}$, podrazumevamo predikat $C(s')$ sa sledećim karakteristikama:*

(GS_1) $\forall s \forall s'[A(s) \wedge \hat{B}(s, s') \Rightarrow C(s')]$,

(GS_2) *za svaki predikat $D(s')$ za koji važi (GS_1), važi i $C \Rightarrow D$.*

Smisao generalisane supstitucije je sledeći: ako u nekom stanju s važi $A(s)$ i ako postoji prelaz (s, s') koji zadovoljava predikat $\hat{B}(s, s')$, tada završno stanje s' zadovoljava predikat $A \otimes \hat{B}$. Supstitucija uvek postoji, jer osobina (GS_1) iz definicije 4.3.8 važi za

$C \Leftrightarrow \top$. Supstitucija je jedinstvena do nivoa ekvivalencije, jer ako je $A \otimes \hat{B} \equiv C_1$, zatim $A \otimes \hat{B} \equiv C_2$ i najzad $\neg(C_1 \Leftrightarrow C_2)$, tada C_1 i C_2 nisu vrednosti supstitucije nego je to $C_1 \wedge C_2$. Osobinu (GS_2) iz definicije 4.3.8 ćemo zvati minimalnost supstitucije. Usvojićemo da se po prioritetu operacija $\otimes$ nalazi odmah iza operacije negacije $\neg$, odnosno da operacije $\wedge$, $\vee$, $\Rightarrow$ i $\Leftrightarrow$ imaju manji prioritet.

Teorema 4.3.9 *Neka je s trenutno stanje koje zadovoljava predikat W. Tada supstitucija $W \otimes \hat{JS}_K$ obuhvata sva završna stanja dostupna iz s i samo njih.*

Dokaz.

Neka je $Q \equiv W \otimes \hat{JS}_K$ i neka je s stanje za koje važi $W(s)$. Prvo, prema teoremi 4.3.4, za svako stanje s' dostupno iz s mora da postoji prelaz (s, s') koji zadovoljava $\hat{JS}_K$. Shodno definiciji 4.3.8 važi predikat $Q(s')$. Neka je s'_0 stanje u kojem važi $Q(s'_0)$, ali koje nije dostupno iz s. Tada ne važi $\hat{JS}_K(s, s'_0)$. Neka je Z predikat koji je tačan samo za s'_0, odnosno važi $Z(s'_0) \wedge \forall s', s' \neq s'_0, \neg Z(s')$. Iz ovog sledi da važi $\forall s \forall s'(W(s) \wedge \hat{JS}_K(s, s') \Rightarrow Q(s') \wedge \neg Z(s'))$, tj. $\neg(Q \equiv W \otimes \hat{JS}_K)$.

□

Teorema 4.3.10 *Za svako A, $\hat{B}_1$ i $\hat{B}_2$ važi*

$$(\hat{B}_1 \Rightarrow \hat{B}_2) \Rightarrow (A \otimes \hat{B}_1 \Rightarrow A \otimes \hat{B}_2).$$

Dokaz.

1. *Neka je (s_0, s'_0) prelaz za koji važi $B_1(s_0, s'_0) \equiv \top$. Ako je tačno $(\hat{B}_1 \Rightarrow \hat{B}_2)$ mora biti tačno i $\hat{B}_2(s_0, s'_0)$. Ako je $A(s_0) \equiv \top$ mora biti i $[A \otimes \hat{B}_1](s_0, s'_0) \equiv \top$, kao i $[A \otimes \hat{B}_2](s_0, s'_0) \equiv \top$. Ako je pak $A(s_0) \equiv \bot$ tada je, zbog minimalnosti supstitucije, $[A \otimes \hat{B}_1](s_0, s'_0) \equiv \bot$.*

2. *Neka važi $\hat{B}_1(s_0, s'_0) \equiv \bot$. Tada, zbog minimalnosti supstitucije, važi $[A \otimes \hat{B}_1](s_0, s'_0) \equiv \bot$.*

□

Teorema 4.3.11 *Za A_1, A_2 i $\hat{B}$ važi*

$$(A_1 \Rightarrow A_2) \Rightarrow (A_1 \otimes \hat{B} \Rightarrow A_2 \otimes \hat{B}).$$

Dokaz.

1. *Neka je s_0 stanje za koje je $A(s_0) \equiv \top$. Po antecedenti teoreme mora biti tačno i $A_2(s_0)$. Ako postoji prelaz (s_0, s_0') takav da je $\hat{B}(s_0, s_0') \equiv \top$, mora važiti i $[A_1 \otimes \hat{B}](s_0, s_0')$ kao i $[A_2 \otimes \hat{B}](s_0, s_0')$. Ako takav prelaz ne postoji, tada, zbog minimalnosti supstitucije, važi $[A_1 \otimes \hat{B}](s_0, s_0') \equiv \bot$.*

2. *Neka je $A_1(s_0) \equiv \bot$. Tada, zbog minimalnosti supstitucije, važi $[A_1 \otimes \hat{B}](s_0, s_0') \equiv \bot$.*

□

Teorema 4.3.12 (Monotonost supstitucije) *Za A_1, A_2, $\hat{B}_1$ i $\hat{B}_2$ važi*

$$[(A_1 \Rightarrow A_2) \wedge (\hat{B}_1 \Rightarrow \hat{B}_2)] \Rightarrow [A_1 \otimes \hat{B}_1 \Rightarrow A_2 \otimes \hat{B}_2].$$

Dokaz.

Na osnovu teorema 4.3.10 i 4.3.11 direktno sledi
$(A_1 \Rightarrow A_2) \wedge (\hat{B}_1 \Rightarrow \hat{B}_2)$
$\Rightarrow (A_1 \otimes \hat{B}_1 \Rightarrow A_2 \otimes \hat{B}_1) \wedge (A_2 \otimes \hat{B}_1 \Rightarrow A_2 \otimes \hat{B}_2)$
$\Rightarrow (A_2 \otimes \hat{B}_1 \Rightarrow A_2 \otimes \hat{B}_2)$.

□

Teorema 4.3.13 *Za A_1, A_2 i $\hat{B}$ važi*

$$(A_1 \wedge A_2) \otimes \hat{B} \Leftrightarrow A_1 \otimes \hat{B} \wedge A_2 \otimes \hat{B}.$$

Dokaz.

a.) Neka je (s_0, s_0') prelaz za koji važi $\hat{B}(s_0, s_0') \equiv \top$, tada važi:

- *ako je $A_1(s_0) \wedge A_2(s_0) \equiv \top$, tada zbog minimalnosti supstitucije važi $(A_1 \wedge A_2) \otimes \hat{B} \equiv \top$, $A_1 \otimes \hat{B} \equiv \top$ i $A_2 \otimes \hat{B} \equiv \top$,*
- *ako je $A_1(s_0) \wedge A_2(s_0) \equiv \bot$, tada zbog minimalnosti supstitucije važi $(A_1 \wedge A_2) \otimes \hat{B} \equiv \bot$, a s obzirom da važi $(A_1(s_0) \equiv \bot) \vee (A_2(s_0) \equiv \bot)$, dobijamo $A_1 \otimes \hat{B} \wedge A_2 \otimes \hat{B} \equiv \bot$.*

b.) Neka je (s_0, s_0') prelaz za koji važi $\hat{B}(s_0, s_0') \equiv \bot$, tada, zbog minimalnosti supstitucije, važi $(A_1 \wedge A_2) \otimes \hat{B} \equiv \bot$, $A_1 \otimes \hat{B} \equiv \bot$ i $A_2 \otimes \hat{B} \equiv \bot$.

□

Teorema 4.3.14 *Za A_1, A_2 i $\hat{B}$ važi*

$$(A_1 \vee A_2) \otimes \hat{B} \Leftrightarrow A_1 \otimes \hat{B} \vee A_2 \otimes \hat{B}.$$

Dokaz.

a.) Neka je (s_0, s_0') *prelaz za koji važi* $\hat{B}(s_0, s_0') \equiv \top$*, tada važi:*

- *ako je* $A_1(s_0) \vee A_2(s_0) \equiv \top$*, tada zbog minimalnosti supstitucije važi* $(A_1 \vee A_2) \otimes \hat{B} \equiv \top$*, a s obzirom da važi* $(A_1(s_0) \equiv \top) \vee (A_2(s_0) \equiv \top)$*, dobijamo* $A_1 \otimes \hat{B} \vee A_2 \otimes \hat{B} \equiv \top$*,*
- *ako je* $A_1(s_0) \vee A_2(s_0) \equiv \bot$*, tada zbog minimalnosti supstitucije važi* $(A_1 \vee A_2) \otimes \hat{B} \equiv \bot$*,* $(A_1 \otimes \hat{B} \equiv \bot)$ *i* $(A_2 \otimes \hat{B} \equiv \bot)$*.*

b.) Neka je (s_0, s_0') *prelaz za koji važi* $\hat{B}(s_0, s_0') \equiv \bot$*, tada zbog minimalnosti supstitucije, važi* $(A_1 \vee A_2) \otimes \hat{B} \equiv \bot$*,* $A_1 \otimes \hat{B} \equiv \bot$ *i* $A_2 \otimes \hat{B} \equiv \bot$*.*

□

Teorema 4.3.15 *Za* A*,* $\hat{B}_1$ *i* $\hat{B}_2$ *važi*

$$A \otimes (\hat{B}_1 \wedge \hat{B}_2) \Leftrightarrow A \otimes \hat{B}_1 \wedge A \otimes \hat{B}_2.$$

Dokaz.

a.) Neka je (s_0, s_0') *prelaz za koji važi* $\hat{B}_1(s_0, s_0') \wedge \hat{B}_2(s_0, s_0') \equiv \top$*, tada važi:*

- *ako je* $A(s_0) \equiv \top$*, tada zbog minimalnosti supstitucije važi* $A \otimes (\hat{B}_1 \wedge \hat{B}_2) \equiv \top$*,* $A \otimes \hat{B}_1 \equiv \top$ *i* $A \otimes \hat{B}_2 \equiv \top$*,*
- *ako je* $A(s_0) \equiv \bot$*, tada zbog minimalnosti supstitucije važi* $A \otimes (\hat{B}_1 \wedge \hat{B}_2) \equiv \bot$*,* $A \otimes \hat{B}_1 \equiv \bot$ *i* $A \otimes \hat{B}_2 \equiv \bot$*.*

b.) Neka je (s_0, s_0') *prelaz za koji važi* $\hat{B}_1(s_0, s_0') \wedge \hat{B}_2(s_0, s_0') \equiv \bot$*, odnosno* $(\hat{B}_1(s_0, s_0') \equiv \bot) \vee (\hat{B}_2(s_0, s_0') \equiv \bot)$*, pa zbog minimalnosti supstitucije zaključujemo da važi* $A \otimes (\hat{B}_1 \wedge \hat{B}_2) \equiv \bot$ *i* $A \otimes \hat{B}_1 \wedge A \otimes \hat{B}_2 \equiv \bot$*.*

□

Teorema 4.3.16 *Za* A*,* $\hat{B}_1$ *i* $\hat{B}_2$ *važi*

$$A \otimes (\hat{B}_1 \vee \hat{B}_2) \Leftrightarrow A \otimes \hat{B}_1 \vee A \otimes \hat{B}_2.$$

Dokaz.

a.) Neka je (s_0, s_0') *prelaz za koji važi* $\hat{B}_1(s_0, s_0') \vee \hat{B}_2(s_0, s_0') \equiv \top$*, odnosno* $(\hat{B}_1(s_0, s_0') \equiv \top) \wedge (\hat{B}_2(s_0, s_0') \equiv \top)$*, pa zbog minimalnosti supstitucije zaključujemo:*

- *ako je* $A(s_0) \equiv \top$*, tada važi* $A \otimes (\hat{B}_1 \vee \hat{B}_2) \equiv \top$ *i* $A \otimes \hat{B}_1 \equiv \top$ *i* $A \otimes \hat{B}_2 \equiv \top$*,*
- *ako je* $A(s_0) \equiv \bot$*, tada važi* $A \otimes (\hat{B}_1 \vee \hat{B}_2) \equiv \bot$ *i* $A \otimes \hat{B}_1 \equiv \bot$ *i* $A \otimes \hat{B}_2 \equiv \bot$*.*

b.) Neka je (s_0, s_0') *prelaz za koji važi* $\hat{B}_1(s_0, s_0') \vee \hat{B}_2(s_0, s_0') \equiv \bot$, *odnosno* $(\hat{B}_1(s_0, s_0') \equiv \bot) \vee (\hat{B}_2(s_0, s_0') \equiv \bot)$, *pa zbog minimalnosti supstitucije zaključujemo da važi* $A \otimes (\hat{B}_1 \vee \hat{B}_2) \equiv \bot$, $A \otimes \hat{B}_1 \equiv \bot$ *i* $A \otimes \hat{B}_2 \equiv \bot$.

□

Na kraju, sledeće dve jednostavne teoreme mogu da posluže prilikom određivanja supstitucije.

Teorema 4.3.17 *Predikat* Q *je supstitucija* $\hat{B}(s, s')$ *iz* $A(s)$ *ako i samo ako važi* $[A \wedge \hat{B} \Rightarrow Q] \wedge [\forall s' \exists s Q(s') \Rightarrow A(s) \wedge \hat{B}(s, s')$.

Dokaz.

Neka je $Q \equiv A \otimes \hat{B}$. *Ako je drugi deo konjunkcije netačan, znači da postoji stanje* s_0' *za koje nema stanja* s_0 *takvog da je* $A(s_0) \wedge \hat{B}(s_0, s_0')$ *tačno. Neka je stanje* s_0' *jednoznačno određeno predikatom* R. *U tom slučaju, međutim, mora važiti* $A \wedge \hat{B} \Rightarrow Q \wedge \neg R$, *što znači* $Q \neq A \otimes \hat{B}$. *Obrnuto, neka je drugi deo konjunkcije tačan. Neka je* R *predikat za koji važi* $A \otimes \hat{B} \Rightarrow R$, $R \Rightarrow Q$ *i* $\neg(R \Leftrightarrow Q)$. *Neka je* s_0' *stanje u kojem je* $R(s_0') \equiv \bot$ *i* $Q(s_0') \equiv \top$. *Ako je* $Q(s_0') \equiv \top$ *tada, po pretpostavci, postoji stanje* s_0 *za koje važi* $A(s_0) \wedge \hat{B}(s_0, s_0') \equiv \top$. *No, tada formula* $A(s_0) \wedge \hat{B}(s_0, s_0') \Rightarrow R(s_0')$ *nije tačna, te tako nije tačna ni formula* $A \otimes \hat{B} \Rightarrow R$.

□

Teorema 4.3.18 *Neka je*

$$A(s) \wedge \hat{B}(s, s') \Leftrightarrow \bigvee_{i=1}^{n} \alpha_i(s) \wedge \beta_i(s'), \tag{1}$$

gde su $\alpha_i(s)$ *i* $\beta_i(s'), i = 1, 2, \ldots, n$ *predikati definisani nad prostorom stanja. Neka važi*

$$\alpha_i(s) \wedge \beta_i(s') \Rightarrow \gamma_i(s'), i = 1, 2, \ldots, n, \tag{2}$$

pri čemu je $\gamma_i(s')$ *minimalni predikat po* s' *za koji važi* (2). *Tada je*

$$A \otimes \hat{B} \Leftrightarrow \bigvee_{i=1}^{n} \gamma_i(s'). \tag{3}$$

Dokaz.

Prvo, iz (1) *i* (2) *sledi*

$$\forall s \forall s' A(s) \wedge \hat{B}(s, s') \Rightarrow \bigvee_{i=1}^{n} \gamma_i(s'), \tag{4}$$

Neka je $\delta(s')$ *predikat za koji važi*

$$\forall s \forall s' A(s) \wedge \hat{B}(s, s') \Rightarrow \delta(s'),$$

tj.

$$\forall s \forall s' [\bigvee_{i=1}^{n} \alpha_i(s) \wedge \beta_i(s')] \Rightarrow \delta(s').$$

No, tada je

$$\forall s \forall s' [\bigwedge_{i=1}^{n} \alpha_i(s) \wedge \beta_i(s') \Rightarrow \delta(s')].$$

Po definiciji $\gamma_i(s')$*, to znači*

$$\forall s \forall s' [\bigwedge_{i=1}^{n} \gamma_i(s') \Rightarrow \delta(s')],$$

odakle sledi

$$\forall s \forall s' [\textstyle\bigwedge_{i=1}^{n} \gamma_i(s')] \Rightarrow \delta(s'). \tag{5}$$

Konačno, iz (4) *i* (5) *sledi* (3) *što smo i želeli da dokažemo.*

□

4.3.2 Algoritam za izračunavanje stroge invarijante

Neka je S_0 neki podskup skupa stanja U_K. Počev od skupa S_0 pravimo sledeću sekvencu:

$$S_0 \subseteq S_1 \subseteq S_2 \subseteq \dots$$

tako da je $S_i = S_{i-1} \cup \Delta S_{i-1}$, $i = 1, 2, \dots$, gde je ΔS_{i-1} skup svih stanja koja su dostupna iz skupa S_{i-1}, odnosno $\Delta S_{i-1} = \{s' | (s' \in U_K) \wedge (s \in S_{i-1}) \wedge \hat{JS}_K(s, s')\}$. Skup svih tako dobijenih skupova S_i, $i = 1, 2, \dots$ označićemo sa S, odnosno $S = \{S_0, S_1, \dots\}$. Neka su $Z_0, Z_1, Z_2, \dots$ predikati koji opisuju redom skupove stanja $S_0, S_1, \dots$, što znači da važi

$$Z_0 \Rightarrow Z_1 \Rightarrow Z_2 \Rightarrow \dots$$

i neka je $Z = \{Z_0, Z_1, \dots\}$. Skup S je kompletan lanac s obzirom na to da svaki njegov podskup ima najveću donju i najmanju gornju granicu. Posmatrajmo preslikavanje $\sigma_0 : Z \to Z$ oblika

$$\sigma_0(z) = z \vee z \otimes \hat{JS}_K, \ z \in Z.$$

Lema 4.3.19 *Važi* $Z_{i+1} \Leftrightarrow \sigma_0(Z_i), i = 0, 1, 2, \ldots.$

Dokaz.

