Logička analiza hibridnih sustava

Ivan Gavran[1] i Mladen Vuković[2]

Sažetak. Suvremena razina znanosti i inženjerstva omogućuje razvoj vrlo složenih sustava. Uz razvoj takvih sustava, jednako važnu ulogu predstavlja i njihova analiza. Ona omogućuje – već na razini modela – donošenje zaključaka o ponašanju budućih sustava, ispravljanje potencijalnih nedostataka i upotrebu optimalne tehnologije. Pojam *hibridni sustavi* označava sustave koji u sebi sadrže diskretnu (primjerice digitalnu) i kontinuiranu (primjerice fizikalnu) komponentu. Takvi su sustavi česti u automobilskoj industriji, zračnom i željezničkom prometu, medicinskim uređajima... U ovom članku razmatramo tzv. logičku analizu hibridnih sustava.

1 Uvod

Osamdesetih godina prošlog stoljeća došlo je do velikog napretka u analizi konačnih sustava: tada su Clarke i Emerson predstavili metodu *model checking*. Osnovna je ideja efikasno pretražiti sva moguća stanja u kojima se sustav može naći i detektirati ona *nepoželjna*. Ta je tehnika kasnije proširena i na konačne apstrakcije beskonačnih sustava. Zbog svog principa rada, *model checking* je prikladan za pronalaženje konkretnih protuprimjera. Drugi princip često korišten u analizi je tzv. deduktivna verifikacija. Ideja je pritom pomoću automatskih dokazivača dokazati poželjna svojstva sustava. Oba ova principa pokazala su se neodgovarajućima za analizu hibridnih sustava; prvi zbog prevelike složenosti (*eksplozija* broja stanja, što je i inače problem pri *model–checkingu*, u hibridnim se sustavima ne može nikako kontrolirati), a drugi zbog premalene izražajnosti temporalnih logika.

Američki matematičar André Platzer u svojoj disertaciji bavi se problemom učinkovite analize hibridnih sustava. Analizu provodi kompozicijskim računom - naime, dokaz poželjnog svojstva razlaže na manje složene dokaze. Za potrebe analize uvodi i tri različite logike. U ovom članku mi se bavimo samo najjednostavnijom od njih – **diferencijalnom dinamičkom logikom**. Za tu logiku koristimo oznaku *dL*. Na temelju logike *dL* Platzer je razvio i alat za automatsku verifikaciju *KeYmaera*.

[1]Ivan Gavran, Ballebeat, Inc, Zagreb
[2]Mladen Vuković, PMF – Matematički odsjek, Zagreb

KeYmaera je testirana na nekoliko pokaznih primjera. Jedan od njih je ETCS (eng. European Train Control System) – europski sustav mreže željeznica, koji se postupno implementira, i to će biti glavni motivacijski primjer u ovom članku.

Članak je podijeljen u četiri točke. Nakon Uvoda, u drugoj točki dajemo primjere hibridnih sustava. Na tim ćemo primjerima demonstrirati definicije i tehnike koje će biti uvedene u nastavku članka. Logiku *dL* uvodimo u trećoj točki gdje definiramo jezik logike – terme i formule, ali i hibridne programe kao dio logike *dL*. U toj točki predstavljamo i formalni račun za analizu hibridnih sustava. U četvrtoj točki dajemo ilustraciju korištenja logike *dL* na jednostavnom primjeru u kontekstu sustava ETCS.

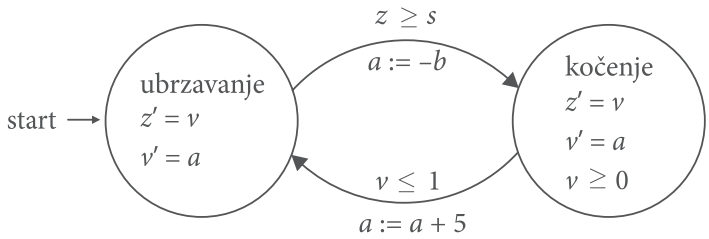
Ovaj članak zapravo je sažetak diplomskog rada [2]. Zbog ograničenog prostora ovdje nisu dani dokazi, a dano je i vrlo malo primjera. Dokaze, a i više detalja, možete pronaći u navedenom diplomskom radu.

2 Primjeri hibridnih sustava

U ovoj točki uvodimo nekoliko radnih, pokaznih primjera hibridnih sustava. Tim ćemo se pojednostavljenim primjerima uvijek iznova vraćati u narednim točkama kako bismo uz njihovu pomoć ilustrirali novouvedene definicije ili tehnike. Također, da bi svrha razvoja *dL* bila jasnija, odmah uz primjere neformalno ćemo predstaviti pojmove koji će biti definirani tek kasnije, i uz tako dobivenu širu sliku postaviti pitanja na koja u ostatku članka želimo dati odgovore.

