Restriction on cut in cyclic proof system for symbolic heaps

Kenji Saotome[1], Koji Nakazawa[1], and Daisuke Kimura[2]

[1] Nagoya University, Japan, saotomekenji@sqlab.jp, knak@i.nagoya-u.ac.jp
[2] Toho University, Japan, kmr@is.sci.toho-u.ac.jp

Abstract. It has been shown that some variants of cyclic proof systems for symbolic heap entailments in separation logic do not enjoy the cut elimination property. To construct complete system, we have to consider the cut rule, which requires some heuristics to find cut formulas in bottom-up proof search. Hence, we hope to achieve some restricted variant of cut rule which does not change provability and does not interfere with automatic proof search without heuristics. This paper gives a limit on this challenge. We propose a restricted cut rule, called the presumable cut, in which cut formula is restricted to those which can occur below the cut. This paper shows that there is an entailment which is provable with full cuts in cyclic proof system for symbolic heaps, but not with only presumable cuts.

1 Introduction

Separation logic is an extension of Hoare logic for verifying programs manipulating heap memories. It is successful as theoretical base to achieve (semi-)automatic verification systems for low level languages and low-level programming languages which is often developed with C language.

One of the keys for automation is to solve the entailment checking problem for separation logic formulas with inductive predicates, which represent recursively structured data on heaps. To obtain decidability, some restrictions are required. One of them is to restrict formulas to (extended) symbolic heaps, which are considered sufficient to represent assertions in real verification systems, and for which some decidability results have been obtained [1, 2, 10, 11, 14, 17, 12]. Some of them [1, 2, 17] are based on proof theoretic approaches, which have some advantages: it provides evidences of correctness for valid entailments, and it can be easily extended by adding inference rules.

Cyclic proof systems are adopted for the entailment checking for symbolic heaps with general inductive predicates [4, 5, 17]. Cyclic proofs are proofs with cycles, which correspond to use of induction hypotheses. They are considered suitable for automated inductive reasoning since we need not to decide formulas to be proved by induction a priori. However, there are few results on fundamental properties of cyclic proof systems.

Cut-elimination property is one of such properties of logical systems. It is important not only from theoretical interest, but also from application, since

it ensures that provability is not changed by existence of the cut rule, which is not good for bottom-up proof search since we have to find cut formulas. However, it is shown in [13] that we cannot eliminate cuts from the cyclic proof system for symbolic heaps. The existing cyclic proof systems [4, 5, 17] are cut-free systems, and neither completeness nor decidability has been proved for the systems in [4, 5], and the system in [17] requires additional mechanism called the spatial factorization and a relatively strong restriction for inductive predicates to achieve completeness and decidability. Hence we hope that we can restrict the cut rule as not to interfere with automated proof search.

This paper gives a limit on this challenge for a cyclic proof system for symbolic heaps. It shows that it seems hard to restrict the cut rule as not to interfere with automated proof search. In other words, some heuristics seem unavoidable to find cut formulas for full power of provability.

First, we propose a restricted variant of the cut rule, called *presumable cuts*. A formula is called *presumable* from a sequent if it is obtained by bottom-up applications of inference rules except for cut from the sequent, and a cut is called presumable if the cut formula is presumable. Presumable cuts can be applied without heuristics, so it is preferable that any conclusion which is provable with cuts can be proved with only presumable cuts. We call this property *quasi cut-elimination property*.

One counterexample to the cut elimination in [13] is $\mathrm{lsne}(x, y) \vdash \mathrm{slne}(x, y)$, where the predicates are defined as

$$\mathrm{lsne}(x, y) = x \mapsto y \mid \exists z.(x \mapsto z * \mathrm{lsne}(z, y)),$$
$$\mathrm{slne}(x, y) = x \mapsto y \mid \exists z.(\mathrm{slne}(x, z) * z \mapsto y).$$

They showed that the entailment can be proved with cuts, but is not provable without cut. Its cyclic proof with a cut is

$$\dfrac{\dfrac{\dfrac{}{x \mapsto y \vdash x \mapsto y}}{x \mapsto y \vdash \mathrm{slne}(x, y)} \quad \dfrac{\dfrac{\vdots\ (1)}{x \mapsto z * \mathrm{lsne}(z, y) \vdash \underline{x \mapsto z * \mathrm{slne}(z, y)}} \quad \dfrac{\vdots\ (2)}{\underline{x \mapsto z * \mathrm{slne}(z, y)} \vdash \mathrm{slne}(x, y)}}{x \mapsto z * \mathrm{lsne}(z, y) \vdash \mathrm{slne}(x, y)}\ (\mathrm{Cut})}{\mathrm{lsne}(x, y) \vdash \mathrm{slne}(x, y)},$$

where both (1) and (2) are cut free. The underlined cut formula $x \mapsto z * \mathrm{slne}(z, y)$ is presumable since it is obtained by unfolding $\mathrm{slne}(x, y)$. From this example, we may expect that any entailment provable with cuts can be proved with only presumable cuts, that is, the quasi cut-elimination property holds for the cyclic proof system for symbolic heaps.

The main result of this paper is that the cyclic proof system for symbolic heaps does not satisfy the quasi cut-elimination property. It is proved by a counterexample entailment which is provable with cuts but not provable with only presumable cuts. It means that the proof of this entailment requires some heuristics to find a cut formula.

Related work There are some results on cut-elimination property for proof systems with infinite paths and cyclic proof systems. For proof systems with infinite

paths, cut elimination holds for several logics such as first-order predicate logic [6] and multiplicative additive liner logic with fixed point operators (μ-MALL) [9]. On the other hand, for cyclic proof systems, it does not hold for several logics such as symbolic heap separation logic [13], μ-MALL [9], and sequent style system for Kleene algebra [8]. These results suggest that there is an essential difference between proof systems with infinite paths and cyclic proof systems, which are obtained by restricting proofs in the former systems to regular trees.

Hence, for automated reasoning, it is important to achieve cyclic proof systems with restricted cut rule which is sufficient for provability, that is, the quasi cut-elimination property holds, and which does not require any heuristics to find cut formulas. However, there are few studies on such restriction to the cut rule in cyclic proof systems. The proof system introduced by Chu [7] is implicitly based on the cyclic proof system with restricted form of the cut rule where one of the assumptions must be a bud in the cyclic proof of which the companion stands below the bud. This restriction is a special case of the presumable cut.

We can also find this kind of restriction in the sequent calculi of some modal logics [16], for which the cut elimination does not hold but the cut rule can be restricted, without changing provability, to that with a cut formula which is a subformula of the bottom sequent of the cut. This restricted cut is also a special case of the presumable cut.

Structure of the paper We introduce a cyclic proof system for symbolic heaps in Section 2. In Section 3, we propose the presumable cuts and the quasi cut-elimination property. In Section 4, we show that the cyclic proof system defined in Section 2 does not satisfy the quasi cut-elimination property. We give concluding remark in Section 5.

2 Cyclic proof system for Symbolic heaps

We define the logic SL_1 of symbolic heaps in separation logic, and the cyclic proof system CSL_1ID^ω for symbolic heaps.

2.1 Symbolic heaps

We define the formulas of the separation logic with the singleton heap $\mapsto$ and the separating conjunction $*$ in a standard way [15].

We use the metavariables x, y, z,... for variables, and P, Q,... for predicate symbols. Each predicate symbol is supposed to have a fixed arity.

Definition 1 (Symbolic heaps). *Let* nil *be a term constant, and* $\top$ *and* emp *be propositional constants. A* term*, denoted by* t, u,..., *is defined as either a variable or* nil*. We use the boldface metavariables* $\boldsymbol{x}$ *and* $\boldsymbol{t}$ *for finite sequences of variables and terms, respectively. The* pure formulas Π *and the* spatial formulas Σ *are defined as follows.*

$$\Pi ::= \top \mid t = u \mid t \neq u \mid \Pi \wedge \Pi, \qquad \Sigma ::= \mathrm{emp} \mid P(\boldsymbol{t}) \mid t \mapsto u \mid \Sigma * \Sigma.$$

The formulas of SL_1 are (quantifier-free) symbolic heaps A, which is defined as $\Pi \wedge \Sigma$. We define $FV(A)$ as the set of free variables in A.