Na osnovu teoreme 4.3.9, zaključujemo da za predikate iz skupa Z *važi*

$$Z_{i+1} \Leftrightarrow \sigma_0(Z_i), i = 0, 1, 2, \ldots.$$

□

Lema 4.3.20 *Funkcija* σ_0 *je monotona na lancu* Z*.*

Dokaz.

S obzirom na monotonost supstitucije (teorema 4.3.12) važi

$$(z \Rightarrow y) \Rightarrow (\sigma_0(z) \Rightarrow \sigma_0(y)),\ z, y \in Z,$$

tj. funkcija σ_0 *je monotona na lancu* Z*.*

□

Lema 4.3.21 *Funkcija* σ_0 *ima najmanju nepokretnu tačku na lancu* Z*.*

Dokaz.

Iz teoreme Tarskog [102] i leme 4.3.20 sledi da funkcija σ_0 *na kompletnom lancu* Z *ima najmanju nepokretnu tačku* Z_{fix} *za koju važi*

$$\sigma_0(Z_{fix}) \Leftrightarrow Z_{fix}.$$

□

Na osnovu leme 4.3.21 kojom smo pokazali da postoji nepokretna tačka Z_{fix}, zaključujemo da važi

$$(\forall j \geq fix) Z_j \Leftrightarrow Z_{fix},$$

gde $fix, j \in 0, 1, 2, \ldots$, a odatle zaključujemo da dobijeni predikat Z_{fix} opisuje sva stanja dostupna iz S_0 i samo njih.

Neka predikat IS_K^0 opisuje sva inicijalna stanja objekta i samo njih. Posmatramo specijalan slučaj, a to je da skup S_0 sadrži inicijalna stanja i samo njih, odnosno da važi $Z_0 \equiv IS_K^0$. U cilju jednostavnijeg izražavanja, uvodimo funkciju $fixp$ na ovaj način:

$$Z_{fix} \Leftrightarrow fixp(Z_0, \sigma_0).$$

Teorema 4.3.22 *Predikat $fixp(IS_K^0, \sigma_0)$ jeste stroga invarijanta u klase K.*

Dokaz.

Pošto važi $Z_0 \Leftrightarrow IS_K^0$, predikat Z_0 opisuje inicijalna stanja objekta i samo njih, pa na osnovu Lema 4.3.19 i 4.3.21 postoji najmanja nepokretna tačka Z_{fix} koja opisuje sva dostupna stanja počev od stanja koja opisuje Z_0. Sada ćemo dokazati ekvivalenciju $\forall s Z_{fix}(s) \Leftrightarrow (s \in V_K)$.

Skup S_0 sadrži validna inicijalna stanja objekta i samo njih, pa prema tome sva stanja koja su dostupna počev od njih posle fix prelaza, takođe jesu validna. Na osnovu toga zaključujemo da važi implikacija s leva u desno, odnosno da važi $\forall s Z_{fix}(s) \Rightarrow (s \in V_K)$.

Pretpostavimo sledeće $\exists t \in V_K$, takvo da važi $Z_{fix}(t) \equiv \perp$. To znači da stanje t nije dostupno posle fix prelaza počev od inicijalnih stanja iz skupa S_0, pa ne može biti validno. Na osnovu toga zaključujemo da važi implikacija s desna u levo, odnosno važi $\forall s Z_{fix}(s) \Rightarrow (s \in V_K)$.

Pošto smo dokazali implikaciju u oba smera zaključujemo da važi ekvivalencija $\forall s Z_{fix}(s) \Leftrightarrow (s \in V_K)$ i time je teorema dokazana.

□

Uvedimo skraćenicu $\hat{JS}_K^i$ na ovaj način:

$$\hat{JS}_K^i \leftrightarrow \underbrace{\hat{JS}_K \otimes \hat{JS}_K \otimes \cdots \otimes \hat{JS}_K}_{i}$$

i neka je po definiciji $\hat{JS}_K^i \equiv \top$. Tada važi

$$IS_K \Leftrightarrow O \vee \bigvee_{i=0}^{fix} (IS_K^0 \otimes \hat{JS}_K^i),$$

gde je redosled izvođenja operacije $\otimes$ s leva u desno, tj. $A \otimes B \otimes C \equiv ((A \otimes B) \otimes C)$. Iz ove formule direktno se izvodi rekurzivni algoritam za određivanje stroge statičke invarijante:

Algoritam 4.3.23
$Z_0 \equiv W_0 \equiv IS_K^0$
$- - - - - - - - - - - - - -$
$REPEAT$
$W_{i+1} \equiv W_i \otimes \hat{JS}_K$
$Z_{i+1} \equiv Z_i \vee W_{i+1}$
$i = 0, 1, \ldots$
$UNTIL\ Z_{i+1} \Leftrightarrow Z_i$
$- - - - - - - - - - - - - -$
$fix = i$
$\boxed{IS_K \equiv Z_i}$

4.3.3 Algoritam za izračunavanje invarijante koja nije nužno stroga

Neka je S_0 neki podskup skupa stanja U_K. Počev od skupa S_0 pravimo sledeću sekvencu:

$$S_0 \subseteq S_1 \subseteq S_2 \subseteq \dots$$

tako da je $S_i = S_{i-1} \cup \Delta S_{i-1}$, $i = 1, 2, \dots$, gde je ΔS_{i-1} skup svih stanja koja su dostupna iz skupa S_{i-1}, odnosno $\Delta S_{i-1} = \{s' | (s' \in U_K) \wedge (s \in S_{i-1}) \wedge \hat{J}_K(s, s')\}$. Skup svih tako dobijenih skupova S_i, $i = 1, 2, \dots$ označićemo sa S, odnosno $S = \{S_0, S_1, \dots\}$. Neka su $Z_0, Z_1, Z_2, \dots$ predikati koji opisuju redom skupove stanja $S_0, S_1, \dots$, što znači da važi

$$Z_0 \Rightarrow Z_1 \Rightarrow Z_2 \Rightarrow \dots$$

i neka je $Z = \{Z_0, Z_1, \dots\}$. Skup S je kompletan lanac s obzirom na to da svaki njegov podskup ima najveću donju i najmanju gornju granicu. Posmatrajmo familiju preslikavanja čiji su članovi oblika

$$\sigma_y(z) = z \vee y, \ z, y \in Z.$$

gde važi $z \wedge \hat{J}_K \Rightarrow y$.

Lema 4.3.24 *Važi $Z_{i+1} \Leftrightarrow \sigma_{y_i}(Z_i)$, $i = 0, 1, 2, \dots$.*

Dokaz.

Na osnovu teoreme 4.3.9, zaključujemo da za predikate iz skupa Z važi

$$Z_{i+1} \Leftrightarrow \sigma_{y_i}(Z_i), i = 0, 1, 2 \dots .$$

□

Lema 4.3.25 *Svaka funkcija oblika σ_y je monotona na odgovarajućem lancu Z.*

Dokaz.

Neka je tačno $Z_i \Rightarrow Z_j$. To znači da je $i \leq j$ i da je

$$Z_j \Leftrightarrow Z_i \vee y_i \vee y_{i+1} \vee \dots \vee y_{j-1},$$

te je

$$\sigma_{y_j}(Z_j) \Leftrightarrow Z_j \vee yj \Leftrightarrow Z_i \vee y_i \vee \dots \vee y_j \Leftrightarrow (Z_i \vee y_i) \vee (y_{i+1} \vee \dots \vee y_j) \Leftrightarrow \sigma_{y_i}(Z_i) \vee (y_{i+1} \vee \dots \vee y_j),$$

odakle dobijamo

$$\sigma_{y_i}(Z_i) \Rightarrow \sigma_{y_j}(Z_j).$$

□

Lema 4.3.26 *Svaka funkcija oblika σ_y ima najmanju nepokretnu tačku na lancu Z.*

Dokaz.

Iz teoreme Tarskog i Leme 4.3.25 sledi da funkcija σ_y ima na lancu Z najmanju nepokretnu tačku Z_{fix} za koju važi

$$\sigma_y(Z_{fix}) \Leftrightarrow Z_{fix}.$$

□

Na osnovu Leme 4.3.26 kojom smo pokazali da postoji nepokretna tačka Z_{fix}, zaključujemo da važi

$$(\forall j \geq fix) Z_j \Leftrightarrow Z_{fix},$$

gde $fix, j \in \{0, 1, 2, \dots\}$.

Neka skup S_0 sadrži inicijalna stanja objekta i samo njih, odnosno važi $Z_0 \equiv IS_K^0$. U cilju jednostavnijeg izražavanja, uvodimo funkciju $fixp$ na ovaj način:

$$Z_{fix} \Leftrightarrow fixp(Z_0, \sigma_y).$$

Teorema 4.3.27 *Predikat $fixp(IS_K^0, \sigma_y)$ jeste invarijanta u klasi K koja ne mora nužno biti stroga.*

Dokaz.

Posmatrajmo lanac predikata $Z = \{Z_0, Z_1, \dots\}$, gde je $Z_0 \Leftrightarrow IS_K^0$ i $Z_{i+1} \Leftrightarrow Z_i \vee y_i$, $i = 0, 1, 2, \dots$. Prvo, Z_0 opisuje validna inicijalna stanja.

Dalje, neka predikat $Z_i, i \geq 0$ opisuje sva validna stanja, a pored njih eventualno još i neka invalidna stanja. Za sledeći predikat u lancu Z_{i+1} važi

$$Z_{i+1} \Leftrightarrow Z_i \vee y_i,$$

gde

$$Z_i \otimes \hat{J}_K \Rightarrow y_i.$$

Kako važi $\hat{JS}_K \Rightarrow \hat{J}_K$, prema teoremi 4.3.12 (ili 4.3.10) biće

$$Z_i \otimes \hat{JS}_K \Rightarrow Z_i \otimes \hat{J}_K,$$

odnosno

$$Z_i \otimes \hat{JS}_K \Rightarrow y_i.$$

Prema teoremi 4.3.9 supstitucija $Z_i \otimes \hat{JS}_K$ obuhvata sva stanja dostupna iz Z_i u jednoj promeni stanja, pa stoga y_i obuhvata bar sva validna stanja. Dakle, ako Z_i opisuje bar sva validna stanja dostupna u i-tom koraku tada $Z_i \vee y_i$ opisuje bar sva validna stanja dostupna u $i+1$-vom koraku, čime je teorema dokazana.

□

Iz teoreme 4.3.27 neposredno se izvodi rekurzivni algoritam za određivanje statičke invarijante koja nije nužno stroga:

Algoritam 4.3.28
$Z_0 \equiv W_0 \equiv IS_K^0$
$- - - - - - - - - - - - - -$
$REPEAT$
$W_{i+1} \Rightarrow W_i \wedge \hat{J}_K$
$Z_{i+1} \Rightarrow Z_i \vee W_{i+1}$
$i = 0, 1, \ldots$
$UNTIL\ Z_{i+1} \Leftrightarrow Z_i$
$- - - - - - - - - - - - - -$
$fix = i$
$\boxed{I_K \equiv Z_i}$

4.4 Interpretacija

Apstraktni prostor stanja U_K interpretiramo na skupu polja $\Phi_K = \{\phi_1, \ldots, \phi_k\}$ koja mogu biti kako nestatička tako i statička. Interpretaciju izvodimo tako što svakom apstraktnom stanju iz skupa U_K pridružujemo vektor polja $\bar{\phi} = (\phi_1, \ldots, \phi_k)$. Predikat P kojeg smo definisali na apstraktnom skupu stanja sada interpretiramo kao bulov izraz $P(\bar{\phi})$. Na osnovu skupa polja Φ_K, skupa metoda M_K i njihovih dinamičkih postuslova, a primenom Algoritama 4.3.23 i/ili 4.3.28 možemo formirati različite vrste invarijanata. U ovom poglavlju ćemo razmotriti primenu DP-analize na konkretnim primerima.

Prvo ćemo reći nešto o interpretaciji najužeg dinamičkog postuslova. Neka je apstraktni prostor stanja klase interpertiran na skupu polja klase $\Phi_K = \{\phi_1, \ldots, \phi_k\}$. Neka je m metoda koja referencira polja $F_m = \{f_1, \ldots, f_r\}$ i ne referencira polja $G_m = \{g_1, \ldots, g_s\}$, $F_m \bigcup G_m = \Phi_K$, $F_m \bigcap G_m = \emptyset$. Posmatrajmo vektor polja $\bar{f} = (f_1, \ldots, f_r)$. Bulov izraz za strogi dinamički postuslov preduslova $\Gamma(m)$ jeste

$$\hat{sdp}(m, \Gamma(m))(\bar{\phi}, \bar{\phi}') \equiv \Gamma(m)(\bar{f}) \wedge \hat{\beta}(m, \Gamma(m))(\bar{f}, \bar{f}') \wedge \bigwedge_{g \in G_m} (g' = g),$$

, a bulov izraz za minimalni predikat prelaza je

$$\hat{\alpha}(m, \Gamma(m))(\bar{\phi}, \bar{\phi}') \equiv \hat{\beta}(m, \Gamma(m))(\bar{f}, \bar{f}') \wedge \bigwedge_{g \in G_m} (g' = g).$$

Teorema 4.3.18 neposredno omogućuje primenu supstitucije na intepretiranom prostoru stanja. Neka je prostor stanja interpretiran na skupu polja $\Phi = \{\phi_1, \ldots, \phi_k\}$. Neka su D_{ϕ_j}, $j = 1, \ldots, k$ domeni polja $\phi_1, \ldots, \phi_k$ respektivno. Uvedimo sledeće oznake:

$$\bar{\phi} = (\phi_1, \ldots, \phi_k),$$

$$D_{\bar{\phi}} = D_{\phi_1} \times \cdots \times D_{\phi_k},$$

$$C_i \subseteq D_{\bar{\phi}},$$

$$\rho_i \subseteq D_{\bar{\phi}} \times D_{\bar{\phi}},$$

$i = 1, \ldots, n$, gde je n neki prirodan broj.

Teorema 4.4.1 (Primena supstitucije) *Neka je $A \wedge \hat{B}$ oblika*

$$A(\bar{\phi}) \wedge \hat{B}(\bar{\phi}, \bar{\phi}') \Leftrightarrow \bigvee_{i=1}^{n} \alpha_i(\bar{\phi}) \wedge \beta_i(\bar{\phi}')$$

i neka je

$$C_i' = \{\bar{c}' | (\bar{c} \in C_i) \wedge ((\bar{c}, \bar{c}') \in \rho_i)\},\ i = 1, \ldots, n,$$

tada je

$$A \otimes \hat{B} \Leftrightarrow \bigvee_{i=1}^{n} (\bar{\phi}' \in C_i'),$$

gde je n neki prirodan broj.

Dokaz.

Direktno, iz teoreme 4.3.18. Prvo, po definiciji C_i' sledi

$$(\bar{\phi} \in C_i) \wedge ((\bar{\phi}, \bar{\phi}') \in \rho_i) \Rightarrow (\bar{\phi}' \in C_i'),\ i = 1, \ldots, n.$$

Neka je $\bar{\phi_0}'$ vrednost $\bar{\phi}'$ za koju ne postoji $\phi \in C_i,\ i = 1, \ldots, n$ takvo da važi

$$(\bar{\phi} \in C_i) \wedge ((\bar{\phi}, \bar{\phi_0}') \in \rho_i),\ i = 1, \ldots, n.$$

Tada, na osnovu definicije C_i' mora važiti

$$\bar{\phi_0}' \notin C_i',\ i = 1, \ldots, n,$$

što znači da je $\bar{\phi}' \in C_i',\ i = 1, \ldots, n$ minimalna konsekventa izraza

$$(\bar{\phi} \in C_i) \wedge ((\bar{\phi}, \bar{\phi}') \in \rho_i).$$

□

Evo nekoliko primera primene teoreme 4.4.1:
$(x = 1) \otimes (x' = x + 1) \Leftrightarrow (x' = 2)$,
$(1 \leq x \leq a) \otimes (x' < x + 1) \Leftrightarrow (x' < a + 1)$,
$(1 \leq x \leq a) \otimes (x' > x + 1) \Leftrightarrow (x' > 2)$.

U osnovi, invarijante se dele na invarijante objekta i invarijante klase. *Invarijanta objekta* podrazumeva da se sve metode aktiviraju preko jednog i samo jednog objekta-predstavnika (tzv. *this*) i opisuju sva validna stanja u kojima se taj objekat može naći. Invarijanta objekta može biti čista kada su sva polja i metode nestatičke, odnosno mešovita kada među poljima i metodama ima i statičkih. Invarijanta klase opisuje sva validna stanja u kojima se može naći data klasa. Takođe, invarijanta klase može biti čista (promene stanja vrše se statičkim modifikatorima, konstruktorima i destruktorom) ili mešovita (promene stanja izazivaju svi modifikatori, konstruktori i destruktor). Konačno, invarijanta može biti parcijalna kada interpretirani prostor stanja obuhvata samo neka polja i totalna kada ih obuhvata sva.

Invarijanta objekta

Čista invarijanta objekta dobija se na sledeći način:

1.) Skup polja Φ_K sadrži isključivo nestatička polja od kojih formiramo vektor polja $\bar{\phi}$.

2.) Skup metoda M_K sastoji se od nestatičkih modifikatora.

3.) Nulti predikat se interpretira bulovim izrazom $O(\bar{\phi}) \equiv (this = null)$.

4.) Struktura dinamičkog postuslova za konstruktor $c \in C_K$ ima oblik

$$\hat{sdp}(c, \Gamma(c)) \equiv O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge \hat{\alpha}(\bar{\phi}, \bar{\phi}').$$

5.) Struktura dinamičkog postuslova za destruktor d ima oblik

$$\hat{sdp}(d, \Gamma(d)) \equiv \neg O(\bar{\phi}) \wedge O(\bar{\phi}').$$

6.) Struktura dinamičkog postuslova za modifikator $m \in M_K$ jeste

$$\hat{sdp}(m, \Gamma(m)) \equiv \neg O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge \Gamma(m)(\bar{f}) \wedge \hat{\beta}(\bar{f}, \bar{f}') \wedge \bigwedge_{g \in G_m} (g' = g),$$

gde je $\bar{f}$ vektor polja koja m referencira i koja čine skup F_m, a G_m je skup polja koje m ne referencira, $F_m \cup G_m = \Phi_K$, $F_m \cap G_m = \emptyset$.

7.) Stroga dinamička invarijanta jeste:

$$\hat{JS}_K \equiv \bigwedge_{m \in M_K} \hat{sdp}(m, \Gamma(m)).$$

8.) Bulov izraz $IS^0_K(\bar{\phi})$ opisuje sva inicijalna stanja i samo njih.

