LIETUVOS MATEMATIKOS RINKINYS
Proc. of the Lithuanian Mathematical Society, Ser. A
Vol. 65, 2024, pages 18–24
https://doi.org/10.15388/LMR.2024.37368

Grouping based calculus for propositional linear temporal logic

Kostas Ragauskas, Adomas Birštunas

Faculty of Mathematics and Informatics, Vilnius University
Naugarduko str. 24, LT-03225 Vilnius, Lithuania
E-mail: kostas.ragauskas@mif.stud.vu.lt; adomas.birstunas@mif.vu.lt

Received May 22, 2024; published online December 10, 2024

Abstract. In this paper, the authors research the problem of loops in linear temporal logic PLTL. The task involves defining the standard rule application process for the derivation procedure (as used in [4] and [5]), determining and proving properties for the absence of a loop beneath some sequent, and creating a new calculus G*TL, which uses the proposed sequent grouping method, along with the method of marks (similar marking concepts were proposed in [5] and [6]). A new type of structural rule (GROUP), along with a modification of the rule ($\circ$) to ($\circ^*$) is introduced. Finally, it is shown that the loop checking mechanism used in calculus G*TL is efficient, comparing it with other known calculi for logic PLTL.

Keywords: loop checking; sequent calculi; marked sequents

AMS Subject Classification: 03B44

Introduction

Sequent calculi for temporal logic can describe the reasoning about how the truth values of assertions change over time. There are quite a few calculi created for such logic (see the works of [4] and [5]), however, none of them have any efficient loop elimination methods proposed. For this reason, loops and loop checking are still a problem in temporal logic.

In this paper, a new sequent grouping procedure is presented, from which an optimized calculus for loop checking will be created.

1 Logic PLTL – calculus $\mathrm{G_L PLTL}$

1.1 Syntax and semantics

Propositional linear temporal logic (PLTL) uses standard propositional logic operators and modal operators for the time – $\Box$ (“always”) and $\circ$ (“next”). The modal operator $\Diamond$ is not used, since it can be expressed as $\Diamond p = \neg\Box\neg p$.

- $\circ p$ is understood as “p is true in the next time moment”.
- $\Box p$ is understood as “p is true now and in all other time moments”.

In the paper [5] a sequent calculus $\mathrm{G_L PLTL}$ for logic PLTL is described, in which both positive (loop-axioms) and negative (non-derivable) loops can be constructed.

Definition 1. $S \to S'$ is a loop in a sequent derivation tree if S and S' are sequents presented in same branch of the tree (S' is above S), and S' and S contains the same formulas on their antecedents and succedents.

Definition 2. S' is a loop axiom in a sequent calculus for PLTL if there is a loop $S \to S'$, there exists such a rule $(\vdash \Box)$ application between S and S', that S' is on the right branch of this application, and there is no such a rule $(\vdash \Box)$ application between S and S', that S' is on the left branch of this application.

Definition 3. Sequent calculus $\mathrm{G_L PLTL}$ for logic PLTL is a calculus with 2 axioms – traditional $A, \Gamma \vdash A, \Delta$ and a loop-axiom, classical rules for propositional logic (see calculus $\mathrm{LK_0}$ [1]) and specific temporal rules:

$$\frac{\Gamma_1 \vdash \Gamma_2}{\Sigma_1, \circ\Gamma_1 \vdash \Sigma_2, \circ\Gamma_2}(\circ) \quad \frac{A, \circ\Box A, \Gamma \vdash \Delta}{\Box A, \Gamma \vdash \Delta}(\Box \vdash) \quad \frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, \circ\Box A}{\Gamma \vdash \Delta, \Box A}(\vdash \Box)$$

here, A is any formula. $\Gamma, \Gamma_1, \Gamma_2, \Delta, \Sigma_1, \Sigma_2$ are the multisets of formulas.

Definition 4. Sequent S is called **primary** if it is of the shape $\Sigma_1, \circ\Gamma_1 \vdash \Sigma_2, \circ\Gamma_2$, where multisets $\Sigma_1, \circ\Gamma_1, \Sigma_2, \circ\Gamma_2$ may be empty. Σ_1, Σ_2 consist of only atomic formulas, $\Sigma_1 \cap \Sigma_2 = \emptyset$. $\circ\Gamma_1, \circ\Gamma_2$ consist of only formulas of shape $\circ F$, $\circ\Gamma_1 \cap \circ\Gamma_2 = \emptyset$.

