Sveučilište u Zagrebu
Prirodoslovno-matematički fakultet
Matematički odsjek

Mateo Dujić i Laura Horvat

Algoritam za provjeru mješovito periodičkih modela linearne temporalne logike

Zagreb, 2022.

Ovaj rad izrađen je na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta pod vodstvom prof. dr. sc. Mladena Vukovića i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2021./2022.

Sadržaj

1 Uvod

Problem provjere modela neke modalne logike za danu formulu provjerava njenu istinitost (u smislu promatrane logike). Logika prvog reda nije odlučiva po Churchovom teoremu, što je dokazano u [15], a problem ispunjivosti za logiku sudova je NP-potpun po Cook-Levinovom teoremu (*Theorem 1.8.14.* u [11]). Uz to, nisu sve modalne logike odlučive, a njihov problem ispunjivosti formula često je vrlo složen. Provjera proizvoljnog modela linearne temporalne logike nije računski dohvatljiva, no u ovome se radu bavimo problemom provjere konačnih i beskonačnih mješovito periodičkih modela te logike, koji je efikasno rješiv. Ovaj problem jedan je od ključnih problema "formalne verifikacije", o čemu se može pročitati u [12], a pojavljuje se i u dosta drugih područja.

1.1 Jezik linearne temporalne logike

Temporalna logika način je formalizacije razmišljanja i promatranja događaja u vremenu. U ovome ćemo se radu koncentrirati na posebnu vrstu temporalne logike - **linearnu temporalnu logiku** (u daljnjem tekstu pišemo kratko LTL). Čitatelja zainteresiranog za općenitu temporalnu logiku i njezine primjene upućujemo na [1, 9, 16].

Kako bismo LTL uopće mogli proučavati, potrebno je definirati njezin jezik. Temelj na kojem počiva svaki jezik njegov je **alfabet**, odnosno proizvoljan neprazan skup **znakova**, konačnim nizanjem kojih gradimo **riječi**.

Definicija 1.1. Alfabet linearne temporalne logike *unija je skupova* A_1, A_2, A_3, A_4 *i* A_5, *pri čemu je:*

$A_1 = \{p_0, p_1, p_2, \ldots\}$	*prebrojiv skup čije elemente nazivamo* ***propozicionalnim varijablama***
$A_2 = \{\bot\}$	*logička konstanta* ***laž***
$A_3 = \{\neg, \vee, \wedge, \rightarrow, \leftrightarrow\}$	*skup* ***logičkih veznika***
$A_4 = \{\mathsf{X}, \mathsf{F}, \mathsf{G}, \mathsf{U}, \mathsf{R}\}$	*skup* ***temporalnih operatora***
$A_5 = \{(,)\}$	*skup pomoćnih simbola - zagrade*

Nerijetko se kao oznake za propozicionalne varijable umjesto $p_0, p_1, p_2, \ldots$ koriste slova p, q, r. Također, taj skup označavat ćemo i kao PROP. Operatori X, F, G su unarni, dok na U, R gledamo kao na binarne operatore (to će biti i jasno nakon definicije pojma formule).

Baš kao što u proizvoljnom jeziku nije suvisao svaki niz slova, tako i u jeziku LTL-a ima smisla promatrati samo određene riječi koje nazivamo LTL–formulama, a koje ćemo sada definirati.

Definicija 1.2. Atomarna LTL–formula *je svaka propozicionalna varijabla i logička konstatna* $\bot$. *Pojam* **LTL–formule** *definiramo rekurzivno na sljedeći način:*

a. svaka atomarna LTL–formula je LTL-formula;

b. ako su ϕ *i* ψ *LTL–formule, tada su LTL–formule i sljedeće riječi:* $(\neg\phi)$, $(\phi \vee \psi)$, $(\phi \wedge \psi)$, $(\phi \rightarrow \psi)$, $(\phi \leftrightarrow \psi)$*;*

c. ako su ϕ *i* ψ *LTL–formule, tada su LTL–formule i sljedeće riječi:* $(\mathsf{X}\phi)$, $(\mathsf{F}\phi)$, $(\mathsf{G}\phi)$, $(\phi\mathsf{U}\psi)$, $(\phi\mathsf{R}\psi)$*;*

d. riječ alfabeta je LTL–formula ako je nastala primjenom konačno mnogo koraka a., b. i c.

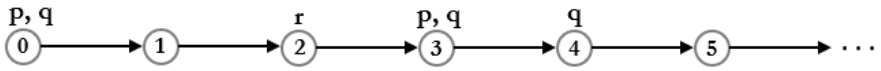
Napomena. U računarstvu se često za opisivanje sintakse jezika koristi Backus-Naurova forma (kratko: BNF) gdje je sintaksa definirana rekurzivno uz pomoć produkcijskih pravila (Usp. Napomena 1.1.4. u [4], Definition 3.1. iz [10]), no u ovome smo je radu definirali po uzoru na [17].

U daljnjem ćemo tekstu umjesto "LTL–formula" pisati samo "formula". Primjere nekih formula navodimo u Primjeru 1.6.

Kako bismo barem donekle osigurali preglednost prilikom pisanja formula, dogovorno logičkim veznicima i temporalnim operatorima pridružujemo prioritete. Najveći prioritet imaju $\neg, \mathsf{X}, \mathsf{F}, \mathsf{G}$, koje slijede U, R. Nakon njih jednak prioritet imaju $\wedge$ i $\vee$ te naposljetku $\rightarrow$ i $\leftrightarrow$.

Za LTL promatramo posebne modele koje sada definiramo.

Definicija 1.3. Linearni model *(u nastavku pišemo kratko:* ***model****) za LTL je svaki niz* $\sigma\colon \mathbb{N} \to \mathscr{P}(\mathrm{PROP})$. *Vrijednosti* $\sigma(j)$ *niza kratko ćemo nazivati* ***stanjima****.*



Slika 1: Primjer restrikcije nekog linearnog modela σ na prvih 6 elemenata skupa $\mathbb{N}$.

Konkretno, za primjer sa slike vrijedi $\sigma(0) = \{p,q\}$, $\sigma(1) = \emptyset$, $\sigma(2) = \{r\}$, $\sigma(3) = \{p,q\}$, $\sigma(4) = \{q\}$, $\sigma(5) = \emptyset$.

Definicija 1.4. *Neka je* σ *model,* $i \in \mathbb{N}$ *te neka su* ϕ *i* ψ *formule. Definiramo* ***relaciju istinitosti*** $\vDash$ *rekurzivno na sljedeći način:*