9.) Uzimamo da važi $Z_0 \equiv W_0 \equiv IS^0_K(\bar{\phi})$ i dalje primenjujemo Algoritam 4.3.23.

Postupak za određivanje invarijante koja nije stroga dobija se zamenom $\hat{JS}_K$ sa $\hat{J}_K$ i primenom Algoritma 4.3.28.

Mešovita invarijanta objekta dobija se ovako:

1.) Skup polja Φ_K sadrži nestatička i statička polja od kojih formiramo vektor polja $\bar{\phi}$.

2.) Skup metoda M_K sastoji se od nestatičkih i statičkih modifikatora.

3.) Nulti predikat se interpretira bulovim izrazom $O(\bar{\phi}) \equiv (this = null)$.

4.) Koraci 4.), 5.), 6.) i 7.) su isti kao kod čiste invarijante objekta.

5.) Bulov izraz $IS^0_K(\bar{\phi})$ opisuje sva inicijalna stanja i samo njih (opisuje inicijalne vrednosti statičkih i nestatičkih polja).

6.) Uzimamo da važi $Z_0 \equiv W_0 \equiv IS^0_K(\bar{\phi})$ i dalje primenjujemo Algoritam 4.3.23.

Postupak za određivanje invarijante koja nije stroga dobija se zamenom $\hat{JS}_K$ sa $\hat{J}_K$ i primenom Algoritma 4.3.28.

Invarijanta klase

Invarijanta klase odnosi se na njena statička polja koja se mogu modifikovati statičkim, ali i nestatičkim metodama. Invarijanta klase dobija se sledećim postupkom:

1.) Skup polja Φ_K sadrži samo statička polja od kojih formiramo vektor polja $\bar{\phi}$.

2.) Skup metoda M_K sadrži modifikatore uključujući, po potrebi, i konstruktore i destruktor.

3.) Struktura postuslova svih modifikatora uključujući i konstruktore i destruktor oblika je

$$\hat{sdp}(m, \Gamma(m)) \equiv \Gamma(m)(\bar{f}) \wedge \hat{\beta}(\bar{f}, \bar{f}') \wedge \bigwedge_{g \in G_m} (g' = g),$$

gde je $\bar{f}$ vektor statičkih polja koja m referencira i koja čine skup F_m, a G_m je skup statičkih polja koje m ne referencira, $F_m \cup G_m = \Phi_K$, $F_m \cap G_m = \emptyset$.

4.) Bulov izraz $IS^0_K(\bar{\phi})$ opisuje inicijalne vrednosti svih statičkih polja.

5.) Uzimamo da važi $Z_0 \equiv W_0 \equiv IS^0_K(\bar{\phi})$ i dalje primenjujemo Algoritam 4.3.23.

Postupak za određivanje invarijante koja nije stroga dobija se zamenom $\hat{JS}_K$ sa $\hat{J}_K$ i primenom Algoritma 4.3.28.

4.4.1 Parametri metoda

Parametri metoda ponešto usložnjavaju postupak dinamičke analize. Glavni razlog jeste činjenica da su njihovi stvarni oblici (tj. argumenti) poznati tek u trenutku poziva, a da sami parametri imaju prirodu lokalnih promenljivih. Njihova osnovna osobina u odnosu na modelovanje dinamičkog postuslova metode jeste to što nisu deo opisa stanja, što znači da se moraju na neki način eliminisati iz dinamičkog postuslova. Pokazaćemo kako se primenjuje teorema 4.3.2 za slučaj da metoda ima parametre. Neka je apstraktni prostor stanja klase K interpretiran na skupu polja $\Phi_K = \{\phi_1, \ldots, \phi_k\}$ i neka je $\bar{\phi} = (\phi_1, \ldots, \phi_k)$. Posmatrajmo metodu $m(\bar{\pi})$ gde je $\bar{\pi}$ vektor (formalnih, ulaznih i ulazno-izlaznih) parametara. Neka je $\Gamma(m)(\bar{\pi}, \bar{\phi})$ zaštitni preduslov te metode. Najuži dinamički postuslov metode odeđuje se u dva koraka. U prvom koraku se iz izvornog koda određuje najuži dinamički postuslov $\hat{Q}_{sdp}(\bar{\pi}, \bar{\phi}, \bar{\phi}')$, pri čemu se parametri metode tretiraju kao parametri postuslova. U drugom koraku eliminišu se parametri putem restrikcije $(\bar{\pi}, \bar{\phi}, \bar{\phi}')$ na $(\bar{\phi}, \bar{\phi}')$. Neka je $\Pi(\bar{\pi})$ izraz koji opisuje sve moguće vrednosti parametara metode, odnosno opisuje tip svakog parametra pojedinačno. Tada važi

$$\hat{sdp}(m, \Gamma(m)) \equiv (\Pi \wedge \hat{Q}_{sdp})[(\bar{\pi}, \bar{\phi}, \bar{\phi}')\backslash(\bar{\phi}, \bar{\phi}')],$$

gde $[(\bar{\pi}, \bar{\phi}, \bar{\phi}')\backslash(\bar{\phi}, \bar{\phi}')]$ označava restrikciju $(\bar{\pi}, \bar{\phi}, \bar{\phi}')$ na $(\bar{\phi}, \bar{\phi}')$, tj. $(\Pi \wedge \hat{Q}_{sdp})[(\bar{\pi}, \bar{\phi}, \bar{\phi}')\backslash(\bar{\phi}, \bar{\phi}')]$ je minimalna konsekvenca $\Pi \wedge \hat{Q}_{sdp}$ koja ne zavisi od $\bar{\pi}$.

Analogno teoremi 4.3.2, $\hat{Q}_{sdp}(\bar{\pi}, \bar{\phi}, \bar{\phi}')$ opisuje sve dozvoljene kombinacije $(\bar{\pi}, \bar{\phi}, \bar{\phi}')$ koje ostvaruje metoda m. Dalje, ako je $(\bar{\phi}_0, \bar{\phi}'_0)$ prelaz koji se ne može ostvariti iz preduslova $\Gamma(m)$, tada odgovarajući izraz nije minimalna konsekvenca i treba ga suziti isključivanjem tog prelaza (opet analogno teoremi 4.3.2).

Ilustrovaćemo postupak jednostavnim primerom. Neka je $add(L\ obL)$ metoda klase K, koja uvećava polje $int\ y$ za iznos $obL.getX()$, gde je $int\ x$ polje klase L za koje znamo da je veće od 0, tj.

```
public void add(L obL) {
     if(obL==null) throw RuntimeException();
     y=y+obL.getX();
}
```

U prvom koraku određujemo $\hat{Q}_{sdp}$, najuži dinamički postuslov koji sadrži obL kao parametar:

$$\hat{Q}_{sdp}(\bar{\pi}, \bar{\phi}, \bar{\phi}') \equiv \neg O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (obL \neq null) \wedge (y' = y + obL.getX()),$$

gde je $\bar{\pi} = (obL)$ i $\bar{\phi} = (y)$. Iz činjenice da obL pripada klasi L, tj. važi predikat $\Pi(\bar{\pi}) \equiv (L\ obL)$, sledi da je $obL.getX() > 0$. Očigledno, važi $\Gamma(add)(\bar{\pi}, \bar{\phi}) \equiv \neg O(\bar{\phi}) \wedge (obL \neq null)$.

Svođenjem prostora stanja $(\bar{\pi}, \bar{\phi})$ na $(\bar{\phi})$ dobijamo

$$\hat{sdp}(add, \Gamma(add))(\bar{\phi}) \equiv \neg O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (y' > y).$$

4.4.2 Osvrt na interpretirani prostor stanja

Posmatrajmo neku klasu K koja sadrži dva celobrojna polja, a i b. Posmatrajmo metodu m:

```
public void m() {
     if(a!=b) throw RuntimeException();
     a++;
     b++;
}
```

Neka se prostor stanja U_K interpretira pomoću vektora $\bar{\phi} = (a)$, a polje b ćemo pokušati da tretiramo kao parametar, tako da važi

$$\hat{sdp}(m, \Gamma(m)) \equiv \neg O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (a = b) \wedge (a' = a + 1),$$

gde je $\bar{\pi} = (b)$.

Svođenjem prostora stanja $(\bar{\pi}, \bar{\phi})$ na $(\bar{\phi})$ naziru se dva problema:

- Ne može se u svakoj situaciji ustanoviti da li važi $(a = b)$, s obzirom na činjenicu da je b parametar.
- U interpretaciji pomoću vektora $\bar{\phi} = (a)$ ne može se detektovati promena vrednosti b metodom m.

Neka je potrebno odrediti strogu invarijantu objekta za klasu K. Neka je čak poznato da u nekom stanju važi $a = b$. Primena metode m daće za rezultat

$$(\neg O(\bar{\phi}) \wedge (a = b)) \otimes \hat{sdp}(m, \Gamma(m)) \equiv \neg O(\bar{\phi}) \wedge (a = b + 1).$$

Već sledeća primena iste metode daće

$$(\neg O(\bar{\phi}) \wedge (a = b + 1)) \otimes \hat{sdp}(m, \Gamma(m)) \equiv \bot,$$

što nije korektan rezultat jer nije uzeta u obzir i promena vrednosti polja b, tj. činjenica da posle prve primene m i da i dalje važi $(a = b)$, a ne $(a = b + 1)$. Inače, situacija može biti još i gora – naime, može se dogoditi da se pre primene metode m uopšte ne zna da li važi $(a = b)$ ili ne, što znači da je supstitucija neprimenljiva. Da bismo izbegli ovakav i slične probleme, moramo prihvatiti još jednu, logičnu, pretpostavku:

Pretpostavka 4.4.2 *Interpretacija prostora stanja U_K mora biti zatvorena u odnosu na metode, što znači da sva polja koja se referenciraju u $\hat{JS}_K$ moraju pripadati vektoru $\bar{\phi}$.*

4.4.3 Primer: klasa $IntFactory$

Posmatrajmo klase $IntFactory$ i $MyInt$:

```
class IntFactory {
    private myInt[] ints;
    public IntFactory() {
          ints= new MyInt[1000];
     }
     public MyInt get Int(int x) {
          if(ints[x] == null)
               ints[x] = new MyInt(x);
          return ints[x];
     }
}

class MyInt {
     private int x;
     public MyInt(int x) {
          this.x = x;
     }
     public boolean equals(MyInt i) {
          return i==this;
     }
}
```

Odredićemo čistu invarijantu objekta. S obzirom na činjenicu da se prostor stanja može proizvoljno interpretirati, odlučićemo se za vektor koji sadrži polja $ints[a]$ i $ints.length$, gde je $ints[a]$ proizvoljan element niza $ints$, pri čemu se a tretira kao konstanta i podrazumeva da je $0 \leq a \leq ints.length$. Neka je

$$\bar{\phi} = (ints[a], ints.length).$$

Za nulti predikat važi:

$$O(\bar{\phi}) \equiv (this = null).$$

Prvo, iz izvornog koda konstruktora $IntFactory$ i metode $getInt$ zaključujemo

$\Gamma(IntFactory) \equiv O(\bar{\phi})$,
$\hat{sdp}(IntFactory, O) \equiv O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (ints[a]' = null) \wedge (ints.length' = 1000)$,
$IS^0_K(\bar{\phi}) \equiv \neg O(\bar{\phi}) \wedge (ints[a] = null) \wedge (ints.length = 1000)$,

$\Gamma(getInt) \equiv \neg O(\bar{\phi})$,

$\hat{sdp}(getInt, \Gamma(getInt)) \equiv \neg O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (ints.length' = ints.length) \wedge [(ints[a]' = ints[a]) \vee ((ints[a] = null) \wedge (ints[a]' = i(a)))]$,

gde je $i(a)$ objekat klase $MyInt$ konstruisan sa argumentom a. Na osnovu toga dobijamo:

$\hat{JS} \equiv \hat{sdp}(IntFactory, \Gamma(IntFactory)) \vee \hat{sdp}(getInt, \Gamma(getInt))$
$\equiv \{O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (ints[a]' = null) \wedge (ints.length' = 1000)\} \vee \{\neg O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (ints.length' = ints.length) \wedge [(ints[a]' = ints[a]) \vee ((ints[a] = null) \wedge (ints[a]' = i(a)))]\}$

Usvajamo da je:

$Z_0 \equiv W_0 \equiv IS_K^0(\bar{\phi})$

i primenom algoritma 4.3.23 određujemo strogu invarijantu:

$Z_0 \equiv W_0 \equiv IS_K^0(\bar{\phi})$
$W_0 \wedge \hat{JS} \equiv \neg O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (ints.length = 1000) \wedge (ints.length' = ints.length) \wedge (ints[a] = null) \wedge [(ints[a]' = ints[a]) \vee ((ints[a] = null) \wedge (ints[a]' = i(a)))]$
$W_1 \equiv W_0 \otimes \hat{JS} \equiv \neg O(\bar{\phi}) \wedge (ints.length = 1000) \wedge [(ints[a] = null) \vee (ints[a] = i(a))]$
$Z_1 \equiv Z_0 \vee W_1$
$W_1 \wedge \hat{JS} \equiv \neg O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (ints.length = 1000) \wedge (ints.length' = ints.length) \wedge [(ints[a] = null) \vee (ints[a] = i(a))] \wedge [(ints[a]' = ints[a]) \vee ((ints[a] = null) \wedge (ints[a]' = i(a)))]$
$W_2 \equiv W_1 \otimes \hat{JS} \equiv \neg O(\bar{\phi}) \wedge (ints.length = 1000) \wedge [(ints[a] = null) \vee (ints[a] = i(a))]$
$Z_2 \equiv Z_1 \vee W_2$

Pošto važi $Z_1 \equiv Z_2$ zaključujemo da je nepokretna tačka predikat Z_1 i da važi

$Z_1 \equiv Z_2 \equiv Z_3 \equiv Z_4 \ldots$

pa je:
$IS_{IntFactory} \equiv Z_1$
$IS_{IntFactory} \equiv O(\bar{\phi}) \vee \{\neg O(\bar{\phi}) \wedge (ints.length = 1000) \wedge [(ints[a] = null) \vee (ints[a] = i(a))]\}$
$IS_{IntFactory} \equiv (this = null) \vee \{(this \neq null) \wedge (ints.length = 1000) \wedge [(ints[a] = null) \vee (ints[a] = i(a))]\}$.

Uočimo da je ova invarijanta nešto preciznija od one date u [86], jer potonja ne obuhvata slučaj $ints[a] = null$.

4.4.4 Primer: klasa *Singleton*

Razmotrimo strogu invarijantu za patern *Singleton* koji je analiziran u [86]:

```
class Singleton {
     private static Singleton unique;
     private Singleton();
     public static Singleton getInstance() {
          if(unique == null)
               unique=new Singleton();
          return unique;
     }
}
```

Osnovna osobina paterna *Singleton* jeste ta da u klijentu klase može da egzistira najviše jedna i to stalno ista instanca klase. Prostor stanja interpretira vektor koji sadrži statičko polje *unique*. Neka je

$$\bar{\phi} = (unique).$$

U ovom primeru određujemo čistu invarijantu klase. Bulov izraz IS_K^0 opisuje inicijalnu vrednost statičkog polja *unique*, odnosno:
$IS_K^0(\bar{\phi}) \equiv (unique = null)$.

Prvo, zaključujemo
$\Gamma(Singleton) \equiv (unique = null)$,
$\hat{sdp}(Singleton, \Gamma(Singleton)) \equiv (unique = null) \wedge (unique' = null)$,
$\Gamma(getInstance) \equiv (unique = null)$,
$\hat{sdp}(getInstance, \Gamma(getInstance)) \equiv (unique = null) \wedge (unique' = null) \wedge [((unique = null) \wedge (unique' = ob)) \vee ((unique = ob) \wedge (unique' = unique))]$,

odakle dobijamo:

$\hat{JS} \equiv \{(unique = null) \wedge (unique' = null)\} \vee \{(unique = null) \wedge (unique' = null) \wedge [((unique = null) \wedge (unique' = ob)) \vee ((unique = ob) \wedge (unique' = unique))]\}$
$\equiv ((unique = null) \wedge (unique' = ob)) \vee ((unique = ob) \wedge (unique' = unique))$.

Usvajamo da je:

$Z_0 \equiv W_0 \equiv IS_K^0(\bar{\phi})$

i primenjujemo algoritam 4.3.23:

$Z_0 \equiv W_0 \equiv IS_K^0(\bar{\phi})$
$W_0 \wedge \hat{JS} \equiv (unique = null) \wedge [((unique = null) \wedge (unique' = ob)) \vee ((unique = ob) \wedge (unique' = unique))]$
$W1 \equiv W_0 \otimes \hat{JS} \equiv (unique = ob)$
$Z_1 \equiv Z_0 \vee W_1$
$W_1 \wedge \hat{JS} \equiv (unique = ob) \wedge [((unique = null) \wedge (unique' = ob)) \vee ((unique = ob) \wedge (unique' = unique))]$
$W_2 \equiv W_1 \otimes \hat{JS} \equiv (unique = ob)$
$Z_2 \equiv Z_1 \vee W_2$

Pošto važi $Z_1 \equiv Z_2$ zaključujemo da je nepokretna tačka predikat Z_1 i da važi

$Z_1 \equiv Z_2 \equiv Z_3 \equiv Z_4 \ldots$

pa je:

$IS_{Singleton} \equiv Z_1$
$IS_{Singleton} \equiv (unique = null) \vee (unique = ob)$,

gde je $ob = const$. Očigledno, klasa *Singleton* je ili u stanju u kojem ne postoji nijedan kreirani objekat ili je u stanju sa kreiranim jednim i to uvek istim objektom.

