Failure of Cut-Elimination in Cyclic Proofs of Separation Logic

Daisuke Kimura[1], Koji Nakazawa[2], Tachio Terauchi[3], and Hiroshi Unno[4]

[1] Toho University kmr@is.sci.toho-u.ac.jp
[2] Nagoya University knak@i.nagoya-u.ac.jp
[3] Waseda University terauchi@waseda.jp
[4] Tsukuba University uhiro@cs.tsukuba.ac.jp

Abstract This paper studies the role of the cut rule in cyclic proof systems for separation logic. A cyclic proof system is a sequent-calculus style proof system for proving properties involving inductively defined predicates. Recently, there has been much interest in using cyclic proofs for proving properties described in separation logic with inductively defined predicates. However, it is not known whether such systems has the cut-elimination property, an important meta-theoretic property of proof systems. Cut-elimination is not only of theoretical interest, because needing arbitrary cuts would mean that there is a limit to what one would be able to prove by a naïve mechanical proof search.

In this paper, we answer the question by showing that the cut-elimination property fails in cyclic proof systems for separation logic. We present two systems, one for sequents with single-conclusion, and another for sequents with multiple-conclusions. To show the cut-elimination failure, we present concrete counter-example sequents which the systems can prove with cuts but not without cuts. The counter-examples are reasonably simple formulas about singly-linked lists, and therefore, suggest that the cut rule is important for a practical application of cyclic proofs to separation logic.

1 Introduction

Separation logic [13] is a popular program logic for reasoning about programs that use pointer data structures. In separation logic, reasoning about recursive data structures is made possible by augmenting the logic with *inductively defined predicates*. For example, a predicate which says that a pointer points to a list may be written as follows:

$$\mathbf{ls}(x, y) := x \mapsto y \mid x \mapsto z * \mathbf{ls}(z, y).$$

Here, $x \mapsto y$ means that the memory cell at address x contains the value y, and $A * B$ is a *separating conjunction* which says that the memory is a union of two memory regions h_A and h_B with disjoint domains such that h_A satisfies A and h_B satisfies B. Therefore, $\mathbf{ls}(x, y)$ says that x points to a singly-linked list that ends in y. (We refer to Section 2 for the formal definition of separation logic.)

In a sequent-calculus style proof system, it is customary to handle inductively defined predicates like $\mathbf{ls}(x, y)$ by adding a set of rules that introduce them to left and right sides of sequents. For $\mathbf{ls}(x, y)$, the right introduction rules are:

$$\frac{A \vdash x \mapsto y}{A \vdash \mathbf{ls}(x, y)} \qquad \frac{A \vdash x \mapsto z * \mathbf{ls}(z, y)}{A \vdash \mathbf{ls}(x, y)},$$

and the left introduction rule is:

$$\frac{A * x \mapsto y \vdash C[x, y] \quad A * x \mapsto z * C[z, y] \vdash C[x, y] \quad A * C[x, y] \vdash B}{A * \mathbf{ls}(x, y) \vdash B}\ (\text{Ind}),$$

where z is a fresh variable and $C[x, y]$ is a formula that may have free variables x and y. Note that the formula C in the rule (Ind) is an *induction hypothesis*. That is, the left premise of the rule says that C holds in the base case, and the middle premise encodes the inductive case which roughly says that if C holds for the smaller list from z to y then it also holds for the larger list from x to y. The rule is an obstacle to a mechanical proof search because one needs to guess an appropriate C.

A *cyclic proof system* [3, 5, 4] offers an alternative approach to doing proofs about inductively defined predicates in a sequent calculus. In this approach, the left introduction rule of an inductively defined predicate is replaced by a rule that directly encodes the inductive definition. For instance, the left introduction rule of $\mathbf{ls}(x, y)$ is given as

$$\frac{A * x \mapsto y \vdash B \quad A * x \mapsto z * \mathbf{ls}(z, y) \vdash B}{A * \mathbf{ls}(x, y) \vdash B}\ (\mathrm{UL}).$$

A proof search in a cyclic proof system starts from the root goal sequent, mechanically building the proof tree upwards by applying an applicable rule at each node. The search may stop by reaching a sequent (called a *bud*) that it has seen before (called a *companion*), thereby forming a "cyclic" proof that has an edge from a leaf bud node to an internal companion node. To ensure correctness, a certain condition, called *global trace condition*, is imposed on the cyclic structure (cf. Section 3 for details).

Importantly, in the absence of the cut rule, the possible children of a node can be syntactically determined from the (finitely many) rules applicable at the node, which substantially expediates the mechnichal proof search.[1] The property has been used to a great advantage by researchers of cyclic proofs, and they have proposed automatic theorem provers based on cyclic proofs [6, 7]. Furthermore, as we discuss below, some automatic induction-based provers for separation logic [11, 8, 16, 15] can also be seen as cyclic proof systems with restricted forms of cuts.

Meanwhile, cyclic proof systems are being intensively studied in the theoretical computer science community (for various logics, such as separation logic, first-order logic, and linear logic) [5, 3, 6, 7, 1, 2, 14, 12]. The research has lead to some remarkable results, such as showing that the cyclic proof system (with cuts) is strictly more powerful than the standard inductive first-order logic sequent calculus (i.e., that with a rule analogous to the (Ind) rule above) [5, 1]. However, some fundamental proof-theoretic properties, such as cut-elimination and completeness, still remain open.

As the main contribution of this paper, we show that *cut-elimination fails in cyclic proof systems for separation logic*. As remarked above, the result is not only of theoretical interest since the presence of the cut rule substantially affects a mechanical proof search process. We prove the result for two cyclic proof systems for separation logic: $\mathrm{CSL}_0\mathrm{ID}\omega$ which deduces sequents with single conclusions, and $\mathrm{CSL}_0^M\mathrm{ID}\omega$ which deduces sequents with multiple conclusions. We show the cut-elimination failure by presenting concrete counter-example sequents which the systems can prove with cuts but not without cuts. The counter-examples are fairly simple formulas about singly-linked lists. They contain three kinds of semantically equivalent predicates $\mathbf{ls}(x, y)$, $\mathbf{lsX}(x, y)$ and $\mathbf{sl}(x, y)$ each describing a singly-linked list from x to y. The predicate $\mathbf{ls}(x, y)$ is the usual list definition shown above, whereas $\mathbf{lsX}(x, y)$ defines a list to be either a list of an odd length or a list of an even length where odd and even length lists are defined inductively in a manner analogous to $\mathbf{ls}(x, y)$. The predicate $\mathbf{sl}(x, y)$ is a "backward" definition of a list whereby a list is constructed by adding an element to the tail rather than to the head. We show that the sequent $\mathbf{ls}(x, y) \vdash \mathbf{lsX}(x, y)$ is a counter-example to cut-elimination for $\mathrm{CSL}_0\mathrm{ID}\omega$, and the sequent $\mathbf{ls}(x, y) \vdash \mathbf{sl}(x, y)$ is that for $\mathrm{CSL}_0^M\mathrm{ID}\omega$. Thus, a practical implication of our results is that (some form of) a cut rule is necessary for designing useful theorem provers based on cyclic proofs, at least for separation logic.

[1] Technically, this is only true if structural rules such as weakening, contraction, and substitution are made implicit.

The rest of the paper is organized as follows. We discuss related work next. Section 2 introduces SL_0, a simple fragment of separation logic used in the rest of the paper. Section 3 presents $\mathrm{CSL}_0\mathrm{ID}\omega$ and shows our first main result which says that cut-elimination fails for $\mathrm{CSL}_0\mathrm{ID}\omega$. $\mathrm{CSL}_0\mathrm{ID}\omega$ is closely related to the system proposed by Brotherston et al. [6], and the section also shows that cut-elimination fails in their system as well. In Section 3, we present $\mathrm{CSL}_0^M\mathrm{ID}\omega$, and show that the counter-example for $\mathrm{CSL}_0\mathrm{ID}\omega$ (i.e. $\mathbf{ls}(x,y) \vdash \mathbf{lsX}(x,y)$) is cut-free provable in $\mathrm{CSL}_0^M\mathrm{ID}\omega$. Then, we show that $\mathrm{CSL}_0^M\mathrm{ID}\omega$ still fails to satisfy cut-elimination by presenting the cut-free unprovability of the counter-example $\mathbf{ls}(x,y) \vdash \mathbf{sl}(x,y)$. Section 5 concludes the paper with a discussion on future work.

Related Work: As remarked above, there has been much work on meta-theoretic properties of cyclic proof systems for various logics [5, 3, 6, 7, 1, 2, 14, 12, 10, 9]. Among them, some papers discuss the cut-elimination property for cyclic proof systems. The paper [10] considers a cyclic proof system for μMALL (linear logic with least and greatest fixed-point operators), and discusses non-preservability of the cyclic structure by the ordinary cut-elimination procedure. It discusses the behavior of cut-elimination procedure, but our paper considers admissibility of the cut rule. The paper [9] proposes a sequent-style system for Kleene algebra, and shows the cut rule is not admissible in the system.