We use the following notation. For $I = \{1, \cdots, n\}$, we write $\bigwedge_{i \in I} \Pi_i$ for $\Pi_1 \wedge \cdots \wedge \Pi_n$. Similarly, we write $\ast_{i \in I} \Sigma_i$ for $\Sigma_1 \ast \cdots \ast \Sigma_n$. When A is $\Pi_1 \wedge \Sigma_1$ and B is $\Pi_2 \wedge \Sigma_2$, we write $A \ast B$ for $\Pi_1 \wedge \Pi_2 \wedge \Sigma_1 \ast \Sigma_2$. Similarly, we write $A \wedge \Pi$ for $\Pi_1 \wedge \Pi \wedge \Sigma_1$ and $A \ast \Sigma$ for $\Pi_1 \wedge \Sigma_1 \ast \Sigma$.

Each predicate P is supposed to be accompanied with its definition clauses as

$$P(\boldsymbol{x}) := \exists \boldsymbol{y}_1.A_1 \mid \exists \boldsymbol{y}_2.A_2 \mid ... \mid \exists \boldsymbol{y}_n.A_n,$$

where each A_i is a quantifier-free symbolic heap whose free variables are in $\boldsymbol{x}$ or $\boldsymbol{y}_i$.

Substitutions of terms are finite mappings from term variables to terms. A substitution θ is called a renaming, if θ is injective and $\theta(x)$ is a variable for any x in its domain.

We identify formulas up to commutativity and associativity for $\ast$, commutativity, associativity, and idempotence for $\wedge$, and symmetry for $=$ and $\neq$. $\top$ is a unit for $\wedge$. We use $\equiv$ for syntactic identity modulo these identifications.

Definition 2 (Extended subformula). *The* extended subformulas *of a symbolic heap $\bigwedge_{i \in I} \Pi_i \wedge \ast_{j \in J} \Sigma_j$ are defined as formulas obtained by renaming from the formulas of the form $\bigwedge_{i \in I'} \Pi_i \wedge \ast_{j \in J'} \Sigma_j$ for $I' \subseteq I$ and $J' \subseteq J$.* $\mathrm{ExSub}(A)$ *is defined as the set of extended subformulas of A.*

2.2 Semantics of symbolic heap

In the following, for a mapping f, the domain and the image of f are denoted as $\mathrm{dom}(f)$ and $\mathrm{img}(f)$, respectively.

A store, denoted by s, is a function from variables to natural numbers $\mathbb{N}$. It is extended to a mapping on terms by $s(\mathrm{nil}) = 0$. We write $s[\boldsymbol{x} := \boldsymbol{a}]$ for the store which is the same as s except the values of $\boldsymbol{x}$ are $\boldsymbol{a}$. A heap, denoted by h, is a finite partial function from $\mathbb{N} \setminus \{0\}$ to $\mathbb{N}$. $h = h_1 + h_2$ means $\mathrm{dom}(h_1) \cap \mathrm{dom}(h_2) = \emptyset$, $\mathrm{dom}(h) = \mathrm{dom}(h_1) \cup \mathrm{dom}(h_2)$, and $h(n) = h_i(n)$ for $n \in \mathrm{dom}(h_i)$ $(i \in \{1, 2\})$. A pair (s, h) is called a *heap model*.

Definition 3 (Interpretation of formulas). *The interpretation of a formula A in (s,h), denoted by $s,h \models A$, is inductively defined as follows.*

$$
\begin{array}{lll}
s \models t = u & \stackrel{def}{\iff} & s(t) = s(u), \\
s \models t \neq u & \stackrel{def}{\iff} & s(t) \neq s(u), \\
s \models \Pi_1 \wedge \Pi_2 & \stackrel{def}{\iff} & s \models \Pi_1 \text{ and } s \models \Pi_2, \\
s,h \models t \mapsto u & \stackrel{def}{\iff} & \mathrm{dom}(h) = \{s(t)\} \text{ and } h(s(t)) = s(u), \\
s,h \models P^{(0)}(\boldsymbol{t}) & & \text{never holds}, \\
s,h \models P^{(m+1)}(\boldsymbol{t}) & \stackrel{def}{\iff} & s[\boldsymbol{y} := \boldsymbol{b}], h \models A[P_1^{(m)}, ..., P_k^{(m)}/P_1, ..., P_K](\boldsymbol{t}, \boldsymbol{y}), \\
 & & \text{for some definition clause } \exists \boldsymbol{y}.A \text{ of } P \text{ containing } P_1, ..., P_K, \\
s,h \models P(\boldsymbol{t}) & \stackrel{def}{\iff} & s,h \models P^{(m)}(\boldsymbol{t}) \text{ for some } m, \\
s,h \models \Sigma_1 * \Sigma_2 & \stackrel{def}{\iff} & \text{there exist } h_1 \text{ and } h_2 \text{ such that } h = h_1 + h_2, \\
 & & s,h_1 \models \Sigma_1, \text{ and } s,h_2 \models \Sigma_2, \\
s,h \models \Pi \wedge \Sigma & \stackrel{def}{\iff} & s \models \Pi \text{ and } s,h \models \Sigma,
\end{array}
$$

where $P^{(m)}$ is an auxiliary notation for defining $s,h \models P(\boldsymbol{t})$ and $A[P_1^{(m)}, ..., P_K^{(m)}/P_1, ..., P_K]$ is the formula obtained by replacing each P_i by $P_i^{(m)}$.

2.3 Inference rules of CSL_1ID^ω

The cyclic proof system CSL_1ID^ω consists of standard inference rules [4, 14].

Definition 4 (Sequent). *Let A and B be symbolic heaps. $A \vdash B$ is called a sequent. A is called the antecedent of $A \vdash B$, and B is called the succedent of $A \vdash B$. We use the metavariable e for sequents. A sequent $A \vdash B$ is defined to be* valid*, denoted by $A \models B$, if and only if, for any heap model (s,h) such that $s,h \models A$, $s,h \models B$ holds.*

Definition 5 (Inference rules of CSL_1ID^ω). *The inference rules of CSL_1ID^ω are the following.*

$$\frac{}{A \vdash A}\ (Id) \qquad \frac{}{A * t \mapsto u_1 * t \mapsto u_2 \vdash B}\ (\mapsto L) \qquad \frac{}{t \neq t \wedge A \vdash B}\ (NEQL)$$

$$\frac{A \vdash C \quad C \vdash B}{A \vdash B}\ (cut) \qquad \frac{A \vdash B}{\Pi \wedge A \vdash B}\ (Wk) \qquad \frac{A \vdash C \quad B \vdash D}{A * B \vdash C * D}\ (*)$$

$$\frac{t \neq u \wedge A \vdash B}{t \neq u \wedge A \vdash t \neq u \wedge B}\ (NEQR)$$

$$\frac{t = u \wedge A[u/x] \vdash B[u/x]}{t = u \wedge A[t/x] \vdash B[t/x]}\ (EQL) \qquad \frac{A \vdash B}{A \vdash t = t \wedge B}\ (EQR)$$

$$\frac{A \vdash B}{A * \mathrm{emp} \vdash B}\ (EL1) \qquad \frac{A * \mathrm{emp} \vdash B}{A \vdash B}\ (EL2)$$

$$\frac{A \vdash B}{A \vdash B * \mathrm{emp}}\ (ER1) \quad \frac{A \vdash B * \mathrm{emp}}{A \vdash B}\ (ER2)$$

$$\frac{C_1(\boldsymbol{x}, \boldsymbol{y}_1) * A \vdash B \quad \cdots \quad C_n(\boldsymbol{x}, \boldsymbol{y}_n) * A \vdash B}{P(\boldsymbol{x}) * A \vdash B}\ (Case) \quad \frac{A \vdash C_i(\boldsymbol{u}, \boldsymbol{t}) * B}{A \vdash P(\boldsymbol{u}) * B}\ (PR),$$

where the definition clauses of the predicate P are the following

$$P(\boldsymbol{x}) := \exists \boldsymbol{y}_1.C_1(\boldsymbol{x}, \boldsymbol{y}_1) \mid \cdots \mid \exists \boldsymbol{y}_n.C_n(\boldsymbol{x}, \boldsymbol{y}_n).$$

In (PR), i satisfies $1 \leq i \leq n$, and the terms $\boldsymbol{t}$ are arbitrary. In (Case), the variables $\boldsymbol{y}_i$ are fresh. The formula C in (cut) is called the cut formula.