4.4.5 Primer: klasa *Stack*

Razmotrićemo sekvencijalni stek sa najviše $CAPACITY$ elemenata. Realizacija steka ima sledeći oblik:

```
/**
Sequential stack.
*/
public class Stack {
     private int t, c;
     private Object[] body;

/**
Constructs an instance of the class Stack.
@param CAPACITY Stack c (max.  number of elements).
@throws RuntimeException if CAPACITY <= 0.
*/
```

```
public Stack(int CAPACITY) {
     if(CAPACITY<=0) throw new RuntimeException("Stack constructor underflow.");
     body=new Object [CAPACITY];
     c=CAPACITY;
     t= -1;
}

/**
Checks if stack is empty.
*/
public boolean empty() { return t<0; }

/**
Checks if stack is full.
*/
public boolean full() { return t==c; }

/**
Reads the top element.
@throws RuntimeException if the stack is empty.
*/
public Object top() {
     if(empty()) throw new RuntimeException("Stack underflow.");
     return body[t];
}

/**
Pops the stack.
@throws RuntimeException if t<0.
*/
public void pop() {
     if(empty()) throw new RuntimeException("Stack underflow.");
     t--;
}

/**
Pushes an element into the stack.
@throws RuntimeException if t=c.
*/
public void push(Object el) {
     if(full()) throw new RuntimeException("Stack overflow");
     body[++t]=el;
```

}
}

Odredićemo čistu invarijantu objekta za prostor stanja interpretiran poljima t i c. Neka je

$$\bar{\phi} = (t, c).$$

Svrha ovog primera je da se prikaže slučaj kada metode imaju parametre. Naime, konstruktor ima parametar $CAPACITY$. Za nulti predikat važi:

$$O(\bar{\phi}) \equiv (this = null).$$

Za početak, zaključujemo:
$\Gamma(Stack) \equiv O(\bar{\phi}) \wedge (CAPACITY > 0)$,
$\hat{sdp}(Stack, \Gamma(Stack)) \equiv O(\bar{\phi}) \wedge (CAPACITY > 0) \wedge \neg O(\bar{\phi}') \wedge (t' = -1) \wedge (c' = CAPACITY)$,

odakle uvođenjem restrikcije vektora $(CAPACITY, t, c)$ na (t, c), kao što je opisano u poglavlju 4.4.1, dobijamo:

$\hat{sdp}(Stack, \Gamma(Stack)) \equiv O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (t' = -1) \wedge (c' > 0)$,
$IS^0_K(\bar{\phi}) \equiv \neg O(\bar{\phi}) \wedge (t = -1) \wedge (c > 0)$.

Dalje, zaključujemo:

$\Gamma(pop) \equiv \neg O(\bar{\phi}) \wedge (t > -1)$,
$\hat{sdp}(pop, \Gamma(pop)) \equiv \neg O(\bar{\phi}) \wedge (t > -1) \wedge \neg O(\bar{\phi}') \wedge (t' = t - 1) \wedge (c' = c)$,
$\Gamma(push) \equiv \neg O(\bar{\phi}) \wedge (t < c - 1)$,
$\hat{sdp}(push, \Gamma(push) \equiv \neg O(\bar{\phi}) \wedge (t < c - 1) \wedge \neg O(\bar{\phi}') \wedge (t' = t + 1) \wedge (c' = c)$,

odakle dobijamo:

$\hat{JS} \equiv [O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (t' = -1) \wedge (c' > 0)] \vee [\neg O(\bar{\phi}) \wedge (t > -1) \wedge \neg O(\bar{\phi}') \wedge (t' = t-1) \wedge (c' = c)] \vee [\neg O(\bar{\phi}) \wedge (t < c - 1) \wedge \neg O(\bar{\phi}') \wedge (t' = t + 1) \wedge (c' = c)]$.

Usvajamo da je:

$Z_0 \equiv W_0 \equiv IS^0_K(\bar{\phi})$

i primenjujemo algoritam 4.3.23:

$Z_0 \equiv W_0 \equiv IS^0_K(\bar{\phi})$
$W_0 \wedge \hat{JS} \equiv \neg O(\bar{\phi}) \wedge (t = -1) \wedge (t < c-1) \wedge (c > 0) \wedge (t < c-1) \wedge \neg O(\bar{\phi}') \wedge (t' = t+1) \wedge (c' = c)$,
$W_1 \equiv W_0 \otimes \hat{JS} \equiv \neg O(\bar{\phi}) \wedge (t = 0) \wedge (t < c) \wedge (c > 0)$
$Z_1 \equiv Z_0 \vee W_1$

očigledno, dalje bi dobili sledeće:

$W_2 \equiv W_1 \otimes \hat{JS} \equiv \neg O(\bar{\phi}) \wedge ((t = -1) \vee (t = 1)) \wedge (t < c) \wedge (c > 0)$
$Z_2 \equiv Z_1 \vee W_2$
$W_3 \equiv W_2 \otimes \hat{JS} \equiv \neg O(\bar{\phi}) \wedge ((t = 0) \vee (t = 2)) \wedge (t < c) \wedge (c > 0)$
$Z_3 \equiv Z_2 \vee W_3$
$W_4 \equiv W_3 \otimes \hat{JS} \equiv \neg O(\bar{\phi}) \wedge ((t = -1) \vee (t = 1) \vee (t = 3)) \wedge (t < c) \wedge (c > 0)$
$Z_4 \equiv Z_3 \vee W_4$
$W_5 \equiv W_4 \otimes \hat{JS} \equiv \neg O(\bar{\phi}) \wedge ((t = 0) \vee (t = 2) \vee (t = 4)) \wedge (t < c) \wedge (c > 0)$
$Z_5 \equiv Z_4 \vee W_5$

Indukcijom dokazujemo

$$W_{2j} \equiv \neg O(\bar{\phi}) \wedge (t \in \{0, 2, \ldots, 2j-2\}) \wedge (t < c) \wedge (c > 0),$$

$$W_{2j+1} \equiv \neg O(\bar{\phi}) \wedge (t \in \{-1, 1, 3, \ldots, 2j-1\}) \wedge (t < c) \wedge (c > 0),$$

odakle dobijamo

$$Z_n \equiv \neg O(\bar{\phi}) \wedge (c > 0) \wedge (t \in \{-1, 0, 1, 2, \ldots, c-1\}).$$

Pošto važi $Z_n \equiv Z_{n+1}$ zaključujemo da je nepokretna tačka predikat Z_n i da važi

$$Z_n \equiv Z_{n+1} \equiv Z_{n+2} \equiv Z_{n+3} \ldots$$

pa je

$$IS_{Stack} \equiv (this = null) \vee [(this \neq null) \wedge (c > 0) \wedge (t \in \{-1, 0, 1, 2, \ldots, c-1\})].$$

4.5 Vlasnik i komponente

Neka je Φ skup polja klase relevantnih za određivanje invarijante. U ovaj skup mogu se uključiti i polja koja pripadaju komponentama klase, ali pod određenim uslovima diktiranim dostupnošću komponente. Pre nego što razmotrimo te uslove dokazaćemo dve teoreme.

Teorema 4.5.1 *Dat je skup polja* $\Phi = (f_1, ..., f_n, g_1, \ldots, g_k)$, *gde je* $k \geq 1$. *Neka je* $\bar{\phi} = (f_1, \ldots, f_n, g_1, \ldots, g_k)$ *uređena* $(n+k)$*-torka i neka je* $\bar{g} = (g_1, \ldots, g_k)$ *uređena* k*-torka polja klase* K. *Neka su* $IS_K(\bar{\phi})$ *i* $IS_K(\bar{g})$ *stroge invarijante u klasi u prostorima*

stanja određenim redom sa $\bar{\phi}$ *i* $\bar{g}$*. Neka je* $\bar{\phi}\backslash\bar{g}$ *restrikcija* $(n+k)$*-torki* $\bar{\phi}$ *na* k*-torke* $\bar{g}$*. Tada važi:*

$$IS_K(\bar{\phi}\backslash\bar{g}) = IS_K(\bar{g}).$$

Dokaz.

Dokaz direktno sledi iz definicije validnog stanja, kao stanja u kojem se objekat nalazi kada ni jedna metoda nije aktivna, pri čemu definicija ne zavisi od interpretacije apstraktnog skupa stanja. Shodno tome, svakom validnom stanju odgovara tačno jedna k*-torka* $(g_1, \ldots, g_k)$*. Kako* $IS_K(\bar{\phi}\backslash\bar{g})$*, odnosno* $IS_K(\bar{g})$ *opisuju sva validna stanja i samo njih, svaka od pomenutih* k*-torki obuhvaćena je kako sa* $IS_K(\bar{\phi}\backslash\bar{g})$ *tako i sa* $IS_K(\bar{g})$*.*

□

Teorema 4.5.2 *Za vektore* $\bar{\phi}$ *i* $\bar{g}$ *iz prethodne teoreme važi*

$$IS_K(\bar{\phi}) \Rightarrow IS_K(\bar{g}).$$

Dokaz.

Direktno, iz teoreme 4.5.1 i činjenice da za svaku restrikciju $\bar{\phi}\backslash\bar{g}$ *važi* $IS_K(\bar{\phi}) \Rightarrow IS_K(\bar{\phi}\backslash\bar{g})$*.*

□

Neka je data klasa K koja sadrži polje p klase L i čije se metode referišu na statičke članove klase N, pri čemu L i N može i ne mora biti ista klasa. Neka je stroga invarijanta u klasi K oblika

$$IS_K(f_1, \ldots, f_m, p.g_1, \ldots, p.g_n, N.h_1, \ldots, N.h_t),$$

gde su $f_1, \ldots, f_m$ polja klase K, $p.g_1, \ldots, p.g_n$ nestatička polja klase L i $N.h_1, \ldots, N.h_t$ statička polja klase N. Međusobni odnos invarijanata u klasama K, L i N diktiran je dostupnošću klasa L odnosno N. U zavisnosti od toga da li se radi o polju p ili statičkim članovima klase N razlikujemo 4 slučaja:

1. Polje p je nedostupno za treću stranu uključujući i rekurzivni pristup iz klase K.
2. Polje p je dostupno za treću stranu.
3. Klasa N je nedostupna za treću stranu uključujući i rekurzivni pristup iz metoda klase K.
4. Klasa N je dostupna za treću stranu.

Slučaj 1.
U ovom slučaju smatra se da je svaki objekat klase K, u oznaci obK, ekskluzivni vlasnik objekta $obK.p$, tj. p predstavlja *rep field* [9] [84]. Objekat p je jako inkapsuliran poštujući Demetrin zakon [13] [62] [63] ili je pak zaštićen *packing* mehanizmom [9] [61].

S obzirom na to da invarijanta u klasi L mora biti poštovana, a da invarijanta u klasi K ne može biti narušena od treće strane (jer treća strana ne postoji), invarijanta u klasi K može pooštriti invarijantu u klasi L. Neka je

$$\bar{\phi} = (f_1, \ldots, f_m, p.g_1, \ldots, p.g_n, N.h_1, \ldots, N.h_t),$$

$$p.\bar{g} = (p.g_1, \ldots, p.g_n).$$

$$N.\bar{h} = (N.h_1, \ldots, N.h_t).$$

Zbog napred navedenog važi

$$IS_K(\bar{\phi} \backslash p.\bar{g}) \Rightarrow IS_L(p.\bar{g}).$$

S obzirom na teoremu 4.5.1 dobija se $IS_K(p.\bar{g}) \Rightarrow IS_L(p.\bar{g})$, odnosno na osnovu teorema 4.5.2 dobijamo:

$$IS_K(\bar{\phi}) \Rightarrow IS_L(p.\bar{g}).$$

Do sada prikazane relacije ne uzimaju u obzir (realnu) pretpostavku da je *rep* komponenta p inkapsulirana u klasu. Inkapsuliranje ove komponente, u stvari, ima za posledicu to da pozivi metoda klase L kojoj pripada komponenta nisu proizvoljni, nego su locirani unutar metoda klase K. Dakle, postoji podskup M_{KL} skupa M_K metoda klase K koje koriste metode klase L i koji ne mora da se poklopi sa skupom M_K.

Komponenta p definisana u klasi K može i ne mora biti konstruisana unutar klase, odnosno može biti prosleđena i kao argument neke metode. Da bismo dobili što opštiju formulu uvešćemo predikat $Q_p(p.\bar{g})$ koji opisuje sva početna stanja (i nulto, po potrebi) u kojima se objekat p može naći kada je u ulozi komponente klase K. Najzad, neka je

$$\hat{JS}_{KL} = \bigvee_{m \in M_{KL}} \hat{sdp}(m, \Gamma(m)).$$

Sada možemo neznatno modifikovati Algoritam 4.3.23 tako da započne od predikata $Q_p(p.\bar{g})$.

Algoritam 4.5.3
$Z_0 \equiv W_0 \equiv Q_p(p.\bar{g})$
$- - - - - - - - - - - - - -$
$REPEAT$
$W_{i+1} \equiv W_i \otimes \hat{JS}_{KL}(p.G)$
$Z_{i+1} \equiv Z_i \vee W_{i+1}$
$i = 0, 1, \ldots$
$UNTIL\ Z_{i+1} \Leftrightarrow Z_i$
$- - - - - - - - - - - - - -$
$fix = i$
$\boxed{IS_K L \equiv Z_i}$

Sada predikat $IS_{KL}(p.\bar{g})$ obuhvata sva stanja u kojima se komponenta p može naći, a s obzirom na prostor stanja određen vektorom $p.\bar{g}$ i, shodno do sada rečenom, mora da važi

$$IS_K(\bar{\phi}) \Rightarrow IS_{KL}(p.\bar{g}).$$

Slučaj 2.
Ako je komponenta dostupna za treću stranu (uključujući i rekurzivni pristup), tada invarijanta u klasi K ne sme da pooštri invarijantu u klasi L (videti [9]). To znači da mora važiti $IS_K(\bar{\phi}\backslash p.\bar{g}) \Leftrightarrow IS_L(p.\bar{g})$ tj.

$$IS_K(p.\bar{g}) \Leftrightarrow IS_L(p.\bar{g})$$

Drugim rečima, važi $IS_K(\bar{\phi}) \Rightarrow IS_L(p.\bar{g})$, ali ne postoji predikat $Q(p.\bar{g})$ različit od $IS_L(p.\bar{g})$ takav da bude

$$[IS_K(\bar{\phi}) \Rightarrow Q(p.\bar{g})] \wedge [Q(p.\bar{g}) \Rightarrow IS_L(p.\bar{g})].$$

Ovde treba zapaziti da gornji uslov obezbeđuje sigurnost, ali nije uvek ostvarljiv. Naime, komponenta p može biti inkapsulirana u vlasniku K što znači da metode u klasi K mogu aktivirati samo neke od metoda u klasi L. Samim tim, pošto nema mogućnosti izbora, nema ni načina da se obezbedi uslov $IS_K(\bar{\phi}\backslash p.\bar{g}) \Leftrightarrow IS_L(p.\bar{g})$.

U posebnom slučaju, kada su svi vlasnici komponente p unapred poznati, rešenje nalazimo u postizanju tzv. ekvilibrijuma svih vlasnika [51] [54] [56]. Neka su klase $K_1, \ldots, K_r$ vlasnici istog objekta p iz klase L. Tada su vlasnici u ekvilibrijumu ako važi

$$IS_{K_i}(\bar{\phi}_i) \Rightarrow IS_{K_1}(p.\bar{g}) \wedge \cdots \wedge IS_{K_r}(p.\bar{g}),$$

$$IS_{K_i}(p.\bar{g}) \Rightarrow IS_L(p.\bar{g}), i = 1, \ldots, r.$$

Slično slučaju 1, a pod pretpostavkom da je komponenta p inkapsulirana u svim vlasnicima, možemo stanje ekvilibrijuma učiniti nešto preciznijim. Neka je $M_{\{K\}L}$ skup svih metoda klasa-vlasnica $K_1, \ldots, K_r$ koje pozivaju metode klase L kojoj pripada komponenta p. Neka je $IS_{\{K\}L}$ predikat dobijen iz $M_{\{K\}L}$ Algoritmom 4.5.3 na osnovu predikata

$$\hat{JS}_{\{K\}L} = \bigvee_{m \in M_{\{K\}L}} \hat{sdp}(m, \Gamma(m)).$$

Predikat $IS_{\{K\}L}$ u potpunosti definiše skup stanja u kojima se može naći objekat p kao zajednička komponenta svih navedenih vlasnika. Shodno tome, ekvilibrijum se postiže kada je ispunjeno

$$IS_{K_i}(\bar{\phi}_i) \Rightarrow IS_{\{K\}L}(p.\bar{g}), i = 1, \ldots, r,$$

pri čemu $IS_{\{K\}L}(p.\bar{g}) \Rightarrow IS_L(p.\bar{g})$.

Treba, ipak, imati na umu da se uvođenjem zavisnosti invarijante u klasi od invarijante deljene komponente narušava modularnost i da vredi razmisliti o tome da se invarijanta IS_L radije uključi u invarijantu programa [81].

Slučaj 3.
U ovom slučaju nedostupnost klase N iz trećih klasa znači da je klasa N u celosti inkapsulirana u klasu K, odnosno da je zaštićena od pristupa treće strane mehanizmom expose [61]. Tipičan slučaj je unutrašnja klasa koja je u celosti inkapsulirana u spoljašnju. Postupkom koji je u potpunosti analogan slučaju 1 dobijamo

$$IS_K(\bar{\phi}\backslash N.\bar{h}) \Rightarrow IS_N(N.\bar{h}),$$

gde je $N.\bar{h} = (N.h_1, \ldots, N.h_t)$. Isto tako,

$$IS_K(N.\bar{h}) \Rightarrow IS_N(N.\bar{h}),$$

$$IS_K(\bar{\phi}) \Rightarrow IS_N(N.\bar{h}).$$

Ako je M_{KN} skup metoda klase K koje referenciraju statička polja klase N, i ako je

$$\hat{JS}_{KN} = \bigvee_{m \in M_{KN}} \hat{sdp}(m, \Gamma(m)),$$

postupkom analognim slučaju 1 dobija se

$$IS_K(\bar{\phi}) \Rightarrow IS_{KN}(N.\bar{h}),$$

gde se statička invarijanta $IS_{KN}(N.\bar{h})$ dobija iz $\hat{JS}_{KN}$ Algoritmom 4.5.3.