Primjer 2.1 *Promotrimo jedan (primitivan) mogući sustav za upravljanje vlakom. Na Slici 1. nalazi se hibridni automat koji ga opisuje. Čvorovi predstavljaju dva različita stanja sustava (ubrzavanje i kočenje). Diferencijalne jednadžbe u njima opisuju gibanje vlaka. Bridovi prikazuju upravljanje sustavom od strane logičke jedinice. Diferencijalna jednadžba u oba čvora je ista (naime, $z' = v$, $v' = a$) i opisuje da je (vremenska) derivacija položaja z brzina v, a derivacija brzine akceleracija a. Osim ovih jednadžbi, u čvoru kočenje nalazi se i ograničenje $v \geq 0$. Ono opisuje dokle se gibanje u skladu s jednadžbama smije odvijati (to odgovara i predodžbi o kretanju vlaka: samo kočenjem smjer kretanja ne može se mijenjati). Sustav nikako ne smije ostati u čvoru (stanju) u kojem ograničenje nije ispoštovano, već mora prijeći u drugo. Svejedno, to ne znači da u čvoru ostaje sve dok ograničenje vrijedi: može ga napustiti čim je zadovoljen uvjet na nekom od izlaznih bridova. (Da naglasimo još jednom – to je samo nužan uvjet. Kad je uvjet na bridu zadovoljen, tada sustav smije promijeniti čvor, ali i ne mora. Kada će se čvor zaista promijeniti, nije precizirano, radi se o nedeterminističkom sustavu.) Na Slici 1 uvjet za prelazak iz čvora kočenje u čvor ubrzavanje je da brzina padne ispod 1. To vidimo iznad brida. Ispod brida nalazi se pridruživanje $a := a + 5$. Za vrijeme promjene čvora akceleracija a mijenja svoju vrijednost, a gibanje u čvoru ubrzavanje se nastavlja uz tako definiranu akceleraciju. Takve diskretne transformacije nazivat ćemo skokovima. Za prelazak iz ubrzavanja u kočenje zahtijevamo da*

položaj pređe neku unaprijed zadanu veličinu i tada se događa skok akceleracije koja postaje –b. (Primijetimo da nikakvo ograničenje unutar čvora ubrzavanje ne postoji, tako da sustav u tom čvoru može ostati po volji dugo.) Početni čvor sustava je ubrzavanje (što je naznačeno riječju start). Kako bismo u potpunosti opisali automat, moramo zadati početne vrijednosti varijabli z i v, početnu vrijednost akceleracije a, kao i konstante s i b.



Slika 1. Hibridni automat za (pojednostavljeni) sustav upravljanja vlakom

U ovako definiranom automatu lako je pronaći manjkavosti: ako je konstanta b dovoljno velika ($b > 5$), u čvoru ubrzavanje akceleracija će biti negativna (i nikakvog ubrzanja neće biti). Štoviše, padne li brzina ispod nule, čak i ako je uvjet za prelazak u čvor kočenje ispunjen, sustav će biti blokiran u čvoru ubrzavanje i nikada više neće moći iz njega izaći (zbog ograničenja $v \geq 0$ unutar čvora kočenje koje mora biti zadovoljeno i na ulasku u čvor).

Zbog jednostavnosti modela, lako smo uočili greške i lako bismo odabrali dobre parametre. Ipak, zanima nas kako u općenitom, složenijem, slučaju biti siguran da sustav neće ostati blokiran u nekom čvoru. Također, kako znati koji su parametri odgovarajući, a koji nisu? Koja je maksimalna brzina koju će sustav razviti i do koje točke će najdalje doći? Odgovore na ta pitanja pružit će nam analiza hibridnih sustava pomoću logike dL.

U sljedećem primjeru neformalno uvodimo pojam *hibridnog programa*.

Primjer 2.2 *Zamislimo loptu koja je ispuštena s određene visine H (na primjer košarkaška lopta kad se provjerava je li dobro napumpana). Taj vrlo jednostavan primjer nije pravi hibridni sustav, ali se može promatrati kao takav (u njemu je logička komponenta zemlja koja odbija loptu i mijenja joj smjer). Na Slici 2 je hibridni automat za tu loptu. Dakle, lopta se ispušta iz zraka i giba se pod utjecajem gravitacije ($h'' = -g$). Zadano je ograničenje $h \geq 0$, pa smo kombinacijom spomenutog ograničenja i uvjeta na bridu osigurali da se promjena stanja događa točno onda kad je $h = 0$. Dok se stanje mijenja, brzina mijenja smjer, uz $0 < c < 1$, faktor prigušenja. Isti ovaj sustav, osim hibridnim automatom, mogli bismo opisati sljedećim hibridnim programom.*

