A uniform proof-theoretic foundation for abstract paraconsistent logic programming

Norihiro Kamide
Kansai Collaboration Center,
National Institute of Advanced Industrial Science and Technology.
kamide@mtc.biglobe.ne.jp
June 18, 2007

Abstract

It is known that paraconsistent logic programming, which is usually based upon a paraconsistent logic, is important in dealing with inconsistency-tolerant and non-monotonic reasoning appropriately. Firstly in this paper, a cut-free single succedent sequent system $S_{N4}$ and a cut-free multiple succedent sequent system $M_{N4}$ are introduced for Nelson’s paraconsistent 4-valued logic $N4$ [3], and the uniformity theorem (with respect to the notion of uniform proofs) which is established by Miller et al. [34] is shown for $S_{N4}$ and $M_{N4}$. The framework using $S_{N4}$ provides us with an abstract paraconsistent logic programming language that can express inconsistency-tolerant reasoning and inexact information by using the properties of paraconsistency and constructible falsity. The framework using $M_{N4}$ gives an abstract paraconsistent disjunctive logic programming language that can allow to express disjunctive (indefinite) information in the program clauses. Secondly, a cut-free single succedent sequent system $S_{N16}$ is introduced for an extension $N16$ of $N4$, which is a variant of Shramko and Wansing’s 16-valued logics [42], and the uniformity theorem for $S_{N16}$ is shown. The framework using $S_{N16}$ produces an abstract (extended) paraconsistent logic programming language that can also express a certain kind of synonymous information. Thirdly, a cut-free single succedent sequent system $S_C$ is introduced for a fragment of Wansing’s non-commutative logic COSPL [46], which is a non-commutative version of $N4$, and the uniformity theorem for $S_C$ is shown. The framework using $S_C$ provides us with an abstract paraconsistent ordered linear logic programming language that can represent both ordered and hierarchical information. The results of this paper are regarded as natural extensions of the results by Miller et al. [34] and by Harland et al. [16].
1 Introduction

1.1 Paraconsistent logic programming languages

It is known that negation handling in logic programming is a very important issue (see e.g. [5]). In such an issue, paraconsistent logic programming is especially important in dealing with inconsistency-tolerant and non-monotonic reasoning appropriately, and thus a number of interesting results in this area have been obtained by many researchers (see e.g. [6, 10, 11, 23]). Paraconsistent logic programming languages and semantics are usually based upon some paraconsistent logics such as Belnap’s 4-valued logic, bilattice logics, annotated logics and Nelson’s logics. A number of generalized or extended versions of Belnap’s 4-valued logic and Nelson’s logics have also been widely studied as various bilattice- and trilattice-based logics and logic programming semantics (see e.g. [6, 7, 12, 42]). In particular, trilattice-based logics have recently been studied by Shramko and Wansing [42] as some useful 16-valued logics. The base logics adopted in this paper are Nelson’s paraconsistent 4-valued logic N4 [3], a new extension N16 of N4, which is regarded as a variant of Shramko and Wansing’s 16-valued logics [42], and Wansing’s non-commutative logic COSPL (constructive sequential propositional logic) [46, 47], which is a non-commutative version of N4.

The logic N4 (or equivalently called N¬), which is the paraconsistent variant of Nelson’s constructive logic N with strong negation [32], has been studied by many researchers (see e.g. [35, 44, 46]). It is observed that N4 is an extension of both positive intuitionistic logic and Belnap’s 4-valued logic, and it has two desirable properties of negation connective: paraconsistency and constructible falsity. Logic programming languages based on some Nelson’s logics were introduced and studied by many researchers (see e.g. [2, 35, 36, 44]).

Although bilattices and their logical counterparts have been used as logic programming semantics in a broad range of knowledge representation in AI, trilattices [41] and their logical counterparts [42] have not yet been used as logic programming tools. However, such a logic programming approach using trilattice-based 16-valued logics may be promising, because trilattices are known as a natural generalization of bilattices and also as a good representation tool for constructive truth value spaces [41]. The present paper’s approach based on N16 may thus be interesting as the first attempt.