2.4 Cyclic proof in CSL_1ID^ω

The cyclic proofs in CSL_1ID^ω are defined in a similar way to [6, 4, 5].

A derivation tree (denoted by D) in CSL_1ID^ω of a sequent e is defined in a usual way by the inference rules of CSL_1ID^ω.

Definition 6 (Bud, companion, and pre-proof). *A bud is a leaf sequent of a CSL_1ID^ω derivation tree that is not an axiom. A companion for a bud e is an occurrence of a sequent of which e is a substitution instance.*

A pre-proof P is defined as a pair (D,R) where D is a derivation tree and R is a function such that, for each bud occurrence e, $R(e)$ is a companion for e.

Definition 7 (Path). *A proof-graph $G(P)$ of a pre-proof $P = (D, R)$ is a directed graph structure D in the bottom-up manner with additional edges from buds to companions assigned by R. A path in P is a path in $G(P)$.*

Definition 8 (Trace). *Let $(e_i)_{i\geq 0}$ be a path in $P = (D, R)$. A* trace *along $(e_i)_{i\geq 0}$ is a sequence of inductive predicates $(C_i)_{i\geq 0}$ such that each C_i occurs in the antecedent of e_i, and satisfies the following conditions:*

(a) If e_i is the conclusion of (Case) in D, then either $C_i = C_{i+1}$ or C_i is unfolded in the rule instance and C_{i+1} appears as a subformula of the unfolding result. In the latter case, i is called a progressing point.
(b) If e_i is the conclusion of a rule other than (Case), then C_{i+1} is the subformula occurrence in e_{i+1} corresponding C_i in e_i.
(c) If e_i is a bud, C_{i+1} is the corresponding occurrence of the predicate to C_i in e_i.

If a trace contains infinitely many progressing points, it is called an infinitely progressing trace.

Definition 9 (Cyclic proof). *A pre-proof P of CSL_1ID^ω is called a cyclic proof if it satisfies the global trace condition: for any infinite path $(e_i)_{i\geq 0}$ in P, there is a number j and an infinitely progressing trace along the path $(e_i)_{i\geq j}$.*

Note that when e_i is the conclusion of (cut) and e_{i+1} is its right assumption, there is no trace along the path containing e_i and e_{i+1} since e_{i+1} contains no formula corresponding C_i in e_i. Hence, if a pre-proof contains an infinite path which passes through right assumptions of cuts infinitely many times, then it is not a cyclic proof since it cannot satisfy the global trace condition.

Example 1. A cyclic proof of $\mathrm{ls}(x,y) * \mathrm{ls}(y,z) \vdash \mathrm{ls}(x,z)$ in CSL_1ID^ω is given as follows, where the predicate ls is defined by

$$\mathrm{ls}(x,y) := x = y \wedge \mathrm{emp} \mid \exists x'.(x \neq y \wedge x \mapsto x' * \mathrm{ls}(x',y)).$$

Intuitively, $\mathrm{ls}(x,y)$ represents linear list segments.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{ls}(y,z) \vdash \mathrm{ls}(y,z)}}{x = y \wedge \mathrm{ls}(y,z) \vdash \mathrm{ls}(y,z)}\,(Wk)}{x = y \wedge \mathrm{ls}(y,z) \vdash \mathrm{ls}(x,z)}\,(EQL)}{x = y \wedge \mathrm{emp} * \mathrm{ls}(y,z) \vdash \mathrm{ls}(x,z)}\,(EL) \quad \dfrac{\dfrac{\dfrac{\dfrac{}{x \mapsto x' \vdash x \mapsto x'} \quad \underline{\mathrm{ls}(x',y)} * \mathrm{ls}(y,z) \vdash \mathrm{ls}(x',z)\ (\dagger)}{x \mapsto x' * \underline{\mathrm{ls}(x',y)} * \mathrm{ls}(y,z) \vdash x \mapsto x' * \mathrm{ls}(x',z)}\,(*)}{x \neq y \wedge x \mapsto x' * \underline{\mathrm{ls}(x',y)} * \mathrm{ls}(y,z) \vdash x \mapsto x' * \mathrm{ls}(x',z)}\,(Wk)}{x \neq y \wedge x \mapsto x' * \underline{\mathrm{ls}(x',y)} * \mathrm{ls}(y,z) \vdash x \neq y \wedge x \mapsto x' * \mathrm{ls}(x',z)}\,(NEQ)}{x \neq y \wedge x \mapsto x' * \underline{\mathrm{ls}(x',y)} * \mathrm{ls}(y,z) \vdash \mathrm{ls}(x,z)}\,(PR)$$

$$\dfrac{\text{(left)} \qquad \text{(right)}}{\underline{\mathrm{ls}(x,y)} * \mathrm{ls}(y,z) \vdash \mathrm{ls}(x,z)\ (\dagger)}\,(Case)$$

The sequents marked (†) are corresponding bud and companion. This pre-proof contains only one infinite path, which contains the trace, indicated by underlines, progressing at (Case) infinitely many times. Hence, it satisfies the global trace condition.

Example 2. A cyclic proof of $\mathrm{ls}^3(x_1,y_1,x_1) \vdash \mathrm{ls}^3(y_1,x_1,y_1)$ is given as follows, where ls^3 is a ternary variant of list-segment predicate defined by

$$\begin{aligned}\mathrm{ls}^3(x,y,z) := {} & x = y \wedge y = z \wedge \mathrm{emp} \\ & \mid \exists x'.(x = y \wedge y \neq z \wedge x \mapsto x' * \mathrm{ls}^3(x',x',z)) \\ & \mid \exists x'.(x \neq y \ \wedge \ x \mapsto x' * \mathrm{ls}^3(x',y,z)).\end{aligned}$$

Intuitively, $\mathrm{ls}^3(x,y,z)$ represents a list segment from x to z through y, and has the same meaning as $\mathrm{ls}(x,y) * \mathrm{ls}(y,z)$.

$$\dfrac{\overset{(1)}{\mathrm{ls}^3(x_1,y_1,x_1) \vdash \mathrm{ls}(x_1,y_1) * \mathrm{ls}(y_1,x_1)} \quad \overset{(2)}{\mathrm{ls}(x_1,y_1) * \mathrm{ls}(y_1,x_1) \vdash \mathrm{ls}^3(y_1,x_1,y_1)}}{\mathrm{ls}^3(x_1,y_1,x_1) \vdash \mathrm{ls}^3(y_1,x_1,y_1)}\,(cut),$$

where the subproofs (1) and (2) are in Fig. 1.