Hibridni program 1

$(h' = v, v' = -g \;\&\; h \geq 0$

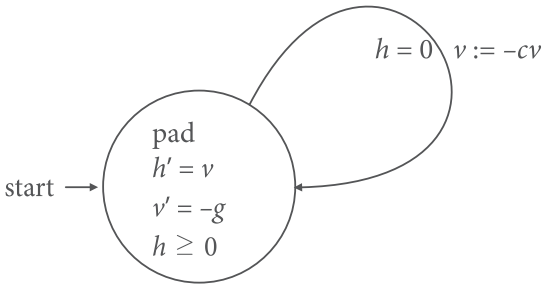
if $(h = 0)$ then

$v := -cv$

endif

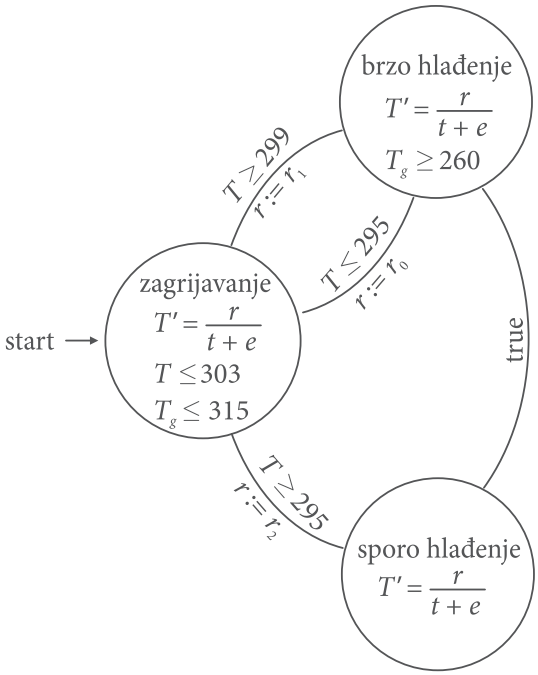
$)^*$

Prva linija programa opisuje kontinuirano gibanje lopte. Ograničenje tog gibanja dano je nakon znaka &. Simbol „točka-zarez" odvaja taj dio od dijela u kojem je test za prijelaz (if ... then). Ako je uvjet zadovoljen, događa se diskretna promjena vrijednosti varijable v. *Na kraju programa nalazi se i regularni operator *, znak da se program može ponoviti po volji mnogo puta. Na ovom primjeru može se pratiti veza između automata i programa, a definicija hibridnog programa doći će u točki koja slijedi.*



Slika 2. Hibridni automat za loptu koja se odbija od zemlje

Primjer 2.3 *Automat prikazan na Slici 3 prikazuje sustav za reguliranje temperature u prostoriji. Sustav se sastoji od jednostavnog uređaja koji može biti ili na maksimalnoj temperaturi (zagrijavanje), ili se samo isključiti i prirodno se hladiti (sporo hlađenje) ili se na neki način iznutra hladiti (brzo hlađenje). Simbol T odnosi se na temperaturu prostorije (u Kelvinima, na primjer), a* T_g *na temperaturu uređaja. Ograničenja za* T_g *odnose se na maksimalnu i minimalnu temperaturu koju uređaj može postići.*



Slika 3. Hibridni automat za grijalicu koja zagrijava sobu

Novo u ovom automatu je da se iz čvora zagrijavanje može prijeći i u brzo hlađenje i u sporo hlađenje, i tu se očituje nova mogućnost nedeterminizma u oblikovanju hibridnih automata.

3 Diferencijalna dinamička logika *dL*

U ovoj točki opisat ćemo sintaksu logike *dL*, te navesti najvažnije činjenice o njoj. Ovdje nećemo dati formalne definicije pojmova jer bi to zahtijevalo dosta prostora. Sve detalje možete vidjeti u [4], odnosno u [2].

Slično kao i kod zadavanja sintakse logike prvog reda (vidi primjerice u [5]), i ovdje polazimo od skupa individualnih varijabli i signature, zatim gradimo terme i na kraju formule logike. Za razliku od logike prvog reda, u sistemu *dL* definiramo i hibridne programe koji su ravnopravni sintaktički elementi i koji grade formule ove logike.

Alfabet logike *dL* osim prebrojivog skupa V individualnih varijabli i skupa pomoćnih simbola, sadrži prebrojiv skup relacijskih simbola, te skup funkcijskih simbola kojih ima koliko i realnih brojeva. Skup funkcijskih simbola sadrži dva istaknuta

nul–mjesna simbola koje označavamo 0 i 1, a sadrži još i neprebrojiv skup $\sum$ nul–mjesnih funkcijskih simbola čije elemente nazivamo varijable stanja. Zatim pretpostavljamo da su u alfabetu dvomjesni funkcijski simboli koje redom označavamo sa $+, \cdot, -, /$.[3] U alfabetu logike *dL* još se nalaze logički simboli $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall$ i $\exists$, te simboli hibridnih programa $:=, ', \&, ?, \cup$ i $*$. Pojam terma definira se na standardni način. U daljnjem tekstu sa $Trm(\sum, V)$ označavamo skup svih terma. Kako bismo mogli definirati pojam formule logike *dL*, prvo moramo definirati pojam hibridnog programa. U daljnjem tekstu sa $Fml_{FO}(\sum, V)$ označavamo skup svih formula logike prvog reda nad skupovima varijabli $\sum$ i V.

Sada ćemo dati definiciju hibridnih programa. Radi se o modelima za hibridne sustave koji objedinjuju kontinuiranu komponentu (izraženu kroz diferencijalne jednadžbe, dio hibridnih programa) i logičku, upravljačku komponentu (koja se ostvaruje kombiniranjem hibridnih programa operacijama $\cup, ^*$ i ;, testiranjem vrijednosti i promjenom vrijednosti varijabli).

Skup **hibridnih programa** $HP(\sum, V)$ je najmanji skup takav da vrijedi:

1. ako su $x_1, x_2, ..., x_n \in \sum$ međusobno različite varijable stanja, a $\theta_i \in Trm\ (\sum, V)$ term za $1 \leq i \leq n$ onda je $(x_1 := \theta_1, x_2 := \theta_2, ..., x_n := \theta_n) \in HP(\sum, V)$.

   Izraz $(x_1 := \theta_1, x_2 := \theta_2, ..., x_n := \theta_n)$ nazivamo **diskretni skok**.