Slučaj 4.
Klasa N je dostupna za treću klasu. Ovaj slučaj može biti od interesa kada je sistem klasa koji obuhvata K i N unapred poznat, tj. kada su unapred poznate sve klase koje su vlasnici klase N. Primer za to jeste slučaj kada je klasa N nalazi inkapsulirana u neki modul (jedinicu u Pascalu, Javin paket ili C++ biblioteku), tj. kada je dostupna samo klasama iz istog modula. Tada postoji potpuna analogija sa deljenom komponentom iz slučaja 2, te slede praktično istovetni zaključci. Prvo, za svaku klasu K koja je vlasnica klase N važi $IS_K(\bar{\phi}\backslash N.\bar{h}) \Leftrightarrow IS_N(N.\bar{h})$ odakle

$$IS_K(N.\bar{h}) \Leftrightarrow IS_N(N.\bar{h}).$$

Neka skup vlasnika klase N čini skup $\{K_1, \ldots, K_r\}$. Stanje ekvilibrijuma vlasnika u odnosu na statička polja $N.\bar{h}$ određeno je implikacijama

$$IS_{K_i}(\bar{\phi}_i) \Rightarrow IS_{K_1}(N.\bar{h}) \wedge \cdots \wedge IS_{K_r}(N.\bar{h}),$$

$$IS_{K_i}(N.\bar{h}) \Rightarrow IS_N(N.\bar{h}), i = 1, \ldots, r.$$

Pod pretpostavkom da se statičkim poljima $N.\bar{h}$ pristupa isključivo putem (statičkih i nestatičkih) metoda klase N, možemo i u ovom slučaju precizirati uslov ekvilibrijuma, slično slučaju 2. Neka je $M_{\{K\}N}$ skup svih metoda klasa $K_1, \ldots, K_r$ koje referenciraju

statička polja $N.\bar{h}$. Neka je $IS_{\{K\}N}$ predikat dobijen iz $M_{\{K\}N}$ Algoritmom 4.5.3 na osnovu predikata

$$\hat{JS}_{\{K\}N} = \bigvee_{m \in M_{\{K\}N}} \hat{sdp}(m, \Gamma(m)).$$

Kao i u slučaju 2, predikat $IS_{\{K\}N}$ u potpunosti definiše sve vrednosti koje mogu dobiti polja iz $N.\bar{h}$ pod dejstvom klasa-vlasnica $K_1, \ldots, K_r$. Dakle, ekvilibrijum se postiže kada važi

$$IS_{K_i}(\bar{\phi}_i) \Rightarrow IS_{\{K\}N}(N.\bar{h}), i = 1, \ldots, r,$$

pri čemu $IS_{\{K\}N}(N.\bar{h}) \Rightarrow IS_N(N.\bar{h})$.

Rezime.
Zapažamo da se svi navedeni slučajevi mogu sažeti u jedan jedini. Neka je data klasa Y i neka je $\{X_1, \ldots, X_r\}$ skup svih klasa koje koriste usluge klase Y. Neka je, dalje, $\bar{a} = (a_1, \ldots, a_n)$ uređena n-torka čiji je svaki element ili polje objekta p koji dele klase $X_1, \ldots, X_r$ ili statičko polje klase Y. Neka je $\bar{\phi}_i$ torka polja klase $X_i, i = 1, \ldots, r$ koja opisuje njen prostor stanja, pri čemu $\bar{\phi}_i$ obuhvata sve elemente torke $\bar{a}$. Neka je $M_{\{X\}Y}$ skup svih metoda klasa-vlasnica $X_1, \ldots, X_r$ koje pozivaju metode klase Y i neka je $IS_{\{X\}Y}$ predikat dobijen iz $M_{\{X\}Y}$ Algoritmom 4.5.3, na osnovu predikata

$$\hat{JS}_{\{X\}Y} = \bigvee_{m \in M_{\{X\}Y}} \hat{sdp}(m, \Gamma(m)).$$

Tada mora da važi

$$IS_{X_i}(\bar{\phi}_i) \Rightarrow IS_{\{X\}Y}(\bar{a}), i = 1, \ldots, r,$$

pri čemu $IS_{\{X\}Y}(\bar{a}) \Rightarrow IS_Y(\bar{a})$.

Ownership
U određenim slučajevima komponente imaju pristup vlasniku (vlasnicima). Ovde je situacija ista kao u slučaju 2, jer se može smatrati da je vlasnik uvek dostupan trećoj strani. Neka vlasnik objekta klase K pripada klasi Z i neka je dostupan kroz polje *owner* klase K. Neka su polja $owner.i_1, \ldots, owner.i_j$ uključena u postuslove metoda klase K. Tada mora da važi

$$IS_K(\bar{\phi} \backslash owner.\bar{i}) \Leftrightarrow IS_Z(owner.\bar{i}),$$

gde je sada

$$owner.\bar{i} = (owner.i_1, \ldots, owner.i_j),$$

$$F = (f_1, \ldots, f_m, p.g_1, \ldots, p.g_n, N.h_1, \ldots, N.h_t, owner.i_1, \ldots, owner.i_j).$$

Analognim rezonovanjem kao u slučaju 2 dolazimo do formule

$$IS_K(owner.\bar{i}) \Leftrightarrow IS_Z(owner.\bar{i})$$

iz koje sledi $IS_K(\bar{\phi}) \Rightarrow IS_Z(owner.\bar{i})$. Inače, ovaj slučaj bitno se razlikuje od prethodna četiri zahvaljujući činjenici da je potpuno nerealno pretpostaviti da je vlasnik nedostupan.

Neka je $IS'_K(\bar{\phi})$ invarijanta u klasi K dobijena Algoritmom **A1**. Ako $\bar{\phi}$ obuhvata sva polja iz torke $owner.\bar{i}$ na osnovu upravo izvedenog mora važiti $IS'_K(\bar{\phi}) \Rightarrow IS_Z(owner.\bar{i})$, a da bi se obezbedila još i ekvivalencija $IS_K(owner.\bar{i}) \Leftrightarrow IS_Z(owner.\bar{i})$ za invarijantu IS_K treba usvojiti

$$IS_K(\bar{\phi}) = IS'_K(\bar{\phi}) \wedge IS_Z(owner.\bar{i}).$$

4.5.1 Primer

Posmatrajmo sledeće klase:

```
public class K {
     private int x;
     public K() { x=0; }
     public void setX(int xx) { x=xx; }
     public int getX(int xx) { return x; }
}

public class L1 {
     private K p;
     public L1(K pp) { p=pp; p.setX(10); }
     public void incX() { int xx=p.getX(); p.setX(xx+1) }
}

public class L2 {
     private K p;
     public L2(K pp) { p=pp; p.setX(5); }
     public void incX() { int xx=p.getX(); p.setX(xx+1) }
}
```

Usvojićemo da je

$$\bar{\phi} = (k, x),$$

$$p.\bar{g} = (x).$$

Primenom algoritma 4.3.23 iz klase K dobijamo:
$IS_K \equiv O(p.\bar{g}) \vee (\neg O(p.\bar{g}) \wedge (x \in D_{int}))$.

Dalje, očigledno važi:
$\hat{JS}_{KL_1} \equiv (O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (x' = 10)) \vee (O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (x \in D_{int}) \wedge (x' = x + 1))$,
$\hat{JS}_{KL_2} \equiv (O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (x' = 5)) \vee (O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (x \in D_{int}) \wedge (x' = x + 1))$.

Uzimamo da je $Q_p \equiv (x = 10)$ i pomoću predikata $\hat{JS}_{KL1}$ iz algoritma 4.5.3 dobijamo:
$IS_{KL_1} \equiv O(\bar{\phi}) \vee (\neg O(\bar{\phi}) \wedge (x \geq 10))$.

Uzimamo da je $Q_p \equiv (x = 5)$ i pomoću predikata $\hat{JS}_{KL_2}$ iz algoritma 4.5.3 dobijamo:
$IS_{KL_2} \equiv O(\bar{\phi}) \vee (\neg O(\bar{\phi}) \wedge (x \geq 5))$.

Predikat
$IS_{\{K\}L} \equiv IS_{KL_1} \wedge IS_{KL_2}$
$IS_{\{K\}L} \equiv [O(\bar{\phi}) \vee (\neg O(\bar{\phi}) \wedge (x \geq 10))] \wedge [O(\bar{\phi}) \vee (\neg O(\bar{\phi}) \wedge (x \geq 5))]$
$IS_{\{K\}L} \equiv O(\bar{\phi}) \vee (\neg O(\bar{\phi}) \wedge (x \geq 10))$

opisuje sva stanja u kojima se komponenta p može naći kao zajednička komponenta.

4.6 Natklasa i potklasa

Na početku ovog poglavlja uvešćemo sledeće pretpostavke:

1. Nasleđivanje je jednostruko.
2. Nema redefinisanih polja.

Samo po sebi, nasleđivanje - onakvo kako je realizovano u programskim jezicima - ne postavlja nikakva ograničenja u pogledu stvarnog odnosa klasa koje vezujemo nasleđivanjem. Korišćenjem private nasleđivanja u C++ [29] [49] ili kombinovanjem nasleđivanja i implementacije interfejsa u Javi [98] [30] moguće je svaku klasu tehnički povezati nasleđivanjem sa svakom drugom klasom.

Zbog toga je neophodno tražiti ograničenja na drugoj strani. Rešenje nalazimo u Zakonu supstitucije [64], odnosno u *subtyping*-u [10] [46] gde se insistira na tom da polimorfna zamena pretka potomkom ne sme izazvati nikakvu promenu u klijentskom softveru. Takođe, Mejer [77] navodi da redefinisana metoda može oslabiti preduslov originala, kao i pojačati postuslov - i to je sve.

Neka klasa L nasleđuje klasu K. Neka je IS_K stroga invarijanta u klasi K nad poljima $\bar{f} = (f_1, \ldots, f_m)$. Neka je prostor stanja klase L definisan nad poljima $\bar{g} = (g_1, \ldots, g_n)$ koja sadrže sva polja iz $\bar{f}$. Dokažimo da tada mora važiti

$$IS_L(\bar{g} \backslash \bar{f}) \Rightarrow IS_K(\bar{f}).$$

Neka je s stanje koje zadovoljava $IS_L(\bar{g} \backslash \bar{f})$ i ne zadovoljava $IS_K(\bar{f})$. Ako se objekat klase L, na primer obL, nađe u tom stanju i izvrši polimorfna dodela

$$obK = obL$$

gde je obK objekat klase K, tada će se obK naći u invalidnom stanju. Dalje, na osnovu teoreme 4.5.2 važi

$$IS_L(\bar{g}) \Rightarrow IS_L(\bar{g} \backslash \bar{f}),$$

odakle

$$IS_L(\bar{g}) \Rightarrow IS_K(\bar{f}).$$

Kao specijalan slučaj $\bar{g} = \bar{f}$, sledi poznata relacija [55]:

$$IS_L(\bar{f}) \Rightarrow IS_K(\bar{f}).$$

4.6.1 Dinamička analiza

Poslednja implikacija ne garantuje da će se klase K i L ponašati korektno, jer ne uzima u obzir inkluzioni polimorfizam u dinamičkim uslovima. Da bismo analizirali ponašanje metoda, podelićemo ih u tri grupe:

- redefinisane metode što čine skup M_L^1,
- preuzete (nasleđene) metode, skup M_L^2,
- dodate metode, skup M_L^3,
- konstruktori iz skupa C_L.

Neka je m_L redefinisana metoda, a m_K njena verzija iz natklase K. Posmatrajmo sledeće dve klase:

```
public class K {
     private int x;
     public K() {
          x= 1;
     }
     public void incX() {
          x++;
     }           public int getX() {
          return x;
     }
}


public class L extends K {
     public void incX() {
          x+= 2;
     }
}
```

Lako se uočava da važi

$$IS_K = (x \geq 1),$$

$$IS_L = (x \geq 1) \wedge (x \; je \; neparno) = IS_K \wedge (x \; je \; neparno),$$

što znači da važi $IS_L \Rightarrow IS_K$ i naizgled je sve u redu. Nažalost, to nije tačno. Neka je obL objekat klase L, a obK objekat klase K. Posmatrajmo neku rutinu f proizvoljnog tipa T koja za parametar ima objekat klase K, i takođe lokalni objekat loc iste klase:

```
T f(K ob) { K loc = ob; ...  }
```

U opštem slučaju, a u zavisnosti od toga da li je rutina pozvana sa $f(obK)$ ili sa $f(obL)$, lokalni objekat loc se ne ponaša jednako jer se verzije $incX$ ponašaju različito. Ovo, dalje, izaziva promenu ponašanja cele rutine, što je u direktnoj suprotnosti sa Zakonom supstitucije. Ako malo više obratimo pažnju na to šta se dešava u ovom slučaju, zapazićemo da redefinisana verzija metode $incX()$ menja ponašanje ne zato što uvodi eventualno nova završna stanja (ne uvodi ih) nego uvodi nove prelaze.

Redefinisane metode

Pođimo od glavnog problema – redefinisanih metoda. Neka je, ponovo, klasa L izvedena iz klase K i neka je m_L redefinisana verzija metode m_K. Neka su obK i obL instance redom klasa K i L. Da bi se obezbedilo korektno ponašanje redefinisane metode m_L potrebno je voditi računa o dve, međusobno povezane, pojedinosti:

- polimorfizmu operacije pridruživanja, gde se mora garantovati da, neposredno posle pridruživanja potomka obL pretku obK, stanje obK bude validno (validno u klasi K),
- dinamičkom povezivanju, gde se mora garantovati da će se metoda m_L ponašati korektno u odnosu na objekat obK preko kojeg je aktivirana posle polimorfnog pridruživanja.

Da bi se ovi uslovi postigli potrebno je poštovati tradicionalno pravilo (navedeno npr. u [77]), naravno primenjeno na preduslov $\Gamma(m)$ i najuži dinamički postuslov $\hat{sdp}$:

$$\Gamma(m_K) \Rightarrow \Gamma(m_L),$$

$$\hat{sdp}(m_L, \Gamma(m_L)) \Rightarrow \hat{sdp}(m_K, \Gamma(m_K)).$$

Razmotrimo ove uslove na nivou prostora stanja klasa. Prvo, poćićemo od notorne činjenice da izvedena klasa preuzima polja od bazne klase i da, shodno tome, ima smisla govoriti o deljenju prostora stanja. Neka je U_K jedan prostor stanja klase K i neka je on definisan i za klasu L. Neka je U_L prostor stanja klase L takav da postoji homomorfizam

$$h_{LK} : U_L \to U_K,$$

takav da ako u U_L postoji prelaz (s, s'), tada u U_K postoji prelaz $(h_{LK}(s), h_{LK}(s'))$. Inače, prostor stanja U_K opisan poljima $(f_1, \ldots, f_m)$ koja postoje u obe klase i prostor stanja U_L opisan poljima $(f_1, \ldots, f_m, g_1, \ldots, g_n)$ zadovoljavaju navedeni uslov uz homomorfizam oblika

$$h_{LK}(f_1, \ldots, f_m, g_1, \ldots, g_n) = (f_1, \ldots, f_m).$$

Gore navedeno pravilo sada dobija nešto precizniji oblik

(RM_1) $\quad \Gamma(m_K)[U_K] \Rightarrow \Gamma(m_L)[U_L]$,
(RM_2) $\quad \hat{sdp}(m_L, \Gamma(m_L))[U_L] \Rightarrow \hat{sdp}(m_K, \Gamma(m_K))[U_K]$,

gde sufiksi $[U_K]$ i $[U_L]$ označavaju prostor stanja u odnosu na koji se zadaju predikati i važi $m_L \in M_L^1$.

Neka je $s \in U_L$ validno stanje klase L takvo da je $h_{LK}(s)$ stanje koje je validno u prostoru stanja U_K. Neka je (s, s') prelaz ostvaren metodom m_L. Implikacija (RM_2) garantuje da se među validnim prelazima verzije m_K mora nalaziti prelaz $(h_{LK}(s), h_{LK}(s'))$, iz čega sledi da je i $h_{LK}(s')$ validno stanje u klasi K. Drugim rečima,

$$\{IS_K \wedge \Gamma(m_L)\} m_L() \{IS_K\},$$

tj. iz implikacije (RM_2) sledi da redefinisana metoda čuva invarijantu bazne klase.

Implikacija (RM_1) garantuje da će posle polimorfnog pridruživanja, redefinisana verzija sigurno terminirati ako terminira i osnovna. Dakle, u ma kojem stanju obK iz kojeg m_K terminira, posle pridruživanja npr. $obK = obL$, terminiraće i m_L.

Neka metoda m_L uzrokuje prelaz (s, s') gde $s, s' \in U_L$. Prema definiciji homomorfizma h_{KL}, u klasi K postoji prelaz $(h_{KL}(s), h_{KL}(s'))$. U uslovima inkluzionog polimorfizma treba ispitati dva slučaja:

Slučaj 1:
Neposredno pre polimorfnog pridruživanja oblika $obK = obL$ izvršena je operacija $obL.m_L()$ te se, dakle, objekat obL nalazi u stanju s' koje je validno u prostoru stanja U_L. Tada stanje $h_{LK}(s')$ mora biti validno u prostoru stanja U_K, da bi objekat obK posle pridruživanja bio u validnom stanju. Pošto redefinisana metoda m_L čuva invarijantu u klasi K, sve što je potrebno jeste da m_L otpočne u stanju koje poštuje IS_K. Uočimo, da ovaj uslov nije sâm po sebi zadovoljen jer zavisi od primene drugih metoda pre operacije $obK.m_L()$. Na primer, ako konstruktor klase L ostavi objekat obL u stanju t takvom da $h_{LK}(t)$ nije validno stanje u U_K, tada nema garancije da će $h_{LK}(t')$ biti validno u U_K.

Slučaj 2:
Posle polimorfnog pridruživanja $obK = obL$ izvršava se operacija $obK.m_L()$. Ako je obK pre primene m_L bio u stanju koje je validno u U_K, tada, zbog implikacije (2), on ostaje u stanju koje je validno u U_K.

Preuzete metode

Preuzete (nasleđene u smislu Amadi-Kardeli [2]) metode iz skupa M_L^2 mogu promeniti ponašanje u klasi L i to zbog dinamičkog povezivanja, u slučaju da koriste neku od metoda iz skupa M_L^1. Dokaz za ovo je sledeća jednostavna situacija. Neka je $n()$ metoda koja se preuzima u klasu L i neka je m metoda koja se redefiniše, dakle metoda sa verzijama m_K i m_L. Neka metoda n ima sledeći oblik:

```
tip n() { m(); }
```

Očigledno je da, usled dinamičkog povezivanja (zamene m_K sa m_L) metoda $n()$ menja ponašanje, kao da je redefinisana. Shodno tome, i od preuzetih metoda očekuje se da ispune uslove analogne redefinisanim:

(PM_1) $\quad \Gamma(m)[U_K] \Rightarrow \Gamma(m)[U_L]$,
(PM_2) $\quad \hat{sdp}(m, \Gamma(m))[U_L] \Rightarrow \hat{sdp}(m, \Gamma(m))[U_K]$,

za $m \in M_L^2$.