In the context of (semi-)automated deduction, several cyclic-proof-based theorem provers for separation logic have been proposed [6, 11, 8, 16, 15]. Some of them allow restricted forms of cut. For instance, SLEEK [11] allows cuts, but only against user-provided lemmas. The theorem provers proposed in [8, 16, 15] synthesize induction hypotheses during the proof search by following a certain set of rules. They can be seen as a kind of a cyclic proof system in which the cuts are restricted to be only against the synthesized induction hypotheses. None of these papers investigate the effect of having or not having the cut rule nor discuss relation to restricted forms of cuts.

2 Simple fragment of separation logic SL_0

This section defines a simple fragment of separation logic (SL_0), which has the minimum necessary connectives $\mapsto$ and $*$ to define our counter-examples.

2.1 Syntax of SL_0

We assume a finite set $\{P_1, \ldots, P_K\}$ of *inductive predicates*. Each inductive predicate P has its arity $\mathbf{ar}(P)$. *Terms* of SL_0 (denoted by $t, u, \ldots$) consist of variables ($x, y, z \cdots$) and **nil**. We sometimes write $x \in \overrightarrow{x}$ if x appears in $\overrightarrow{x}$.

Formulas (denoted by $A, B, C, \ldots$) of SL_0 are defined as follows.

$$A ::= t \mapsto u \mid P(\overrightarrow{t}) \mid A * A,$$

where the length of $\overrightarrow{t}$ is $\mathbf{ar}(P)$. We sometimes write $A(\overrightarrow{x})$ to denote variables occurring in A explicitly. We implicitly suppose associativity and commutativity of the separating conjunction $*$, that is, $(A * B) * C$ and $B * (A * C)$ are identified.

The set of free variables in A is written as $\mathrm{FV}(A)$. The union of $\mathrm{FV}(A_1), \ldots, \mathrm{FV}(A_n)$ is written as $\mathrm{FV}(A_1, \ldots, A_n)$.

A substitution (denoted by θ) has the form $x_1 := t_1, \ldots, x_k := t_k$, where x_i and x_j are different variables if $i \neq j$. The formula obtained by replacing each x_i by t_i ($i = 1, \ldots, k$) in A is written by $A[x_1 := t_1, \ldots, x_k := t_k]$.

Each inductive predicate P has its own definition, which is given as follows:

$$P(\overrightarrow{x}) := A_1 \mid \ldots \mid A_s$$

Each A_i is called a definition clause of P. Intuitively, this means that $P(\overrightarrow{x})$ is defined by the disjunction of $A_1, \ldots, A_s$. We note that variables of A_j not appearing in $\overrightarrow{x}$ are implicitly existentially quantified. Namely, $P(\overrightarrow{x})$ is defined by $\exists \overrightarrow{y_1} A_1(\overrightarrow{x}, \overrightarrow{y_1}) \vee \ldots \vee \exists \overrightarrow{y_s} A_s(\overrightarrow{x}, \overrightarrow{y_s})$.

In this paper we will consider several kinds of list predicates **ls**, **sl**, **lsO**, **lsE** and **lsX** given below.

Definition 1. The definitions of **ls**, **sl**, **lsO**, **lsE**, and **lsX** are given as follows.

$\mathbf{ls}(x, y) := x \mapsto y \mid x \mapsto z * \mathbf{ls}(z, y)$
$\mathbf{sl}(x, y) := x \mapsto y \mid \mathbf{sl}(x, z) * z \mapsto y$
$\mathbf{lsO}(x, y) := x \mapsto y \mid x \mapsto z * \mathbf{lsE}(z, y)$
$\mathbf{lsE}(x, y) := x \mapsto z * \mathbf{lsO}(z, y)$
$\mathbf{lsX}(x, y) := \mathbf{lsO}(x, y) \mid \mathbf{lsE}(x, y)$

Both $\mathbf{ls}(x, y)$ and $\mathbf{sl}(x, y)$ mean singly-linked list segments of positive lengths from x to y. The former and the latter predicates represent list segments constructed by adding cells repeatedly to the head position and the tail position, respectively. $\mathbf{lsO}(x, y)$ and $\mathbf{lsE}(x, y)$ mean list segments with odd and positive even lengths, respectively. They are defined by a mutual induction. $\mathbf{lsX}(x, y)$ means list segments with odd or positive even length, that is, list segments of positive length. The formulas $\mathbf{ls}(x, y)$, $\mathbf{sl}(x, y)$, and $\mathbf{lsX}(x, y)$ are semantically equivalent (see Lemma 2).

2.2 Semantics of $\mathbf{SL}_0$

Let N be the set of natural numbers. A *store* (denoted by s) is a function from variables to N. It is extended to a function on terms by $s(\mathbf{nil}) = 0$. We define update $s[x_1 := a_1, \ldots, x_n := a_n]$ of s by the store s' such that $s'(x_i) = a_i$ and $s'(y) = s(y)$ if $y \notin \{x_1, \ldots, x_n\}$. It is sometimes abbreviated by $s[\overrightarrow{x} := \overrightarrow{a}]$. A *heap* (denoted by h) is a finite partial function from $N \setminus \{0\}$ to N. The domain of h is written by $\mathrm{dom}(h)$. We write $h_1 + h_2$ for disjoint union of h_1 and h_2, namely, it is defined when $\mathrm{dom}(h_1)$ and $\mathrm{dom}(h_1)$ are disjoint, and $(h_1 + h_2)(a) = h_i(a)$ if $a \in \mathrm{dom}(h_i)$ for $i = 1, 2$. We sometimes write $[a_1 \mapsto b_1, \ldots, a_m \mapsto b_m]$ for the heap h defined by $\mathrm{dom}(h) = \{a_1, \ldots, a_m\}$ and $h(a_j) = b_j$ $(1 \leq j \leq m)$.

A pair (s, h) is called a *heap model*.

Definition 2 (Interpretation of formulas)**.** The interpretation of a formula A in (s, h) (denoted by $s, h \models A$) is inductively defined as follows.

$$
\begin{array}{lcl}
s, h \models t \mapsto u & \stackrel{\text{def}}{\iff} & h = [s(t) \mapsto s(u)] \\
s, h \models P^{(0)}(\overrightarrow{t}) & \stackrel{\text{def}}{\iff} & \text{never} \\
s, h \models P^{(m+1)}(\overrightarrow{t}) & \stackrel{\text{def}}{\iff} & s[\overrightarrow{y} := \overrightarrow{b}], h \models A[P_1^{(m)}, \ldots, P_K^{(m)}/P_1, \ldots, P_K](\overrightarrow{t}, \overrightarrow{y}) \\
 & & \text{for some } \overrightarrow{b} \text{ and definition clause } A \text{ of } P. \\
s, h \models P(\overrightarrow{t}) & \stackrel{\text{def}}{\iff} & s, h \models P^{(m)}(\overrightarrow{t}) \text{ for some } m \\
s, h \models A_1 * A_2 & \stackrel{\text{def}}{\iff} & \text{there exist } h_1 \text{ and } h_2 \text{ such that } h = h_1 + h_2, \\
 & & s, h_1 \models A_1 \text{ and } s, h_2 \models A_2,
\end{array}
$$

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

Intuitively $P^{(m)}$ corresponds the m-time unfolding of P, that is, $P^{(0)}(\overrightarrow{x})$ means $\bot$ and $P^{(m+1)}(\overrightarrow{x})$ means $\bigvee_{i=1}^{s} \exists \overrightarrow{y_i} A_i[P_1^{(m)}, \ldots, P_K^{(m)}/P_1, \ldots, P_K](\overrightarrow{x}, \overrightarrow{y_j})$, where $A_1, \ldots, A_s$ are the definition clauses of P.

The following lemma explains how the above definition for inductive predicates works. This result will be used in the proof of Theorem 2 and Lemma 5.

Lemma 1. Let h_n be $[a_0 \mapsto a_1, a_1 \mapsto a_2, \ldots, a_{n-1} \mapsto a_n]$ for $n \geq 1$. That is, h_n forms a singly-linked list of length n. Take a store s that satisfies $s(x) = a_0$ and $s(y) = a_n$. Then we have

$s, h_n \models \mathbf{ls}^{(n)}(x, y)$ by induction on n. Hence $s, h_n \models \mathbf{ls}(x, y)$ holds for any $n \geq 1$. We can also show $s, h_n \models \mathbf{sl}(x, y)$ and $s, h_n \models \mathbf{lsX}(x, y)$.

We say A and B are *logically equivalent* if for any heap model (s, h), $s, h \models A$ and $s, h \models B$ are equivalent. Then we have the following claim.

Lemma 2. $\mathbf{ls}(x, y)$, $\mathbf{sl}(x, y)$, *and* $\mathbf{lsX}(x, y)$ *are logically equivalent.*

3 Cyclic proof system $\mathrm{CSL}_0\mathrm{ID}\omega$ for SL_0

This subsection defines a cyclic proof system $\mathrm{CSL}_0\mathrm{ID}\omega$ for SL_0, which handles single-conclusion sequents defined as follows.