2. neka je $x_i \in \sum$ varijabla stanja, a $\theta_i \in Trm\ (\sum, V)$ za $1 \leq i \leq n$.

   Ako je $\chi \in Fml_{FO}(\sum, V)$, onda je $(x_1' = \theta_1, x_2' = \theta_2, ..., x_n' = \theta_n \,\&\, \chi) \in HP(\sum, V)$.

   Izraz $(x_1' = \theta_1, x_2' = \theta_2, ..., x_n' = \theta_n \,\&\, \chi)$ nazivamo **neprekidna evolucija**.

3. ako je $\chi \in Fml_{FO}(\sum, V)$, onda je $(?\chi) \in HP(\sum, V)$. Izraz $(?\chi)$ zove se **test.**

4. ako su $\alpha, \beta \in HP(\sum, V)$, onda je $(\alpha \cup \beta) \in HP(\sum, V)$. Izraz $(\alpha \cup \beta)$ nazivamo **nedeterministički izbor**.

5. ako su $\alpha, \beta \in HP(\sum, V)$, onda je $(\alpha; \beta) \in HP(\sum, V)$. Izraz $(\alpha; \beta)$ nazivamo **nizanje**.

6. ako je $\alpha \in HP(\sum, V)$, onda je $(\alpha^*) \in HP(\sum, V)$. Izraz (α^*) nazivamo **nedeterminističko ponavljanje**.

[3]Skup funkcijskih simbola sadrži i prebrojivo mnogo tzv. Skolemovih funkcijskih simbola. Budući da u ovom članku nećemo dati niti jedan dokaz, odlučili smo ne definiriti pojmove kao što su Skolemov term i Skolemova funkcija. Sve detalje o tome možete primjerice vidjeti u [1].

U napomeni koja slijedi navest ćemo intuitivno značenje šest izraza iz definicije 3 (nazivat ćemo ih operacijama).

Napomena 3.1 *U ovoj napomeni dajemo intuitivno značenje operacija koje čine hibridne programe.*

Prva operacija o kojoj govorimo je diskretni skok, odnosno izraz koji ima oblik: $(x_1 := \theta_1, x_2 := \theta_2, ..., x_n := \theta_n)$. *Željeni efekt diskretnog skoka je istovremeno mijenjanje interpretacije varijabli* x_1 *u odgovarajuće* θ_i. *(Naglasak na riječi istovremeno odnosi se na to da se svi* θ_i, *koje mogu ovisiti o nekom* x_j, *izvrijedne na početku, prije promjene bilo kojeg* x_j.*)*

Slijedi intuitivni opis operacije neprekidna evolucija. To je operacija koja opisuje kontinuirano mijenjanje hibridnog sustava. Njen formalni oblik je: $(x_1' = \theta_1, x_2' = \theta_2, ..., x_n' = \theta_n \,\&\, \chi)$. *Ovdje označava vremensku derivaciju varijable* x_i, *tj.* $\frac{dx_i(t)}{dt} = \theta_i(t)$. *Neprekidna evolucija označava promjenu vrijednosti varijabli* $x_1, x_2, ..., x_n$ *za različite vrijednosti t u skladu sa zadanim diferencijalnim jednadžbama, a početne vrijednosti su vrijednosti varijabli* $x_1, ..., x_n$ *neposredno prije početka neprekidne evolucije. Evolucija se može odvijati sve dok vrijedi* χ.

Operacija test omogućuje nam grananje programa. Semantika namijenjena operaciji test oblika $(?\chi)$ *je nepromijenjeno izvršavanje hibridnog programa ako formula* χ *vrijedi u danom stanju. U suprotnom, program je blokiran i ne može nastaviti izvršavanje.*

Namijenjena semantika operacije nedeterministički izbor, čiji formalni zapis ima oblik: $(\alpha \cup \beta)$, *jest da se nedeterministički odabire hoće li hibridni sustav slijediti program* α *ili* β.

Za operaciju nizanje, koja formalno zapisano izgleda: $(\alpha; \beta)$, *želimo da bude potpuno deterministička; program* β *započinje s izvršavanjem tek nakon što je program* α *završio.*

Posljednja operacija koju opisujemo je nedeterminističko ponavljanje. Oznaka za nju motivirana je sličnom operacijom iz regularnih izraza: (α^*). *Namijenjena joj je semantika da se hibridni program* α *izvršava n puta, gdje je* $n \in \mathbb{N}_0$ *nedeterministički odabran broj.*

Sada dajemo neke primjere hibridnih programa.

Primjer 3.1 *Promotrimo vrlo jednostavan hibridni program koji opisuje gibanje vlaka po jednadžbi $z' = v$, $v' = a$.*

Hibridni program 2

$((a := -b) \cup (?v < 8; a := A)); z' = v, v' = a)$

Na početku sustav nedeterministički odabire (što je određeno simbolom $\cup$) hoće li postaviti akceleraciju na $-b$, akceleraciju kočenja, $(a := -b)$ ili će, samo ako je brzina manja od 8, postaviti akceleraciju na pozitivnu vrijednost $A((?v < 8; a := A))$. Nakon toga se sustav ravna prema početnoj diferencijalnoj jednadžbi po volji dugo. Operacija nedeterminističkog ponavljanja na kraju (simbol *) označava da se cijeli postupak može ponoviti po volji mnogo puta.