1. *ako je* ϕ *logička konstanta (*$\phi \equiv \bot$*):* $\sigma, i \nvDash \bot$*, to jest, ne vrijedi* $\sigma, i \vDash \bot$*,*
2. *ako je* ϕ *propozicionalna varijabla, tj.* $\phi \equiv p$ *za neki* $p \in PROP$*, tada definiramo* $\sigma, i \vDash p$ *ako* $p \in \sigma(i)$*,*
3. *inače definiramo rekurzivno:*

$$
\begin{aligned}
\sigma, i \vDash \neg\phi \quad &\text{ako} \quad \sigma, i \nvDash \phi, \\
\sigma, i \vDash \phi \wedge \psi \quad &\text{ako} \quad \sigma, i \vDash \phi \text{ i } \sigma, i \vDash \psi, \\
\sigma, i \vDash \phi \vee \psi \quad &\text{ako} \quad \sigma, i \vDash \phi \text{ ili } \sigma, i \vDash \psi, \\
\sigma, i \vDash \phi \to \psi \quad &\text{ako} \quad \sigma, i \nvDash \phi \text{ ili } \sigma, i \vDash \psi, \\
\sigma, i \vDash \phi \leftrightarrow \psi \quad &\text{ako} \quad (\sigma, i \vDash \phi \text{ i } \sigma, i \vDash \psi) \text{ ili } (\sigma, i \nvDash \phi \text{ i } \sigma, i \nvDash \psi), \\
\sigma, i \vDash \mathsf{X}\phi \quad &\text{ako} \quad \sigma, i+1 \vDash \phi, \\
\sigma, i \vDash \mathsf{F}\phi \quad &\text{ako} \quad \exists j,\ j \geq i, \text{ takav da } \sigma, j \vDash \phi, \\
\sigma, i \vDash \mathsf{G}\phi \quad &\text{ako} \quad \forall j,\ j \geq i, \text{ vrijedi } \sigma, j \vDash \phi,
\end{aligned}
$$

$$\sigma, i \vDash \psi \mathsf{U} \phi \quad \text{ako} \quad \exists j,\ j \geq i, \text{ takav da } \sigma, j \vDash \phi \text{ i}$$
$$\forall k,\ i \leq k \leq j:\ \sigma, k \vDash \psi,$$
$$\sigma, i \vDash \phi \mathsf{R} \psi \quad \text{ako} \quad \forall j,\ j \geq i,\ (\sigma, j \nvDash \phi \text{ i } \sigma, j \vDash \psi) \text{ ili } \exists j,\ j \geq i, \text{ takav da}$$
$$\sigma, j \vDash \phi \text{ i } \forall k,\ i \leq k \leq j:\ \sigma, j \vDash \psi.$$

Relacija istinitosti komutira s preostalim logičkim veznicima.

Definicija 1.5. *Kažemo da je formula ϕ* ***istinita na linearnom modelu*** *σ ako vrijedi $\sigma, 0 \vDash \phi$. To označavamo sa $\sigma \vDash \phi$.*

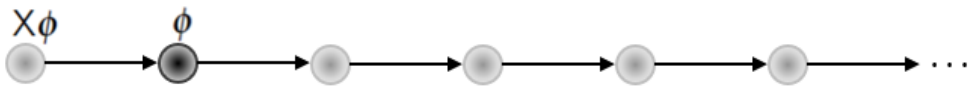
Neka je σ model i $i \geq 0$. Pišemo $\sigma[i, +\infty\rangle$ da bismo označili LTL model dobiven iz σ odbacivanjem prvih i elemenata domene. Dakle, za svaki $j \geq 0$ ima smisla definirati $\sigma[i, +\infty\rangle(j) := \sigma(i + j)$. Analogno definiramo oznaku $\sigma[i, j] := \{\sigma(k), i \leq k \leq j\}$

Napomena. Ponekad se koriste i sljedeće pokrate:

- $\mathsf{F}^{\infty}\phi := GF\phi$,
- $\mathsf{G}^{\infty}\phi := FG\phi$,
- $\phi \mathsf{R} \psi := \neg(\neg\phi \mathsf{U} \neg\psi)$,

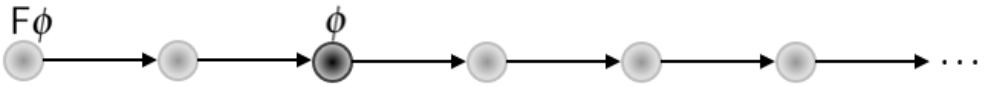
Napomena (O temporalnim operatorima). "Svijet" u kojem promatramo linearnu temporalnu logiku možemo zamisliti kao niz stanja od kojih svako ima jedinstvenog neposrednog sljedbenika. Iz tog razloga smisleno je stanje koje trenutno promatramo na neki način smatrati sadašnjošću, a stanja koja slijede iza njega budućim stanjima.
Tako su uz logičke veznike, sastavni dio jezika svake logičke teorije, ključnu ulogu u sintaksi linearne temporalne logike pronašli temporalni operatori. Radi lakšeg razumijevanja definicija i rezultata koji slijede, navodimo intuitivni opis semantičkog značenja svakog od njih.

X (Oznaka je uzeta iz sljedećih engleskih riječi: "neXt time")
Neka je ϕ proizvoljna LTL-formula. Formula oblika $X\phi$ govori nam da će formula ϕ biti istinita u sljedećem stanju.
Na sljedećoj slici ilustriramo istitinost formule $X\phi$ u stanju 0.



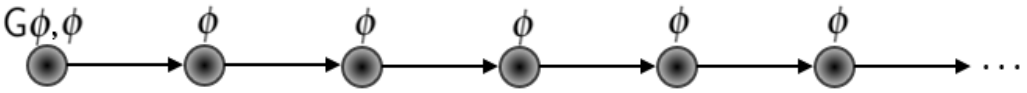
Slika 2: $\sigma, 0 \vDash X\phi$

F (Oznaka ovog operatora je uzeta iz sljedećih engleskih riječi: "sometime in the Future")
Formula oblika $F\phi$ govori nam da je formula ϕ istinita u trenutnom ili će biti istinita u nekom budućem stanju. To ilustriramo na sljedećoj slici.



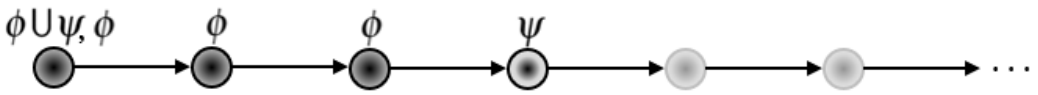
Slika 3: $\sigma, 0 \vDash F\phi$

G (Oznaka ovog operatora je prvo slovo engleske riječi "Globally")
Formula oblika $G\phi$ govori nam da je formula ϕ istinita u trenutnom i u svakom budućem stanju. Na sljedećoj slici dana je ilustracija istinosti formule $G\phi$ u stanju 0.



Slika 4: $\sigma, 0 \vDash G\phi$

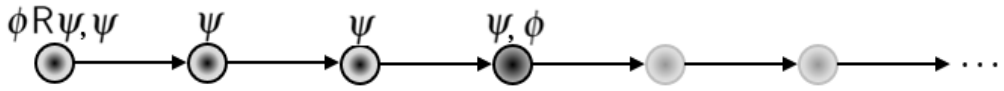
U (Oznaka ovog operatora prvo je slovo engleske riječi "Until")
Za razliku od svojih prethodnika, ovo je binarni temporalni operator. Formula oblika $\phi \mathsf{U} \psi$ govori nam da je formula ϕ istinita u trenutnom i svakom budućem stanju dok ne dođemo do stanja na kojem je istinita formula ψ. U tom trenutku formula ϕ prestaje biti istinita. Jednu takvu situaciju ilustriramo na sljedećoj slici.



Slika 5: $\sigma, 0 \vDash \phi \mathsf{U} \psi$

Primijetimo da formula ψ može biti istinita i u trenutnom stanju.