Definition 3 (Single-conclusion sequents of SL_0)**.** A *sequent* of SL_0 has the form $A \vdash B$. The formula on the left-hand side and the right-hand side of a sequent are called its *antecedent* and *succedent* (or *conclusion*), respectively. A sequent $A \vdash B$ is called *valid* if, for any heap model (s, h), $s, h \models A$ implies $s, h \models B$.

In the next subsection, the word "conclusion" for an inference rule will be defined. In order to avoid confusion, for sequents, we will not use this word alone, but use it as "single-conclusion" or "multiple-conclusion".

3.1 Cyclic proof system $\mathrm{CSL}_0\mathrm{ID}\omega$

The derivation rules of $\mathrm{CSL}_0\mathrm{ID}\omega$ consists of the basic rules and the unfolding rules. The basic rules are given as follows.

$$\frac{}{A \vdash A}\ (\mathrm{Id}) \qquad \frac{A \vdash C \quad C \vdash B}{A \vdash B}\ (\mathrm{Cut}) \qquad \frac{A_1 \vdash B_1 \quad A_2 \vdash B_2}{A_1 * A_2 \vdash B_1 * B_2}\ (*)$$

The unfolding rules consist of UL and UR. In the following, we assume $P(\overrightarrow{x}) := A_1 \mid \ldots \mid A_s$ is the definition of P.

$$\frac{B \vdash A_j(\overrightarrow{t}, \overrightarrow{u})}{B \vdash P(\overrightarrow{t})}\ (\mathrm{UR}) \qquad \frac{B * A_1(\overrightarrow{t}, \overrightarrow{z_1}) \vdash C \quad \cdots \quad B * A_s(\overrightarrow{t}, \overrightarrow{z_s}) \vdash C}{B * P(\overrightarrow{t}) \vdash C}\ (\mathrm{UL})$$

where $\overrightarrow{z_1}, \ldots, \overrightarrow{z_s}$ are fresh

For each inference rule, sequents above the horizontal line are called its *premises*, and a sequent below the line is called its *conclusion*.

Example 1. The unfolding rules for **ls**, **sl**, **lsO**, **lsE** and **lsX** are as follows.
Rules for **ls**:

$$\frac{B \vdash t \mapsto u}{B \vdash \mathbf{ls}(t, u)}\ (\mathrm{UR}) \quad \frac{B \vdash t \mapsto t_1 * \mathbf{ls}(t_1, u)}{B \vdash \mathbf{ls}(t, u)}\ (\mathrm{UR}) \quad \frac{B * t \mapsto u \vdash C \quad B * t \mapsto z * \mathbf{ls}(z, u) \vdash C}{B * \mathbf{ls}(t, u) \vdash C}\ (\mathrm{UL})$$

Rules for **sl**:

$$\frac{B \vdash t \mapsto u}{B \vdash \mathbf{sl}(t, u)}\ (\mathrm{UR}) \quad \frac{B \vdash \mathbf{sl}(t, u_1) * u_1 \mapsto u}{B \vdash \mathbf{sl}(t, u)}\ (\mathrm{UR}) \quad \frac{B * t \mapsto u \vdash C \quad B * \mathbf{sl}(t, z) * z \mapsto u \vdash C}{B * \mathbf{sl}(t, u) \vdash C}\ (\mathrm{UL})$$

Rules for **lsO**:

$$\frac{B \vdash t \mapsto u}{B \vdash \mathbf{lsO}(t, u)}\ (\mathrm{UR}) \quad \frac{B \vdash t \mapsto t_1 * \mathbf{lsE}(t_1, u)}{B \vdash \mathbf{lsO}(t, u)}\ (\mathrm{UR}) \quad \frac{B * t \mapsto u \vdash C \quad B * t \mapsto z * \mathbf{lsE}(z, u) \vdash C}{B * \mathbf{lsO}(t, u) \vdash C}\ (\mathrm{UL})$$

Rules for **lsE**:

$$\frac{B \vdash t \mapsto t_1 * \mathbf{lsO}(t_1, u)}{B \vdash \mathbf{lsE}(t, u)}\ (\mathrm{UR}) \quad \frac{B * t \mapsto z * \mathbf{lsO}(z, u) \vdash C}{B * \mathbf{lsE}(t, u) \vdash C}\ (\mathrm{UL})$$

Rules for **lsX**:

$$\frac{B \vdash \mathbf{lsO}(t,u)}{B \vdash \mathbf{lsX}(t,u)}\ (\mathrm{UR}) \quad \frac{B \vdash \mathbf{lsE}(t,u)}{B \vdash \mathbf{lsX}(t,u)}\ (\mathrm{UR}) \quad \frac{B * \mathbf{lsO}(t,u) \vdash C \quad B * \mathbf{lsE}(t,u) \vdash C}{B * \mathbf{lsX}(t,u) \vdash C}\ (\mathrm{UL})$$

We define a *cyclic proof* of $\mathrm{CSL}_0\mathrm{ID}\omega$ in the similar way to [4, 5].

Definition 4 (Derivation tree)**.** A *derivation tree* (denoted by $\mathcal{D}$) of a sequent e is a finite tree structure whose nodes are labeled by sequents of SL_0, the label of the root node is e, and a node labeled with e' has children labeled with $e'_1, \ldots, e'_k$ when there is an instance $\frac{e'_1 \quad \cdots \quad e'_k}{e'}$ of an inference rule of $\mathrm{CSL}_0\mathrm{ID}\omega$. A leaf node which is the conclusion of (Id)-rule is called *closed*. An open (not closed) leaf is called a *bud*. A *companion* for a bud e_b is an occurrence of a sequent e_c in $\mathcal{D}$ of which e_b is an substitution instance, namely, $e_b = e_c[\theta]$ for some θ.

In a derivation tree, if e appears as a conclusion of a rule instance and e' is a premise of the rule, e' is called a *premise* of e. In this case, e is called a *parent* of e'. Similary we also use the usual terminology of the tree structure such as child and descendant.

Definition 5 (Pre-proof)**.** A pre-proof of e is given by $(\mathcal{D}, \mathcal{R})$, where $\mathcal{D}$ is a derivation tree of e and $\mathcal{R}$ is a function assigning a companion to every bud of $\mathcal{D}$. A *proof-graph* $\mathcal{G}(\mathcal{P})$ of a pre-proof $\mathcal{P} = (\mathcal{D}, \mathcal{R})$ is a graph structure $\mathcal{D}$ with additional edges from buds to their companions assigned by $\mathcal{R}$. A *path* in $\mathcal{P}$ is a path in $\mathcal{G}(\mathcal{P})$.

Definition 6 (Trace)**.** Let $(e_i)_{i\in\omega}$ be an infinite path in $\mathcal{P}$. A *trace* following $(e_i)_{i\in\omega}$ is a sequence of $(C_i)_{i\in\omega}$ such that each C_i is a subformula occurrence of the form $P(\overrightarrow{t})$ in the antecedent of e_i, and satisfies the following conditions:
(a) if e_i is the conclusion of (UL) in $\mathcal{D}$, 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* of the trace;
(b) if e_i is the conclusion of a rule other than (UL), 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 and $e_{i+1}[\theta] = e_i$ for some substitution θ, then C_{i+1} is the corresponding subformula occurrence in e_{i+1}, which satisfies $C_{i+1}[\theta] = C_i$.

Definition 7 (Cyclic proof)**.** Let $\mathcal{P}$ be a pre-proof of e. $\mathcal{P}$ is called a *cyclic proof* of e if it satisfies the *global trace condition*: for any infinite path $(e_i)_{i\in\omega}$ in $\mathcal{P}$, there exists a trace $(C_i)_{i\in\omega}$ following the path that have infinitely many progressing points.

The global trace condition is a sufficient condition for soundness, that is, we have the following theorem. This theorem is shown by the similar way to [4, 5, 6].

Theorem 1 (Soundness of $\mathrm{CSL}_0\mathrm{ID}\omega$)**.** *If $A \vdash B$ has a cyclic proof $\mathcal{P}$ of $CSL_0ID\omega$, then all sequents in $\mathcal{P}$ are valid. In particular, $A \vdash B$ is valid.*

In the following subsection we will prove that $\mathbf{ls}(x,y) \vdash \mathbf{lsX}(x,y)$ is a counter-example for the cut-elimination property of $\mathrm{CSL}_0\mathrm{ID}\omega$. We first show that this example has a cyclic proof with (Cut).

Proposition 1. *(1) $x \mapsto z * \mathbf{lsX}(z,y) \vdash \mathbf{lsX}(x,y)$ is cut-free provable in $CSL_0ID\omega$.*
(2) $\mathbf{ls}(x,y) \vdash \mathbf{lsX}(x,y)$ is provable in $CSL_0ID\omega$ with (Cut).