Primjer 3.2 *Prisjetimo se hibridnog automata iz Primjera 2.3 koji modelira rad upravljačke jedinice sobne grijalice. Hibridni program za njega izgleda ovako:*

Hibridni program 3

$$((T' = \frac{r}{t+e} \& T \le 203 \wedge T_g \le 215); (?T \ge 199; r := r_1) \cup$$

$$\cup (?T \ge 195; r := r_2; T' = \frac{r}{t+e}); (T' = \frac{r}{t+e} \& T_g \ge 160); ?T \ge 199; r := r_1)^*$$

Primjer 3.3 *U ovom primjeru ilustriramo kako se neke od klasičnih naredbi za kontrolu toka programa izražavaju pomoću hibridnih programa. Redom dajemo hibridne programe za operacije if-then-else, if-then, while-do, repeat-until (koje su standardne u svim programskim jezicima i čije je značenje jasno) te abort. Operacija abort (pojavljuje se npr. u programskom jeziku Java) prekida izvršavanje programa.*

- *provjera if χ then α else β izražava se hibridnim programom $(?\chi; \alpha) \cup (?\neg\chi; \beta)$. Iako se ovdje prividno radi o nedeterminističkom izboru, nije tako: točno jedan od uvjeta χ ili $\neg\chi$ može vrijediti pa je sustav prisiljen odabrati njega.*
- *provjera if χ then α izražava se hibridnim programom $(?\chi; \alpha) \cup (?\neg\chi)$.*

  Primijetimo da je bilo bitno dodati i dio nakon znaka $\cup$, tj. $(?\neg\chi)$. U suprotnom bi sustav, kad χ ne vrijedi, samo stao i program se ne bi mogao izvršiti.
- *petlja while χ do α izražava se hibridnim programom $(?\chi; \alpha)^*; ?\neg\chi$.*
- *petlja repeat α until χ izražava se hibridnim programom $\alpha(?\neg\chi; \alpha)^*; ?\chi$.*
- *operacija abort može se izraziti hibridnim programom ?false.*

Zaista, vrijedi i više: hibridnim programima može se simulirati rad Turingovog stroja. Dokaz te tvrdnje dan je u [4].

Formule logike *dL* definiraju se na isti način kao formule logike prvog reda[4]. U definiciji formula logike *dL* imamo još sljedeći dodatak: ako je F neka formula logike prvog reda i α neki hibridni program, tada su $[\alpha]F$ i $\langle\alpha\rangle F$ formule logike *dL*[3].

Ovdje nećemo formalno definirati, a niti posebno objašnjavati pojam semantike sistema *dL*. Sada formalno definiramo račun logike *dL*.

Definicija 3.1 ***Račun logike*** *dL zadan je sa sljedeće 32 sheme pravila izvoda:*

$$(\neg r)\ \frac{\phi \vdash}{\vdash \neg\phi} \qquad (\vee r)\ \frac{\vdash \phi, \psi}{\phi \vee \psi} \qquad (\wedge r)\ \frac{\vdash \phi \quad \vdash \psi}{\vdash \phi \wedge \psi} \qquad (\to r)\ \frac{\phi \vdash \psi}{\vdash \phi \to \psi} \qquad (\mathrm{ax})\ \frac{*}{\phi \vdash \phi}$$

$$(\neg l)\ \frac{\vdash \phi}{\neg\phi \vdash} \qquad (\vee l)\ \frac{\phi \vdash \quad \psi \vdash}{\phi \vee \psi \vdash} \qquad (\wedge l)\ \frac{\phi, \psi \vdash}{\phi \wedge \psi \vdash} \qquad (\to l)\ \frac{\vdash \phi \quad \psi \vdash}{\phi \to \psi \vdash} \qquad (cut)\ \frac{\vdash \phi \quad \phi \vdash}{\vdash}$$

$$(\langle ; \rangle)\ \frac{\langle\alpha\rangle\langle\beta\rangle\phi}{\langle\alpha;\beta\rangle\phi} \qquad (\langle {}^{*n} \rangle)\ \frac{\phi \vee \langle\alpha\rangle\langle\alpha^*\rangle\phi}{\langle\alpha^*\rangle\phi} \qquad (\langle := \rangle)\ \frac{\phi^{\theta_1 \ldots \theta_n}_{x_1 \ldots x_n}}{\langle x_1 := \theta_1, \ldots, x_n := \theta_n \rangle}$$

$$([;])\ \frac{[\alpha][\beta]\phi}{[\alpha;\beta]\phi} \qquad ([{}^{*n}])\ \frac{\phi \vee [\alpha][\alpha^*]\phi}{[\alpha^*]\phi} \qquad ([:=])\ \frac{\phi^{\theta_1 \ldots \theta_n}_{x_1 \ldots x_n}}{[x_1 := \theta_1, \ldots, x_n := \theta_n]}$$

$$(\langle \cup \rangle)\ \frac{\langle\alpha\rangle\phi \vee \langle\beta\rangle\phi}{\langle\alpha \cup \beta\rangle\phi} \qquad (\langle ? \rangle)\ \frac{\chi \wedge \psi}{\langle ?\chi\rangle\psi} \qquad (\langle' \rangle)\ \frac{\exists t \geq 0((\forall t : 0 \leq \tilde{t} \leq t\langle \mathcal{S}_i \rangle\chi) \wedge \langle \mathcal{S}_t \rangle\phi)}{\langle x_1' = \theta_1, \ldots, x_n' = \theta_n \,\&\, \chi\rangle\phi}$$