The logic COSPL has not yet been applied to computer science, but an extended logic WILL (= COSPL with exchange rule), which is a conservative extension of the propositional fragment of non-modal intuitionistic linear logic, has been applied in [24, 18]. For example, Petri nets with inhibitor arcs and electric circuits can

\footnote{A logic that can include such a 16-valued logic has recently been posed by Kamide in [22] as a Gentzen-type multiple-succedent sequent system, which is an extension of a system for implication-free positive classical logic. For more information on some relationships between the logics mentioned above, see e.g. [20, 21].}
be described and verified using WILL. In the present paper, a COSPL-based logic programming framework is introduced in order to represent bipolar preferences in decision aiding theory and taxonomic trees appearing in real life situations.

1.2 Uniform proof-theoretic approaches

It is known that the notion of uniform proofs is very useful in formalizing and designing logic programming languages naturally. This notion was originally established by Miller et al. [34], and various extensions, generalizations and implementations of the notion have been studied by many computer scientists (see, e.g. [16, 17, 30, 31]). Roughly speaking, uniform proofs are based on a goal-directed bottom-up proof search using Gentzen-type sequent systems for the so-called hereditary Harrop formulas. The hereditary Harrop formulas are regarded as natural extensions and generalizations of the Horn clauses used in the language Prolog. Uniform proofs give us not only a concrete way of presenting richer logic programming languages, but also an operational way of evaluating logic programs as an interpreter.

Disjunctive logic programming, in which the program clauses set allows disjunctive formulas, is known as very useful in expressing indefinite information, and thus a number of interesting results have been obtained (see e.g. [27, 28]). Uniform proof-theoretic characterizations of disjunctive logic programming were studied by Nadathur et al. [31] and by Harland et al. [16]. In [31], by introducing a more general logic where classical provability coincides with intuitionistic provability, a uniform proof-theoretic disjunctive logic programming language was described by identifying an appropriate class of formulas that can permit disjunctive program clauses. In [16], a similar class of formulas as in [31] was identified by using a multiple-succedent sequent system LM for intuitionistic logic, and the uniformity theorem for LM was proved based on a comprehensive investigation of the property of inference permutability [25].

Ordered linear logic programming, which is a suitable refinement of linear logic programming, has been studied by Polakow et al. (see [37, 38] and the references therein). The notion of “ordered hypothesis” in ordered linear logic programs was introduced by Polakow and Pfenning, and the uniform-proof theoretic foundation for this notion was also established [37]. The COSPL-based framework presented is, on the other hand, not an extension of Polakow and Pfenning’s one, but the spirit of the “ordered hypothesis” seems to be inherited.

1.3 The results of this paper

In this paper, a proof-theoretic foundation for paraconsistent logic programming languages is obtained using the notion of uniform proofs, and this foundation is mainly based on four cut-free Gentzen-type sequent systems $S_{N4}$ (single-succedent system for N4), $M_{N4}$ (multiple-succedent system for N4), $S_{N16}$ (single-succedent system for N16), and $S_C$ (single-succedent system for a fragment of COSPL).
The contents of this paper are then summarized as follows.

In Section 2, the systems $S_{N4}$, $M_{N4}$, $S_{N16}$, $S_{N16\#}$ and $S_{C}$ are introduced and discussed. $S_{N4}$ is an extension of the $\{\to, \land, \lor\}$-fragment of LJ for intuitionistic logic, $M_{N4}$ is an extension of the $\{\to, \land, \lor\}$-fragment of LM [16] for intuitionistic logic, $S_{N16}$ is an extension of $S_{N4}$, $S_{N16\#}$ is a simplification of $S_{N16}$, and $S_{C}$ is a system of Wansing’s COSPL [46]. Along the lines of [35, 46], cut-elimination theorem, paraconsistency and constructible falsity are presented for $S_{N4}$. By using Rautenberg’s embedding [40], cut-elimination theorem is proved for $M_{N4}$ as a new result. By using a similar method, cut-elimination theorem and paraconsistency are also proved for $S_{N16}$ and $S_{N16\#}$. The cut-elimination theorem and paraconsistency for $S_{C}$ are known [46].