Proof. (1) is shown as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x \mapsto z * \mathbf{lsO}(z,y) \vdash x \mapsto z * \mathbf{lsO}(z,y)}}{x \mapsto z * \mathbf{lsO}(z,y) \vdash \mathbf{lsE}(x,y)}\ (\mathrm{UR})}{x \mapsto z * \mathbf{lsO}(z,y) \vdash \mathbf{lsX}(x,y)}\ (\mathrm{UR}) \qquad \dfrac{\dfrac{\dfrac{}{x \mapsto z * \mathbf{lsE}(z,y) \vdash x \mapsto z * \mathbf{lsE}(z,y)}}{x \mapsto z * \mathbf{lsE}(z,y) \vdash \mathbf{lsO}(x,y)}\ (\mathrm{UR})}{x \mapsto z * \mathbf{lsE}(z,y) \vdash \mathbf{lsX}(x,y)}\ (\mathrm{UR})}{x \mapsto z * \mathbf{lsX}(z,y) \vdash \mathbf{lsX}(x,y)}\ (\mathrm{UL})$$

By using this result, (2) is shown as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x \mapsto y \vdash x \mapsto y}}{x \mapsto y \vdash \mathbf{lsO}(x,y)}\,(\mathrm{UR})}{x \mapsto y \vdash \mathbf{lsX}(x,y)}\,(\mathrm{UR}) \quad \dfrac{\dfrac{\dfrac{}{x \mapsto z \vdash x \mapsto z} \quad (\dagger)\ \underline{\mathbf{ls}(z,y)} \vdash \mathbf{lsX}(z,y)}{x \mapsto z * \underline{\mathbf{ls}(z,y)} \vdash x \mapsto z * \mathbf{lsX}(z,y)}\,(*) \quad (1)\ x \mapsto z * \mathbf{lsX}(z,y) \vdash \mathbf{lsX}(x,y)}{x \mapsto z * \underline{\mathbf{ls}(z,y)} \vdash \mathbf{lsX}(x,y)}\,(\mathrm{Cut})}{(\dagger)\ \underline{\mathbf{ls}(x,y)} \vdash \mathbf{lsX}(x,y)}\,(\mathrm{UL})$$

The topmost entailment $\mathbf{ls}(z,y) \vdash \mathbf{lsX}(z,y)$ marked by $(\dagger)$ is a bud whose companion is the entailment $\mathbf{ls}(z,y) \vdash \mathbf{lsX}(z,y)$ with the same mark at the root position. The only infinite path which is created from the bud-companion has an infinitely progressing trace (the sequence of the underlined predicates). Hence the above pre-proof is a cyclic proof. □

3.2 Counter-example for cut-elimination of $\mathrm{CSL}_0\mathrm{ID}\omega$

We define $\sharp_{\mapsto} A$ by the number of $\mapsto$ in A. We also define $\sharp_{\mapsto}(A \vdash B)$ by $\sharp_{\mapsto} A$.

Next theorem is our first main result, namely, the cut-elimination property fails in $\mathrm{CSL}_0\mathrm{ID}\omega$.

Theorem 2. *The sequent* $\mathbf{ls}(x,y) \vdash \mathbf{lsX}(x,y)$ *is not cut-free provable in* $CSL_0ID\omega$.

Proof. Suppose that $\mathbf{ls}(x,y) \vdash \mathbf{lsX}(x,y)$ has a cut-free cyclic proof $(\mathcal{D}, \mathcal{R})$. We will show contradiction.

We construct a finite path $\mathbf{e} = (e_0, e_1, \ldots, e_m)$ of $\mathcal{D}$ such that each e_j has the form (up to permutation of $*$)

$$z_0 \mapsto z_1 * \ldots * z_{k_j - 1} \mapsto z_{k_j} * \mathbf{ls}(z_{k_j}, w) \vdash \mathbf{lsX}(z_0, w)$$

for some k_j and pairwise distinct variables $z_0, \ldots, z_{k_j}, w$. Let e_0 be $\mathbf{ls}(x,y) \vdash \mathbf{lsX}(x,y)$ at the root position. Assume that $e_0, \ldots, e_j$ are already defined. If e_j is the conclusion of a (UL), then define e_{j+1} by the unique premise of the rule instance that contains the **ls**-predicate in the antecedent. We note that e_{j+1} has the required form and $\sharp_{\mapsto} e_j < \sharp_{\mapsto} e_{j+1}$. Otherwise finish constructing the path $\mathbf{e}$.

Claim1. For any sequent e in $\mathcal{D}$, if the antecedent of e contains the **ls**-predicate, then $e \in \mathbf{e}$ or e is a descendant of e_m.

This claim is shown by induction on the height $ht(e)$, namely the length of the path from the root node to e, of e in $\mathcal{D}$. If $ht(e) = 0$, then $e = e_0 \in \mathbf{e}$. We show the case $ht(e) > 0$. Assume $e \notin \mathbf{e}$. We show that e is a descendant of e_m. In this case, e is an premise of an instance of a rule (r), which is not (Cut). Let e' be the parent of e, that is the conclusion of the rule instance. Then the antecedent of e' contains the **ls**-predicate. Hence $e' \in \mathbf{e}$ or e' is a descendant of e_m by the induction hypothesis. If the latter case holds, we have the expected result. Otherwise, e' must be e_m since e contains **ls**, $e' \in \mathbf{e}$ and $e \notin \mathbf{e}$. Thus e is a descendant of e_m. Hence we have Claim1.

Claim2. e_m is not a bud.

We show this claim. Assume that e_m is a bud. Then the antecedent of the companion e contains the **ls**-predicate. By Claim1, we have $e \in \mathbf{e}$ since e_m is a bud. Hence there is an infinite path $e, \ldots, e_m, e, \ldots$ of the cyclic proof $(\mathcal{D}, \mathcal{R})$. By the global trace condition, there is a trace following the path with infinitely many progressing points This means $\sharp_{\mapsto} e < \sharp_{\mapsto} e_m < \sharp_{\mapsto} e$. Hence we have contradiction. Thus we obtain Claim2.

By Claim2, e_m is a conclusion of an instance of a rule (r). Then (r) must be (UR) by case analysis of the inference rules. Let $\tilde{e}$ be the unique child of e_m in $\mathcal{D}$. The form of $\tilde{e}$ is either

(a) $z_0 \mapsto z_1 * \ldots * z_{k_m - 1} \mapsto z_{k_m} * \mathbf{ls}(z_{k_m}, w) \vdash \mathbf{lsO}(z_0, w)$, or
(b) $z_0 \mapsto z_1 * \ldots * z_{k_m - 1} \mapsto z_{k_m} * \mathbf{ls}(z_{k_m}, w) \vdash \mathbf{lsE}(z_0, w)$

Consider the case (a). Take a store s_1 such that $s_1(z_i) = i+1$ and $s_1(w) = 2 * k_m + 1$. Define h_1 by $\mathrm{dom}(h_1) = \{1, 2, \ldots, 2 * k_m\}$ and $h_1(i) = i+1$. Then $s_1, h_1 \models z_0 \mapsto z_1 * \ldots * z_{k_m - 1} \mapsto z_{k_m} * \mathbf{ls}(z_{k_m}, w)$ and $s_1, h_1 \not\models \mathbf{lsO}(z_0, w)$. Hence (a) is invalid. We can also show that (b) is invalid by taking s_2 such that $s_2(z_i) = i+1$ and $s_2(w) = 2 * k_m + 2$, and defining h_2 by $\mathrm{dom}(h_2) = \{1, 2, \ldots, 2 * k_m + 1\}$

and $h_2(i) = i + 1$. Therefore $\tilde{e}$ is invalid. This contradicts the soundness theorem. Hence we conclude that $\mathbf{ls}(x, y) \vdash \mathbf{lsX}(x, y)$ is not cut-free provable. □

By combining proposition 1 and theorem 2, we can obtain our first main result.

Corollary 1. *The cyclic proof system $CSL_0ID\omega$ does not enjoy the cut-elimination property.*

3.3 Failure of Cut-elimination in Brotherston's CADE2011-system

Our $\mathrm{CSL_0ID}\omega$ is designed as a simpler system as much as possible in order to simplify our discussion. In this subsection we demonstrate that the proof given in the previous subsection also works for some extended systems including the cyclic proof system of separation logic given by Brotherston's CADE2011 paper [6].

We first extend SL_0 by adding the empty predicate **emp**, that is, the formulas of the extended system $\mathrm{SL'}_0$ are given as follows:

$$A ::= t \mapsto t \mid A * A \mid P(\overrightarrow{t}) \mid \mathbf{emp}$$

The interpretation of the empty predicate is given as follows:

$$s, h \models \mathbf{emp} \quad \overset{\text{def}}{\iff} \quad \mathrm{dom}(h) = \emptyset$$

Then we extend $\mathrm{CSL_0ID}\omega$ by adding the following derivation rules:

$$\frac{A \vdash B}{A \vdash B * \mathbf{emp}}\ (\mathrm{EmpR}) \qquad \frac{A \vdash B}{\mathbf{emp} * A \vdash B}\ (\mathrm{EmpL})$$

$$\frac{A \vdash B * \mathbf{emp}}{A \vdash B}\ (\mathrm{EmpR'}) \qquad \frac{\mathbf{emp} * A \vdash B}{A \vdash B}\ (\mathrm{EmpL'})$$

$$\frac{}{x \mapsto u * x \mapsto u' * A \vdash B}\ (\mathrm{Unsat}\mapsto)$$

We call this extended system $\mathrm{CSL'}_0\mathrm{ID}\omega$.