(1)

$$\dfrac{\dfrac{\dfrac{}{x_1 \mapsto y_2 \vdash x_1 \mapsto y_2}\,(Id) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x_2 \mapsto x_3 \vdash x_2 \mapsto x_3}\,(Id) \qquad \dfrac{\underline{\mathrm{ls}^3(x_3, y_1, x_1)} \vdash \mathrm{ls}(x_3, y_1) * \mathrm{ls}(y_1, x_1) \cdots (\alpha)}{x_2 \mapsto x_3 * \underline{\mathrm{ls}^3(x_3, y_1, x_1)} \vdash x_2 \mapsto x_3 * \mathrm{ls}(x_3, y_1) * \mathrm{ls}(y_1, x_1)}\,(*)}{\vdots\ (PR), (NEQR), and (Wk)}}{x_2 \neq y_1 \wedge x_2 \mapsto x_3 * \underline{\mathrm{ls}^3(x_3, y_1, x_1)} \vdash \mathrm{ls}(x_2, y_1) * \mathrm{ls}(y_1, x_1)}\,(Case)}{\underline{\mathrm{ls}^3(x_2, y_1, x_1)} \vdash \mathrm{ls}(x_2, y_1) * \mathrm{ls}(y_1, x_1) \cdots (\alpha)}\,(Wk)}{x_1 \neq y_1 \wedge \mathrm{ls}^3(x_2, y_1, x_1) \vdash \mathrm{ls}(x_2, y_1) * \mathrm{ls}(y_1, x_1)}\,(*)}{x_1 \neq y_1 \wedge x_1 \mapsto x_2 * \mathrm{ls}^3(x_2, y_1, x_1) \vdash x_1 \mapsto x_2 * \mathrm{ls}(x_2, y_1) * \mathrm{ls}(y_1, x_1)}\,(NEQR)}{x_1 \neq y_1 \wedge x_1 \mapsto x_2 * \mathrm{ls}^3(x_2, y_1, x_1) \vdash x_1 \neq y_1 \wedge x_1 \mapsto x_2 * \mathrm{ls}(x_2, y_1) * \mathrm{ls}(y_1, x_1)}\,(PR)}{x_1 \neq y_1 \wedge x_1 \mapsto x_2 * \mathrm{ls}^3(x_2, y_1, x_1) \vdash \mathrm{ls}(x_1, y_1) * \mathrm{ls}(y_1, x_1)}\,(Case)}{\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}(x_1, y_1) * \mathrm{ls}(y_1, x_1)}$$

(other cases) (other cases)

(2)

$$\dfrac{\dfrac{\dfrac{}{y_1 \mapsto y_2 \vdash y_1 \mapsto y_2}\,(Id) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\underline{\mathrm{ls}(x_2, y_1)} \vdash \mathrm{ls}^3(x_2, x_2, y_1) \cdots (\gamma)}{\vdots}}{(\text{case } x_1 = y_1)} \qquad \dfrac{}{(\text{case } x_1 \mapsto x_2)}\,(Case)}{\dfrac{\underline{\mathrm{ls}(x_1, y_1)} \vdash \mathrm{ls}^3(x_1, x_1, y_1) \cdots (\gamma)}{y_2 = x_1 \wedge \mathrm{ls}(x_1, y_1) \vdash \mathrm{ls}^3(x_1, x_1, y_1)}\,(Wk)}{y_2 = x_1 \wedge \mathrm{ls}(x_1, y_1) \vdash \mathrm{ls}^3(y_2, x_1, y_1)}\,(EQL) \qquad \dfrac{\underline{\mathrm{ls}(y_3, x_1)} * \mathrm{ls}(x_1, y_1) \vdash \mathrm{ls}^3(y_3, x_1, y_1) \cdots (\beta)}{\vdots\ (\text{case } y_2 \mapsto y_3)}}{\underline{\mathrm{ls}(y_2, x_1)} * \mathrm{ls}(x_1, y_1) \vdash \mathrm{ls}^3(y_2, x_1, y_1) \cdots (\beta)}\,(Case)}{y_1 \mapsto y_2 * \mathrm{ls}(y_2, x_1) * \mathrm{ls}(x_1, y_1) \vdash y_1 \mapsto y_2 * \mathrm{ls}^3(y_2, x_1, y_1)}\,(*)}{\vdots\ (PR),\ (NEQR), \text{and } (Wk)}}{y_1 \neq x_1 \wedge y_1 \mapsto y_2 * \mathrm{ls}(x_1, y_1) * \mathrm{ls}(y_2, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)}\,(Case)}{\mathrm{ls}(x_1, y_1) * \mathrm{ls}(y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)}$$

(case $x_1 = y_1$)

Fig. 1. Subproofs for $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$

The sequents marked (α), (β), and (γ) are each corresponding bud and companion, which make three infinite paths separately. Since each infinite path contains infinitely progressing trace, this satisfies the global trace condition.

This cyclic proof of $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ contains the (cut) with the cut formula $\mathrm{ls}(x_1, y_1) * \mathrm{ls}(y_1, x_1)$, which contains the predicate ls which does not occur in the conclusion. However, we can also construct another proof with the cut formula $\mathrm{ls}^3(x_1, y_1, y_1) * \mathrm{ls}^3(y_1, x_1, x_1)$ since $\mathrm{ls}^3(x, y, y)$ has the same meaning as $\mathrm{ls}(x, y)$.

Theorem 1 (Soundness of CSL_1ID^ω). *If (D, R) is a cyclic proof in CSL_1ID^ω, then every sequent in D is valid.*

The proof of this theorem is similar to [6, 4].

3 Quasi cut-elimination property

3.1 Failure of cut elimination in CSL_1ID^ω

As [13] shows, we cannot eliminate cut for CSL_1ID^ω.

Theorem 2 ([13]). *CSL_1ID^ω does not satisfy cut-elimination property.*

For the predicates lsne and slne defined in Section 1, the sequent $\mathrm{lsne}(x, y) \vdash \mathrm{slne}(x, y)$ is provable, but it cannot be proved in CSL_1ID^ω without cut.

The cut rule cannot be completely removed from CSL_1ID^ω, so, as a workaround, it is expected to achieve some restriction on the cut rule which does not interfere with automatic proof search.

3.2 Presumable cut and quasi cut-elimination property

In this section, we propose a restriction on the cut rule. We can consider this restriction not only for CSL_1ID^ω, but also for usual sequent-calculus-style proof system S and some suitable notion of subformulas $\mathrm{ExSub}(A)$. We call inference rules except for cut *non-cut rules*.

Definition 10 (Presumable formula). *Let $A \vdash B$ be a sequent in S. We inductively define the set $\mathcal{R}_S(A \vdash B)$ of* reachable sequents *from $A \vdash B$ in S, and the set $\mathcal{P}_S(A \vdash B)$ of* presumable formulas *from $A \vdash B$ simultaneously as the smallest set which satisfies the following property:*

(a) $A \vdash B \in \mathcal{R}_S(A \vdash B)$.

(b) If $\dfrac{A_1'' \vdash B_1'' \quad \cdots \quad A_n'' \vdash B_n''}{A' \vdash B'}\,(r)$ *where (r) is either a non-cut rule in S or the cut rule of which cut formula is in $\mathcal{P}_S(A \vdash B)$, then $A_i'' \vdash B_i'' \in \mathcal{R}_S(A \vdash B)$ for $1 \le i \le n$.*

(c) *If* $A' \vdash B' \in \mathcal{R}_S(A \vdash B)$ *and* $C \in \mathrm{ExSub}(A') \cup \mathrm{ExSub}(B')$*, then* $C \in \mathcal{P}_S(A \vdash B)$.

A presumable cut (pcut) *in a proof of* $A \vdash B$ *is a cut of which the cut formula is presumable from* $A \vdash B$.