Dodate metode

Dodate metode iz M_L^3 mogu, naravno, uvoditi nove prelaze jer nisu direktno dostupne metodama klase K, te tako ne postoje problemi sa dinamičkim povezivanjem. Međutim, time nije rešen drugi problem: problem polimorfnog pridruživanja. Neka je $m() \in M_L^3$ dodata metoda i neka je neposredno pre polimorfnog pridruživanja $obK = obL$ izvršena operacija $obL.m()$. Da bi objekat obK posle pridruživanja bio u validnom stanju metoda m mora poštovati strogu invarijantu u klasi K, tj.

(DM_1) $\quad \{IS_K \wedge \Gamma(m)\}m()\{IS_K\}$,

za $m \in M_L^3$.

Konstruktori

U svrhu poštovanja inkluzionog polimorfizma, objekat klase L odmah posle kreiranja mora biti u stanju s takvom da je $h_{LK}(s)$ validno stanje u klasi K. Dakle, za sve konstruktore mora važiti

(K_1) $\quad \hat{sdp}(c, O) \Rightarrow IS_K$,

gde $c \in C_L$.

Algoritam za izračunavanje stroge invarijante potklase
Neka je klasa L izvedena iz klase K uz poštovanje svih prethodno navedenih pravila. Uvodimo predikat:

$$\hat{JS}_L \Leftrightarrow \bigvee_{m\in M_L^1} s\hat{d}p(m,\Gamma(m)) \vee \bigvee_{m\in M_L^2} s\hat{d}p(m,\Gamma(m)) \vee \bigvee_{m\in M_L^3} s\hat{d}p(m,\Gamma(m)).$$

Predikat IS_L^0 opisuje sva inicijalna stanja potklase. Sada se može formulisati algoritam za izračunavanje stroge invarijante potklase:

Algoritam 4.6.1
$Z_0 \equiv W_0 \equiv IS_L^0$
$- - - - - - - - - - - - -$
$REPEAT$
$W_{i+1} \equiv W_i \otimes \hat{JS}_L$
$Z_{i+1} \equiv Z_i \vee W_{i+1}$
$i = 0, 1, \ldots$
$UNTIL\ Z_{i+1} \Leftrightarrow Z_i$
$- - - - - - - - - - - - -$
$fix = i$
$\boxed{IS_L \equiv Z_i}$

Primenom teoreme 4.3.15 se može uprostiti izračunavanje supstitucije $A \otimes \hat{JS}_L$ tako što dobijamo:

$$A\otimes \hat{JS}_L \Leftrightarrow A\otimes[\bigvee_{m\in M_L^1} s\hat{d}p(m,\Gamma(m))]\vee A\otimes[\bigvee_{m\in M_L^2} s\hat{d}p(m,\Gamma(m))]\vee A\otimes[\bigvee_{m\in M_L^3} s\hat{d}p(m,\Gamma(m))]$$

$$A \otimes \hat{JS}_L \Leftrightarrow \bigvee_{m\in M_L^1} A \otimes s\hat{d}p(m,\Gamma(m)) \vee \bigvee_{m\in M_L^2} A \otimes s\hat{d}p(m,\Gamma(m)) \vee \bigvee_{m\in M_L^3} A \otimes s\hat{d}p(m,\Gamma(m)).$$

Matematičkom indukcijom ćemo pokazati da stroga invarijanta potklase L dobijena algoritmom 4.6.1 povlači strogu invarijantu njene natklase K:

- S obzirom da za konstruktore klase L važi pravilo K_1, sledi da u svim stanjima koja opisuje predikat Z_0 važi IS_K.
- Pretpostavimo da u svim stanjima koja opisuje predikat Z_n važi IS_K, gde je predikat Z_n dobijen algoritmom 4.6.1 posle n-prelaza.

- Na osnovu pravila (RM_1), (PM_1) i (DM_1) sledi da će sve metode iz skupova M_L^1, M_L^2 i M_L^3 terminirati, a na osnovu pravila (RM_2), (PM_2) i (DM_2) sledi da će sve metode iz skupova M_L^1, M_L^2 i M_L^3 proizvesti stanja u kojima važi IS_K. Dakle, sledi da će u svim stanjima koja opisuje predikat Z_{n+1} važiti IS_K.

Dakle, zaključujemo:

a.) Zbog polimorfizma moraju da važe pravila (RM_1), (RM_2), (PM_1), (PM_2), (DM_1) i (K_1).

b.) Pošto pomenuta pravila važe, onda mora da važi $IS_L \Rightarrow IS_K$ ili drugim rečima, ako je nasleđivanje izvedeno korektno, tada nužno važi $IS_L \Rightarrow IS_K$.

4.6.2 Primer

Posmatrajmo sledeće klase:

```
public class K {
     protected int x;
     public K() { x=1; }
     public void m() { x++; }
}

public class L extends K {
     protected int y;
     public L() { super(); y=1; }
     public void n() { y++; }
     public void m() { x+=2; }
}
```

Usvajamo da je:

$$\bar{g} = (x),$$
$$\bar{f} = (x, y).$$

Primenom algoritma 4.3.23 dobijamo:

$$IS_K \equiv O(\bar{g}) \vee [\neg O(\bar{g}) \wedge (x \geq 1)].$$

Iz klase L dobijamo:

$\hat{sdp}(L, \Gamma(L)) \equiv O(\bar{f}) \wedge \neg O(\bar{f}) \wedge (x = 1) \wedge (y = 1)$,
$IS_L^0(\bar{f}) \equiv (x = 1) \wedge (y = 1)$,

$\hat{sdp}(n, \Gamma(n)) \equiv \neg O(\bar{f}) \wedge \neg O(\bar{f}) \wedge (x' = x) \wedge (y' = y + 1)$,
$\hat{sdp}(m, \Gamma(m)) \equiv \neg O(\bar{f}) \wedge \neg O(\bar{f}) \wedge (x' = x + 2) \wedge (y' = y)$,

odakle je:
$\hat{JS}_L \equiv \hat{sdp}(L, \Gamma(L)) \vee \hat{sdp}(n, \Gamma(n)) \vee \hat{sdp}(m, \Gamma(m))$.

Usvajamo da je $Z_0 \equiv W_0 \equiv IS_L^0$, primenjujemo algoritam 4.6.1 i dobijamo:

$$IS_L \equiv O(\bar{f}) \vee [\neg O(\bar{f}) \wedge (x \geq 1) \wedge (x \textit{ je neparanbroj}) \wedge (y \geq 1)].$$

Očigledno, važi $IS_L \Rightarrow IS_K$.

4.6.3 Veza dinamičkih invarijanata natklase i potklase

Uočimo odmah najvažnije – s obzirom na to da konstruktori klase L uspostavljaju invarijantu u klasi K i da sve ostale metode klase L čuvaju invarijantu u klasi K sledi

$$IS_L \Rightarrow IS_K.$$

Zanimljivija je situacija sa dinamičkom invarijantom. **Prvo**, ne postoji garancija da važi $\hat{JS}_L \Rightarrow \hat{JS}_K$. Dokaz za to su sledeće klase K i L:

```
public class K {
     protected int x;
     public K() {
          x=1;
     }
     public void m() {
          x++;
     }
}

public class L extends K {
     protected int y;
     public L() {
          y=1;
     }
     public void n() {
          y++;
     }
}
```

Ipak veza između dinamičkog ponašanja klasa K i L postoji. Neka je M skup svih metoda klase L osim dodatih, tj.

$$M = M_L^1 \cup M_L^2 \cup C_L$$

Neka je $\hat{JS}_{LK}$ predikat oblika

$$\hat{JS}_{LK} = \bigvee_{m \in M} \hat{sdp}(m, \Gamma(m)).$$

Na osnovu izvedenih relacija za konstruktore, redefinisane i nasleđene metode lako se pokazuje da važi

$$\hat{JS}_{LK} \Rightarrow \hat{JS}_K.$$

Drugo, razmotrićemo ideju da se invarijanta potklase računa bez uključivanja nasleđenih metoda i pokazati da je, u opštem slučaju, to neizvodljivo. Naime, neka je M skup svih metoda klase L osim nasleđenih. Neka je $\hat{J}_L$ predikat oblika

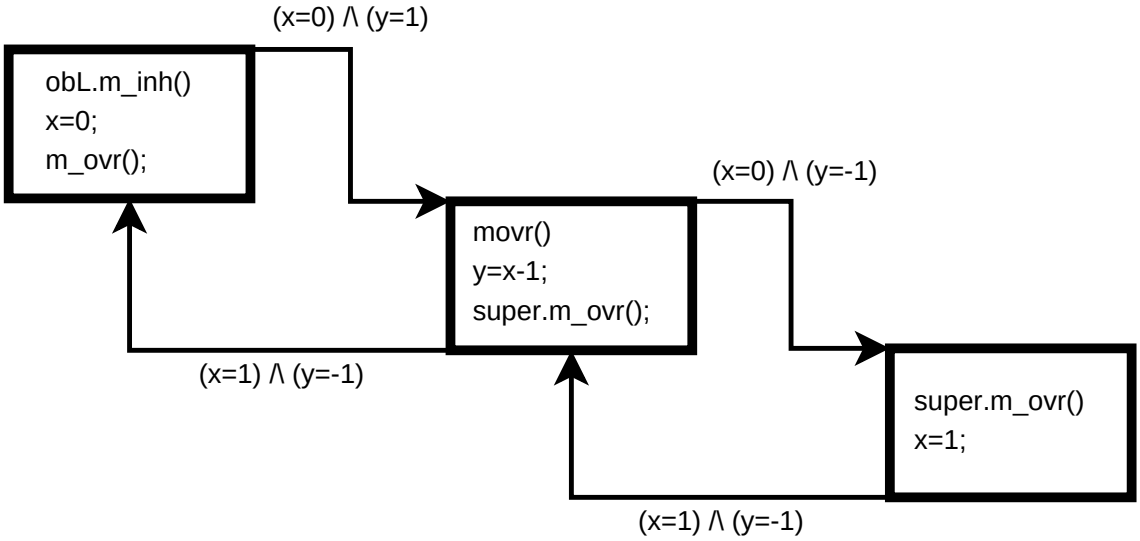
$$\hat{J}_L = \bigvee_{m \in M} \hat{sdp}(m, \Gamma(m)).$$

Neka je, dalje, I_L stroga invarijanta klase L dobijena na osnovu $\hat{J}_L$ (dakle, bez uključivanja nasleđenih metoda). Konačno, neka je IS_K originalni oblik stroge invarijante natklase K. U opštem slučaju ne važi

$$IS_L \Leftrightarrow IS_K \wedge I_L,$$

gde bi $\hat{IS}_L$ bila stroga invarijanta klase L (u prostoru stanja U_L). Dokazaćemo ovo kontraprimerom. Posmatrajmo sledeći par klasa, K i L gde je L potklasa:

```
public class K {
     protected int x;
     public K() {
          x=1;
     }
     public void m_inh() {
          x=0;
     m_ovr();
     }
     public void m_ovr() {
          x=1;
     }
}
```



Slika 4.1: Primer 4.6.2.

```
public class L extends K {
     protected int y;
     public L() {
          y=1;
     }
     public void m_ovr { //overriden method
          y=x-1;
          super.m_ovr();
     }
}
```

Lako se uočava da važi

$$IS_K = (x = 0)$$

$$I_L = (x = 1) \wedge (y \in \{0, 1\})$$

što bi značilo da je

$$IS_K \wedge I_L = (x = 1) \wedge (y \in \{0, 1\}).$$

ali to nije stroga invarijanta klase L. Posmatrajmo sledeći par naredbi:

```
L obL = new L(); // sada je obL.x = obL.y = 1
obL.m_inh();
```

kojima se kreira objekat i poziva nasleđena metoda (koja poziva redefinisanu metodu *m_ovr*). Ono što se dešava prikazano je na slici 4.1.

Kako vidimo, na kraju gornjeg segmenta biće $(x = 1) \wedge (y = -1)$, što je validno stanje, ali ne zadovoljava predikat $IS_K \wedge I_L$.

4.7 Varijabilna struktura objekta (pokazivači/reference)

Normalno je da se, u toku izvršenja programa, dati objekat povezuje (pokazivačima ili referencama) sa drugim objektima kao posledica postojanja različitih veza: asocijacije, agregacije, pa i kompozicije [97] [80]. Rezultat toga je da se struktura objekta može menjati u toku izvršenja programa, a neki od takvih slučajeva proučavani su u Odeljku 4.5, kao i u [4] [100] [79].

Ono što nas ovde interesuje nisu objekti što se u toku izvršenja programa povezuju sa drugim objektima i razvezuju od njih. Poseban i sasvim drukčiji slučaj predstavljaju objekti koji nisu povezani sa drugim objektima, nego kod kojih reference (pokazivači) povezuju delove istog objekta u logičku celinu. Posmatrajmo objekat klase *List* koji predstavlja jednostruko spregnutu listu (slična situacija je i kod drugih kontejnerskih klasa). U osnovnom memorijskom prostoru objekta ove klase nalazi se samo referenca na prvi element (možda i ništa drugo), dok se lista, u stvari, nalazi na hipu, kao spregnuta struktura. Sam objekat nije ništa više do zaglavlje (header) liste i dobija smisao tek kada se logički objedini u celinu sa preostalim delom. Može se, dakle, uočiti da ovakvi i slični objekti u logičkom smislu menjaju strukturu u toku izvršenja programa i da je ta struktura rekurzivna, što je uočio Mejer govoreći o "run-time object structure", [79].

Objekti sa varijabilnom strukturom ne mogu se kvalitetno analizirati samo na osnovu sadržaja polja. Kao što smo rekli, objekat klase *List* sadrži samo referencu na prvi element, pa bi se sva analiza svela na to da li ta referenca ima vrednost *null* ili nema. Takođe, ni analiza proizvoljnog elementa liste ne može pružiti mnogo više podataka: sve što bi se moglo ustanoviti jeste da je neki element prvi ili poslednji u listi, odnosno da je negde u sredini. Suštinski, ovaj problem je posledica šireg problema lokalnosti (locality, Mejer [78]) i treba ga napasti upravo na toj tački.

Da bismo rešili problem lokalnosti iskoristićemo ponovo činjenicu da se apstraktni prostor stanja može interpretirati proizvoljno, pod uslovom da bude zatvoren za dinamičku invarijantu. Ovo znači da apstraktni prostor stanja ne moramo interpretirati preko vrednosti stvarnih, fizičkih, polja - možemo ga interpretirati i logički, prema ideji iz [46], gde se takođe koriste logička polja.

Izložićemo, bez pretenzija na opštost, jednu mogućnost za analizu invarijanata u klasama poput *List*, tj. u klasama sa rekurzivnom strukturom. S obzirom na to da želimo da analiziramo strukturu objekta, kao prirodno rešenje za interpretaciju, a koje eliminiše problem lokalnosti, jeste interpretacija apstraktnog prostora stanja na skupu nelabeliranih konačnih digrafa. Neka je $\mathbb{G}_N$ skup svih konačnih digrafa bez petlji sa najviše N čvorova (pri čemu se ne pravi razlika između izomorfnih digrafa) i neka je $\mathbb{G}$ skup svih takvih skupova. Tada apstraktni prostor stanja interpretiramo kao element skupa $\mathbb{G}$. Dalje, neka je $first$ član klase *List* koji predstavlja referencu ili pokazivač na prvi element liste.

Osnovna ideja jeste da se polje $first$ zameni logičkim poljem, npr. LST što pripada intepretiranom prostoru stanja (tj. LST jeste konačni digraf). Mejer [79] tvrdi: "To study the semantics of O-O computation, it seems more productive to start by modeling the run-time object structure, and the associated operations such as feature call, then work our way up - in a second step of the effort, only sketched in the present article - to classes and other program-level mechanisms such as inheritance."

Upravo uvođenjem logičkih polja, kakvo je LST mi, u stvari, objedinjujemo pomenuta dva koraka u jedan: umesto da prvo modelujemo run-time strukturu objekata, pa iz nje modelujemo klasu, primenom logičkih polja mi unapred prenosimo run-time strukturu objekta na klasu.

Skup linearnih, acikličnih digrafa sa $N, N \geq 0$ čvorova ćemo označiti sa $\mathbb{L}_N$. Neka je

$$\bar{\phi} = (LST).$$

Određujemo čistu invarijantu objekta. Za nulti predikat važi:

$$O(\bar{\phi}) \equiv (this = null).$$

Pretpostavićemo da je klasa $List$ realizovana tako da konstruktor konstruiše praznu listu, tj. listu za koju važi $LST \in \mathbb{L}_0$:

$\hat{sdp}(List, \Gamma(List)) \equiv O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (LST' \in \mathbb{L}_0)$,
$IS^0_K(\bar{\phi}) \equiv \neg O(\bar{\phi}) \wedge (LST \in \mathbb{L}_0)$.

Takođe, pretpostavićemo da je klasa $List$ realizovana tako da ako metode za dodavanje/uklanjanje ne mogu da se izvrše, onda ostavljaju listu u zatečenom stanju:

$\hat{sdp}(add, \Gamma(add)) \equiv \neg O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge [((LST \in \mathbb{L}_N) \wedge (LST' \in \mathbb{L}_N)) \vee ((LST \in \mathbb{L}_N) \wedge (LST' \in \mathbb{L}_{N+1})) \vee ((LST \notin \mathbb{L}_N) \wedge (LST' \in \mathbb{G}))]$,

$\hat{sdp}(remove, \Gamma(remove)) \equiv \neg O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge [((LST \in \mathbb{L}_N) \wedge (LST' \in \mathbb{L}_N)) \vee ((LST \in \mathbb{L}_N) \wedge (LST \notin \mathbb{L}_0) \wedge (LST' \in \mathbb{L}_{N-1})) \vee ((LST \notin \mathbb{L}_N) \wedge (LST' \in \mathbb{G}))]$,

$\hat{JS} \equiv \hat{sdp}(List, \Gamma(List)) \vee \hat{sdp}(add, \Gamma(add)) \vee \hat{sdp}(remove, \Gamma(remove))$,

gde je add zajedničko ime za sve metode za dodavanje, a $remove$ zajedničko ime za sve metode za uklanjanje. Član $(LST' \in \mathbb{G})$ nije minimalna konsekventa, ali to nema uticaja jer, kao što ćemo videti, neće nigde biti korišćen, pošto uslov $(LST \notin \mathbb{L}_N)$ neće biti ispunjen.