The definitions of pre-proofs, traces, and cyclic proofs of $\mathrm{CSL'}_0\mathrm{ID}\omega$ are given in the similar way to those of $\mathrm{CSL_0ID}\omega$. The soundness theorem for this extended system also holds.

Then we can show the similar claim to Theorem 2.

Proposition 2. *The sequent $\mathbf{ls}(x, y) \vdash \mathbf{lsX}(x, y)$ is not cut-free provable in $CSL'_0ID\omega$.*

Proof (sketch). Suppose that $\mathbf{ls}(x, y) \vdash \mathbf{lsX}(x, y)$ has a cut-free cyclic proof $(\mathcal{D}, \mathcal{R})$. We construct a finite path $\mathbf{e} = (e_0, e_1, \ldots, e_m)$ of $\mathcal{D}$ such that each e_j has the form

$$z_0 \mapsto z_1 * \ldots * z_{k_j-1} \mapsto z_{k_j} * \mathbf{ls}(z_{k_j}, w) * \overrightarrow{\mathbf{emp}} \vdash \mathbf{lsX}(z_0, w) * \overrightarrow{\mathbf{emp}}$$

for some k_j and pairwise distinct variables $z_0, \ldots, z_{k_j}, w$, and e_0 is $\mathbf{ls}(x, y) \vdash \mathbf{lsX}(x, y)$ at the root position. If e_j is the conclusion of a (UL), define e_{j+1} by the unique premise of e_j which contains **ls**. If e_j is the conclusion of EmpL, EmpR, EmpL', or EmpR', define e_{j+1} by the unique premise. We claim that $\sharp_{\mapsto} e_j \leq \sharp_{\mapsto} e_{j+1}$ for any $j < m$, and, in particular, $\sharp_{\mapsto} e_j < \sharp_{\mapsto} e_{j+1}$ if e_j is the conclusion of a rule instance of (UL). Then we have the same claims as Claim1 and Claim2 of Theorem 2. We also have the following claim:

Claim3 e_m is not the conclusion of a rule instance of $(\mathrm{Unsat}\mapsto)$.

This claim is obtained by investigating that the antecedents of $\mathbf{e}$ are satisfiable. Hence e_m cannot be the conclusion of $(\mathrm{Unsat}\mapsto)$.

By using Claim1, Claim2, and Claim3, we can show the expected result in a similar way to the proof of Theorem 2. □

We introduce an extended system $\mathrm{CSL}_B\mathrm{ID}\omega$, which is a variant of the system given in [6]. The difference between these two systems is not essential: terms of the system in [6] are only variables. The formulas of $\mathrm{CSL}_B\mathrm{ID}\omega$ are given as follows:

$$A ::= t \mapsto t \mid A * A \mid P(\overrightarrow{t}) \mid \mathbf{emp} \mid t = t \mid t \neq t \mid t \overset{2}{\mapsto} (t, t) \mid \top \mid \bot \mid A \vee A$$

The derivation rules of $\mathrm{CSL}_B\mathrm{ID}\omega$ are those of $\mathrm{CSL'}_0\mathrm{ID}\omega$ with the following rules:

$$\frac{}{\bot * A \vdash B}\ (\bot) \qquad \frac{}{A \vdash \top}\ (\top) \qquad \frac{}{A \vdash t = t}\ (=\mathrm{R})$$

$$\frac{}{t = u * t \neq u * A \vdash B}\ (\mathrm{Unsat}=) \qquad \frac{}{t \overset{2}{\mapsto} (u_1, u_2) * t \overset{2}{\mapsto} (u_1', u_2') * A \vdash B}\ (\mathrm{Unsat}\overset{2}{\mapsto})$$

$$\frac{A_1 * B \vdash C \quad A_2 * B \vdash C}{(A_1 \vee A_2) * B \vdash C}\ (\vee\mathrm{L}) \qquad \frac{A \vdash B_i * C}{A \vdash (B_1 \vee B_2) * C}\ (\vee\mathrm{R})\ (i = 1, 2)$$

The following lemma shows that the above additional inference rules cannot be applied in proof-search procedures of $\mathbf{ls}(x, y) \vdash \mathbf{lsX}(x, y)$.

Lemma 3. *Let A and B be formulas of $\mathrm{CSL}_B\mathrm{ID}\omega$ whose connectives and predicates are* $\mathbf{emp}$, $*$, $\mapsto$, $\mathbf{ls}$, $\mathbf{lsX}$, $\mathbf{lsE}$, *and* $\mathbf{lsO}$. *Assume $A \vdash B$ has a cut-free derivation tree of $\mathrm{CSL}_B\mathrm{ID}\omega$. Then all inference rules used in the derivation tree are those of $\mathrm{CSL'}_0\mathrm{ID}\omega$.*

This lemma can be shown by induction on the derivation tree.

As the result of this subsection, we can show the failure of cut-elimination in $\mathrm{CSL}_B\mathrm{ID}\omega$.

Theorem 3 (Failure of cut-elimination in $\mathrm{CSL}_B\mathrm{ID}\omega$)**.** *$\mathrm{CSL}_B\mathrm{ID}\omega$ does not enjoy the cut-elimination property.*

Proof. Note that $\mathbf{ls}(x, y) \vdash \mathbf{lsX}(x, y)$ can be shown in $\mathrm{CSL}_B\mathrm{ID}\omega$ with (Cut), since $\mathrm{CSL}_B\mathrm{ID}\omega$ is an extension of $\mathrm{CSL}_0\mathrm{ID}\omega$. Then we show $\mathbf{ls}(x, y) \vdash \mathbf{lsX}(x, y)$ is not cut-free provable in $\mathrm{CSL}_B\mathrm{ID}\omega$. Assume that it has a cut-free cyclic proof of $\mathrm{CSL}_B\mathrm{ID}\omega$. Then the cyclic proof is also a proof of $\mathrm{CSL'}_0\mathrm{ID}\omega$ since all inference rules are those of $\mathrm{CSL'}_0\mathrm{ID}\omega$ by the previous lemma. Hence we have contradiction by proposition 2. □

Remark 1. In [6], an automated prover for sequents (that does not contain $\mathbf{nil}$) in $\mathrm{CSL}_B\mathrm{ID}\omega$, which is based on a proof-search procedure in $\mathrm{CSL}_B\mathrm{ID}\omega$, is proposed. Although the system contains (Cut), the tool uses the rule for managing basic properties about $*$, such as associativity, commutativity, and unit of $*$. Hence the prover cannot find non-trivial cut-formulas (e.g. the cut-formula $x \mapsto z * \mathbf{lsX}(z, y)$ used in the proof of Proposition 1) during its proof-search procedure.

Recall that each sequent considered in this section has a single-conclusion. It is the reason why $\mathbf{ls}(x, y) \vdash \mathbf{lsX}(x, y)$ works as a counter-example for the cut-elimination, that is, we are forced to choose either $\mathbf{lsO}(x, y)$ or $\mathbf{lsE}(x, y)$ in unfolding $\mathbf{lsX}(x, y)$ on the right-hand side of a sequent. This situation can be avoided if we consider sequents with multiple-conclusions.

4 Cyclic proof system $\mathrm{CSL}_0^M\mathrm{ID}\omega$ for multiple-conclusion SL_0

This section presents the second cyclic proof system $\mathrm{CSL}_0^M\mathrm{ID}\omega$ for sequents with multiple-conclusions defined below.

Let Δ be a multiset of formulas. We sometimes write $B_1, B_2, \ldots, B_n$ instead of $\{B_1, B_2, \ldots, B_n\}$. We also write Δ, B for $\Delta \cup \{B\}$. We define $\Delta * \Delta'$ by $\{B * B' \mid B \in \Delta \text{ and } B' \in \Delta'\}$. We also define $s, h \models \bigvee \Delta$ by $\exists B \in \Delta (s, h \models B)$.

Definition 8 (Multiple-conclusion sequents of SL_0)**.** The *multiple-conclusion sequents* of SL_0 have the form $A \vdash \Delta$. A sequent $A \vdash \Delta$ is valid if, for any (s, h), $s, h \models A$ implies $s, h \models \bigvee \Delta$.