2 Optimization method for loop checking

2.1 Rule application process

From [4] and [5] follows, that rules may be applied in a specific order without losing sequent derivability in $\mathrm{G_L PLTL}$. This order is as follows:

- Logical rules for propositional logic and modal rules $(\Box \vdash)$, $(\vdash \Box)$ are applied while possible.
- Finally each sequent will contain only atomic formulas, or formulas of the shape $\circ F$. Therefore, a **primary sequent** will be constructed (see Definition 4), for which the rule $(\circ)$ will have to be applied.

2.2 Qualities for loop absence beneath some sequent

Definition 5. If formula F is $H \to G$, $G \to H$, $H \wedge G$, $G \wedge H$, $H \vee G$, $G \vee H$, $\neg G$, $\circ G$ or $\Box G$, then G is a *subformula* of F. All subformulas of G are subformulas of F.

Definition 6. If formula $F = \Box G$, then $\circ\Box G$ is the *extended subformula* of F if it has the same entry as F. Positive and negative entry of the formula is defined in [2].

Definition 7. Subformula FP of formula F will be called *fundamental subformula*, if FP has the same entry as F.

Definition 8. Subformula FP of formula F will be called $\Box$-*subformula*, if FP is in the scope of $\Box$ operator (while being inside F).

Definition 9. Formula G, which is not a fundamental $\Box$-subformula in the current sequent, will be called *ground formula.* This is a modified description taken from [4].

Definition 10. Ground formula, which is in the shape $\Box F$ and is in the current sequent's antecedent, will be called *super-ground formula.* We will mark it $\blacksquare G$.

Lemma 1. *The construction of a loop* $S_1 \to S_{n+1} = S_1, S_2, \dots, S_n, S_{n+1}$ *is impossible if an atomic ground formula* f *is presented in the primary sequent* S_n*.*

Proof.

1. If a formula g is in the sequent S_1, then it will also be in sequent S_{n+1}, since $S_1 \to S_{n+1}$ is a loop. For this reason, formula $\circ g$ will be in sequent S_n, since all formulas without the $\circ$ operator are removed during rule ($\circ$) application.
2. If a formula f is a fundamental $\Box$-subformula of some other formula g, which is in sequent S_1, then formula g will be in sequent S_{n+1} and formula $\circ g$ will be in sequent S_n (from item 1). In this case, formula f would be a fundamental $\Box$-subformula of some formula in sequent S_n. Hence, in our case, ground formula f is not a fundamental $\Box$-subformula of any other formula in sequent S_1.
3. If a formula f is a subformula of some other formula g (which is in sequent S_1) and f is not a fundamental $\Box$-subformula of any other formula in sequent S_1, then f is either i) an extended subformula of g (that is $f = \circ g$), or ii) $f = g$ (then formula g is one of the forms: $\Box G$, $\circ\Box G$), or iii) g is none of the forms $\Box G$, $\circ\Box G$, since then formula f would be a fundamental $\Box$-subformula.
4. If a formula f is in sequent S_n, then f is a subformula of some other formula, which is in sequent S_1. Suppose g is the **longest** formula in sequent S_1, and f is a subformula of g. Sequent length is defined in a standard way (see [3]).
5. From item 1, formula $\circ g$ is in sequent S_n, hence in sequent S_1 there is such a formula h, that $h = \circ g$, or $\circ g$ is either an extended subformula, or a fundamental $\Box$-subformula of formula h.

If $\circ g$ is not an extended subformula of h, then formula h will be longer than g. If f is a subformula of g, and $\circ g$ is a subformula of h, then f is also a subformula of h. We have a **contradiction** since g should be the longest formula in S_1, that f is a subformula of g.

If $\circ g$ is an extended subformula of h, then $h = \Box G = g$. We know that f is a subformula of g (from item 4), hence either $f = g$, or f is a fundamental $\Box$-subformula of g.