Definition 11 (Quasi cut-elimination property). *If every* $A \vdash B$ *which is provable in* S *with cuts can be proved only with presumable cuts, we say that* S *satisfies the quasi cut-elimination property.*

Remark 1. The counterexample $\mathrm{lsne}(x, y) \vdash \mathrm{slne}(x, y)$ to the cut elimination of CSL_1ID^ω can be proved only with presumable cuts as follows.

$$\dfrac{\dfrac{\dfrac{}{x \mapsto y \vdash x \mapsto y}}{x \mapsto y \vdash \mathrm{slne}(x, y)}\,(PR) \qquad \dfrac{\dfrac{\dfrac{}{x \mapsto x' \vdash x \mapsto x'} \quad \mathrm{lsne}(x', y) \vdash \mathrm{slne}(x', y)\ (\dagger)}{x \mapsto x' * \mathrm{lsne}(x', y) \vdash x \mapsto x' * \mathrm{slne}(x', y)}\,(*) \qquad \dfrac{\dfrac{}{x \mapsto x' * \mathrm{slne}(x', y) \vdash x \mapsto x' * \mathrm{slne}(x', y)}\,(PR)}{x \mapsto x' * \mathrm{slne}(x', y) \vdash \mathrm{slne}(x, y)}\,(pcut)}{x \mapsto x' * \mathrm{lsne}(x', y) \vdash \mathrm{slne}(x, y)}\,(Case)}{\mathrm{lsne}(x, y) \vdash \mathrm{slne}(x, y)\ (\dagger)}$$

We use $x \mapsto x' * \mathrm{slne}(x', y)$ as the cut formula in this proof. By applying (PR) to $\mathrm{lsne}(x, y) \vdash \mathrm{slne}(x, y)$, we have $\mathrm{lsne}(x, y) \vdash x \mapsto x' * \mathrm{slne}(x', y)$ as its assumption. Hence $x \mapsto x' * \mathrm{slne}(x', y)$ is a presumable from $\mathrm{lsne}(x, y) \vdash \mathrm{slne}(x, y)$.

3.3 Restricting cuts in sequent calculi

In the bottom-up proof search, it is hard to apply the cut rule since we have to find a cut formula. If candidates of the cut formula are not limited, we have to find a suitable cut formula by heuristics. The notion of the presumable cuts is intended to limit the range of the cut formulas to the formulas which can occur in a sequent under the cut.

Such restriction is not very new, and we can find a similar restriction in the proof system for symbolic heaps in [7]. They did not explicitly describe, but their system can be seen as a cyclic proof system with a restricted cut rule, where one of assumptions is a bud whose companion is located below the cut. We call this restricted cut the *normal bud cut*. The cut formula must occur below the cut, and hence the normal bud cut can be seen as a stricter restriction than the presumable cut. They showed that $\mathrm{lsne}(x, y) \vdash \mathrm{slne}(x, y)$ is provable with normal bud cuts.

One may wonder what happens if we remove the condition "normal" from the normal bud cut, that is, if we consider only cuts where one of assumptions is a bud whose companion is located anywhere in the proof. In fact such restriction does not restrict anything on cut formulas, since any cut in a cyclic proof can be transformed to a "bud cuts" in a cyclic proof forest with the same cut formula as in Fig. 2, where both of two cuts in the right proof forest are cuts where one of assumptions is a bud.

We can also find another similar restriction on cuts for sequent calculus of propositional modal logics [16]. For modal logics S5, K4B, and so on, the sequent calculi do not satisfy the cut elimination property. Takano [16] proposed the

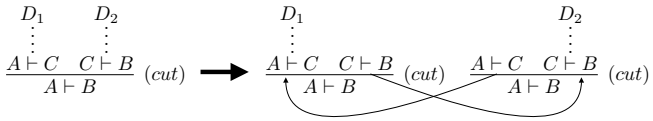

Fig. 2. Cut to bud cuts

restriction on cuts where the cut formula must be a subformula of the bottom sequent, and the restricted cuts and full cuts have the same provability power. In those systems (and usual sequent calculi for propositional logics), every non-cut rule satisfies the subformula property in a local sense, that is, every formula in the assumptions occurs in the conclusion as its subformula, so such restriction on cut formulas is stricter than the presumable cuts. Hence, Takano's results shows that these systems are examples of sequent calculi satisfying the quasi cut-elimination whereas not satisfying the full cut-elimination.

4 Failure of Quasi cut elimination in CSL_1ID^ω

In this section, we show that $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ is a counterexample to the quasi cut-elimination property for CSL_1ID^ω, where ls^3 is defined in example 2.

Before the proof, we explain why $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ is a counterexample. Similarly to the counterexample $\mathrm{lsne}(x, y) \vdash \mathrm{slne}(x, y)$ to the cut-elimination property, $\mathrm{ls}^3(x_1, y_1, x_1)$ and $\mathrm{ls}^3(y_1, x_1, y_1)$ have the same meaning but are syntactically different. We can unfold the former only at x_1 and the latter at y_1, and hence, when we prove $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ its proof has the following form with the cut formula such as $\mathrm{ls}^3(x_1, y_1, y_1) * \mathrm{ls}^3(y_1, x_1, x_1)$, which can be unfolded at both x_1 and y_1:

$$\dfrac{\begin{array}{cc} \begin{array}{c} D_1 \\ \vdots \\ \mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(x_1, y_1, y_1) * \mathrm{ls}^3(y_1, x_1, x_1) \end{array} & \begin{array}{c} D_2 \\ \vdots \\ \mathrm{ls}^3(x_1, y_1, y_1) * \mathrm{ls}^3(y_1, x_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1) \end{array} \end{array}}{\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)}\ (cut),$$

where we can construct a cycle within D_1 by unfolding at x_1, and another cycle within D_2 by unfolding at y_1. However, such a formula $\mathrm{ls}^3(x_1, y_1, y_1) * \mathrm{ls}^3(y_1, x_1, x_1)$ is not presumable from $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$. Hence, we cannot construct the proof of this form with only presumable cuts.

In the following, we prove that $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ is a counterexample to the quasi cut-elimination property. We have already shown in Example 2 that it is provable with cuts, so we will prove that $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ is not provable with only presumable cuts.

The rules (Case) and (PR) for ls^3 are the following:

$$\dfrac{e_a \quad e_b \quad e_c}{A * \mathrm{ls}^3(x, y, z) \vdash B}\ (Case),$$

where e_a is $x = y \land y = z \land A * \mathrm{emp} \vdash B$, e_b is $x = y \land y \neq z \land A * x \mapsto x' * \mathrm{ls}^3(x', x', z) \vdash B$, and e_c is $x \neq y \land A * x \mapsto x' * \mathrm{ls}^3(x', y, z) \vdash B$ for fresh x'.

$$\frac{A \vdash x = y \land y = z \land \mathrm{emp}}{A \vdash B * \mathrm{ls}^3(x, y, z)}\ (PR1),$$

$$\frac{A \vdash x = y \land y \neq z \land x \mapsto t * \mathrm{ls}^3(t, t, z)}{A \vdash B * \mathrm{ls}^3(x, y, z)}\ (PR2),$$

$$\frac{A \vdash B * x \neq y \land x \mapsto t * \mathrm{ls}^3(t, y, z)}{A \vdash B * \mathrm{ls}^3(x, y, z)}\ (PR3),$$

where t is an arbitrary term.

We suppose for a contradiction that there is a cyclic proof (D,R) of $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ in CSL_1ID^ω with only presumable cuts. In the following Lemma 1, we prove that the presumable formulas for $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ is limited to the form defined in Definition 12. In Lemma 6, we will see that (D,R) has an infinite path $(e_i)_{i\geq 0}$, which passes through right branches of (pcut) infinitely many times, which contradicts the global trace condition. Therefore, CSL_1ID^ω does not satisfy the quasi cut-elimination property.

We call a spatial formula *predicate-free* if it contains no predicate symbol except $\mapsto$.