R (Oznaka ovog operatora je prvo slovo engleske riječi "Release")
Formula oblika $\phi \mathsf{R} \psi$ govori nam da istinitost formule ϕ "oslobađa" formulu ψ. Malo točnije, formula ψ je istinita dok ne dođemo do stanja gdje je istinita formula ϕ. Formulu $\phi \mathsf{R} \psi$ možemo smatrati pokratom formule $\mathsf{G}\psi \vee \psi \mathsf{U}(\phi \wedge \psi)$. Na sljedećoj slici ilustriramo istinitost formule $\phi \mathsf{R} \psi$ na stanju 0.



Slika 6: $\sigma, 0 \vDash \phi \mathsf{R} \psi$

Primjer 1.6. Navedimo nekoliko primjera formula gdje koristimo sugestivne oznake za propozicionalne varijable (npr. uzbuna, alarm, mir..). Iza svake formule navodimo intuitivni opis.

a. $\mathsf{G}(\text{uzbuna} \rightarrow (\text{alarm } \mathsf{U} \text{ mir}))$

Prethodnu formulu možemo čitati ovako: "Uvijek za vrijeme uzbune alarm je aktiviran dok se uzbuna ne zaustavi."

b. $\mathsf{G}(P_{poslana} \rightarrow \mathsf{F}P_{isporucena})$

Intuitvni opis navedene formule je sljedeći: "Kada pošaljemo poruku, ona će u nekom budućem trenutku biti isporučena."

c. $\mathsf{G}(\text{crveno} \rightarrow \neg\mathsf{X}\ \text{zeleno})$

Dana formula izražava sljedeće: "Ako semafor svijetli crveno, ne može neposredno nakon toga zasvijetliti zeleno."

d. $\mathsf{G}(\text{crveno} \rightarrow \mathsf{F}\ \text{zeleno} \wedge (\neg\text{zeleno } \mathsf{U}\ \text{žuto})$

Prethodnu formulu možemo čitati ovako: "Ako semafor svijetli crveno, svjetlo u nekom trenutku postaje zeleno i u tom trenutku prestaje svijetliti žuto."

Lema 1.7. *Za svaku formulu ϕ i svaki linearni model σ te $i \in \mathbb{N}$ vrijedi sljedeća ekvivalencija:*

$$\sigma, i \vDash \phi \ \text{ ako i samo ako } \ \sigma[i, +\infty) \vDash \phi$$

Dokaz. Definiramo složenost formule ϕ kao ukupan broj logičkih i temporalnih operatora koji se u njoj nalaze (oznaka: $s(\phi)$). Tvrdnju dokazujemo indukcijom po složenosti formule.

B.I. Ako je $s(\phi) = 0$, onda ϕ može biti:

1° $\phi \equiv \bot$

Iz definicije vrijedi $\forall \sigma, \forall i$: $\sigma, i \nvDash \bot$. To je ekvivalentno $\sigma[i,+\infty), 0 \nvDash \bot$, to jest $\sigma[0,+\infty) \nvDash \bot$.

2° $\phi \equiv p \in \text{PROP}$

$\sigma[i,+\infty) \vDash p \iff \sigma[i,+\infty), 0 \vDash p \iff p \in \sigma[i,+\infty)(0) \iff p \in \sigma(i+0) = \sigma(i) \iff \sigma, i \vDash p.$

P.I. Pretpostavimo da postoji $n \in \mathbb{N} \setminus \{0\}$ takav da za svaku formulu složenosti strogo manje od n vrijedi tvrdnja.

K.I. Neka je ϕ formula takva da $s(\phi) = n$. Tada ϕ može biti nekog od sljedećih oblika:

1° $\phi \equiv \neg\psi$ za neku formulu ψ očito vrijedi $s(\psi) = n - 1 < n$ pa za nju možemo primijeniti pretpostavku indukcije.

Tvrdnji $\sigma, i \vDash \phi$ ekvivalentno je $\sigma, i \nvDash \psi$, što je po pretpostavci ekvivalentno $\sigma[i,+\infty) \nvDash \psi$, što je pak ekvivalentno $\sigma[i,+\infty) \vDash \phi$.

2° Iz tvrdnje $\phi \equiv \psi \wedge \rho$ slijedi $s(\psi), s(\rho) < n$ pa za te dvije formule možemo primijeniti pretpostavku indukcije.

Tvrdnji $\sigma, i \vDash \psi \wedge \rho$ ekvivalentno je $\sigma, i \vDash \psi$ i $\sigma, i \vDash \rho$, što je po pretpostavci ekvivalentno $\sigma[i,+\infty) \vDash \psi$ i $\sigma[i,+\infty) \vDash \rho$, što je ekvivalentno $\sigma[i,+\infty), 0 \vDash \psi$ i $\sigma[i,+\infty), 0 \vDash \rho$, što je ekvivalentno $\sigma[i,+\infty), 0 \vDash \psi \wedge \rho$, što je ekvivalentno $\sigma[i,+\infty) \vDash \psi \wedge \rho$.

3° Iz tvrdnje $\phi \equiv \mathsf{X}\psi$ slijedi $s(\psi) < n$ pa možemo primijeniti pretpostavku indukcije. Uočimo ovdje suptilnost: pretpostavku ćemo primijeniti za broj $i+1$, ne za i kao inače, no to je ekvivalentno jer ionako pretpostavka vrijedi $\forall i \in \mathbb{N}$. U ovome dijelu dokaza koristimo pomoćnu tvrdnju koja je dokazana poslije ove leme.

Tvrdnji $\sigma, i \vDash \phi$ ekvivalentno je $\sigma, i+1 \vDash \psi$, što je po pretpostavci ekvivalentno $\sigma[i+1,+\infty) \vDash \psi$, što je po pomoćnoj tvrdnji ekvivalentno $\sigma[i,+\infty), 1 \vDash \psi$, što je ekvivalentno $\sigma[i,+\infty), 0 \vDash \mathsf{X}\psi$,

što je ekvivalentno $\sigma[i,+\infty) \vDash \mathsf{X}\psi$, što je ekvivalentno $\sigma[i,+\infty) \vDash \phi$.

4° Iz tvrdnje $\phi \equiv \mathsf{F}\psi$ slijedi $s(\psi) < n$ pa zaključujemo kao u 3°. Uz to, vršimo supstituciju.

Tvrdnji $\sigma, i \vDash \phi$ ekvivalentno je $\exists j \geq i : \sigma, j \vDash \psi$, što je po pretpostavci ekvivalentno, $\exists j \geq i : \sigma[j,+\infty) \vDash \psi$, što je po pretpostavci uz supstituciju $k = j - i$ ekvivalentno $\exists k \geq 0 : \sigma[i+k,+\infty) \vDash \psi$, što je ekvivalentno $\exists k \geq 0 : \sigma[i+k,+\infty), 0 \vDash \psi$ što je po pomoćnoj tvrdnji ekvivalentno $\exists k \geq 0 : \sigma[i,+\infty), k \vDash \psi$, što je ekvivalentno $\sigma[i,+\infty), 0 \vDash \mathsf{F}\psi$, što je ekvivalentno $\sigma[i,+\infty) \vDash \mathsf{F}\psi$, što je ekvivalentno $\sigma[i,+\infty) \vDash \phi$.

5° Tvrdnja $\phi \equiv \mathsf{G}\psi$ dokazuje se kao 4°, samo zamijenimo $\exists$ sa $\forall$.

6° Iz tvrdnje $\phi \equiv \psi \mathsf{U} \rho$ slijedi $s(\psi), s(\rho) < n$ pa zaključujemo kao prije.