In Section 3, it is explained that the sequent-based paradigm presented is useful in describing medical reasoning from the point of view of paraconsistency, constructible falsity and synonymous falsity. Some illustrative examples including bipolar preference modeling and taxonomic tree handling are also discussed as a virtue of non-commutativity of $S_{C}$.

In Section 4, firstly, extended notions of the uniform proofs and hereditary Harrop formulas are introduced by adding some conditions on the strong negation connective $\sim$. These notions are regarded as natural extensions of the original notions by Miller et al. [34]. By using the results of Section 2 (and Appendix), the uniformity theorem for $S_{N4}$ with respect to these notions is proved. It is thus shown that the framework based on $S_{N4}$ provides us with an abstract paraconsistent logic programming language. The uniformity theorem for $M_{N4}$ is presented as a natural extension of the result by Harland et al. [16], and hence this framework gives a proof-theoretic interpretation of paraconsistent disjunctive logic programming. The uniformity theorem for $S_{N16}$ (and $S_{N16\#}$) is also presented as an extension of the theorem for $S_{N4}$. This result can produce various negation expressions in logic programming using several kinds of paraconsistent quasi-negation connectives. Using the same way, the uniformity theorem for $S_{C}$ is presented. This framework gives an abstract paraconsistent ordered linear logic programming language.

In Section 5, this paper is concluded, and in Appendix, the inference permutabilities, which are used to show the uniformity theorems for the underlying systems, are briefly discussed.

Since the results presented in this paper can straightforwardly be extended to the first-order predicate versions, such extended results are omitted. Although other important issues including the issues of language design (such as backchaining) and of the notions of simple proof and maximality are not discussed in this paper, analogous results may hold for the proposed frameworks.

Since all logics discussed in this paper are formulated as sequent systems, we will occasionally identify a sequent system with the logic determined by it.
2  Sequent systems $S_{N4}$, $M_{N4}$, $S_{N16}$ and $S_C$

2.1 Preliminaries w.r.t. $S_{N4}$, $M_{N4}$ and $S_{N16}$

The usual propositional language with the strong negation connective $\sim$ and without falsum and truth constants is used in this paper. Greek lower-case letters $\alpha, \beta, \gamma, ...$ are used to denote formulas. Greek capital letters $\Gamma, \Delta, ...$ are used to represent finite (possibly empty) sets of formulas. A sequent is an expression of the form $\Gamma \Rightarrow \gamma$ or $\Gamma \Rightarrow \Delta$, where $\Gamma$ and $\gamma$ ($\Delta$) are called the antecedent and the succedent, respectively. If a sequent $\Gamma \Rightarrow \gamma$ or $\Gamma \Rightarrow \Delta$ is provable in a sequent system $L$, then such a fact is denoted as $L \vdash \Gamma \Rightarrow \gamma$ ($L \vdash \Gamma \Rightarrow \Delta$). Sometimes $\vdash L$ means the provability of $L$ or the system $L$ itself. An expression $\alpha \leftrightarrow \beta$ means $L \vdash \alpha \Rightarrow \beta$ and $L \vdash \beta \Rightarrow \alpha$ for a sequent system $L$.

**Definition 2.1 (Paraconsistency)** A sequent system $L$ is called explosive with respect to a negation-like connective $\sharp$ if $L \vdash \alpha, \sharp\alpha \Rightarrow \beta$ for any formulas $\alpha$ and $\beta$. A sequent system $L$ is called paraconsistent with respect to $\sharp$ if $L$ is not explosive with respect to $\sharp$.

2.2 $S_{N4}$

First, we present a cut-free sequent calculus $S_{N4}$ for N4, which was introduced and discussed in [35, 46].