- In the case $f = g$, we can infer that $f = \Box G$. We get a **contradiction** of the original assumption that f is an atomic formula.
- In the case f is a fundamental $\Box$-subformula of g, formula f would also be a fundamental $\Box$-subformula of $\circ g$. We get a **contradiction** of the original assumption that f is a ground formula. $\Box$

Lemma 2. *If a ground formula, which is not in the shape $\Box H$ and not an extended subformula, is present in sequent S' – a loop $S \to S'$ cannot be constructed.*

Proof. The proof goes straightforward from Lemma 1. $\Box$

2.3 Sequent grouping

Each primary sequent can be grouped into distinct sets by using the operation
$GROUP(S) = \Xi_1, \Xi_2, \circ\Lambda_1, \circ\Lambda_2, \circ\Box\Pi, \circ\blacksquare\Omega \vdash \circ\Box\Psi, \circ\Box\Delta, \circ\Box\Theta, \Phi_1, \Phi_2, \circ X_1, \circ X_2$
in which:

- $\circ\blacksquare\Omega$ – super-ground formulas of the shape $\circ\Box H$.
- $\circ\Box\Pi$ – formulas of the shape $\circ\Box H$, which are not ground formulas in S.
- $\circ\Box\Psi$ – formulas of the shape $\circ\Box H$, which are fundamental $\Box$-subformulas of $\circ\blacksquare\Omega$ or $\circ\Box\Pi$ formulas.
- $\circ\Box\Delta$ – ground formulas of the shape $\circ\Box H$.
- $\circ\Box\Theta$ – formulas of the shape $\circ\Box H$, which are fundamental $\Box$-subformulas of formulas in $\circ\Box\Delta$.
- $\Xi_1, \Phi_1, \circ\Lambda_1, \circ X_1$ – ground formulas of the shape H or $\circ H$ accordingly.
- $\Xi_2, \Phi_2, \circ\Lambda_2, \circ X_2$ – non-ground formulas of the shape H or $\circ H$ accordingly.

From the previous lemmas, it is evident that a loop cannot be constructed (in the ancestor sequents) if at least one of the sets from $\Xi_1, \Phi_1, \circ\Lambda_1, \circ X_1$ is not empty.

It is important to note that this grouping method can be applied in other loop checking optimization methods since it **does not change the derivation tree** itself.

2.4 Calculus G*TL

For calculus G*TL we use marked sequents and marked modal operator $\Box$. The same approach is used in [6] and [5]. To determine the absence of loops below some sequent, a new calculus G*TL with the incorporated ($GROUP$) rule was created:

Definition 11. Sequent calculus G*TL for logic PLTL is a calculus with 2 axioms – traditional $A, \Gamma \vdash A, \Delta$, and a loop-axiom, classical rules for propositional logic (the mark '$-$' of the sequents should be deleted if such exists) and specific temporal rules:

$$\frac{A, \circ\Box A, \Gamma \vdash \Delta}{\Box A, \Gamma \vdash^{\alpha} \Delta}(\Box\vdash) \quad \frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, \circ\Box A}{\Gamma \vdash^{\alpha} \Delta, \Box A}(\vdash\Box) \quad \frac{\Gamma \vdash^{-} \Delta, A \quad \Gamma \vdash \Delta, \circ\Box A}{\Gamma \vdash^{\alpha} \Delta, \Box^{*} A}(\vdash\Box^{*})$$

$$\frac{\Xi_1, \Xi_2, \circ\Lambda_1, \circ\Lambda_2, \circ\Box\Pi, \circ\blacksquare\Omega \vdash^{\alpha} \circ\Box\Psi, \circ\Box\Delta, \circ\Box\Theta, \Phi_1, \Phi_2, \circ X_1, \circ X_2}{\Sigma_1, \circ\Gamma_1 \vdash^{\alpha} \Sigma_2, \circ\Gamma_2}(GROUP)$$

$$\frac{\Lambda_1, \Lambda_2, \Box\Pi, \blacksquare\Omega \vdash^{\delta} \Box\Psi, \Box^{*}\Delta, \Box\Theta, X_1, X_2}{\Xi_1, \Xi_2, \circ\Lambda_1, \circ\Lambda_2, \circ\Box\Pi, \circ\blacksquare\Omega \vdash^{\alpha} \circ\Box\Psi, \circ\Box\Delta, \circ\Box\Theta, \Phi_1, \Phi_2, \circ X_1, \circ X_2}(\circ^{*})$$

here, δ is '$-$', if $\Xi_1 \cup \Phi_1 \cup \circ\Lambda_1 \cup \circ X_1 \neq \emptyset$, δ is $\emptyset$ otherwise, $\alpha \in \{\emptyset, -\}$.