Tvrdnji $\sigma, i\phi$ ekvivalentno je $\exists j \geq i : \sigma, j \vDash \rho$ i $\forall k, i \leq k < j : \sigma, k \vDash \psi$, što je po pretpostavci ekvivalentno $\exists j \geq i : \sigma[j,+\infty) \vDash \rho$ i $\forall k, i \leq k < j : \sigma[k,+\infty) \vDash \psi$, što je uz supstituciju $l = j - i$ ekvivalentno $\exists l \geq 0 : \sigma[l+i,+\infty), 0 \vDash \rho$ i
$\forall k, 0 \leq k < l : \sigma[i+k,+\infty), 0 \vDash \psi$, što je po pomoćnoj tvrdnji ekvivalentno $\exists l \geq 0 : \sigma[i,+\infty), l \vDash \rho$ i $\forall k, 0 \leq k < l : \sigma[i,+\infty), k \vDash \psi$, što je ekvivalentno $\sigma[i,+\infty), 0 \vDash \psi \mathsf{U} \rho$, što je ekvivalentno $\sigma[i,+\infty) \vDash \phi$.

Preostale tvrdnje za logičke veznike svode se na svojstvo komutiranja relacije zadovoljivosti s logičkim veznicima.

□

Dokaz sljedeće leme sasvim je analogan dokazu Leme 1.7 pa ga ovdje ispuštamo.

Lema 1.8. *Za svaki model σ i svaku formulu ϕ te za sve $i, j \in \mathbb{N}$ vrijedi sljedeća ekvivalencija:*

$$\sigma\,[i+j,+\infty) \vDash \phi \;\; \textit{ako i samo ako} \;\; \sigma\,[i,+\infty)\,, j \vDash \phi\,.$$

2 Problem logičkog odlučivanja: *Provjera modela*

Osnovno pitanje koje se nameće u bilo kojoj logici je pitanje je li dana formula te logike istinita, ili valjana na danom modelu logike. Ako smo zainteresirani za *algoritamsko* rješenje, ovo se svodi na problem odluke provjere modela (eng. Model checking). **Provjera modela puta** u LTL problem je koji ispituje je li proizvoljna formula ϕ istinita na danom modelu. Da bismo ovaj problem mogli algoritamski rješavati, možemo ga riješiti jedino na konačnom modelu, ali linearni modeli su općenito po definiciji beskonačni. Međutim, važna klasa takvih modela, koji se pažljivom interpretacijom mogu svesti na konačne modele, zovu se *mješovito periodički* modeli.

Definicija 2.1. *Linearni model* $\sigma\colon \mathbb{N} \to \mathscr{P}(\text{PROP})$ *zovemo* ***mješovito periodički*** *ako postoje prirodni brojevi* $k, l > 0$ *takvi da* $\sigma(i) = \sigma(i+l)$ *za svaki* $i \geq k$.

Konačni niz (moguće i prazan) $\sigma(0), \ldots, \sigma(k-1)$ *naziva se* ***pretperiod*** *a konačni niz skupova* $\sigma(k), \ldots, \sigma(k+l)$ *naziva se* ***period*** *modela* σ. *U tom slučaju još kažemo da model* σ *ima* ***duljinu pretperioda*** k *te* ***duljina perioda*** l.

Mješovito periodički linearni model $\sigma\colon \mathbb{N} \to \mathscr{P}(\text{PROP})$ *je* ***konačan*** *ako je skup* $\cup_{i \geq 0} \sigma(i)$ *konačan.*

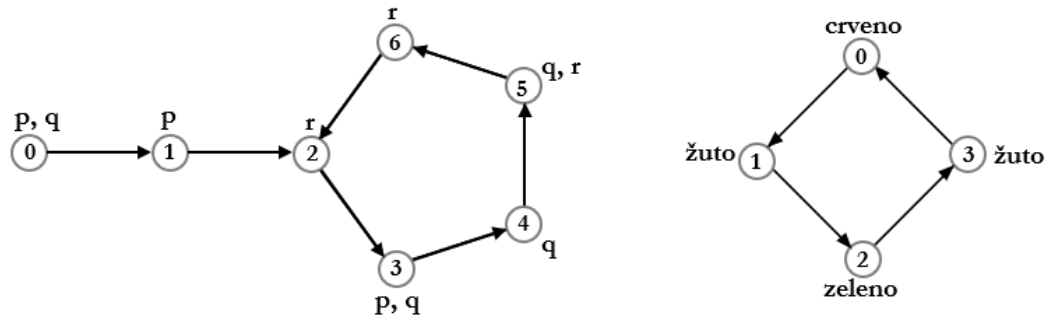
Definiramo ***veličinu*** *konačnog mješovito periodičkog modela kao broj*

$$(i+l) \cdot card(\cup_{i \geq 0} \sigma(i))$$

Mješovito periodički model zadan je konačnim nizom skupova $\Gamma_0, \ldots, \Gamma_{i+l}$ te dva broja i i l, pri čemu je svaki skup Γ_k podskup skupa propozicionalnih varijabli. Model se tada može definirati i ovako:

$$\sigma(j) = \begin{cases} \Gamma_j, & \text{ako } j < i+l \\ \Gamma_k, \text{ gdje je } k \text{ najmanji takav da } k \geq i \text{ i } k \equiv (j-i) \bmod l, & \text{inače.} \end{cases}$$

Primjer 2.2. Na sljedećim slikama dajemo dva primjera mješovito periodičkih modela.



Slika 7: Mješovito periodički modeli

a. Na lijevoj slici pretperiod je niz $\sigma(0), \sigma(1)$, a petlja $\sigma(2), \ldots, \sigma(7)$.
Za formulu $F \equiv \mathsf{XXG}r \vee q$ vrijedi $\sigma \vDash F$, odnosno F je istinita na zadanom modelu. S druge strane, formula $H \equiv \mathsf{X}(q \vee (r \wedge p))$ nije istinita, to jest vrijedi $\sigma \nvDash H$.

b. Na desnoj slici prikazan je model "semafora" na kojem su istinite formule c. i d. iz Primjera 1.6. Skup propozicionalnih varijabli u ovome modelu je {crveno, žuto, zeleno}. U ovom slučaju nema pretperioda, dok je petlja niz $\sigma(0), \ldots, \sigma(4)$.

Problem provjere puta modela LTL–a definiramo kao sljedeći problem odluke:

> za zadani konačan mješovito periodički model σ i zadanu LTL formulu ϕ treba odrediti vrijedi li $\sigma \vDash \phi$.

3 Algoritam

Problem provjere puta mješovito periodičkog modela poseban je slučaj provjere modela za LTL. Za razliku od općenitog problema, koji nije deterministički, pokazat ćemo da ovaj problem jest. Neka je σ konačan mješovito periodički model duljine pretperioda k i duljine perioda n koji je definiran pomoću niza $\Gamma_0, \ldots, \Gamma_{k+n}$ pri čemu $\Gamma_j = \sigma(j)$, gdje je $j \in \{0, 1, \ldots, k+n\}$. Sada opisujemo algoritam.

Neka su $\psi_1, \ldots, \psi_m$ potformule od ϕ poredane po duljini. Preciznije:

- $\psi_m \equiv \phi$,
- ψ_1 je propozicionalna varijabla ili konstanta,
- ako je ψ_a stroga potformula od ψ_b, tada je $a < b$,
- $k \leq |\phi|$.