**Definition 2.2 ($S_{N4}$)** The initial sequents of $S_{N4}$ are of the form: $\alpha \Rightarrow \alpha$.

The cut rule of $S_{N4}$ is of the form:

$$\frac{\Gamma \Rightarrow \gamma \quad \alpha, \Gamma \Rightarrow \gamma}{\Gamma \Rightarrow \gamma} \quad \text{(cut).}$$

The inference rules of $S_{N4}$ are of the form:

$$\frac{\alpha, \beta, \Gamma \Rightarrow \gamma}{\alpha \Rightarrow \beta, \Gamma \Rightarrow \gamma} \quad \text{(→left)} \quad \frac{\alpha, \Gamma \Rightarrow \beta}{\Gamma \Rightarrow \alpha \rightarrow \beta} \quad \text{(→right)}$$

$$\frac{\alpha \land \beta, \Gamma \Rightarrow \gamma}{\alpha, \Gamma \Rightarrow \gamma} \quad \text{(∧ left)} \quad \frac{\alpha, \Gamma \Rightarrow \gamma}{\Gamma \Rightarrow \alpha \land \beta} \quad \text{(∧ right)}$$

$$\frac{\alpha, \Gamma \Rightarrow \gamma}{\sim \sim \alpha, \Gamma \Rightarrow \gamma} \quad \text{(¬ left)} \quad \frac{\sim \alpha, \Gamma \Rightarrow \gamma}{\Gamma \Rightarrow \sim \sim \alpha} \quad \text{(¬ right)}$$

$$\frac{\sim \alpha, \sim \beta, \Gamma \Rightarrow \gamma}{\sim (\alpha \rightarrow \beta), \Gamma \Rightarrow \gamma} \quad \text{(¬→ left)} \quad \frac{\Gamma \Rightarrow \sim \alpha}{\sim \alpha, \Gamma \Rightarrow \sim \beta} \quad \text{(¬→ right)} \quad \frac{\sim \sim \alpha, \sim \beta, \Gamma \Rightarrow \gamma}{\sim (\alpha \land \beta), \Gamma \Rightarrow \gamma} \quad \text{(¬ ∧ left)} \quad \frac{\sim \sim (\alpha \land \beta), \Gamma \Rightarrow \gamma}{\Gamma \Rightarrow \sim (\alpha \land \beta)} \quad \text{(¬ ∧ right 1)}$$

$$\frac{\Gamma \Rightarrow \sim \alpha}{\sim (\alpha \lor \beta), \Gamma \Rightarrow \gamma} \quad \text{(¬ ∨ left)} \quad \frac{\sim \alpha, \sim \beta, \Gamma \Rightarrow \gamma}{\sim \sim (\alpha \lor \beta), \Gamma \Rightarrow \gamma} \quad \text{(¬ ∨ right 1)} \quad \frac{\sim \sim (\alpha \lor \beta), \Gamma \Rightarrow \gamma}{\Gamma \Rightarrow \sim (\alpha \lor \beta)} \quad \text{(¬ ∨ right 2)}$$
It is remarked that the exchange and contraction rules are omitted in $S_{N4}$, since the antecedents of the sequents in $S_{N4}$ are sets. It is noted that the $\neg$-free part of $S_{N4}$ is just a sequent calculus for positive intuitionistic logic.

It is noted that the rules ($\neg$-left), ($\neg\rightarrow$-left) and ($\neg\lor$-left) have the same form as ($\land$-left), the rules ($\neg\rightarrow$-right) and ($\neg\lor$-right) have the same form as ($\land$-right), and the rules ($\neg$-right) and ($\neg\land$-right1,2) have the same form as ($\lor$-right1,2). By these similarities, these rules have the same syntactic behavior, e.g. for the permutation properties discussed in [25, 16], ($\neg\lor$-left) has the same permutation property for ($\land$-left).

For any formulas $\alpha$ and $\beta$, the following conditions hold for $S_{N4}$: (1) $\neg\alpha \leftrightarrow \alpha$, (2) $\neg(\alpha\rightarrow\beta) \leftrightarrow \alpha \land \neg\beta$, (3) $\neg(\alpha \land \beta) \leftrightarrow \neg\alpha \lor \neg\beta$ and (4) $\neg(\alpha \lor \beta) \leftrightarrow \neg\alpha \land \neg\beta$. The conditions (2) and (4) mean that $\neg(\alpha\rightarrow\beta)$ and $\neg(\alpha \lor \beta)$ can be treated as conjunction formulas, and the condition (3) means that $\neg(\alpha \lor \beta)$ can be treated as a disjunction formula. These conditions and facts implicitly reflect to Definitions 4.2 and 4.3 in a later section.

The following theorem is known [35, 46].