Usvajamo da je:

$Z_0 \equiv W_0 \equiv IS^0_K(\bar{\phi})$

i primenom algoritma 4.3.23 određujemo strogu invarijantu:

$Z_0 \equiv W_0 \equiv IS_K^0(\bar{\phi})$

$W_0 \wedge \hat{JS} \equiv \neg O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (LST \in \mathbb{L}_0) \wedge [(LST' \in \mathbb{L}_0) \wedge (LST' \in \mathbb{L}_1)]$
$W_1 \equiv W_0 \otimes \hat{JS} \equiv \neg O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge (LST \in \mathbb{L}_0) \wedge [(LST' \in \mathbb{L}_0) \wedge (LST' \in \mathbb{L}_1)]$
$Z_1 \equiv Z_0 \vee W_1$
$W_1 \wedge \hat{JS} \equiv \neg O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge [(LST \in \mathbb{L}_0) \vee (LST \in \mathbb{L}_1)] \wedge [(LST' \in \mathbb{L}_0) \wedge (LST' \in \mathbb{L}_1) \wedge (LST' \in \mathbb{L}_2)]$
$W_1 \equiv W_0 \otimes \hat{JS} \equiv \neg O(\bar{\phi}) \wedge \neg O(\bar{\phi}') \wedge [(LST \in \mathbb{L}_0) \vee (LST \in \mathbb{L}_1)] \wedge [(LST' \in \mathbb{L}_0) \wedge (LST' \in \mathbb{L}_1) \wedge (LST' \in \mathbb{L}_2)]$

itd.

Lako se dokazuje da je invarijanta:

$$IS_{List} \equiv (this = null) \vee [(this \neq null) \wedge (LST \in \bigcup_{i=0}^{N} \mathbb{L}_i)].$$

Zahvaljujući ovakvom pristupu, možemo analizirati čak i objekte sa veoma složenom strukturom, kakvi inače mogu biti u programerskoj praksi, što ilustruje naredni primer.

4.7.1 Primer

Posmatrajmo klase *Akcija*, *Ivestitor* i *Berza*, čiji objekti imaju varijabilnu strukturu i gde koristimo logička polja. Klasa *Akcija* sadrži polja:

- *naziv* tipa *String*,
- *vlasnik* je referenca na objekat klase *Investitor*,
- *vrednost* je logičko polje koje predstavlja jednostrukospregnutu listu realnih pozitivnih brojeva (neka su to objekti klase *PositiveDouble*).

Stroga invarijanta klase *Akcija* jeste:

$IS_{Akcija} \equiv (this = null) \vee [(this \neq null) \wedge (vlasnik \neq null) \wedge (naziv \in D_{String}) \wedge (vrednost \in \bigcup_{i=0}^{N} \mathbb{L}_i^{PositiveDouble})]$.

Klasa *Investitor* sadrži polja:

- *naziv* tipa *String*,

- *akcije* je logičko polje koje predstavlja jednostrukospregnutu listu objekata klase *Akcija*, koji predstavljaju akcije koje su u vlasništvu investitora,
- *prodaja* je logičko polje koje predstavlja jednostrukospregnutu listu objekata klase *Akcija*, koji predstavljaju akcije koje investitor želi da proda.

Klasa *Investitor* sadrži metode:

- *proceniBerzu*(*Berza b*) u kojoj se procenjuje opravdanost ulaganja na berzi b i na osnovu njenog indeksa $b.indeks$, vrednosti akcija koje se prodaju i koje se nalaze u listi $b.prodaja$, kao i na osnovu vrednosti faktora $faktor_1, \ldots, faktor_n$. Izbor relevantnih faktora rizika, njihove značajnosti, kao i formulisanje sâmog algoritma koji obavlja procenu rizika jeste predmet razmatranja veštačke inteligencije. Na primer, u radu [28] se razmatraju faktori: rizik države (*sovereign risk*), oscilatornost tržišta (*market volatility*), stopa inflacije (*inflation rate*), likvidnost tržišta (*market liquidity*), nivo korupcije (*corruption level*), BDP po glavi stanovnika (*GDP per capita*) i broj stanovnika (*demography*). Specijalno, kada se razmatraju berze zemalja u razvoju, koje odlikuje nestabilnost i nepredvidivost, važan faktor jeste koeficijent linerne korelacije datog indeksa u odnosu na najznačajniji berzanski indeks na svetu S&P 500. Sledeća tabela pokazuje rezultate koeficijenata linearnih korelacija za parove berzanskih indeksa država u našem regionu (Rumunija – BET, Mađarska – BUX, Bugarska – SOFIX, Srbija – BELEX LINE, Hrvatska – CROBEX i Slovenija – SBI20) i berzanskog indeksa S&P 500 [28].

Tabela 4.1: Koeficijenti linearne korelacije r^{xy}

Berzanski indeks x	Berzanski indeks y	r^{xy}
BET	S&P 500	0.9725
BUX	S&P 500	0.9574
SOFIX	S&P 500	0.9463
BELEX LINE	S&P 500	0.9448
CROBEX	S&P 500	0.9374
SBI20	S&P 500	0.9011

- *oglasiProdaju*(*Akcija a*, *Berza b*) u kojoj investitor oglašava prodaju akcije a na berzi b, odnosno akcija a se ubacuje u liste $prodaja$ i $b.prodaja$.

Stroga invarijanta klase *Investitor* jeste:

$IS_{Investitor} \equiv (this = null) \vee [(this \neq null) \wedge (naziv \in D_{String}) \wedge (akcije \in \bigcup_{i=0}^{N} \mathbb{L}_i^{Akcija}) \wedge (prodaja \in \bigcup_{i=0}^{N} \mathbb{L}_i^{Akcija})]$.

Klasa *Berza* sadrži polja:

- *naziv* tipa *String*,
- *indeks* je logičko polje koje predstavlja jednostrukospregnutu listu realnih pozitivnih brojeva.
- $faktor_1, \ldots, faktor_n$ predstavljaju faktore rizika ulaganja i faktore ekonomskog rasta, koji su redom tipa $Type_1, \ldots, Type_n$,
- *prodaja* je logičko polje koje predstavlja jednostrukospregnutu listu objekata klase *Akcija*, koji predstavljaju akcije koje se prodaju na berzi.

Klasa *Berza* sadrži metodu:

- *realizujProdaju*(*Akcija a*, *Investitor prodavac*, *Investitor kupac*) koja uklanja akciju *a* iz lista *prodaja*, *prodavac.prodaja* i *prodavac.akcije* i ubacuje u listu *kupac.akcije*.

Stroga invarijanta klase *Berza* jeste:

$IS_{Berza} \equiv (this = null) \vee [(this \neq null) \wedge (naziv \in D_{String}) \wedge (indeks \in \bigcup_{i=0}^{N} \mathbb{L}_i^{PozitiveDouble}) \wedge ((faktor_1 \in D_{Type_1}) \wedge \cdots \wedge (faktor_n \in D_{Type_n})) \wedge (prodaja \in \bigcup_{i=0}^{N} \mathbb{L}_i^{Akcija})]$.

Zaključak

U ovom izlaganju prikazan je razvoj S-računa u čijoj osnovi se nalaze aksiome i teoreme predikatske logike prvog reda i prikazana njegova primena u analizi semantike strukturiranih programa. Pokazano je da Horova logika predstavlja specijalan slučaj S-računa, odnosno predikatske logike prvog reda. Prikazan je način na koji se opšti Horovi zakoni mogu dokazivati automatski (korišćen je automatski dokazivač `Coq`). Uvedene su S-formule koje definišu garantovano i potencijalno terminirajuću *while* petlju, koje omogućavaju da se *while* petlja može razmatrati u okviru predikatske logike prvog reda. Zatim, u S-račun uvedeni su dinamički postuslovi, koji predstavljaju logičke funkcije početnog i završnog stanja, odnosno logičke funkcije prelaza i pokazano je da opšti Horovi zakoni važe i onda kada se u tripletima nalaze dinamički postuslovi. Detaljno su ispitane osobine najužeg statičkog sp i najužeg dinamičkog $\hat{sdp}$ postuslova.

Primenom dinamičkih postuslova razvijena je DP-analiza invarijanata u klasi. U okviru DP-analize formulisani su rekurzivni algoritmi za izračunavanje različitih invarijanata u klasi. Razmatrano je izračunavanje invarijante vlasnika i komponente, natklase i potklase, kao i objekata sa varijabilnom strukturom koji sadrže pokazivače/reference koji pokazuju na neki memorijski prostor izvan objekta. Time je otvorena mogućnost za razmatranje semantike klasa koje su povezane klijentskim vezama, odnosno nasleđivanjem. Ovo izlaganje predstavlja zaokruženu celinu, jer prikazuje mogućnost analize semantike kako strukturiranih, tako i objektno orijentisanih programa.

Glavni naučni doprinos ovog izlaganje jeste u tome što argumentuje da se kompletna analiza semantike strukturiranih i objektno orijentisanih programa može izvesti u okviru predikatske logike prvog reda. Buduća istraživanja će se kretati u pravcu proučavanja invarijanata objektno orijentisanog programa i mogućnosti automatizacije celog postupka.

Prilog A

Dokazivanje opštih Horovih zakona automatskim dokazivačem Coq

Teorema 1.3.1.*a.*):

```
Variable A: Set.
Variables P Q R: A->Prop.
Variable S: A->A->Prop.
Theorem t131a :
((forall x:A,(P x->R x)) /\
(forall x:A, (R x ->((exists y:A, S x y)/\(forall z:A, (S x z ->Q z ))))))
-> (forall x, (P x ->((exists y, S x y)/\(forall z, (S x z ->Q z ))))).
firstorder.
```

Teorema 1.3.1.*b.*):

```
Variable A: Set.
Variables P Q R: A->Prop.
Variable S: A->A->Prop.
Theorem t131b :
((forall x:A, (P x ->((exists y:A, S x y)/\(forall z:A, (S x z ->Q z )))))
/\ (forall z:A,(Q z->R z)))
-> (forall x, (P x ->((exists y, S x y)/\(forall z, (S x z ->R z ))))).
firstorder.
```

Teorema 1.3.1.*c.*):

```
Variable A: Set.
Variables P Q U V: A->Prop.
Variable S: A->A->Prop.
Theorem t131c :
((forall x:A,(U x->P x)) /\
(forall x:A, (P x ->((exists y:A, S x y)/\(forall z:A, (S x z ->Q z )))))
/\ (forall z:A,(Q z->V z)))
-> (forall x, (U x ->((exists y, S x y)/\(forall z, (S x z ->V z ))))).
firstorder.
```

Teorema 1.3.2.*a.*):

```
Variable A: Set.
Variables P R Q W: A->Prop.
Variable S: A->A->Prop.
Theorem t132a :
((forall x:A, (P x -> ((exists y:A, S x y)/\(forall z:A, (S x z->Q z )))))
/\
(forall x:A, (R x -> ((exists y:A, S x y)/\(forall z:A, (S x z->W z ))))))
->
(forall x:A, ((P x \/ R x) -> ((exists y:A, S x y)/\(forall z:A, (S x z->(Q
z \/ W z)))))).
firstorder.
```

Teorema 1.3.2.*b.*):

```
Variable A: Set.
Variables P Q R W: A->Prop.
Variable S: A->A->Prop.
Theorem t132b :
((forall x:A, (P x -> ((exists y:A, S x y)/\(forall z:A, (S x z->Q z )))))
/\
(forall x:A, (R x -> ((exists y:A, S x y)/\(forall z:A, (S x z->W z ))))))
->
(forall x:A, ((P x /\ R x) -> ((exists y:A, S x y)/\(forall z:A, (S x z->(Q
z /\ W z)))))).
firstorder.
```

Teorema 1.3.4:

```
Variable A: Set.
Variables P Q R W: A->Prop.
Variable S: A->A->Prop.
Theorem t133 :
((forall x:A, (P x -> ((exists y:A, S x y)/\(forall z:A, (S x z->Q z )))))
\/
(forall x:A, (R x -> ((exists y:A, S x y)/\(forall z:A, (S x z->W z ))))))
->
(forall x:A, ((P x /\ R x) -> ((exists y:A, S x y)/\(forall z:A, (S x z->(Q
z \/W z)))))).
firstorder.
```

Teorema 1.3.6:

```
Variable A: Set.
Variables P : A->Prop.
Variable S: A->A->Prop.
Definition phi (x:A) := False.
Theorem t136 :
(forall x:A, (P x -> ((exists y:A, S x y)/\(forall z:A, (S x z->phi z)))) <->
(P x<->phi x)).
firstorder.
```

Teorema 1.3.7.a):

```
Variable A: Set.
Variables P Q R: A->Prop.
Variable S: A->A->Prop.
Theorem t137a :
((forall x:A, (P x -> ((exists y:A, S x y)/\(forall z:A, (S x z->Q z)))))
/\
(forall x:A, (R x -> ((exists y:A, S x y)/\(forall z:A, (S x z->(~Q z)))))))
->
(forall x:A, (~(P x /\ R x))).
firstorder.
```

Teorema 1.3.7.*b*):

```
Variable A: Set.
Variables P Q: A->Prop.
Variable S: A->A->Prop.
Theorem t137b :
((forall x:A, (P x -> ((exists y:A, S x y)/\(forall z:A, (S x z->Q z)))))
/\
(forall x:A, (P x -> ((exists y:A, S x y)/\(forall z:A, (S x z->(~Q z)))))))
<-> (forall x:A, (~(P x))).
firstorder.
```

Teorema 1.6.3:

```
Variable A: Set.
Variables P wpSphi:  A->Prop.
Variable S: A->A->Prop.
Definition phi (x:A) := False.
Axiom wpSphi1 :
forall x:A, (wpSphi x -> ((exists y:A, S x y)/\(forall z:A, (S x z->phi z)))).
Axiom wpSphi2 :
forall x:A, (P x -> ((exists y:A, S x y)/\(forall z:A, (S x z->phi z)))) ->
(P x->wpSphi x).
Theorem t163 :  forall x:A, (wpSphi x <-> phi x).
firstorder using wpSphi1 wpSphi2.
```

Teorema 1.6.7:

```
Variable A: Set.
Variables P Q R wpSQ wpSNQ: A->Prop.
Variable S: A->A->Prop.
Axiom wpSQ1 :
forall x:A, (wpSQ x -> ((exists y:A, S x y)/\(forall z:A, (S x z->Q z)))).
Axiom wpSQ2 :
forall x:A, (P x -> ((exists y:A, S x y)/\(forall z:A, (S x z->Q z)))) ->
(P x->wpSQ x).
Axiom wpSNQ1 :
forall x:A, (wpSNQ x -> ((exists y:A, S x y)/\(forall z:A, (S x z->(~(Q z)))))).
Axiom wpSNQ2 :
forall x:A, (P x -> ((exists y:A, S x y)/\(forall z:A, (S x z->(~(Q z))))))
-> (P x->wpSNQ x).
Theorem t167 :  forall x:A, (~((wpSQ x)/\(wpSNQ x))).
firstorder using wpSQ1 wpSQ2 wpSNQ1 wpSNQ2.
```

Teorema 1.6.8:

```
Variable A: Set.
Variables P Q wpSQ: A->Prop.
Variable S: A->A->Prop.
Axiom wpSQ1 :
forall x:A, (wpSQ x -> ((exists y:A, S x y)/\(forall z:A, (S x z->Q z)))).
Axiom wpSQ2 :
forall x:A, (P x -> ((exists y:A, S x y)/\(forall z:A, (S x z->Q z)))) ->
(P x->wpSQ x).
Theorem t168 :
forall x:A, ((P x -> ((exists y:A, S x y)/\(forall z:A, (S x z->Q z)))) <->
((P x)->(wpSQ x))).
firstorder using wpSQ1 wpSQ2.
```

Literatura

[1] Aggarwal, A. and Randall, K. H. 2001. Related field analysis. In *PLDI'01.*, ACM Press, (June, 2001).

[2] Amadi M. and Cardelli L. 1996. *A Theory of Objects*, New York: Springer-Verlag.

[3] Apt, K. R. 1981. Ten Years of Hoare's Logic: A Survey - Part I. *ACM Transactions on Programming Languages and Systems (TOPLAS) 3*, 4 (October), 431–483.

[4] Apt, K. R. and Olderog, E. -R. 1991. *Verification of Sequential and Concurrent Programs.* New York: Springer-Verlag.

[5] Arthan, R., Martin, U., Mathiesen, E. A., and Oliva, P. 2009. A general framework for sound and complete Floyd-Hoare logics. *ACM Transactions on Computational Logic (TOCL) 11*, 1, 1–31.

[6] Back, R. J. and von Wright, J. 1992. Combining angels, demons and miracles in program specifications. *Theoretical Computer Science 100*, 365–383.

[7] Back, R. J., Akademi, A., and von Wright, J. 1998. *Refinement Calculus: A Systematic Introduction.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[8] Backhouse, R. C. 1986. *Program Construction and Verification.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[9] Barnett, M., Deline, R., Fähndrich, M., Leino, K. R. M. and Schulte, W. 2004. Verification of object-oriented programs with invariants. *Journal of Object Technology*, 3:6, 27–56.

[10] Berg, H. K., Boebert, W. E., Franta, W. R., and Moher, T. G. 1982. *Formal Methods of Program Verification and Specification.* Prentice-Hall, Englewood Cliffs, New Jersey.

[11] Bertot, Y. and Castéran, P. 2004. *Interactive Theorem Proving and Program Development, Coq'Art:the Calculus of Inductive Constructions.* Texts in Theoretical Computer Science. An EATCS Series, vol. XXV. Springer-Verlag, Berlin, Heidelberg.

[12] Blass, A. and Gurevich, Y. 2000. The underlying logic of Hoare logic. *Bull. of the Euro. Assoc. for Theoretical Computer Science 70*, 82–110.

[13] Booch, G. 1994. *Object-Oriented Analysis and Design with Applications, (2nd Edition).* Addison-Wesley.

[14] Chatterjee, R., Ryder, B.G. and Landi, W.A. 1999. Relevant context inference. In *26th ACM Symposium on Principles of Programming Languages (POPL'99).* ACM Press, New York, USA, 133-146.