4.1 Cyclic proof system $\mathrm{CSL}_0^M\mathrm{ID}\omega$ for multiple-conclusion sequents

The derivation rules of $\mathrm{CSL}_0^M\mathrm{ID}\omega$ consists of the basic rules and the unfolding rules. The basic rules of $\mathrm{CSL}_0^M\mathrm{ID}\omega$ are given as follows.

$$\frac{}{A \vdash A}\ (\mathrm{Id}) \qquad \frac{A \vdash \Delta_1, C \quad C \vdash \Delta_2}{A \vdash \Delta_1, \Delta_2}\ (\mathrm{Cut}) \qquad \frac{A_1 \vdash \Delta_1 \quad A_2 \vdash \Delta_2}{A_1 * A_2 \vdash \Delta_1 * \Delta_2}\ (*)$$

$$\frac{A \vdash \Delta}{A \vdash \Delta, B}\ (\mathrm{Wk}) \qquad \frac{A \vdash \Delta, B, B}{A \vdash \Delta, B}\ (\mathrm{Ctr})$$

The unfolding rules (UL) and (UR) of $\mathrm{CSL}_0^M\mathrm{ID}\omega$ are straightforward extension of those of $\mathrm{CSL}_0\mathrm{ID}\omega$. We assume that $P(\overrightarrow{x}) := A_1 \mid \ldots \mid A_n$ is the definition of P.

$$\frac{B \vdash \Delta, A_j(\overrightarrow{t}, \overrightarrow{u})}{B \vdash \Delta, P(\overrightarrow{t})}\ (\mathrm{UR}) \qquad \frac{B * A_1(\overrightarrow{t}, \overrightarrow{z_1}) \vdash \Delta \quad \ldots \quad B * A_n(\overrightarrow{t}, \overrightarrow{z_n}) \vdash \Delta}{B * P(\overrightarrow{t}) \vdash \Delta}\ (\mathrm{UL})$$

where $\overrightarrow{z_1}, \ldots, \overrightarrow{z_n}$ are fresh

The pre-proofs, traces, and cyclic proofs of $\mathrm{CSL}_0^M\mathrm{ID}\omega$ are defined similarly to those of $\mathrm{CSL}_0\mathrm{ID}\omega$.

Remark 2. The main difference between $\mathrm{CSL}_0\mathrm{ID}\omega$ and $\mathrm{CSL}_0^M\mathrm{ID}\omega$ is the structural rules, namely the contraction rule (Ctr) and the weakening rule (Wk). These rules change whole proof structure and strengthen provability of the proof system. As we will see in the next proposition, the previous counter-example does not work in the current system. Hence we need a new counter-example and a new proof technique that can capture the changed proof structure.

The soundness theorem of $\mathrm{CSL}_0^M\mathrm{ID}\omega$ is also shown in the similar way to that of $\mathrm{CSL}_0\mathrm{ID}\omega$:

Theorem 4 (Soundness of $\mathrm{CSL}_0^M\mathrm{ID}\omega$)**.** *If $A \vdash \Delta$ has a cyclic proof $\mathcal{P}$ of $CSL_0^M ID\omega$, then all sequents in $\mathcal{P}$ are valid. In particular, $A \vdash \Delta$ is valid.*

The previous example $\mathbf{ls}(x,y) \vdash \mathbf{lsX}(x,y)$ is cut-free provable in the current system.

Proposition 3. *$\mathbf{ls}(x,y) \vdash \mathbf{lsX}(x,y)$ is cut-free provable in $CSL_0^M ID\omega$.*

Proof. $\mathbf{ls}(x,y) \vdash \mathbf{lsX}(x,y)$ is proved without (Cut) as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x \mapsto y \vdash x \mapsto y}\,(\mathrm{UR})}{x \mapsto y \vdash \mathbf{lsO}(x,y)}\,(\mathrm{Wk}) \quad \dfrac{\dfrac{\dfrac{}{x \mapsto z \vdash x \mapsto z}\ (\dagger)\ \underline{\mathbf{ls}(z,y)} \vdash \mathbf{lsE}(z,y), \mathbf{lsO}(z,y)}{x \mapsto z * \underline{\mathbf{ls}(z,y)} \vdash x \mapsto z * \mathbf{lsE}(z,y), x \mapsto z * \mathbf{lsO}(z,y)}\,(*)}{x \mapsto z * \underline{\mathbf{ls}(z,y)} \vdash \mathbf{lsO}(x,y), \mathbf{lsE}(x,y)}\,(\mathrm{UR} \times 2)}{(\dagger)\quad \underline{\mathbf{ls}(x,y)} \vdash \mathbf{lsO}(x,y), \mathbf{lsE}(x,y)}\,(\mathrm{UL})}{\mathbf{ls}(x,y) \vdash \mathbf{lsX}(x,y), \mathbf{lsX}(x,y)}\,(\mathrm{UR} \times 2)}{\mathbf{ls}(x,y) \vdash \mathbf{lsX}(x,y)}\,(\mathrm{Ctr})$$

Note that the two entailments marked by (†) are in a bud-companion relation. The only infinite path in this pre-proof contains an infinitely progressing trace (the sequence of the underlined predicates). Hence this pre-proof is a cyclic proof since it satisfies the global trace condition. □

This proposition shows that $\mathbf{ls}(x,y) \vdash \mathbf{lsX}(x,y)$ does not work as a counter-example for the cut-elimination property of $\mathrm{CSL}_0^M\mathrm{ID}\omega$. However we still have another counter-example $\mathbf{ls}(x,y) \vdash \mathbf{sl}(x,y)$. We first show that this sequent is provable in $\mathrm{CSL}_0^M\mathrm{ID}\omega$.

Proposition 4. *(1) $x \mapsto z * \mathbf{sl}(z,y) \vdash \mathbf{sl}(x,y)$ is (cut-free) provable in $CSL_0^M ID\omega$.*
(2) $\mathbf{ls}(x,y) \vdash \mathbf{sl}(x,y)$ is provable in $CSL_0^M ID\omega$ with (Cut).

Proof. (1) is shown as follows.

$$\dfrac{\dfrac{\dfrac{}{x \mapsto z * z \mapsto y \vdash x \mapsto z * z \mapsto y}}{x \mapsto z * z \mapsto y \vdash \mathbf{sl}(x,z) * z \mapsto y}\,(\mathrm{UR})}{x \mapsto z * z \mapsto y \vdash \mathbf{sl}(x,y)}\,(\mathrm{UR}) \quad \dfrac{(\dagger)\ x \mapsto z * \underline{\mathbf{sl}(z,w)} \vdash \mathbf{sl}(x,w) \quad \dfrac{}{w \mapsto y \vdash w \mapsto y}}{x \mapsto z * \underline{\mathbf{sl}(z,w)} * w \mapsto y \vdash \mathbf{sl}(x,w) * w \mapsto y}\,(*)}{(\dagger)\ x \mapsto z * \underline{\mathbf{sl}(z,y)} \vdash \mathbf{sl}(x,y)}\,(\mathrm{UL})$$

Note that this pre-proof is a cyclic proof since it satisfies the global trace condition.

By using (1), the second claim (2) is shown as follows.

$$\dfrac{\dfrac{}{x \mapsto y \vdash x \mapsto y}\,(\mathrm{UR})}{x \mapsto y \vdash \mathbf{sl}(x,y)} \quad \dfrac{\dfrac{\dfrac{}{x \mapsto z \vdash x \mapsto z} \quad (\ddagger)\ \underline{\mathbf{ls}(z,y)} \vdash \mathbf{sl}(z,y)}{x \mapsto z * \underline{\mathbf{ls}(z,y)} \vdash x \mapsto z * \mathbf{sl}(z,y)}\,(*) \quad \dfrac{(1)}{x \mapsto z * \mathbf{sl}(z,y) \vdash \mathbf{sl}(x,y)}\,(\mathrm{Cut})}{x \mapsto z * \underline{\mathbf{ls}(z,y)} \vdash \mathbf{sl}(x,y)}\,(\mathrm{UL})$$
$$(\ddagger)\ \underline{\mathbf{ls}(x,y)} \vdash \mathbf{sl}(x,y)$$

We can easily check the above pre-proof satisfies the global trace condition. Hence $\mathbf{ls}(x,y) \vdash \mathbf{sl}(x,y)$ has a cyclic proof with (Cut). □

4.2 Counter-example for cut-elimination of $\mathrm{CSL}_0^M\mathrm{ID}\omega$

This subsection shows that the cut-elimination property fails in $\mathrm{CSL}_0^M\mathrm{ID}\omega$. The main result of this section is the following theorem.

Theorem 5. *The sequent* $\mathbf{ls}(x,y) \vdash \mathbf{sl}(x,y)$ *is not cut-free provable in* $CSL_0^M\,ID\omega$.

Remark 3. In the proof of Theorem 2, a contradiction appears at the point when (UR) is applied to the unique succedent. In the current case, however, this idea does not work, since, at the point that (UR) is used, a formula before (UR) may remain in the succedent part because of contraction.

Our basic idea for proving Theorem 5 is as follows: focusing on the path of a cyclic proof that contains both **ls** in the antecedent part and **sl** in the succedent part; and analyzing the form of sequents on the path. To do this, we prepare some notions and show their properties.

In the following we write $\ast_{i=0}^{n} A_i$ for $A_0 * \cdots * A_n$.

Definition 9 (Ls-form)**.** An SL_0 formula is called a *connected Ls-form from x to y*, if it has the form $\ast_{i=0}^{n-1} z_i \mapsto z_{i+1} * \mathbf{ls}(z_n, y)$, z_0 is x, and $\overrightarrow{z}, y$ are pairwise distinct variables. A formula is called an *Ls-form from x to y*, if it is obtained by removing some points-to predicates from a connected Ls-form.