Za svaki $j \in \{0, \ldots, k+n\}$ definiramo skup formula

$$l[j] := \left\{ \psi \in \{\psi_1, \ldots, \psi_m, \neg\psi_1, \ldots, \neg\psi_m\} \;\middle|\; \sigma, j \vDash \psi \right\}.$$

Najprije ćemo idejno objasniti algoritam, a poslije toga napisati potpuni pseudokod. Algoritam je primjer dinamičkog programiranja, a obrada svakog veznika i operatora jedino zahtijeva razumijevanje njihove definicije. Općenito, svi algoritmi za Provjeru modela su slični, primjerice možemo usporediti algoritam za provjeru modela CTLa (eng. Computation tree logic, vidi [2], str. 39 i dalje). Isto tako, algoritmi za Provjeru modela u Modalnim logikama mogu se pronaći na [3], str. $24-27$. Odmah navodimo da je poznat i rezultat o složenosti algoritma: algoritam za provjeru konačnih modela LTLa uz temporalne operatore koje koristimo je PSPACE-potpun (vidi na [7]).

3.1 Opis algoritma

Unos same formule vrši se preko tekstualnog okvira, nakon čega se ona parsira i ulazi u algoritam. Formula se zatim razlomi na potformule $\psi_1, \ldots, \psi_m$ kao što je opisano u uvodu ovog poglavlja, za koje se od najjednostavnijih prema složenijima provjerava jesu li istinite na pojedinim stanjima. Svaka se potformula klasificira prema svom glavnom vezniku ili operatoru, ili kao propozicionalna varijabla.

Logičku konstantu $\perp$ ne provjeravamo posebno jer po definiciji nije istinita ni na kojem stanju.

Krenimo s opisivanjem provjere istinitosti prema vrsti potformula.
Najjednostavniji je slučaj kada je potformula propozicionalna varijabla. Ona je tada istinita na stanju $\sigma(j)$ ako i samo ako se nalazi u skupu Γ_j. U tom je slučaju dodamo u skup $l[j]$, dok u suprotnom u skup $l[j]$ dodajemo njenu negaciju.

Promotrimo sada logičke veznike.
Potformula može biti oblika $\rho \equiv \neg\psi$. Budući da je ψ stroga potformula od ρ, a samim time i od unesene formule, nužno je već provjerena u algoritmu te se za svaki j ona sama ili njena negacija (upravo ρ) nalazi u $l[j]$. Prema tome, ako ρ nije istinita na j. stanju, odnosno ne nalazi se u skupu $l[j]$, tada u taj skup dodajemo formulu $\neg\rho$. U suprotonom je formula ρ već dodana u skup $l[j]$ nakon provjere potformule ψ.

Sljedeća je mogućnost konjunkcija, tj. $\rho \equiv \psi_1 \wedge \psi_2$. Ta formula je istinita na j. stanju i dodajemo je u skup $l[j]$ ako i samo ako su i ψ_1 i ψ_2 istinite na tom stanju, tj. obje potformule su sadržane u skupu $l[j]$. Inače dodajemo njezinu negaciju.

Za slučaj $\rho \equiv \psi_1 \vee \psi_2$ lakše je provjeriti neistinitost formule ρ na j. stanju. Tada imamo da su na j. stanju neistinite obje potformule ψ_1 i ψ_2. U tom slučaju u skup $l[j]$ dodajemo formulu $\neg\rho$ ako su u $l[j]$ već sadržane formule $\neg\psi_1$ i $\neg\psi_2$. Ako nije takva sluča tada u skup $l[j]$ dodajemo potformulu ρ.

Kao i u prethodnom slučaju, ako je $\rho \equiv \psi_1 \rightarrow \psi_2$, tada je također lakše provjeriti neistinitost na j. stanju. To vrijedi ako se u skup $l[j]$ nalaze već formule ψ_1 i $\neg\psi_2$. U tom slučaju u skup $l[j]$ dodajemo formulu $\neg\rho$. U suprotnom, istinita je formula ρ te je dodamo u skup $l[j]$.

Promotrimo i slučaj $\rho \equiv \psi_1 \leftrightarrow \psi_2$. Tada formulu ρ dodajemo u skup $l[j]$ ako se u njemu već nalaze obje potformule ψ_1 i ψ_2 ili pak su u skupu obje njihove negacije. Inače u skup dodajemo formulu $\neg\rho$.

Preostalo je opisati provjeru istinitosti potformula koje na početku imaju neki temporalni operator.

Promotrimo najprije slučaj kada je potformula oblika $\rho \equiv \mathsf{X}\psi$. Budući da smo naš beskonačni model prikazali konačnim nizom skupova, kao i kod ostalih temporalnih operatora, potrebno je posebno provjeriti granične slučajeve. U slučaju da $\sigma(j)$ nije zadnje stanje prije kraja perioda, provjeravamo nalazi li se formula ψ u skupu $l[j+1]$. Ako se formula ψ nalazi u skup $l[j]$ tada u skup dodajemo formulu ρ. U suprotnom dodajemo formulu $\neg\rho$. Ako je $j = k+n$ tada znamo da je zbog periodičnosti $\sigma(k+n) = \sigma(k)$, pa stoga provjeravamo je li formula ψ istinita na $(k+1)$. stanju, odnosno nalazi li se ta formula u skupu $l[k+1]$.

Sljedeća je mogućnost $\rho \equiv \mathsf{G}\psi$. Za nju najprije provjeravamo period. Ako ψ nije istinita niti na jednom stanju perioda, zbog njegovog ponavljanja formula ρ neće biti istinita niti na jednom stanju modela. U tom slučaju za svaki j u skup $l[j]$ dodajemo formulu $\neg\rho$. Ako pak je formula ψ istinita na svakom stanju perioda, tada dodajemo formulu ρ u svaki skup $l[j]$ koji predstavlja stanja perioda. Još je samo potrebno provjeriti pretperiod. To radimo unatrag s obzirom da ako formula ψ nije istinita na j. mjestu, tada ni formula ρ neće biti istinita ni na nekom mjestu prije j., zaključno s njim samim.

Nadalje, potformula može biti oblika $\rho \equiv \mathsf{F}\psi$. Po definiciji ona je istinita ako je formula ψ istinita na provjeravanom stanju ili na bilo kojem stanju nakon njega. Prema tome ponovno najprije promatramo period. Ako je formula ψ istinita na bilo kojem stanju perioda tada zbog njegovog ponavljanja formula ρ će biti istinita na svakom stanju. U suprotnom je na periodu formula ρ neistinita pa u svaki skup $l[j]$ koji predstavlja stanje ubacujemo njenu negaciju. Ponovno unazad provjeravamo istinitost formule ψ na pretperiodu. Ukoliko je formula istinita na nekom stanju tada će formula ρ biti istinita na tom i svakom prethodnom stanju.

Potformula također može biti oblika $\rho \equiv \psi_1 \mathsf{U} \psi_2$. Najprije provjeravamo pos-

toji li stanje nakon j. na kojem je istinita formula ψ_2 te za svako stanje između j. i pronađenog vrijedi da je istinita ψ_1. Ako postoji, u $l[j]$ dodamo formulu ρ. Ukoliko takvo stanje ne postoji između j. i posljednjeg stanja perioda, a provjeravano j. stanje je unutar perioda, period se ponavlja pa tražimo postoji li stanje između početka perioda i j. na kojem je istinita ψ_2. Ukoliko postoji, za svako stanje između j. i završetka perioda, kao i za ona između početka perioda i pronađenog provjeravamo vrijedi li da je istinita ψ_1. Ako to vrijedi, formula ρ istinita je na j. stanju, stoga ju dodajemo u $l[j]$. U suprotnom, ρ nije istinita na j. stanju pa u $l[j]$ dodamo njenu negaciju.