[15] Church, A. 1941. *The Calculi of Lambda Conversion.* Princeton University Press, Princeton, NJ, USA.

[16] Church, A. 1956. *Introduction to Mathematical Logic, (Vol. 1).* Princeton University Press, Princeton, NJ, USA.

[17] Clarke, E. M. 1979. Program Invariants as Fixedpoints. *Computing*, 21, 273–294.

[18] Cook, S. A. 1978. Soundness and completeness of an axiom system for program verification. *SIAM Jour. Comput. 7*, 1 (February), 70–90.

[19] Cousot, P. and Cousot, R. 1977. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th ACM Symposium on Principles of Programming Languages (POPL'77)*, ACM Press, New York, USA, 238–252.

[20] Cox B. 1986. *Object Oriented Programming: An Evolutionary Approach*, Reading MA, Addison–Wesley.

[21] Curry, H. 1963. *Foundations of Mathematical Logic.* McGraw-Hill, New York, USA.

[22] Cvetković, D. i Simić, S. 1990. *Diskretna matematika - matematika za kompjuterske nauke.* Beograd: Naučna knjiga.

[23] Dahl, O. J., Dijkstra, E. W., and Hoare, C. A. R. 1972. *Structured Programming.* Academic Press Ltd., London, UK, UK.

[24] de Bakker, J. W. 1980. *Mathematical Theory of Program Correctness.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[25] Delić, V., Sečujski, M. and Kupusinac, A. 2009. Transformation-Based Part-Of-Speech Tagging For Serbian Language, In *8th WSEAS Intnl. Conf. on Computational Intelligence, Man-Machine Systems and Cybernetics (CIMMACS)*, Peurto de la Cruz: Tenerife, Spain, 14-16 Decembar, 2009, 98–103.

[26] Dijkstra, E. W. 1975. Guarded Commands, Nondeterminacy and Formal Derivation of Programs. *Comunications of the ACM 18*, 8 (August), 453–457.

[27] Dijkstra, E. W. 1976. *A Discipline of Programming.* Prentice-Hall, Englewood Cliffs, NJ, USA.

[28] Dobromirov, D., Radišić, M. and Kupusinac, A. 2010. Emerging Markets Arbitrages' Perception: Risk vs. Growth Potential, *African Journal of Business Management*, Vol. in press (AJBM-10-060 Dobromirov et al), ISSN 1993-8233.

[29] Eckel, B. 2000. *Thinking in C++, (2nd Edition)*, Prentice-Hall, Englewood Cliffs, NJ, USA.

[30] Eckel, B. 2002. *Thinking in Java, (3nd Edition)*, Prentice-Hall, Englewood Cliffs, NJ, USA.

[31] Flanagan, C., Leino, K. R. M., Lillibridge, M., Nelson G., Saxe, J. B., and Stata, R. 2002. Extended static checking for Java. In *PLDI'02.* ACM Press, (June, 2002).

[32] Floyd, R. W. 1967. Assigning meanings to programs. In *Mathematical Aspects of Computer Science, Proceedings of Symposia in Applied Mathematics.* American Mathematical Society, Providence, RI, USA, 19–32.

[33] Gries, D. 1987. *The Science of Programming.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[34] Goldblatt, R. 1982. *Axiomatising the Logic of Computer Programming.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[35] Gordon, M. J. C. 1988. *Programming Language Theory and its Implementation.* Prentice-Hall International (UK) Ltd., Hertfordshire, UK, UK.

[36] Hehner, E. C. R. 1984. Predicative programming. *Communications of the ACM 27*, 2, 134–151.

[37] Hehner, E. C. R. 1999. Specifications, Programs, and Total Correctness. *Science of Computer Programming 34*, 191–205.

[38] Hehner, E. C. R. 2009. *A Practical Theory of Programming.* Department of Computer Science, University of Toronto, Canada, www.cs.utoronto.ca/∼hehner/aPToP.

[39] Hehner, E. C. R. and Gravell A. M. 1999. Refinement semantics and loop rules. In *Proc. FM'99: World Congress on Formal Methods.*, 1497–1510.

[40] Herrmann, D. S. 1999. *Software Safety and RealibilityL techniques, Approaches and Standards of Key Industrial Sectors*, IEEE Computer Society, ISBN 0-7695-0299-7.

[41] Hoare, C. A. R. 1969. An Axiomatic Basis for Computer Programming. *Comunications of the ACM 12*, 10 (October), 576–585.

[42] Hoare, C. A. R. and Lauer, P. E. 1974. Consistent and Complementary Formal Theories of the Semantics of Programming Languages. *Acta Informatica 3*, 135–153.

[43] Hoare, C. A. R. 1992. Programs are predicates. In *Proceedings of the International Conference on Fifth Generation Computer Systems (FGCS'92).* Institute for New Generation Computer Technology (ICOT, Ed.), Tokyo, Japan, 211–218.

[44] Hoare, C. A. R. and Roscoe, A. W. 1984. Programs as executable predicates. In *Proceedings of the International Conference on Fifth Generation Computer Systems (FGCS'84).* Institute for New Generation Computer Technology (ICOT, Ed.), Tokyo, Japan, 220–228.

[45] Hotomski, P. i Malbaški, D. 2006. *Matematička logika i principi programiranja.* Zrenjanin: Tehnički fakultet „Mihajlo Pupin".

[46] Huizing, K. and Kuiper, R. 2000. Verification of Object Oriented Programs Using Class Invariants. In Proceedings of the Third International Conference on Fundamental Approaches to Software Engineering: Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS 2000, Lecture Notes in Computer Science, vol. 1783, Berlin: Springer-Verlag, March 25-April 02, 2000, 208–221.

[47] Janičić, P. 2009. *Matematička logika u računarstvu.* Beograd: Matematički fakultet.

[48] Kleene, S. C. 1967. *Mathematical Logic.* JohnWiley and Sons, Inc., New York, USA.

[49] Kraus, L. 2004. *Programski jezik C++ sa rešenim zadacima.* Beograd: Akademska misao.

[50] Kupusinac, A. 2005. *Rešenje jednodimenzionalnog* bin–packing *problema primenom genetskog algoritma (diplomski rad).* Novi Sad: Fakultet tehničkih nauka.

[51] Kupusinac, A. 2008. *Invarijanta klase u objektno orijentisanom programiranju (magistarska teza).* Novi Sad: Fakultet tehničkih nauka.

[52] Kupusinac, A. 2008. Rešavanje jednodimenzionalnog *bin-packing* problema primenom permutacionog genetskog algoritma, *7. DOGS, Digitalna obrada govora i slike*, Kelebija, 2-3 Oktobar, 2008, 225–228, ISBN 978-86-7892-136-0.

[53] Kupusinac, A. i Malbaški, D. 2007. Invarijanta klase u objektno orijentisanom programiranju i njena primena. *15. Telekomunikacioni forum TELFOR*, Beograd: 20-22. novembar, 2007, 589–592, ISBN 978-86-7466-301-1.

[54] Kupusinac, A. and Malbaški, D. 2008. Class composition and correctness. In *12th Serbian Mathematical Congress*, Novi Sad: PMF, Novi Sad, 28. avgust - 2. septembar, 2008.

[55] Kupusinac, A. i Malbaški, D. Korektnost potklase. *16. Telekomunikacioni forum TELFOR*, Beograd: 25-27. novembar, 2008, 747–750, ISBN 978-86-7466-337-0.

[56] Kupusinac, A. i Malbaški, D. 2009. Korektnost i kompozicija klase, *17. Telekomunikacioni forum TELFOR*, Beograd, 24-26 Novembar, 2009, 1291–1294, ISBN 978-86-7466-375-2.

[57] Kupusinac, A. and Malbaški, D. 2010. *S-Program Calculus*, CoRR, abs/1003.0773, http://arxiv.org/abs/1003.0773 .

[58] Kupusinac, A. i Sečujski, M. 2008. Poređenje dva postupka automatske morfološke anotacije teksta, *7. DOGS, Digitalna obrada govora i slike*, Kelebija, 2-3 Oktobar, 2008, 20–23, ISBN 978-86-7892-136-0.

[59] Kupusinac, A. and Sečujski, M. 2009. Part-of-Speech Tagging Based on Combining Markov Models and Machine Learning. In *3rd Speech and Language*, Beograd, 13-14 Novembar, 2009.

[60] Kusinitz, A. 1997. Software Validation, *Current Issues in Medical Device Quality Systems*, Association for the Advancement of Medical Instrumentation, ISBN 1-57020-075-0.

[61] Leino, K. R. M. and Müller, P. 2005. Modular verification of static class invariants. *FM 2005: Formal Methods*, pp. 26–42.

[62] Lieberherr, K. J. and Holland I. 1988. *Formulations of the Law of Demeter (technical report).* Demeter 2, Northeastern University, Bostion.

[63] Lieberherr, K. J. and Riel A. 1988. Object-oriented programming: an objective sense of style. *OOPSLA '88 Proceedings.*

[64] Liskov, B. and Wing, J. 1994. A behavioral notion of subtyping. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 16, num. 6 (November 1994), pp. 1811–1841.

[65] Logozzo, F. Class-level modular analysis for object oriented languages. In *Proceedings of the 10th static analysis symposium 2003 (SAS 03).* Lecture Notes in Computer Science, vol. 2694. Berlin: Springer-Verlag, 37-54.

[66] Logozzo, F. 2004. Automatic inference of class invariants. In *Proceedings of the fifth conference on verification, model checking and abstract interpretation (VMCAI'04).* Lectures Notes in Computer Science, vol. 2937. Berlin: Springer-Verlag, 211-222.

[67] Logozzo, F. 2009. Class invariants as abstract interpretation of trace semantics. *Computer Languages, Systems & Structures*, Elsevier, vol. 35, num. 2 (July), 100–142.

[68] Malbaški, D. and Obradović, D. 2002. On Some Basic Concepts in Object Orientation, *6th Balcan Conference on Operational Research*, Thessaloniki.

[69] Malbaški, D. 2003. Semantika programskih jezika. Tech. Rep. 021–21/17, Fakultet tehničkih nauka, Novi Sad, Serbia.

[70] Malbaški, D. 2006. *Objekti i objektno programiranje kroz programske jezike C++ i Pascal*, Novi Sad: Fakultet tehničkih nauka, 2006.

[71] Malbaški, D. 2008. *Objektno orijentisano programiranje kroz programski jezik C++*, Novi Sad: Fakultet tehničkih nauka.

[72] Malbaški, D. 2010. O invarijantama u klasi. Tech. Rep. 021–21/24, Fakultet tehničkih nauka, Novi Sad, Serbia.

[73] Malbaški, D. and Kupusinac, A. 2009. Introduction to *S*-program calculus. Tech. Rep. 021–21/128, Faculty of Technical Science, Novi Sad, Serbia.

[74] Marković, M. 1994. *Filozofski osnovi nauke*, Beograd: Prosveta.

[75] Menezes, A. J, van Oorschot, P. C. and Vanstone, S. A. 1997. *Handbook of Applied Cryptography.* CRC Press LLC.

[76] Meyer, B. 1988. *Object–Oriented Software Construction, (1st Edition)*, Prentice-Hall.

[77] Meyer, B. 1997. *Object–Oriented Software Construction, (2nd Edition)*, Prentice-Hall.

[78] Meyer, B. *Proving pointer program properties, Part 1: Context and overwiev.* http://www2.inf.ethz.ch/∼meyer .

[79] Meyer, B., 2003. Towards Practical Proofs of Class Correctness. In *ZB 2003: Formal Specification and Development in Z and B, Lecture Notes in Computer Science vol. 2651*, Berlin: Springer-Verlag, 359–387.

[80] Motschnig-Pitrik, R. and Kaasbøll, J. Part-Whole Relationship Categories and Their Application in Object-Oriented Analysis. *IEEE Transactions on Knowledge and Data Engineering*, 11:5, 779–797.

[81] Müller, P. and Poetzch-Heffter A. 1997. Formal Specification Techniques for Object-Oriented Programs. *GI Jahrestagung*, 602–611.

[82] Nakano, M., Ogata, K., Nakamura, M., and Futatsugi, K. 2006. Automating invariant verification of behavioral specifications. In *Proceedings of the Sixth International Conference on Quality Software (QSIC'06).* IEEE Computer Society, Beijing, China, 49–56.

[83] Neumann, P. G. 1995. *Computer Related Risks*, ACM Press/Addison-Wesley Publishing Co., ISBN 0-201-55805-X.

[84] Noble, J., Vitek, J. and Potter J. 1998. Flexible alias protection. In *Proceedings of the 12th European Conference on Object-Oriented Programming, ed. Eric Jul, ECOOP'98, Lecture Notes in Computer Science, vol. 1445 (July 1998)*, Berlin: Springer-Verlag, 158–185.

[85] Petrović, G. 1981. *Logika*, Zagreb: Školska knjiga.

[86] Pierik, C., Clarke D. and De Boer, F. 2004. Creational invariants. In *Proc. of the Formal Techniques for Java-like Programs Workshop (FTfJP 2004).* The proceedings of the workshop are available as technical report nr. NIII-R046, University of Nijmegen.

[87] Pratt, V. R. 1976. Semantical Considerations on Floyd-Hoare logic. Tech. Rep. MIT/LCS/TR 168, Massachusetts Institute of Technology, Laboratory for Computer Science, Massachusett, USA.

[88] Probst, C. 2002. Modular control flow analysis for libraries. In *Proceedings of the Static Analysys Symposium (SAS 2002).* Springer–Verlag, vol. 2477, 165-179.

[89] Popović, D. i Popović, M. 1997. *Biomedicinska instrumentacija i merenja.* Beograd: Nauka.

[90] Ramalingam, G., Warshavsky, A., Field, J., Goyal, D., and Sagiv, M. 2002. Deriving specialized program analyses for certifying component-client conformance. In *PLDI'02.* (June 2002).

[91] Sečujski, M. i Kupusinac, A. Određivanje položaja leksičkog akcenta u govornoj bazi korišćenjem stabala odluke, *8. Infoteh*, Jahorina, 18-20 Mart, 2009, 228–231.

[92] Sečujski, M. i Kupusinac, A. 2009. Poređenje postupaka automatske morfološke anotacije tekstova na srpskom jeziku. *17. TELFOR*, Beograd: Drutvo za telekomunikacije, 24-26 Novembar, 2009, 1085–1092, ISBN 978-86-7466-375-2.

[93] Sečujski, M., Kupusinac, A. and Pekar, D. 2009. Prediction of phone duration in Serbian language based on decision trees. In *3. Die Unterschiede zwischen dem Bosnischen/ Bosniakischen, Kroatischen und Serbischen*, Graz, 16-18 April, 2009, 229–240.

[94] Shoenfield, J. R. 1967. *Mathematical Logic.* Addison-Wesley Publishing Company, Reading, MA, USA.

[95] Slonneger, K. and Kurtz, B. L. 1995. *Formal Syntax and Semantics of Programming Languages.* Addison-Wesley Publishing Company, Reading, MA, USA.

[96] Smith, M. and Tockey, S. 1988. An Integrated Approach to Software Requirements Definition Using Objects. In *Proceedings of the 10th Structured Development Forum*, August, 1988.

[97] Snoeck, M. and Dedene, G. 1998. Existence dependency: the key to semantic integrity be-tween structural and behavioral aspects of object types. *IEEE Transactions on Software Engineering*, 24:4, 233–251.

[98] Sun Microsystem, Inc. 2002. *javadoc Tool Home Page.* http://java.sun.com/j2se/javadoc/.

[99] Stroustrup, B. 1991. *Programski jezik C++*, Mikro knjiga, Beograd.

[100] Szyperski C. 1998. *Component Software: Beyond Object-Oriented Programming.* Addison-Wesley.

[101] *The Coq Development Team. 2009. The Coq Proof Assistant Reference Manual Version 8.2-bugfix.* INRIA, Orsay, France.

[102] Tarski, A. 1955. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309.

[103] Zwiers, J. and Roever, W. 1989. Predicates are predicate transformers: a unified compositional theory for concurrency. In *Proceedings of the eighth annual ACM Symposium on Principles of distributed computing (PODC'89, Edmonton, Alberta, Canada).* ACM, New York, NY, USA, 265–279.

Biografija autora

Mr Aleksandar Kupusinac je rodjen 02.07.1981. u Novom Sadu. Završio je osnovnu školu i gimnaziju u Odžacima, nižu muzičku školu u Apatinu i srednju muzičku školu u Zrenjaninu. Osnovne studije je upisao 2000. godine na Fakultetu tehničkih nauka u Novom Sadu, oblast: Elektrotehnika i računarstvo, odsek: Računarstvo i automatika. U toku osnovnih studija više puta je nagrađivan od Univerziteta u Novom Sadu (nagrada "Mileva Marić–Ajnštajn", izuzetne nagrade za postignut uspeh u toku školskih 2002/2003, 2003/2004 i 2004/2005 godina i nagrade za urađene temate). Dobio je i nagradnu stipendiju za 500 najboljih studenata u Republici Srbiji od Ambasade Kraljevine Norveške u Beogradu. Diplomirao je 14.04.2005. godine na Fakultetu tehničkih nauka u Novom Sadu, na temu: "Rešenje jednodimenzionalnog *bin–packing* problema primenom genetskog algoritma" sa ocenom 10. Prosečna ocena na osnovnim studijama je: 9,71. Postdiplomske studije upisao je 2005. godine na Fakultetu tehničkih nauka u Novom Sadu, odsek: Opšte discipline u tehnici, smer: Matematika u tehnici (predmeti: Kombinatorika, Teorija grafova, Teorija igara, Statistika, Teorija algoritama i programiranje, Objektno orijentisane tehnologije). Sve ispite predviđene nastavnim planom u okviru postdiplomskih studija položio je u roku i sa ocenama 10. Magistarsku tezu: "Invarijanta klase u objektno orijentisanom programiranju" uspešno je odbranio 06.03.2008. godine. Od 01.04.2007. godine je zaposlen na Fakultetu tehničkih nauka u Novom Sadu, prvo kao saradnik u nastavi, a docnije kao asistent sa magistraturom i predavač strukovnih studija. Na Katedri za primenjene računarske nauke drži vežbe iz predmeta: Objektno programiranje (C++ i Java) i Objektno orijentisano programiranje (C++). Predaje Računarstvo i informatiku talentovanim učenicima u specijalnim matematičkim odeljenjima u Gimnaziji "Jovan Jovanović Zmaj" u Novom Sadu.