**Theorem 2.3 (Cut-elimination theorem for $S_{N4}$)** For any sequent $\Gamma \Rightarrow \gamma$, if $S_{N4} \vdash \Gamma \Rightarrow \gamma$, then $S_{N4} - \text{(cut)} \vdash \Gamma \Rightarrow \gamma$.

Using Theorem 2.3, we can derive:

**Corollary 2.4 (Constructible falsity for $S_{N4}$)** For any formulas $\alpha$ and $\beta$, if $S_{N4} \vdash \Rightarrow \neg(\alpha \land \beta)$, then either $S_{N4} \vdash \Rightarrow \neg\alpha$ or $S_{N4} \vdash \Rightarrow \neg\beta$.

**Corollary 2.5 (Paraconsistency for $S_{N4}$)** $S_{N4}$ is paraconsistent with respect to $\neg$.

### 2.3 $M_{N4}$

We present a new sequent calculus $M_{N4}$, which is an extension of the $\{\rightarrow, \land, \lor\}$-fragment of the (cut-free) multiple-succedent sequent system LM [16, 45] for intuitionistic logic, and prove the cut-elimination theorem for $M_{N4}$. The calculus $M_{N4}$ is useful to obtain an abstract paraconsistent disjunctive logic programming language, which can be used to model certain types of uncertain and indefinite information.

**Definition 2.6 ($M_{N4}$)** The initial sequents of $M_{N4}$ are of the form: $\alpha \Rightarrow \alpha$.

The cut rule of $M_{N4}$ is of the form:

$$
\frac{\Gamma \Rightarrow \Delta, \alpha \quad \alpha, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta} \quad (\text{cut}^m).
$$

---

\[2\] It is known that the connective $\sim$ represents the “refutability” and “disproof” interpretations [46]. It can thus be seen that an intuitive meaning of $\sim\alpha$ is “$\alpha$ is refutable”.
The inference rules of $M_{N4}$ are of the form:

\[
\frac{\Gamma \Rightarrow \Delta}{\alpha, \Gamma \Rightarrow \Delta} \quad (\text{we}^1) \quad \frac{\Gamma \Rightarrow \Delta, \alpha}{\Gamma \Rightarrow \Delta, \alpha} \quad (\text{we}^2)
\]

\[
\frac{\Gamma \Rightarrow \Delta, \alpha, \beta, \Gamma \Rightarrow \Delta}{\alpha \rightarrow \beta, \Gamma \Rightarrow \Delta} \quad (\rightarrow^m) \quad \frac{\alpha, \Gamma \Rightarrow \beta}{\alpha, \Gamma \Rightarrow \beta} \quad (\rightarrow^m)
\]

\[
\frac{\alpha, \beta, \Gamma \Rightarrow \Delta}{\alpha \land \beta, \Gamma \Rightarrow \Delta} \quad (\land^m) \quad \frac{\Gamma \Rightarrow \Delta, \alpha, \Gamma \Rightarrow \Delta, \beta}{\Gamma \Rightarrow \Delta, \alpha \land \beta} \quad (\land^m)
\]

\[
\frac{\alpha, \Gamma \Rightarrow \Delta}{\alpha \lor \beta, \Gamma \Rightarrow \Delta} \quad (\lor^m) \quad \frac{\Gamma \Rightarrow \Delta, \alpha, \beta}{\Gamma \Rightarrow \Delta, \alpha \lor \beta} \quad (\lor^m)
\]

\[
\frac{\alpha, \Gamma \Rightarrow \Delta}{\sim \alpha, \Gamma \Rightarrow \Delta} \quad (\sim^m) \quad \frac{\Gamma \Rightarrow \Delta, \sim \alpha}{\Gamma \Rightarrow \Delta, \sim \alpha} \quad (\sim^m)
\]

\[
\frac{\sim \alpha, \sim \beta, \Gamma \Rightarrow \Delta}{\sim \alpha, \beta, \Gamma \Rightarrow \Delta} \quad (\sim \lor^m) \quad \frac{\Gamma \Rightarrow \Delta, \sim \alpha, \sim \beta}{\Gamma \Rightarrow \Delta, \sim \alpha \lor \beta} \quad (\sim \lor^m)
\]