Posljednji je slučaj koji provjeravamo $\rho \equiv \psi_1 \mathsf{R} \psi_2$. Prema Napomeni 1.1, to je samo pokrata za formulu $\neg(\neg\psi_1 \mathsf{U} \neg\psi_2)$. Prema tome, algoritam provodi isti postupak kao i za prethodno razmotreni operator U, uzimajući u obzir da radimo s negacijama potformula te da istinita formula za operator U znači neistinita formula za R.

Sada navodimo cjelokupan pseudokod algoritma.

Algoritam 1 Provjera $\sigma, 0 \vDash \phi$

1: $\psi_1, \ldots, \psi_m \Leftarrow$ potformule od ϕ poredane po duljini
2: **for** $j \in [0, k+n]$ **do**
3: $\quad l[j] = \emptyset$
4: **end for**
5: **for** $a \in [1, m]$ **do**
6: $\quad$ **if** $\psi_a \in$ PVar **then** ▷ obrađujemo PVar
7: $\quad\quad$ **for** $j \in [0, k+n]$ **do**
8: $\quad\quad\quad$ **if** $\psi_a \in \Gamma_j$ **then**
9: $\quad\quad\quad\quad l[j] = l[j] \cup \{\psi_a\}$
10: $\quad\quad\quad$ **else**
11: $\quad\quad\quad\quad l[j] = l[j] \cup \{\neg\psi_a\}$
12: $\quad\quad\quad$ **end if**
13: $\quad\quad$ **end for**
14:
15: $\quad$ **else if** $\psi_a \equiv \neg\psi_{a_1}$ **then** ▷ obrađujemo $\neg$
16: $\quad\quad$ **for** $j \in [0, k+n]$ **do**
17: $\quad\quad\quad$ **if** $\psi_a \notin l[j]$ **then**
18: $\quad\quad\quad\quad l[j] = l[j] \cup \{\neg\psi_a\}$
19: $\quad\quad\quad$ **end if**
20: $\quad\quad$ **end for**
21:
22: $\quad$ **else if** $\psi_a \equiv \psi_{a_1} \wedge \psi_{a_2}$ **then** ▷ obrađujemo $\wedge$
23: $\quad\quad$ **for** $j \in [0, k+n]$ **do**
24: $\quad\quad\quad$ **if** $\{\psi_{a_1}, \psi_{a_2}\} \subset l[j]$ **then**
25: $\quad\quad\quad\quad l[j] = l[j] \cup \{\psi_a\}$
26: $\quad\quad\quad$ **else**
27: $\quad\quad\quad\quad l[j] = l[j] \cup \{\neg\psi_a\}$
28: $\quad\quad\quad$ **end if**
29: $\quad\quad$ **end for**

30: **else if** $\psi_a \equiv \psi_{a_1} \vee \psi_{a_2}$ **then** ▷ obrađujemo $\vee$
31: **for** $j \in [0, k+n]$ **do**
32: **if** $\{\neg\psi_{a_1}, \neg\psi_{a_2}\} \subset l[j]$ **then**
33: $l[j] = l[j] \cup \{\neg\psi_a\}$
34: **else**
35: $l[j] = l[j] \cup \{\psi_a\}$
36: **end if**
37: **end for**
38:
39: **else if** $\psi_a \equiv \psi_{a_1} \rightarrow \psi_{a_2}$ **then** ▷ obrađujemo $\rightarrow$
40: **for** $j \in [0, k+n]$ **do**
41: **if** $\{\neg\psi_{a_2}, \psi_{a_1}\} \subset l[j]$ **then**
42: $l[j] = l[j] \cup \{\neg\psi_a\}$
43: **else**
44: $l[j] = l[j] \cup \{\psi_a\}$
45: **end if**
46: **end for**
47:
48: **else if** $\psi_a \equiv \psi_{a_1} \leftrightarrow \psi_{a_2}$ **then** ▷ obrađujemo $\leftrightarrow$
49: **for** $j \in [0, k+n]$ **do**
50: **if** $\{\psi_{a_1}, \psi_{a_2}\} \subset l[j]$ **or** $\{\neg\psi_{a_1}, \neg\psi_{a_2}\} \subset l[j]$ **then**
51: $l[j] = l[j] \cup \{\psi_a\}$
52: **else**
53: $l[j] = l[j] \cup \{\neg\psi_a\}$
54: **end if**
55: **end for**

56: **else if** $\psi_a \equiv \mathsf{X}\psi_{a_1}$ **then** ▷ obrađujemo X

57: **for** $j \in [0, k+n]$ **do**

58: **if** $\big((j < k+n$ **and** $\psi_{a_1} \in l[j+1])$ **or**

59: $(j = k+n$ **and** $\psi_{a_1} \in l[k+1])\big)$ **then**

60: $l[j] = l[j] \cup \{\psi_a\}$

61: **else**

62: $l[j] = l[j] \cup \{\neg\psi_a\}$

63: **end if**

64: **end for**

65:

66:

67: **else if** $\psi_a \equiv \mathsf{G}\psi_{a_1}$ **then** ▷ obrađujemo G

68: always = True

69: **for** $j \in [k, k+n]$ **do** ▷ prvo provjeravamo period

70: **if** $\psi_{a_1} \notin l[j]$ **then**

71: always = False

72: **break**

73: **end if**

74: **end for**

75: **if** always = False **then**

76: **for** $j \in [0, k+n]$ **do**

77: $l[j] = l[j] \cup \{\neg\psi_a\}$ ▷ ako na periodu postoji stanje na kojem ψ_{a_1} nije istinita, onda ψ_a nikad nije istina