Definition 12. *We define the set $\mathcal{S}$ of symbolic heaps as $\mathcal{S} = \{\Pi \land \Sigma_{pf} * \mathrm{ls}^3(t_1, t_2, t_3) \mid \Sigma_{pf}$ is predicate free, and t_1, t_2, t_3 are terms$\} \cup \{\Pi \land \Sigma_{pf} \mid \Sigma_{pf}$ is predicate free$\}$.*

Lemma 1 (Presumable formulas). *We have $\mathcal{P}_{CSL_1ID^\omega}(\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)) \subseteq \mathcal{S}$.*

Proof. Let e_0 be $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ and $\mathcal{S}_\vdash$ be $\{A' \vdash B' \mid A', B' \in \mathcal{S}\}$. We can prove $\mathcal{P}_{CSL_1ID^\omega}(e_0) \subseteq \mathcal{S}$ and $\mathcal{R}_{CSL_1ID^\omega}(e_0) \subseteq \mathcal{S}_\vdash$ by induction on $\mathcal{P}_{CSL_1ID^\omega}(e_0)$ and $\mathcal{R}_{CSL_1ID^\omega}(e_0)$. □

Then, we define the path $(e_i)_{i\geq 0}$ in (D, R).

Definition 13. *The path $(e_i)_{i\geq 0}$ in the cyclic proof (D, R) of $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ is defined as follows:*

(a) e_0 is defined as $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$.
(b) When e_i is a conclusion $A * \mathrm{ls}^3(x, y, z) \vdash B$ of (Case), one of the assumptions is of the form $A * x \neq y \land x \mapsto x' * \mathrm{ls}^3(x', y, z) \vdash B$. Define e_{i+1} as this assumption.
(c) When e_i is a conclusion of (∗), the rule application is of the form

$$\frac{A \vdash C \quad B \vdash D}{A * B \vdash C * D}\ (*).$$

If the spatial part of B or D is emp*, define e_{i+1} as $A \vdash C$. Otherwise, define e_{i+1} as $B \vdash D$.*

(d) When e_i is a conclusion of (pcut), the rule application is of the form

$$\frac{A \vdash C \quad C \vdash B}{A \vdash B}\ (pcut).$$

If the spatial formula of C is the same as B, define e_{i+1} as $A \vdash C$. Otherwise, define e_{i+1} as $C \vdash B$.

(e) When e_i is a bud, define e_{i+1} as the companion of e_i.

(f) For the other rules, there is exactly one assumption, and define e_{i+1} as it.

Lemma 2. *Let (s,h) be a heap model.*

1. *If Σ_{pf} is predicate-free, x occurs in Σ_{pf}, and $s, h \models \Sigma_{pf}$, we have $s(x) \in \mathrm{dom}(h) \cup \mathrm{img}(h)$.*
2. *If $s, h \models \mathrm{ls}^3(t_1, t_2, t_3)$, we have either $s(t_i) \in \mathrm{dom}(h) \cup \mathrm{img}(h)$ or $s(t_1) = s(t_2) = s(t_3)$.*

Definition 14. *A pure formula Π is called* injective if $t = u \notin \Pi$ for any $t \not\equiv u$.

Let x_1, x_2,... be fixed variables. We introduce the following abbreviations: For $m \geq 1$, $x_1 \mapsto^* x_m * \mathrm{ls}^3(x_m, t, t')$ denotes either $\mathrm{ls}^3(x_1, t, t')$ or $x_1 \mapsto x_2 * x_2 \mapsto x_3 * \cdots * x_{m-1} \mapsto x_m * \mathrm{ls}^3(x_m, t, t')$ (for $m \geq 2$). **emp** denotes $\mathrm{emp} * \mathrm{emp} * \cdots * \mathrm{emp}$.

Definition 15. *For natural numbers $m < n \leq k$, we define the set $\mathcal{H}_{m,n,k}$ of heap models as follows:*

$$\mathcal{H}_{m,n,k} = \{(s,h) \mid s \text{ is injective, } s(x_i) = i\ (1 \leq i \leq m),\ s(y_1) = n \\ \mathrm{dom}(h) = [1,k],\ h(i) = i+1\ (1 \leq i \leq k-1), \text{ and } h(k) = 1\},$$

where $[n, m]$ denotes the interval $\{i \mid n \leq i \leq m\}$ for natural numbers n and m.

Note that for any $m < n \leq k$, $\mathcal{H}_{n,m,k}$ is trivially non-empty, and the size of $\mathrm{dom}(h)$ is k for any $(s,h) \in \mathcal{H}_{n,m,k}$.

Lemma 3. *Suppose that Π is injective, $m < n \leq k$, and $(s,h) \in \mathcal{H}_{m,n,k}$. We have $s, h \models \Pi \wedge x_1 \mapsto^* x_m * \mathrm{ls}^3(x_m, y_1, x_1)$.*

Lemma 4. *Suppose that $m < n \leq k$, $(s,h) \in \mathcal{H}_{m,n,k}$, $h = h_1 + h_2$, and $s(t_i) \in [1,k]$ for $1 \leq i \leq 3$. $s, h_1 \models \mathrm{ls}^3(t_1, t_2, t_3)$ holds if and only if (s, h_1) satisfies one of the following conditions:*

(1) $s(t_1) = s(t_2) = s(t_3)$ and $\mathrm{dom}(h_1) = \emptyset$,

(2) $s(t_3) = 1$, not $s(t_1) = s(t_2) = s(t_3)$, $\mathrm{dom}(h_1) = [s(t_1), k]$, and $s(t_2) \in \mathrm{dom}(h_1)$,

(3) $1 < s(t_3) \leq s(t_1)$, not $s(t_1) = s(t_2) = s(t_3)$, $\mathrm{dom}(h_1) = [1, s(t_3) - 1] \cup [s(t_1), k]$, and $s(t_2) \in \mathrm{dom}(h_1)$,

(4) $s(t_1) < s(t_3)$, $\mathrm{dom}(h_1) = [s(t_1), s(t_3) - 1]$, and $s(t_2) \in \mathrm{dom}(h_1)$.

Lemma 5. *Let C be a formula of the form $\Pi \wedge \Sigma_{pf} * \mathrm{ls}^3(t_1, t_2, t_3)$, where Σ_{pf} is predicate-free. If Π_1 and Π_2 are injective, and*

$$\Pi_1 \wedge x_1 \mapsto^* x_m * \mathrm{ls}^3(x_m, y_1, x_1) * \mathbf{emp} \vdash C \text{ and}$$

$$C \vdash \Pi_2 \wedge \mathrm{ls}^3(y_1, x_1, y_1) * \mathbf{emp}$$

are valid, then C is satisfiable and of the form either

$$\Pi \wedge x_1 \mapsto^* x_n * \mathrm{ls}^3(x_n, y_1, x_1) * \mathbf{emp} \text{ for some } n \leq m, \text{ or}$$

$$\Pi \wedge \mathrm{ls}^3(y_1, x_1, y_1) * \mathbf{emp},$$

and Π is injective.

Proof. Let A be $\Pi_1 \wedge x_1 \mapsto^* x_m * \mathrm{ls}^3(x_m, y_1, x_1) * \mathbf{emp}$, B be $\Pi_2 \wedge \mathrm{ls}^3(y_1, x_1, y_1) * \mathbf{emp}$, and V be $\{x_1, \ldots, x_m, y_1\}$, which is the set of variables contained in the spatial part of A.

By lemma 3, every $(s, h) \in \mathcal{H}_{m,n,k}$ $(m < n \leq k)$ is a model of A, and $A \vdash C$ is valid, and hence (s, h) is also a model of C. It implies that C is satisfiable.

(i) We show that Π is injective. Assume that there are t and u such that $t = u \in \Pi$ and $t \not\equiv u$. For $(s, h) \in \mathcal{H}_{m,m+1,m+1}$, we have $s \models \Pi$, and hence $s(t) = s(u)$, which contradicts that s is injective.

(ii) We show that either $t_1 \not\equiv t_2$ or $t_2 \not\equiv t_3$ holds. Otherwise, we have $t_1 \equiv t_2 \equiv t_3$. For any $(s, h) \in \mathcal{H}_{m,m+1,k}$, we have $s, h \models \Sigma_{pf} * \mathrm{ls}^3(t_1, t_1, t_1)$. By the definition of ls^3, we have $s, h \models \Sigma_{pf}$. In particular, there are heap models $(s, h_1) \in \mathcal{H}_{m,m+1,m+1}$ and $(s, h_2) \in \mathcal{H}_{m,m+1,m+2}$ such that $s, h_1 \models \Sigma_{pf}$ and $s, h_2 \models \Sigma_{pf}$. However, the sizes of $\mathrm{dom}(h_1)$ and $\mathrm{dom}(h_2)$ are $m + 1$ and $m + 2$, respectively, which contradicts that Σ_{pf} is predicate-free.