$$([\cup])\ \frac{[\alpha]\phi \wedge [\beta]\phi}{[\alpha \cup \beta]\phi} \qquad ([?])\ \frac{\chi \to \psi}{[?\chi]\psi} \qquad ([']) \frac{\exists t \geq 0((\forall t : 0 \leq \tilde{t} \leq t[\mathcal{S}_i]\chi) \to [\mathcal{S}_t]\phi)}{[x_1' = \theta_1, \ldots, x_n' = \theta_n \,\&\, \chi]\phi}$$

$$(\forall r)\ \frac{\vdash \phi(s(X_1, \ldots, X_n))}{\vdash \forall x \phi(x)} \qquad (\exists r)\ \frac{\vdash \phi(X)}{\vdash \exists x \phi(x)}$$

$$(\exists l)\ \frac{\phi(s(X_1, \ldots, X_n)) \vdash}{\exists x \phi(x) \vdash} \qquad (\forall l)\ \frac{\phi(X) \vdash}{\forall x \phi(x) \vdash}$$

$$(i\forall)\ \frac{\vdash QE(\forall X(\phi(X) \vdash \psi(X)))}{\phi(s(X_1, \ldots, X_n)) \vdash \psi(s(X_1, \ldots, X_n))} \qquad (i\exists)\ \frac{\vdash QE(\exists X \bigwedge_i (\Phi_i \vdash \Psi_i))}{\Phi_1 \vdash \Psi_1 \ \ldots \ \Phi_n \vdash \Psi_n}$$

$$([]gen)\ \frac{\vdash \forall^\alpha(\phi \to \psi)}{[\alpha]\phi \vdash [\alpha]\psi} \qquad (\langle\rangle gen)\ \frac{\vdash \forall^\alpha(\phi \to \psi)}{\langle\alpha\rangle\phi \vdash \langle\alpha\rangle\psi}$$

$$(ind)\ \frac{\vdash \forall^\alpha(\phi \to [\alpha]\phi)}{\phi \vdash [\alpha^*]\phi} \qquad (con)\ \frac{\vdash \forall^\alpha \forall v > 0(\rho(v) \to \langle\alpha\rangle\rho(v-1))}{\exists v \rho(v) \vdash \langle\alpha^*\rangle\exists v \leq 0 \rho(v)}$$

[4]Atomarna formula logike prvog reda je izraz oblika $R(t_1,..., t_k)$ gdje su t_i termi, a R relacijski simbol. Složene formule dobivamo korištenjem logičkih veznika i kvantifikatora. Detaljnu definiciju možete vidjeti u [5].

Dokaz u sustavu računa logike *dL* je konačan, acikličan usmjeren graf s jedinstvenim vrhom bez roditelja – korijenom. Vrhovi grafa označeni su sekventama (konačnim skupovima formula) tako da je za svaki vrh skup oznaka njegove djece jednak skupu sekvenata iz premise nekog pravila računa logike *dL*, a skup oznaka roditelja te djece mora biti jednak skupu sekvenata konkluzije tog istog pravila.

Za formulu ψ kažemo da je **dokaziva** iz konačnog skupa formula Φ ako je korijen označen sa ψ, a svi listovi (vrhovi bez djece) sekventama iz skupa Φ. Kažemo da je ϕ **teorem** računa logike *dL* ako je korijen označen s ϕ, a svi vrhovi bez djece s (*).

Ovdje nećemo navesti primjer dokaza u računu *dL* , već ćemo to učiniti na kraju sljedeće točke. Želimo još samo naglasiti da za račun *dL* vrijedi teorem adekvatnosti, ali ne i teorem potpunosti. Nepotpunost sistema slijedi iz mogućnosti definiranja prirodnih brojeva u logici *dL* i Gödelovog prvog teorema nepotpunosti.

4 ETCS – European Train Control System

U ovoj točki predstavljamo jednu značajnu primjenu logike *dL*. Logika *dL* dala je važan doprinos u analizi sigurnosti europske željezničke mreže. Zato dajemo kratak opis novouspostavljenog zajedničkog europskog standarda, sustava ETCS, a potom uz pomoć logike *dL* analiziramo sigurnosna svojstva na pokaznom primjeru.

Mreža željeznica u Europi nije razvijena kao *europska* mreža, već kao više mreža različitih država. Zbog različitih standarda, pri prijelasku (nekadašnjih, za zemlje EU) državnih granica bilo je potrebno promijeniti lokomotivu i strojovođu, što je značajno usporavalo promet. Usto, razvojem brzih vlakova, signalizacija semaforima na pruzi postala je nedovoljno dobra i teško uočljiva.

Sustav sigurnosti željeznice bio je ranije ostvaren pomoću takozvanih *fiksnih blokova* – dijelova pruge koje je mogao zauzeti samo jedan vlak ili podignuta rampa u danom trenutku. Uz pretpostavku o korektnom funkcioniranju pružnih signala, takav sustav ostvaruje sigurnost, ali i bitno usporava promet (zbog statičnosti blokova).