78: **end for**

79: **else**

80: **for** $j \in [k, k+n]$ **do** ▷ na periodu je ψ_a istinita

81: $l[j] = l[j] \cup \{\psi_a\}$

82: **end for**

83: **for** $j \in [k-1, 0]$ **do** ▷ obrnuti for, provjeravamo unazad je li na pretperiodu istinita

```
84:                 if ψ_{a_1} ∉ l[j] then          ▷ čim ψ_{a_1} nije istinita za prvi j idući unazad, ψ_a nije istinita ni za jedan j' ≤ j
85:                     always = False
86:                     break
87:                 else
88:                     l[j] = l[j] ∪ {ψ_a}
89:                 end if
90:                 if always = False then
91:                     for j' ∈ [0, j] do
92:                         l[j'] = l[j'] ∪ {¬ψ_a}
93:                     end for
94:                 end if
95:             end for
96:         end if
97:
98:
99:     else if ψ_a ≡ Fψ_{a_1} then                          ▷ obrađujemo F
100:        event = False
101:        for j ∈ [k, k+n] do                              ▷ prvo provjeravamo period
102:            if ψ_{a_1} ∈ l[j] then
103:                event = True
104:                break
105:            end if
106:        end for
107:        if event == True then    ▷ ako je na periodu istinita, onda će svakako biti istinita u svakom stanju
108:            for j ∈ [0, k+n] do
109:                l[j] = l[j] ∪ {ψ_a}
110:            end for
111:        else
```

```
            for j ∈ [k, k+n] do                          ▷ na periodu nije istinita
                l[j] = l[j] ∪ {¬ψ_a}
            end for
            for j ∈ [k−1, 0] do    ▷ obrnuti for, provjeravamo unazad je li na pretperiodu istinita
                if ψ_{a_1} ∈ l[j] then  ▷ čim je ψ_{a_1} istinita za prvi j idući unazad, ψ_a je istinita za svaki j' ≤ j
                    event = True
                    break
                else
                    l[j] = l[j] ∪ ¬{ψ_a}
                end if
                if event = True then
                    for j' ∈ [0, j] do
                        l[j'] = l[j'] ∪ {ψ_a}
                    end for
                end if
            end for
        end if


    else if ψ_a ≡ ψ_{a_1} U ψ_{a_2} then                  ▷ obrađujemo U
        for j ∈ [0, k+n] do
            if ∃j' ∈ [j, k+n](ψ_{a_2} ∈ l[j'] and
                    ∀j'' ∈ [j, j'−1]ψ_{a_1} ∈ l[j'']) then
                l[j] = l[j] ∪ {ψ_a}
            else
                if k ≤ j and ∃j' ∈ [k, j−1](ψ_{a_2} ∈ l[j'] and
                        ∀j'' ∈ [j, k+n] ∪ [k, j'−1]ψ_{a_1} ∈ l[j'']) then
                    l[j] = l[j] ∪ {ψ_a}
```

```
                else
                    l[j] = l[j] ∪ {¬ψ_a}
                end if
            end if
        end for

    else if ψ_a ≡ ψ_{a_1} R ψ_{a_2} then                    ▷ obrađujemo R
        for j ∈ [0, k+n] do        ▷ prema Napomeni 1.1, ψ_{a_1} R ψ_{a_2} samo je pokrata za ¬(¬ψ_{a_1} U ¬ψ_{a_2})
            if ∃j' ∈ [j, k+n](ψ_{a_2} ∉ l[j'] and
                    ∀j'' ∈ [j, j'−1] ψ_{a_1} ∉ l[j'']) then
                l[j] = l[j] ∪ {¬ψ_a}        ▷ istinita je ¬ψ_{a_1} U ¬ψ_{a_2}, odnosno negacija početne formule
            else
                if k ≤ j and ∃j' ∈ [k, j−1](ψ_{a_2} ∉ l[j'] and
                        ∀j'' ∈ [j, k+n] ∪ [k, j'−1] ψ_{a_1} ∉ l[j'']) then
                    l[j] = l[j] ∪ {¬ψ_a}
                else
                    l[j] = l[j] ∪ {ψ_a}
                end if
            end if
        end for
    end if
end for

if ψ_m ∈ l[0] then
    return True
else
    return false
end if
```

4 Implementacija

U ovom poglavlju opisujemo na koji način je algoritam implementiran. Za te potrebe korišten je programski jezik Python te njegovi moduli: ltl2dfa, abc, typing, re, itertools i slični, a za potrebe grafičkog sučelja korišten je PyQt5.

4.1 Parser

Prvi problem koji treba riješiti je parsiranje formula. Za to smo koristili dio koda iz projekta LTLf2DFA koji je javno dostupan na
https://github.com/whitemech/LTLf2DFA, a instalira se pomoću

```
$ pip install ltlf2dfa
```

String koji parsiramo uvijek je sastavljen od sljedeće sintakse:

Sintaksa	
element abecede	*zapis u računalu*
$\neg$	"!"
$\wedge$	"&"
$\vee$	"|"
$\rightarrow$	"->"
$\leftrightarrow$	"<->"
X	"X"
U	"U"
R	"R"
F	"F"
G	"G"
$\bot$	"true"
(	"("
)	")"

4.2 Glavni algoritam

Prvo što algoritam radi je rastavljanje na potformule. To se može implementirati pomoću reda (kao strukture podataka). Za danu formulu prilažemo sljedeći pseudokod za rastavljanje na potformule

Algoritam 2 Za dani ψ, vraća red svih potformula od ψ

```
1:  q = Queue()
2:  potformule = []
3:  r = parser(ψ)                          ▷ u r je spremljena parsirana ψ
4:  ubaci r na kraj q
5:  while q nije prazan do
6:      r = uzmi prvi element reda q
7:      ubaci r na početak polja potformule
8:      if r nije propozicijska varijabla then
9:          t = izravne potformule od r    ▷ jedna ako je r unarna, dvije ako je
    binarna
10:     else
11:         continue
12:     end if
13:     ubaci elemente iz t na kraj reda q
14: end while
15: return q
```

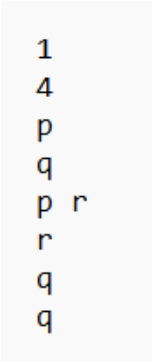
Sada imamo sve potrebno za implementaciju. Kod koji smo napisali u Pythonu nastao je iz dvaju napisanih pseudokodovima. Povrh svega napisano je jednostavno grafičko sučelje za unos formule koje potom vraća istinitosnu vrijednost parsirane formule na zadanom modelu.

4.3 Objašnjenje grafičkog sučelja

Cijeli kod programa nalazi se na javnom repozitoriju $https://github.com/mdujic/ltl-model-checker$ i može se klonirati pomoću:

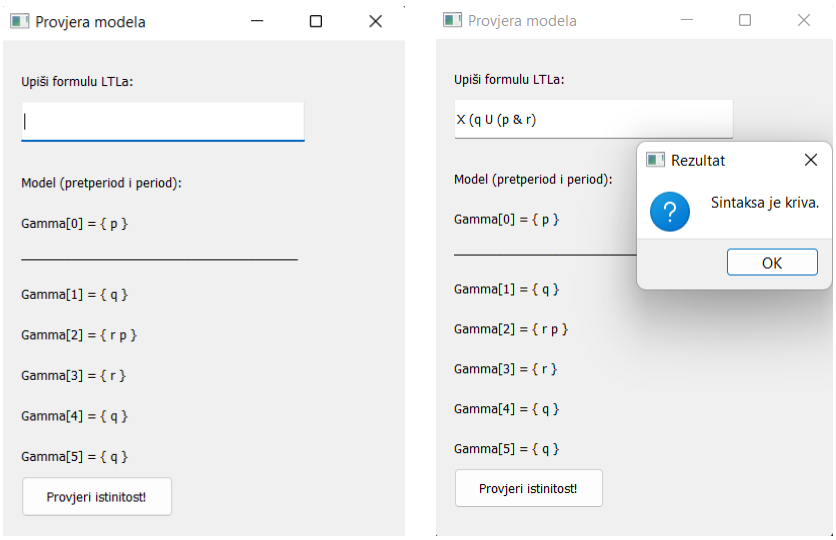
```
$ git clone git@github.com:mdujic/ltl-model-checker.git
```

Kada se repozitorij klonira, u njemu se nalazi datoteka *unos_modela.txt*. Korisnik može mijenjati model nad kojim želi testirati formule tako da mijenja datoteku. Datoteka primjerice izgleda na sljedeći način:



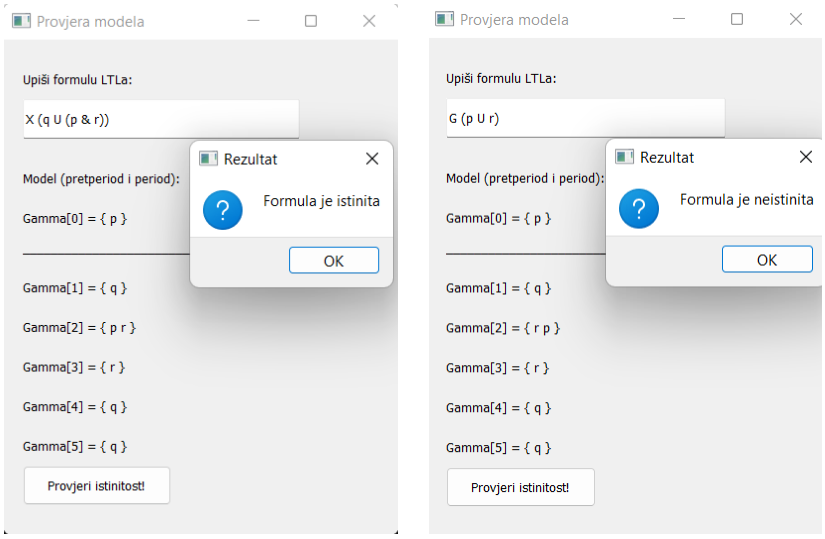
Slika 8: Primjer unosa modela

Prva linija je duljina pretperioda kojeg smo označavali s k. Druga linija je duljina perioda kojeg smo označavali s n. Preostalih $k+n+1$ linija su propozicionalne varijable koje su elementi skupova $\sigma(0),\ldots,\sigma(k+n)$.



(a) (b)

Slika 9: Pokrenuti algoritam za provjeru istinitosti i primjer prepoznavanja pogrešnog unosa



(a) (b)

Slika 10: Provjere istinitosti za istinitu formulu $\mathsf{X}(q\mathsf{U}(p \wedge r))$ i neistinitu $\mathsf{G}(p\mathsf{U}r)$

Napomena. Važno je pripaziti na sljedeće: broj linija mora se poklapati s brojem svjetova i $(k+3)$. linija mora se poklapati sa zadnjom zbog periodičnosti modela. Ako korisnik i zaboravi na te uvjete, program će javiti pogrešku i sučelje se neće pokrenuti.

Cijelo sučelje zatim se pokreće naredbom:

```
$ python3 -m algorithm
```

Literatura

[1] M. BEN-ARI, **Mathematical Logic for Computer Science**, Springer Verlag, 2001.

[2] B. BERARD, M. BIDOIT, A. FINKEL, F. LAROUSSINIE, A. PETIT, L. PETRUCCI, PH. SCHNOEBELEN, **Systems and Software Verification**, Springer Verlag, 2001.

[3] P. BLACKBURN, J. VAN BENTHEM, F. WOLTER (EDS.), **Handbook of Modal Logic**, Elsevier, 2007.

[4] K. BURIĆ, **Linearna temporalna logika**, diplomski rad, PMF-Matematički odsjek, Zagreb, 2014.

[5] A. CHAGROV, M. ZAKHARYASCHEV, **Modal Logic**, Clarendon Press, 1997.

[6] E. M. CLARKE, O. GRUMBERG, D. A. PELED, **Model-Checking**, MIT Press, 2000.

[7] E.M. CLARKE., A.P. SISTLA, **The complexity of linear temporal logic**, Journal of the ACM, 32:733–749, 1985.

[8] S. DEMRI, V. GORANKO, M. LANGE, **Temporal Logic in Computer Science**, Cambridge University Press, 2016.

[9] V. GORANKO, **Temporal logics for specification and verification**, ESSLLI, 2009.

[10] M. HUTH, M. RYAN, **Logic in Computer Science**, Cambridge University Press, 2004.

[11] M. KRACHT, **Tools and Techniques in Modal Logic**, North-Holland, 1999.

[12] K. LUNDQVIST, M. OUIMET, **Formal software verification: model checking and theorem proving**, Embedded Systems Laboratory, Technical Report MIT, USA, 2005.

[13] N. MARKEY, P. SCHNOEBELEN, **Model Checking a Path**, Preliminary Report, CONCUR 3, Marseilles, 2003.

[14] A. NERODE, R. A. SHORE, **Logic for Applications**, Springer, 1997.

[15] U. SCHÖNING, **Logic for Computer Scientists**, Birkhäuser, 2001.

[16] R. SKORIĆ, **Temporalna logika**, diplomski rad, PMF-Matematički odsjek, Zagreb, 2009.

[17] M. VUKOVIĆ, **Matematička logika**, Element, Zagreb, 2009.

Sažetak

Mateo Dujić, Laura Horvat:

Algoritam za provjeru mješovito periodičkih modela linearne temporalne logike

U ovome radu obradili smo svu potrebnu teoriju te implementirali algoritam za provjeru mješovito periodičkih modela linearne temporalne logike. U izvorima koje smo pronašli ([8]), algoritam je bio dan u pseudokodu samo za propozicionalne varijable, konjunkciju, negaciju, temporalne operatore X i U. Nadopunili smo ga i za ostale veznike i operatore: disjunkciju, implikaciju, ekvivalenciju, F, G i R. U cijelosti smo ga implementirali u Pythonu. Ušli smo u srž algoritma koja specifični beskonačan model uspijeva svesti na konačan prepoznavanjem pretperioda i perioda te odgovarajućom interpretacijom istih. Beskonačnost modela, za razliku od konačnosti, ono je što je zanimljivo i u općem slučaju deterministički nedohvatljivo. Pisanjem ovog rada zaronili smo u temu Provjere modela (eng. Model checking) koja je važan dio područja Formalne verifikacije algoritama [6, 13]. Implementaciju smo upotpunili grafičkim sučeljem koje korisniku omogućuje unos mješovito periodičkog modela preko tekstualne datoteke te formule preko tekstualnog okvira. Time smo željeli što je moguće više pojednostaviti korištenje za druge koji bi se željeli baviti ovim ili sličnim problemom.

Ključne riječi: matematička logika, algoritam, linearna temporalna logika, provjera modela, mješovito periodički modeli

Summary

Mateo Dujić, Laura Horvat:

Model checking algorithm for ultimately periodic models in linear temporal logic

In this paper we studied all the necessary theory and implemented an algorithm for model checking of ultimately periodic models in linear temporal logic. In the sources we had studied beforehand ([8]), the algorithm was written in pseudocode for propositional variables, conjunction, negation, and temporal operators X and U. We expanded it so it accepts disjunction, implication, equivalence, and operators F, G and R. We implemented the entire algorithm in Python and succeeded in realizing the main point: specific (ultimately periodic) infinite model can be reduced to finite by recognizing a prefix and a loop and interpreting them accordingly. Infinity of a model, unlike finiteness, is what is interesting and in general case not computationally tractable. Writing this paper, we dove right into topic of Model checking, which is an important part in Formal verification of algorithms [6, 13]. Implementation was extended with graphical user interface which enables users to enter ultimately periodic models through text file and formulas using textbox. Our goal was to simplify the problem for others who might be interested in this or similar topic.

Keywords: mathematical logic, algorithm, linear temporal logic, model checking, ultimately periodic models

Zahvale

Od srca zahvaljujemo mentoru prof. dr. sc. Mladenu Vukoviću na vodstvu i ogromnoj količini podrške tijekom pisanja ovog rada.