(iii) We show that any variable in $\Sigma_{pf} * \mathrm{ls}^3(t_1, t_2, t_3)$, which is the spatial part of C, are in V. Assume that z is a variable in $\Sigma_{pf} * \mathrm{ls}^3(t_1, t_2, t_3)$ but not in V. For any $(s, h) \in \mathcal{H}_{m,m+1,m+1}$ we have $s, h \models C$. Since $z \notin V$, for any $i \notin \mathrm{dom}(h) \cup \mathrm{img}(h) \cup \mathrm{img}(s)$, $s' = s[z \mapsto i]$ is injective, and $s', h \in \mathcal{H}_{m,m+1,m+1}$, and hence $s', h \models C$. Since $s'(z) = i \notin \mathrm{dom}(h) \cup \mathrm{img}(h)$, by Lemma 2.1, z is not contained in Σ_{pf}. Therefore, z is either t_1, t_2, or t_3. By Lemma 2.2, $t_1 \equiv t_2 \equiv t_3$ holds, which contradicts (ii).

(iv) We show that both x_1 and y_1 occur in $\Sigma_{pf} * \mathrm{ls}^3(t_1, t_2, t_3)$. Assume that x_1 does not occur in $\Sigma_{pf} * \mathrm{ls}^3(t_1, t_2, t_3)$. For any $(s, h) \in \mathcal{H}_{m,m+1,m+1}$, we have $s, h \models C$. For any $i \notin \mathrm{dom}(h) \cup \mathrm{img}(h) \cup \mathrm{img}(s)$, $s' = s[x_1 \mapsto i]$ is injective, and we have $s', h \models C$. Since $C \models B$ is valid, $s', h \models B$ holds, and hence $s', h \models \mathrm{ls}^3(y_1, x_1, y_1)$. Since s' is injective, we have $s'(y_1) \neq s'(x_1)$. By Lemma 2, $s'(x_1) \in \mathrm{dom}(h) \cup \mathrm{img}(h)$, which contradicts $s'(x_1) = i \notin \mathrm{dom}(h) \cup \mathrm{img}(h)$. For y_1, it is similarly proved.

(v) We show that $y_1 \mapsto t$ does not occur in C. For any $(s, h) \in \mathcal{H}_{m,m+1,m+2}$, we have $s, h \models C$. If $y_1 \mapsto t$ occurs in C, we have $h(s(y_1)) = s(t)$. By definition of $\mathcal{H}_{m,m+1,m+2}$, we have $s(t) = m + 2$. By (iii), we have $t \in V$, and hence $s(t) \in [1, m] \cup \{m + 1\}$, which contradicts $s(t) = m + 2$.

(vi) We show that $t \mapsto y_1$ does not occur in C. For any $(s,h) \in \mathcal{H}_{m,m+2,m+2}$, we have $s,h \models C$. If $t \mapsto y_1$ occurs in C, we have $h(s(t)) = s(y_1)$. By definition of $\mathcal{H}_{m,m+1,m+2}$, we have $s(t) = m+1$. By (iii), we have $t \in V$, and hence $s(t) \in [1,m] \cup \{m+2\}$, which contradicts $s(t) = m+1$.

(vii) We show that, if $x_i \mapsto x_j$ occurs in C, we have $1 \leq i \leq m-1$ and $j = i+1$. For any $(s,h) \in \mathcal{H}_{m,m+1,m+1}$, we have $s,h \models C$. If $x_i \mapsto x_j$ occurs in C, we have $h(s(x_i)) = s(x_j)$. By definition of $\mathcal{H}_{m,m+1,m+1}$, we have $i+1 = j$ and $1 \leq i \leq m-1$.

By (i)–(vii), we prove the lemma. For $(s,h) \in \mathcal{H}_{m,m+2,m+2}$, since we have $s,h \models C$, we can divide h to h_1 and h_2 such that $h = h_1 + h_2$, $s,h_1 \models \mathrm{ls}^3(t_1,t_2,t_3)$, and $s,h_2 \models \Sigma_{pf}$. Then, (s,h_1) satisfies one of four conditions of Lemma 4. Since Π is injective, $s(t_1) = s(t_2) = s(t_3)$ does not hold by (ii), so (1) of Lemma 4 is not the case. If $m+2 \in \mathrm{dom}(h_2)$, Σ_{pf} contains $z \mapsto t$ for some $z \in V$ by (iii) such that $s(z) = m+2$. Then, z must be y_1, which contradicts (v). Hence $m+2 \in \mathrm{dom}(h_1)$ holds, so (4) of Lemma 4 is not the case since $s(t_3) - 1 < m+2$. Therefore, we have either

(a) $\mathrm{dom}(h_1) = [1, s(t_3)-1] \cup [s(t_1), m+2]$ and $1 \neq s(t_3) \leq s(t_1)$, or
(b) $\mathrm{dom}(h_1) = [s(t_1), m+2]$ and $s(t_3) = 1$.

Case (a). Since $s(x_1) = 1 \notin \mathrm{dom}(h_2) \cup \mathrm{img}(h_2)$ holds by the definition of $\mathcal{H}_{m,m+2,m+2}$, x_1 does not occur in Σ_{pf} by Lemma 2.1, and hence either t_1 or t_2 is x_1 by (iv). Since we have $1 \leq s(t_3) - 1 < s(t_1)$ in this case, we have $t_2 \equiv x_1$. By (iv), (v), (vi), either t_1 or t_3 is y_1. If $t_3 \equiv y_1$ holds, we have $t_1 \equiv y_1$ since $s(t_3) \leq s(t_1)$ holds. If $t_1 \equiv y_1$ and $t_3 \not\equiv y_1$ hold, we have $m+1 \in \mathrm{dom}(h_2)$. However, there is no $z \in V$ such that $s(z) = m+1$. Therefore, we have $t_1 \equiv t_3 \equiv y_1$. In this case, $\mathrm{dom}(h_1) = [1, m+2]$ and $\mathrm{dom}(h_2) = \emptyset$ hold, and hence C is of the form $\Pi \wedge \mathrm{ls}^3(y_1,x_1,y_1) * \mathbf{emp}$.

Case (b). We have $t_3 \equiv x_1$ since $s(t_3) = 1$. If t_1 is y_1, then $m+1 \in \mathrm{dom}(h_2)$ holds, and then Σ_{pf} must contain $z \mapsto t$ for some $z \in V$ by (iii) such that $s(z) = m+1$, but there is no such z. Hence we have $t_1 \not\equiv y_1$, and then we have $t_2 \equiv y_1$ by (iv), (v), (vi). Since $t_1 \in V - \{y_1\}$ holds, we have $t_1 \equiv x_n$ for some $n \leq m$. Then, we have $\mathrm{dom}(h_1) = [n, m+2]$ and $\mathrm{dom}(h_2) = [1, n-1]$. By (iii) and (vii), the form of C is $\Pi \wedge x_1 \mapsto^* x_n * \mathrm{ls}^3(x_n,y_1,x_1) * \mathbf{emp}$.

From the above, the form of C is either

$$\Pi \wedge x_1 \mapsto^* x_n * \mathrm{ls}^3(x_n,y_1,x_1) * \mathbf{emp} \text{ for some } n, \text{ or}$$

$$\Pi \wedge \mathrm{ls}^3(y_1,x_1,y_1) * \mathbf{emp}$$

and Π is injective by (i). □

Lemma 6. *Every sequent e_i in the path $(e_i)_{i \geq 0}$ is of the following form:*

$$\Pi_1 \wedge x_1 \mapsto^* x_m * \mathrm{ls}^3(x_m,y_1,x_1) * \mathbf{emp} \vdash \Pi_2 \wedge \mathrm{ls}^3(y_1,x_1,y_1) * \mathbf{emp}, \quad (\dagger)$$

where Π_1 and Π_2 are injective and the antecedent of e_i is satisfiable. Hence, e_i is not an axiom and the path $(e_i)_{i \geq 0}$ is an infinite path.