We sometimes omit "from x to y" if it is apparent from the context.

Definition 10 (Sl-form)**.** A formula is called a *connected Sl-form from x to y*, if it has the form $\mathbf{sl}(x, z_n) * \ast_{i=0}^{n-1} z_{i+1} \mapsto z_i$ with $z_0 = y$. A formula is called an *Sl-form from x to y*, if it is obtained by removing some points-to predicates from a connected Sl-form. A formula is called a *semi Sl-form*, if it is an Sl-form or contains only $\mapsto$.

A finite multiset Δ of formulas is called a *semi Sl-form*, if all elements of Δ are semi Sl-form.

The following lemma is easily shown from the definition.

Lemma 4. *If A is an Ls-form from x to y, then A is satisfiable.*

Lemma 5. *Assume that A is an Ls-form from x to y, Δ is a semi Sl-form from x to y, and $A \vdash \Delta$ is valid. Then we have the following claims.*

(1) A is a connected Ls-form.

(2) $\mathbf{sl}(x,y)$ is in Δ.

Proof. Suppose the assumption. Since A is an Ls-form from x to y, there is a connected Ls-form $\ast_{i=0}^{n-1} z_i \mapsto z_{i+1} * \mathbf{ls}(z_n, y)$, where $z_0 = x$ and $\overrightarrow{z}, y$ are pairwise distinct. Then A can be written as $\ast_{i \in I} z_i \mapsto z_{i+1} * \mathbf{ls}(z_n, y)$, where $I \subseteq \{0, \ldots, n-1\}$. Let $k = |\mathrm{FV}(A, \Delta)|$. Define s and h by:

$s(z_i) = i+1$ for $i \in \{0,1,\ldots,n\}$, $s(y) = n+k+2$, and $s(w) = 0$ if $w \notin \overrightarrow{z}, y$.
$\mathrm{dom}(h) = \{i+1 \mid i \in I\} \cup \{n+1, n+2, \ldots, n+k+1\}$ and
$h(m) = m+1$ for $m \in \mathrm{dom}(h)$.

Then $|\mathrm{dom}(h)| = |I| + k + 1$ and $n+k+2 \notin \mathrm{dom}(h)$. We also have $s, h \models A$. Hence $s, h \models B$ for some $B \in \Delta$ since $A \vdash \Delta$ is valid by the assumption.

Now B is an Sl-form. We will show this. Suppose that B is not an Sl-form. By the assumption, B has the form $*_{j\in J} x_j \mapsto x'_j$ and $s, h \models B$. Then we obtain contradiction, since $|\mathrm{dom}(h)| = |J| \leq |\mathrm{FV}(\Delta)| \leq k < |\mathrm{dom}(h)|$. Therefore B is an Sl-form.

(1) Suppose that A is not a connected Ls-form. Then the set $\{0,1,\ldots,n-1\} \setminus I$ is not empty. Take the smallest element i_0 of this set. We will show contradiction by the case analysis of i_0.

We show the case $i_0 = 0$. This case means $x = z_0 \notin \mathrm{FV}(A)$ since A does not contain $z_0 \mapsto z_1$. Then $1 = s(z_0) \notin \mathrm{dom}(h)$. Recall the above B. It satisfies $s, h \models B$ and contains $\mathbf{sl}(x, z')$ since B is an Sl-form. Then we have $1 = s(x) \in \mathrm{dom}(h)$. Hence we have contradiction.

We show the case $i_0 > 0$. This case we have $s, h \models *_{j=0}^{i_0-1} z_j \mapsto z_{j+1} * *_{j\in I'} z_j \mapsto z_{j+1} * \mathbf{ls}(z_n, y)$, where $I' = I \setminus \{0,\ldots,i_0-1\}$ and $i_0 \notin I'$. Hence we have $i_0 + k + 1 \leq |\mathrm{dom}(h)|$, since $\{1,\ldots,i_0\} \cup \{n+1,\ldots,n+k+1\} \subseteq \mathrm{dom}(h)$. Now the Sl-form B has the form $\mathbf{sl}(x, z'_m) * *_{j\in J} z'_{j+1} \mapsto z'_j$. Thus there exist h_1 and h_2 such that $h = h_1 + h_2$, $s, h_1 \models \mathbf{sl}(x, z'_m)$ and $s, h_2 \models *_{j\in J} z'_{j+1} \mapsto z'_j$. By the definition of h and i_0, we have $\mathrm{dom}(h_1) \subseteq \{1,\ldots,i_0\}$. Hence we have $|\mathrm{dom}(h_1)| \leq i_0$. We also have $|\mathrm{dom}(h_2)| \leq |\mathrm{FV}(\Delta)| \leq k$. Therefore we obtain $i_0 + k + 1 \leq |\mathrm{dom}(h)| = |\mathrm{dom}(h_1)| + |\mathrm{dom}(h_2)| \leq i_0 + k$. Contradiction.

Finally we conclude that A is a connected Ls-form. Hence (1) is shown.

(2) Since A is a connected Ls-form by (1), we have $I = \{0,\ldots,n-1\}$. Hence we also have $\mathrm{dom}(h) = \{1,\ldots,n+k+1\}$. Recall that $s, h \models B$ and B is an Sl-form. We show B is $\mathbf{sl}(x, y)$. Suppose not. Then B has the form $\mathbf{sl}(x, z'_m) * *_{j\in J} z'_{j+1} \mapsto z'_j$ with $J \neq \emptyset$. Thus there exist h_1 and h_2 such that $h = h_1 + h_2$, $s, h_1 \models \mathbf{sl}(x, z'_m)$ and $s, h_2 \models *_{j\in J} z'_{j+1} \mapsto z'_j$. Note that $\mathrm{dom}(h_1) = \{1,\ldots,s(z'_m)-1\}$ and $\mathrm{dom}(h_2) = \{s(z'_m),\ldots,n+k+1\}$. Moreover $s(z'_m) < n+k+2 = s(y)$ since J is not empty. Hence $z'_m \in \overrightarrow{z}$ by the definition of s and $z'_m \neq y$, We have $|\mathrm{dom}(h_1)| = s(z'_m) - 1 \leq n$ and $|\mathrm{dom}(h_2)| \leq |\mathrm{FV}(\Delta)| \leq k$. From this, we obtain $n+k+1 = |\mathrm{dom}(h)| = |\mathrm{dom}(h_1)| + |\mathrm{dom}(h_2)| \leq n+k$. Contradiction. Therefore $\mathbf{sl}(x, y) = B \in \Delta$. □

Definition 11 (L-form)**.** A sequent $A \vdash \Delta$ is called an *L-form from x to y* if A is an Ls-form from x to y and Δ is a semi Sl-form from x to y.

We define $\sharp_{\mapsto}(e)$ for a sequent e of $\mathrm{CSL}_0^M\mathrm{ID}\omega$ in the similar way to one of $\mathrm{CSL}_0\mathrm{ID}\omega$.

Lemma 6. *Let $\mathcal{P}$ be a cut-free cyclic proof in $CSL_0^M ID\omega$. Assume that an L-form e from x to y appears in $\mathcal{P}$ as the consequence of an inference rule (r). Then there is a unique premise e' of the rule such that e' is an L-form. Moreover, $\sharp_{\mapsto}(e) < \sharp_{\mapsto}(e')$ if (r) is (UL), and $\sharp_{\mapsto}(e) = \sharp_{\mapsto}(e')$ otherwise.*

Proof. This lemma is shown by the case analysis of the rule. Let A and Δ be the antecedent and the succedent of e, respectively.

The rule (Id) is not the case since, by lemma 5, A contains the **ls**-predicate, and Δ does not contain the **ls**-predicate.

The rule (Cut) is not the case since $\mathcal{P}$ is cut-free.

The cases of (Wk), (Ctr) and (Subst) are immediately shown.

The case of (UR). By the premise e has the form $A \vdash \Delta_1, B * \mathbf{sl}(x, z)$, and $B * \mathbf{sl}(x, z)$ is an Sl-form. The unique premise e' of e has the form $A \vdash \Delta_1, B * \mathbf{sl}(x, w) * w \mapsto z$. Then $B * \mathbf{sl}(x, w) * w \mapsto z$ is an Sl-form since $B * Sl(x, z)$ is an Sl-form. Therefore e' is an L-form. We also have $\sharp_{\mapsto}(e) = \sharp_{\mapsto}(e')$.

The case of (UL). By the premise e has the form $A_1 * \mathbf{ls}(z, y) \vdash \Delta$, and $A_1 * \mathbf{ls}(z, y)$ is an Ls-form. Thus e has a unique premise whose antecedent contains the **ls**-predicate. Let e' be the

premise. Then e' has the form $A_1 * z \mapsto z' * \mathbf{ls}(z', y) \vdash \Delta$, where z' is a fresh variable. We note that $A_1 * z \mapsto z' * \mathbf{ls}(z', y)$ is an Ls-form since $A_1 * \mathbf{ls}(z, y)$ is an Ls-form and z' is fresh. Therefore e' is an L-form. We also have $\sharp_{\mapsto}(e) < \sharp_{\mapsto}(e')$.