It is important to mention some aspects of calculus G*TL:

1. The $(GROUP)$ rule can only be applied to primary sequents.
2. The rule application process is the same as defined in Section 2.1. However, after the construction of a primary sequent, it is obligatory to apply the rule $(GROUP)$, after which the rule $(\circ^*)$ will follow.
3. After the application of the rule $(GROUP)$ it is obligatory to check whether at least one of the sets $\Xi_1, \Phi_1, \circ\Lambda_1, \circ X_1$ from the current (grouped) sequent is not empty. In that case, the δ mark will be set to '$-$'. Otherwise, the mark will be left empty.

Lemma 3. *Sequent S is proveable in the calculus G_LPLTL if and only if it is proveable in calculus G*TL.*

Proof. The proof goes straightforward from the Lemma 2 and grouping operation.

Rule $(\circ)$ application in calculus G_LPLTL is replaced by the application of rules $(GROUP) + (\circ^*)$ in calculus G*TL. The rule $(GROUP)$ does not make any changes to the current sequent. In addition, the rule $(\circ^*)$ serves the same purpose as the rule $(\circ)$, only with an addition of certain marks for the current sequent or the $\Box$ operator.

Rule $(\vdash \Box)$ application in calculus G_LPLTL is replaced by the application of rules $(\vdash \Box)$ or $(\vdash \Box^*)$ in calculus G*TL, since they both serve the same purpose as the rule $(\vdash \Box)$ in calculus G_LPLTL, only with an addition of certain marks for the current sequent or the $\Box$ operator.

Therefore, the derivation tree is the same, but the loop check is restricted by comparing the current sequent with sequents presented above the uppermost sequent marked by '$-$'. $\Box$

3 Complexity

3.1 Complexity of the rule (GROUP)

From the initial sequent of the derivation tree, all possible formulas and subformulas G (including extended subformulas) with their entries can be indexed and placed inside a hash table as keys, which will have their values as lists of only the extended parent subformulas $\circ\blacksquare F$ or $\circ\Box F$ (for which G will be their fundamental $\Box$-subformulas). The creation of such a table will be done only once per derivation and it will take a polynomial amount of steps.

Suppose c is the maximum amount of formulas inside some value list from the constructed hash table. If the formula count in the current sequent is equal to m, then the rule $(GROUP)$ will require at most $m \cdot c \cdot (m-1) \leqslant c \cdot m^2$ steps.

Suppose j is $\Box$ operator count in the initial sequent of the derivation tree. It is evident, that $c \leqslant j$, so the operation $(GROUP)$ will require $\leqslant j \cdot m^2$ steps and it will be of polynomial complexity $O(j \cdot m^2)$.

3.2 Complexity of the loop checking method in calculus G*TL

Suppose we have a branch of some derivation tree $S_1, S_2, \ldots, S_n = S_{x+y}$ (S_1 is an initial and S_n is a current sequent), in which the rule $(\circ^*)$ with a non-empty δ mark (rule $(\circ)$ in the case of G_LPLTL) was applied at least once. Suppose:

- x – sequent count from the initial sequent S_1 till sequent S_x – the premise of the uppermost rule $(\circ^*)$ application, with a non-empty δ mark, in the branch.
- y – sequent count from the sequent S_x till the current sequent $S_n = S_{x+y}$.
- s – sequent count from the sequent, which is the premise of the uppermost rule $(\vdash \Box^*)$ application till to the current sequent S_n.
- m – formula count in the current sequent S_n.
- k – $max\{m_1, m_2, \dots, m_{x+y}\}$, where m_i is the formula count in the sequent S_i.