Zbog svega navedenog, EU je 1991. započela s uvođenjem novog željezničkog standarda za signalizaciju i sigurnost koji će omogućiti veću razinu automatizacije prometa, nazvanog ETCS. Sustav se sastoji od četiri razine (0 - 3), a implementiran je tek djelomično (do razine 2). U nastavku opisujemo svaku razinu standarda:

- razina 0
  Odnosi se na vlakove opremljene ETCS opremom koji voze po pruzi bez takve opreme. Tada oprema nadzire samo maksimalnu brzinu vlaka, dok strojovođa upravlja vlakom prema tradicionalnim signalima uz prugu.

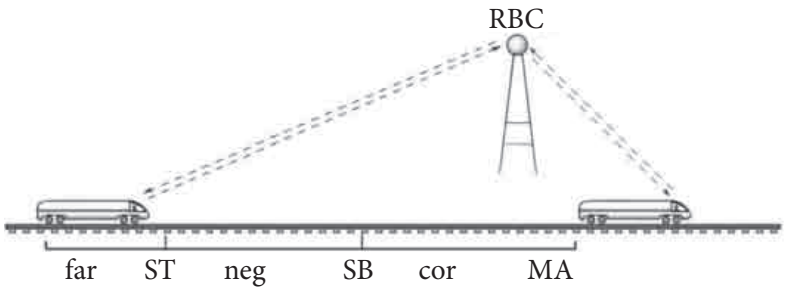
- razina 1
  Na ovoj razini se ETCS oprema nadograđuje na postojeću, transformira signale i šalje ih sustavu koji postoji u vlaku. Sustav na temelju primljenih podataka (koji

uključuju odobrenje prolaska i rutu) izračunava brzinu te po potrebi vrijeme i snagu kočenja. Budući da se podatci prenose samo prijelaskom preko mjesta gdje je signalizacija instalirana, oni se ne dobivaju stalno, već tek po prijelasku tzv. *eurobalize*. Razina 1 omogućava prijelazak granice između država u kojima isti signali imaju različita značenja bez straha.

- razina 2
  Informacije od pruge do vlaka i obrnuto kontinuirano se prenose radio valovima. Na taj način strojovođa u svakom trenutku u svojoj kabini može vidjeti dokle ima slobodan prolazak. Ipak, blokovi i dalje ostaju fiksni.

- razina 3
  Ova se razina još razvija, a uključuje dinamičke blokove za prolazak.

U Hrvatskoj se razina 1 implementira na pruzi Vinkovci – Tovarnik.



Slika 4. Ilustracija ETCS sustava sigurnosti uz korištenje dinamičkih blokova

Promotrimo idealizirani prikaz ETCS sustava razine 3 i pokušajmo postaviti pitanja o njegovim ključnim sigurnosnim pitanjima jezikom logike *dL*. Na Slici 4 dan je shematski prikaz protokola. Svi vlakovi s jednog šireg područja informacije primaju od kontrolne jedinice (RBC – radio block controller) zadužene za to područje. RBC svakom vlaku šalje *dozvolu za prolaz* u obliku najudaljenije točke koju smije dosegnuti, koja je na Slici 4. označena s MA (movement authority). Po primanju te informacije, sustav u vlaku računa od koje točke i koliko treba početi usporavati da bi ostao unutar dopuštenog područja. Dok se vlak nalazi u području označenom s *far*, može mirno voziti. U točki ST već je dopušteno blizu granici područja za koje ima dozvolu za prolaz pa šalje zahtjev za dobivanjem nove dozvole. Ukoliko je ne dobije, u točki SB vlak počinje usporavati (razlog za neproduljivanje dozvole za kretanje može biti što je neki drugi vlak već dobio dozvolu za isti dio pruge ili što na tom dijelu pruge rampa još nije spuštena). Ako kontrolne jedinice daju dozvole za prolaz korektno (tako da se prolasci vlakova ili podignute rampe međusobno isključuju), onda je za sigurnost dovoljno pokazati da svaki pojedini vlak ostaje unutar granica u kojima ima dozvolu za prolaz. Kako to specificirati jezikom logike *dL*, navodimo u sljedećem primjeru.

Primjer 4.1 *Pretpostavimo da se vlak trenutno nalazi u točki z, a da je dobio dozvolu za prolaz do točke m. Trenutna brzina kretanja vlaka je v. Točku SB (od koje vlak mora početi kočiti) predstavljamo sigurnosnom udaljenošću s koja predstavlja udaljenost točke SB od točke m. Uz ove oznake želimo provjeriti hoće li vlak uvijek ostati unutar dopuštenog područja kretanja. Ako ograničenja dana na parametre označimo sa ψ, program koji opisuje komunikaciju vlaka s kontrolnom jedinicom s ctrl, a program koji opisuje kretanje vlaka nakon dobivene informacije s drive, onda je traženo sigurnosno svojstvo opisano sljedećom dL formulom:*

$$\psi \rightarrow (ctrl; drive)^* \, z \leq m \tag{1}$$

Ona govori da, uz dan raspon parametara iz ψ, nakon svakog niza uzastopnih izvođenja programa ctrl i drive vlak ostaje unutar dopuštenih granica. Ključno je, naravno, kako ćemo definirati programe *ctrl i drive*. Program *ctrl* zadajemo sa:

$$ctrl \equiv (?m - z \leq s; a := -b) \cup (?m - z > s; a := A)$$

U ovom se programu radi o jednoj if-then-else provjeri: naime, ako je vlak već prešao točku SB ($m - z \leq s$), on počinje kočiti akceleracijom $-b$. U suprotnom, postavlja akceleraciju na pozitivnu vrijednost A. (Kontrola je ovdje prilično gruba: vlak ili usporava maksimalno, ili ubrzava maksimalno. Ipak, lako je zamisliti kako bi se ova formula profinila s tim da vlak bira proizvoljnu vrijednost usporavanja ili ubrzavanja unutar intervala $[-b, 0]$ odnosno $[0, A]$.) Nakon postavljanja akceleracije u ctrl, slijedi program drive koji se može dizajnirati ovako:

$$drive \equiv \tau := 0; (z' = v, v' = a, \tau' = 1 \& v \geq 0 \wedge \tau \leq \varepsilon)$$