The case of $(*)$. Recall that e contains only one **ls**-predicate. Hence there is a unique premise of e that contains the **ls**-predicate. Let e' be the premise. Let A' and Δ' be the antecedent and succedent of e'. Then A' is an Ls-form. Since Δ is a semi Sl-form, Δ' is also a semi Sl-form. Therefore e' is an L-form. Note that all $\mapsto$ of A must be contained in A'. Otherwise A' is not a connected Ls-form. This contradicts Lemma 5. Hence we have $\sharp_{\mapsto}(e) = \sharp_{\mapsto}(e')$. □

Lemma 7. *Suppose that there is a cut-free cyclic proof $\mathcal{P}$ of $\mathbf{ls}(x, y) \vdash \mathbf{sl}(x, y)$ in $CSL_0^M ID\omega$. Then its graph $\mathcal{G}(\mathcal{P})$ contains an infinite path $(e_i)_{i\in\omega}$ such that (a) e_0 is $\mathbf{ls}(x, y) \vdash \mathbf{sl}(x, y)$, and (b) each e_i is an L-form.*

Proof. Let $\mathcal{D}$ be the underlying derivation tree of $\mathcal{P}$. We inductively define a finite path $e_0, e_1, \ldots, e_n$ of $\mathcal{D}$. Let e_0 be $\mathbf{ls}(x, y) \vdash \mathbf{sl}(x, y)$ at the root position of $\mathcal{D}$. Suppose that $e_0, \ldots, e_k$ are already defined. If e_k is a bud, then finish making the path with $n = k$. Otherwise there exists a unique premise e' of e_k by Lemma 6 such that e' is an L-form. Then we define e_{k+1} by e'. Note that this construction successfully terminates, since any L-form cannot be the conclusion of (Id) and the number of sequents in $\mathcal{D}$ is finite.

We claim that all L-forms of $\mathcal{D}$ are on the path, since the premise e'' of e_k $(k < n)$ other than e_{k+1} does not contain the **ls**-predicate and all sequents of the subtree of $\mathcal{D}$ starting from e'' do not contain the **ls**-predicate. Hence the companion of the bud e_n appears in the path $(e_i)_{i\leq n}$.

Finally we obtain the required infinite path $(e_i)_{i\in\omega}$ of $\mathcal{G}(\mathcal{P})$ by defining $e_{n+1} = e_p$, where e_p is the companion of e_n. □

Finally we show Theorem 5.

Proof of Theorem 5. Suppose that there is a cut-free cyclic proof $\mathcal{P}$ of $\mathbf{ls}(x, y) \vdash \mathbf{sl}(x, y)$ in $\mathrm{CSL}_0^M \mathrm{ID}\omega$. We will show contradiction. By Lemma 7, there is an infinite path $(e_i)_{i\in\omega}$ of $\mathcal{G}(\mathcal{P})$ such that e_0 is $\mathbf{ls}(x, y) \vdash \mathbf{sl}(x, y)$ and each e_i is an L-form. By the global trace condition, there exist $m \in \omega$ and an infinite progressing trace $(\tau_i)_{i\geq m}$ that follows the infinite path $(e_i)_{i\geq m}$. By the construction of the path, the bud appears infinitely many times in the path. Let e_p and e_q be the first and the second occurrences of the bud in the path. Then there is at least one progressing point between τ_p and τ_q. Hence, we have contradiction since $\sharp_{\mapsto}(e_p) < \sharp_{\mapsto}(e_q) = \sharp_{\mapsto}(e_p)$ by Lemma 6. Therefore there is no cut-free cyclic proof of $\mathbf{ls}(x, y) \vdash \mathbf{sl}(x, y)$ in $\mathrm{CSL}_0^M \mathrm{ID}\omega$. □

By combining Proposition 4 and Theorem 5, we can obtain our second main result.

Corollary 2. *The cyclic proof system $CSL_0^M ID\omega$ does not enjoy the cut-elimination property.*

5 Conclusion and future work

In this paper, we have proved that cut-elimination fails in cyclic proof systems for separation logic. We have shown the failure by presenting counter-example sequents that can be proven with cuts but not without cuts. The counter-example sequents are reasonably simple formulas about singly-linked lists, therefore leading one to believe that some form of cuts is necessary for practical uses of cyclic proofs in separation logic. Because it is non-trivial to infer arbitrary cut formulas in general, we envisage automatic provers to include restricted forms of cuts that are suitable for practical proof searches. For instance, the induction hypothesis synthesis pattern employed in Chu et al. [8] may be a reasonable approach to this.

As future work, we plan to investigate the power of various strategies used in cyclic-proof-based provers that can be characterized as restricted forms of cuts. For instance, the approach of [8] can be seen as a cyclic proof system with cuts restricted to only those against buds. An open

question that seems worth investigating is whether all sequents provable in cyclic proof systems are provable with cuts only against buds.

As another line of future work, we would like to investigate whether the cut-elimination property can be recovered by restricting the inductive definition usage. Recall that our counter-examples contain multiple inductive definitions of the same singly-linked list data structure. Perhaps cut-elimination can be recovered if such a situation is avoided.

Finally, we would like to investigate whether cut-elimination fails or not in cyclic proof systems for logics different from separation logic, such as first-order logic and bunched implication logic [5, 4]. An important difference between the cyclic proof systems for separation logic in this paper and those in [5, 4] is the existence of contraction and weakening on antecedents, which precludes a direct application of the proof technique of this paper.

Acknowledgments We wish to thank James Brotherston for valuable discussions about cyclic proofs. We also thank the anonymous reviewers of PPL for their helpful comments. This work is partially supported by JSPS KAKENHI Grant Numbers JP16H05856, JP17H01720, JP18K11161, JP17H01723, JP18K19787, and by JSPS Core-to-Core Program (A. Advanced Research Networks).

References

[1] S. Berardi and M. Tatsuta, Classical System of Martin-Löf's Inductive definitions is not equivalent to cyclic proof system, In: Proceedings of FoSSaCS 2017, *LNCS* 10203 (2017) 301–317.

[2] S. Berardi and M. Tatsuta, Equivalence of inductive definitions and cyclic proofs under arithmetic, In: *Proceedings of 32nd Annual IEEE Symposium on Logic in Computer Science (LICS2017)* (2017) 1–12.

[3] J. Brotherston, Sequent calculus proof systems for inductive definitions, Ph.D thesis, Edinburgh University 2006.

[4] J. Brotherston, Formalised Inductive Reasoning in the Logic of Bunched Implications, In: Proceedings of SAS 2007, *LNCS* 4634 (2007), 87–103.

[5] J. Brotherstonand A. Simpson, Sequent calculi for induction and infinite descent, *Journal of Logic and Computation 21* (6) (2011) 1177–1216.

[6] J. Brotherston, D. Distefano, and R. L. Petersen, Automated cyclic entailment proofs in separation logic, In: Proceedings of CADE-23 (2011) 131–146.

[7] J. Brotherston, N. Gorogiannis, and R. L. Petersen, A Generic Cyclic Theorem Prover, In: Proceedings of APLAS 2012, *LNCS* 7705 (2012), 350–367.

[8] D. H. Chu, J. Jaffar, and M. T. Trinh, Automatic induction proofs of data-structures in imperative programs, In: Proceedings of PLDI 2015, 457–466.

[9] A. Das, D. Pous, Non-wellfounded proof theory for (Kleene+action)(algebras+lattices), In proceedings of CSL 2018, 19:01–19:18.

[10] A. Doumane, On the infinitary proof theory of logics with fixed points, PhD thesis, Paris 7, 2017.

[11] H. H. Nguyen and W. N. Chin, Enhancing Program Verification with Lemmas, In Proceedings of CAV 2008, *LNCS* 5123 (2008), 355–369.

[12] R. Nollet, A. Saurin, and C. Tasson, Local Validity for Circular Proofs in Linear Logic with Fixed Points In Proceedings of CSL 2018, LIPIcs 119 (2018), 35:1–23.

[13] J. C. Reynolds, Separation Logic: A Logic for Shared Mutable Data Structures, In: Proceedings of Seventeenth Annual IEEE Symposium on Logic in Computer Science (LICS2002) (2002) 55–74.

[14] A. Simpson, Cyclic Arithmetic Is Equivalent to Peano Arithmetic, In: Proceedings of FoSSaCS 2017, *LNCS* 10203 (2017) 283–300.

[15] Q. T. Ta, T. C. Le, S. C. Khoo, and W. N. Chin, Automated Mutual Explicit Induction Proof in Separation Logic, In: Proceedings of FM 2016, *LNCS* 9995 (2016) 659–676.

[16] Q. T. Ta, T. C. Le, S. C. Khoo, and W. N. Chin, Automated lemma synthesis in symbolic-heap separation logic, In: Proceedings of POPL 2018.