Proof. First, the antecedent of (†) is satisfiable by Lemma 3.

Secondly, we show that e_i is of the form (†) by induction on i. For $e_0 \equiv \mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$, it is trivial. Under the assumption that e_n is of the form of (†), we prove that for e_{n+1}. We will give only nontrivial cases.

Case (PR). Any assumption of (PR) whose conclusion is e_n must be invalid, so it is not the case.

Case (pcut). By Lemma 1, the cut formula is of the form either $\Pi \wedge \Sigma_{pf} * \mathbf{emp}$ or $\Pi \wedge \Sigma_{pf} * \mathrm{ls}^3(t_1, t_2, t_3) * \mathbf{emp}$, where Σ_{pf} is predicate-free. For the former case, we can see that one of the assumptions is invalid, so it is not the case. For the latter case, e_{n+1} is of the form (†) by Lemma 5. □

Lemma 7. *The path $(e_i)_{i\geq 0}$ passes through right branches of (pcut) infinitely many times.*

Proof. By Lemma 6, $(e_i)_{i\geq 0}$ is an infinite path, hence (Case) must be applied to $(e_i)_{i\geq 0}$ infinitely many times by the global trace condition. When (Case) is applied, the number of $\mapsto$ increases. Any proof trees of cyclic proofs are finite, hence the number of $\mapsto$ in $(e_i)_{i\geq 0}$ must decrease infinitely many times.

By Lemma 6, any sequents on the path $(e_i)_{i\geq 0}$ is form (†) and the number of $\mapsto$ in $(e_i)_{i\geq 0}$ decreases only when it passes through the right branch of (pcut). Hence the path $(e_i)_{i\geq 0}$ must pass through right branches of (pcut) infinitely many times. □

Theorem 3 (Failure of quasi cut elimination for CSL_1ID^ω). *CSL_1ID^ω does not satisfy the quasi cut-elimination property.*

Proof. $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ is a counterexample. First, $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ can be proved in CSL_1ID^ω as in Example 2. Secondly, we show $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ can not be proved in CSL_1ID^ω with only (pcut). Suppose that there is a cyclic proof (D,R) of $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ with only (pcut). By Lemma 7, (D,R) has a path which passes through right branches of (pcut) infinitely many times. However, at each right branch of (pcut), no trace is connected, and hence (D,R) cannot satisfy the global trace condition, which contradicts that (D,R) is a cyclic proof. □

Even if we change the definition of the extended subformula from renaming variables to replacing variables by arbitrary terms, the above proof shows that CSL_1ID^ω does not satisfy the quasi cut-elimination property, since all of the presumable formulas from $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$ still remain in $\mathcal{S}$ after the change of the definition. Hence, the restriction of [7] properly changes the provability from the system with full cuts.

Corollary 1. *In CSL_1ID^ω, if we impose the condition that one of assumption of a cut is a bud whose companion is located below the cut, the provability properly weakens.*

5 Concluding remark

This paper has proposed a restriction for the cut rule, called presumable cut, in sequent-calculus-style proof systems, and a relaxed variant of the cut-elimination, the quasi cut-elimination property. In general, the cut rule requires some heuristics to find the cut formula in the bottom-up proof search, whereas the presumable cut limits the cut formula to those which can occur below the cut.

It has been shown in [13] that the cyclic proof systems for symbolic heaps in separation logic do not enjoy the cut-elimination property, and we hope that those enjoy the quasi cut-elimination property. However, this paper showed that a cyclic proof system CSL_1ID^ω does not enjoy the quasi cut-elimination by giving a counterexample $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$. It shows a limit on automatic proof search in cyclic proof systems.

As future work, we will study on the quasi cut-elimination property for other cyclic proof systems such as the logic of bunched implications [3] and the first-order predicate logic. Another direction is relaxing the condition of presumable cuts. For the example $\mathrm{ls}^3(x_1, y_1, x_1) \vdash \mathrm{ls}^3(y_1, x_1, y_1)$, we can prove it in CSL_1ID^ω with the cut formula $\mathrm{ls}^3(x_1, y_1, y_1) * \mathrm{ls}^3(y_1, x_1, x_1)$, which is a separating conjunction of two presumable formulas. It is an interesting question whether we can restrict the cut formulas to separating conjunctions of some bounded number of presumable formulas.

References

1. Berdine, J., Calcagno, C., O'Hearn, P.W.: A decidable fragment of separation logic. In: Proceedings of FSTTCS 2004. LNCS, vol. 3328, pp. 97–109 (2004)
2. Berdine, J., Calcagno, C., O'Hearn, P.W.: Symbolic execution with separation logic. In: Proceedings of APLAS 2005. LNCS, vol. 3780, pp. 52–68 (2005)
3. Brotherston, J.: Formalised inductive reasoning in the logic of bunched implications. In: Proceedings of SAS '07. LNCS, vol. 4634, pp. 87–103 (2007)
4. Brotherston, J., Distefano, D., Petersen, R.L.: Automated cyclic entailment proofs in separation logic. In: Proceedings of CADE-23. LNAI, vol. 6803, pp. 131–146 (2011)
5. Brotherston, J., Gorogiannis, N., Petersen, R.L.: A generic cyclic theorem prover. In: Proceedings of APLAS 2012. LNCS, vol. 7705, pp. 350–367 (2012)
6. Brotherston, J., Simpson, A.: Sequent calculi for induction and infinite descent. Journal of Logic and Computation **21**(6), 1177–1216 (2011)
7. Chu, D., Jaffar, J., Trinh, M.: Automatic induction proofs of data-structures in imperative programs. In: Proceedings of PLDI 2015. pp. 457–466 (2015)
8. Das, A., Pous, D.: Non-wellfounded proof theory for (kleene+action)(algebras+lattices). In: Proceedings of CSL '18. LIPIcs, vol. 119, pp. 19:01–19:18 (2018)
9. Doumane, A.: On the infinitary proof theory of logics with fixed points. Ph.D. thesis, Paris 7 (2017)
10. Iosif, R., Rogalewicz, A., Simacek, J.: The tree width of separation logic with recursive definitions. In: Proceedings of CADE-24. LNCS, vol. 7898, pp. 21–38 (2013)
11. Iosif, R., Rogalewicz, A., Vojnar, T.: Deciding entailments in inductive separation logic with tree automata. In: Proceedings of ATVA 2014. LNCS, vol. 8837, pp. 201–218 (2014)
12. Katelaan, J., Matheja, C., Zuleger, F.: Effective entailment checking for separation logic with inductive definitions. In: Proceedings of TACAS 2019. LNCS, vol. 11428, pp. 319–336 (2019)
13. Kimura, D., Nakazawa, K., Terauchi, T., Unno, H.: Failure of cut-elimination in cyclic proofs of separation logic. Comupter Software **37**(1), 39–52 (2020)
14. Kimura, D., Tatsuta, M.: Decidability for entailments of symbolic heaps with arrays. Available at `https://arxiv.org/abs/1802.05935` (2018)
15. Reynolds, J.C.: Separation logic: A logic for shared mutable data structures. In: Proceedings of LICS 2002. pp. 55–74 (2002)
16. Takano, M.: Subformula property as a substitute for cut-elimination in modal propositionl logics. Mathematical Japonica **37**, 1129–1145 (1992)
17. Tatsuta, M., Nakazawa, K., Kimura, D.: Completeness of cyclic proofs for symbolic heaps with inductive definitions. In: Proceedings of APLAS 2019 (to appear)