Ovaj program opisuje kretanje prema jednadžbi $z'' = a$ koje traje sve dok je brzina v veća od nule. Novo uvedena varijabla τ služi ograničavanju trajanja nenadzirane vožnje: na početku programa drive ona se inicijalizira na 0, a vožnja može trajati sve dok je $\tau < \varepsilon$. Vrijednost konstante definirana je unutar ψ i označava koliko često vlak prima informacije od kontrolne jedinice. Kako bismo mogli provesti realističnu analizu, moramo pretpostaviti da je ε najveći mogući i dokazati da i uz takav ε formula (1) vrijedi. Parametri zadani u ψ neće se, kao kod provjere modela (model checking) zadavati unaprijed pa onda provjeravati, već se pomoću računa logike dL određuju sve vrijednosti parametara za koje formula vrijedi.

Uz pretpostavku da je vlak prešao u fazu kočenja ($a := -b$, postavljeno u programu ctrl i zatim pokrenut program drive), zanima nas uz koje uvjete je svejedno moguće da vlak pređe granicu m. Pojednostavljeno, zanima nas hibridni program $drive' \equiv z' = v, v' = -b$ i pitanje vrijedi li (i pod kojim uvjetima) formula $\phi \equiv v \geq 0 \wedge z < m \rightarrow drive'\, z > m$. Analiza se provodi pomoću pravila računa dL.

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{v \geq 0, z < m \vdash T \geq 0}
\quad
\dfrac{}{v \geq 0, z < m \vdash -\frac{b}{2}T^2 + vT + z > m}(:=)
}{v \geq 0, z < m \vdash \langle z := -\frac{b}{2}T^2 + vT + z \rangle z > m}
}{v \geq 0, z < m \vdash T \geq 0 \wedge \langle z := -\frac{b}{2}T^2 + vT + z \rangle z > m}\wedge r
}{v \geq 0, z < m \vdash \exists t \geq 0 \langle z := -\frac{b}{2}t^2 + vt + z \rangle z > m}\exists r
}{v \geq 0, z < m \vdash \langle z' = v, v' = -b \rangle z > m}\langle' \rangle
}{\vdash v \geq 0 \wedge z < m \rightarrow \langle z' = v, v' = b \rangle z > m}\rightarrow r, \wedge l
$$

Analizu započinjemo odozdo. Tu izričemo: moguće je dokazati da postoji situacija u kojoj će vlak, iako koči i nije još prešao točku m, nakon nekog vremena prijeći točku m i tako izići iz sigurnog područja. Prvo se primjenom pravila →r i ∧l početna tvrdnja preinačuje pa se u antecedenti nalaze uvjeti, a u konzekventi neprekidna evolucija položaja z. Pravilo <'> zamjenjuje diferencijalnu jednadžbu njenim rješenjem. Egzistencijalni kvantifikator ne možemo eliminirati primjenom pravila QE jer je kvantificirana formula modalna (sadrži diamond operator). Ipak, pravilo ∃r nam omogućuje da uvedemo varijablu T kao svjedoka za ∃t. Pravilo ∧r grana stablo dokaza u dvije grane. Nakon što smo pravilom := eliminirali modalnost iz desne grane, možemo primijeniti i ∃, spojiti obje grane u kojima se pojavljuje T i eliminirati kvantifikator. Konačno, uz ∃r, ∧l dolazimo do $v \geq 0, z < m \quad v^2 > 2b(m - z)$*. To nam govori da će se neželjeni scenarij (prelazak točke m) dogoditi ako kvadrat brzine prijeđe umnožak dvostruke akceleracijske konstante i udaljenosti do točke m.*

Iz ovog primjera vidimo da se upotrebom računa lakše dolazi do puno općenitijih zaključaka nego što smo to mogli ranije.

Još jednom na kraju naglasimo da je formalne dokaze u računu sistema *d*L moguće provoditi na računalu koristeći program KeYmaera koji je kreirao A. Platzer (vidi [3]).

Literatura

1. Encyclopedia of Math, natuknica *Skolem functions*, http://www.encyclopediaofmath.org/index.php/Skolem_function, (srpanj 2013.)
2. I. Gavran, Logička analiza hibridnih sustava, diplomski rad, PMF–MO, Zagreb, 2013. http://web.math.pmf.unizg.hr/ vukovic/Diplomski-radovi
3. KeYmaera, sustav za automatsku analizu hibridnih sustava, verzija 3.3, dostupno na http://symbolaris.com/info/KeYmaera.html, (rujan 2013.)
4. A. Platzer, *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*, Springer, Berlin, 2010.
5. M. Vuković, *Matematička logika*, Element, Zagreb, 2009.