In the case of calculus $\mathrm{G_L}$PLTL, after each application of the rule $(\circ)$ it will be necessary to check all the previous sequents in the derivation branch, which would require $x \cdot m \cdot k$ steps (comparison operations).

In the case of calculus G*TL, after each application of the rule $(\circ^*)$ with an empty δ mark, it will be necessary to check only those sequents, which were constructed above the uppermost rule $(\circ^*)$ application with a non-empty δ mark. This would require $y \cdot m \cdot k$ steps. Hence, the loop check procedure is improved by $\frac{x \cdot m \cdot k}{y \cdot m \cdot k} = \frac{x}{y}$ times. If we have $s < y$, then we can reach an even better improvement of $\frac{x \cdot m \cdot k}{s \cdot m \cdot k} = \frac{x}{s} > \frac{x}{y}$ times.

However, we should also consider the cost of the rule‘s $(GROUP)$ application, which takes $j \cdot m^2$ steps. The optimized loop check, including the cost of the rule $(GROUP)$, in comparison with the loop check in calculus $\mathrm{G_L}$PLTL is improved by $\frac{x \cdot m \cdot k}{y \cdot m \cdot k + j \cdot m^2} = \frac{x \cdot k}{y \cdot k + j \cdot m}$ times. Knowing the fact that in most cases formula count in the derivation tree is getting smaller (that is $m \leqslant k$), we can modify the final improvement to $\frac{x}{j+y}$ (or $\frac{x}{j+s}$).

From this, we can conclude that when x is a sufficiently big value (which occurs quite often, since derivation trees are usually long and rules $(\circ)$ are being applied multiple times), we can get a significant improvement.

Conclusions

1. The created proof, regarding the absence of a loop beneath some sequent, allowed to construct an effective sequent grouping procedure in the propositional linear temporal logic PLTL.
2. It was determined that the presented loop checking procedure is more efficient than the standard one.
3. It was determined that the presented sequent grouping method (for the lower loop limit detection) can be applied in other loop checking mechanisms as well.

References

[1] R. Alonderis. Proof-search of propositional intuitionistic logic sequents by means of classical logic calculus. *Liet. mat. rink. LMD darbai*, **48/49**:256–262, 2008.

[2] J. Andrikonis. Loop-free calculus for modal logic S4. I. *Lith. Math. J.*, **52**:1–12, 2012.

[3] A. Birštunas. PSPACE complexity of modal logic $\mathrm{KD45}_n$. *Lith. Math. J.*, **48**:174–187, 2008.

[4] A. Birštunas. Restrictions for loop-check in sequent calculus for temporal logic. *Liet. mat. rink. LMD darbai*, **48/49**:269–274, 2008.

[5] R. Pliuškevičius. Method of marks for propositional linear temporal logic. *Liet. matem. rink. Proc. LMS, Ser. A*, **55**:46–50, 2014.

[6] M. Sadrzadeh, R. Dyckhoff. Positive logic with adjoint modalities: Proof theory, semantics, and reasoning about information. *Rev. Symb. Log.*, **3**:351–373, 2010.

REZIUMĖ

Grupavimu paremtas tiesinės teiginių laiko logikos skaičiavimas

K. Ragauskas, A. Birštunas

Šiame darbe autoriai tiria ciklų susidarymo problemą tiesinėje laiko logikoje PLTL. Užduotis apima taisyklių taikymo tvarkos apibrėžimą (naudotą [4] bei [5]), savybių ciklų negalimumui nustatymą bei naujo sekvencinio skaičiavimo G*TL sukūrimą, kuris naudoja aprašytus sekvencijų grupavimo bei žymių metodus (panašios žymės buvo naudojamos ir [5] bei [6] darbuose). Pristatoma naujo pobūdžio struktūrinė taisyklė (GROUP), kartu su taisyklės ($\circ$) modifikacija į ($\circ^*$). Galiausiai yra parodoma, jog sukurta ciklų aptikimo procedūra skaičiavime G*TL yra efektyvesnė už įprastą procedūrą, taikomą kituose skaičiavimuose PLTL logikai.

Raktiniai žodžiai: ciklų aptikimas; sekvenciniai skaičiavimai; žymėtos sekvencijos