\[
\frac{\sim \alpha, \sim \beta, \Gamma \Rightarrow \Delta}{\sim \alpha, \sim \beta, \Gamma \Rightarrow \Delta} \quad (\sim \land^m) \quad \frac{\Gamma \Rightarrow \Delta, \sim \alpha, \sim \beta}{\Gamma \Rightarrow \Delta, \sim \alpha \land \beta} \quad (\sim \land^m)
\]

\[
\frac{\sim \alpha, \sim \beta, \Gamma \Rightarrow \Delta}{\sim \alpha, \sim \beta, \Gamma \Rightarrow \Delta} \quad (\sim \lor^m) \quad \frac{\Gamma \Rightarrow \Delta, \sim \alpha, \sim \beta}{\Gamma \Rightarrow \Delta, \sim \alpha \lor \beta} \quad (\sim \lor^m)
\]

The $\sim$-free part of $M_{N4}$ is called here $M_J$.

It is remarked that the rule ($\rightarrow^m$) is the characteristic rule that represents intuitionistic provability. The rule ($\rightarrow^m$) can be replaced by the rule ($\rightarrow^m$) appearing in $S_{N4}$, and such a system was also a cut-free system for $N4$. A multiple-succedent system with ($\rightarrow^m$) was discussed in [43] for intuitionistic logic.

We give an embedding $f$ of $M_{N4}$ into $M_J$, which was introduced by Rautenberg [40].

**Definition 2.7** We fix a set $\Phi$ of propositional variables, and define the set $\Phi' := \{p' | p \in \Phi\}$ of propositional variables. The language $\mathcal{L}^{\mathcal{\sim}}$ is defined by using $\Phi$, $\rightarrow, \land, \lor$ and $\sim$. The language $\mathcal{L}$ is obtained from $\mathcal{L}^{\mathcal{\sim}}$ by adding $\Phi'$ and by deleting $\sim$.

A mapping $f$ from $\mathcal{L}^{\mathcal{\sim}}$ to $\mathcal{L}$ is defined as follows.

1. $f(p) := p$ and $f(\neg p) := p' \in \Phi'$ for any $p \in \Phi$,
2. $f(\alpha \circ \beta) := f(\alpha) \circ f(\beta)$ where $\circ \in \{\rightarrow, \land, \lor\}$,
3. $f(\neg \neg \alpha) := f(\alpha)$,
4. $f(\neg(\alpha \rightarrow \beta)) := f(\alpha) \land f(\neg \beta)$,
5. $f(\neg(\alpha \land \beta)) := f(\neg \alpha) \lor f(\neg \beta)$,
6. $f(\neg(\alpha \lor \beta)) := f(\neg \alpha) \land f(\neg \beta)$.

---

[3] A mapping for intuitionistic logic with strong negation (rather than $N4$), which is similar to Rautenberg’s one, was first introduced by Gurevich [15].
An expression $f(\Gamma)$ denotes the result of replacing every occurrence of a formula $\alpha$ in $\Gamma$ by an occurrence of $f(\alpha)$.

The following lemma is useful to obtain a simple proof of the cut-elimination theorem for $M_{N4}$.

**Lemma 2.8** Let $\Gamma$ and $\Delta$ be sets of formulas in $L\sim$, and $f$ be the mapping defined in Definition 2.7. (1) If $M_{N4} \vdash \Gamma \Rightarrow \Delta$, then $M_J \vdash f(\Gamma) \Rightarrow f(\Delta)$. (2) If $M_J - (\text{cut}^m) \vdash f(\Gamma) \Rightarrow f(\Delta)$, then $M_{N4} - (\text{cut}^m) \vdash \Gamma \Rightarrow \Delta$.

**Proof** (Sketch) The proof of this lemma is similar to that of Lemma 2.15 (for $S_{N16}$), which will precisely proved. A similar proof was also presented in [22] for other logics.

Using Lemma 2.8, we can prove:

**Theorem 2.9** (Cut-elimination theorem for $M_{N4}$) For any sequent $\Gamma \Rightarrow \Delta$, if $M_{N4} \vdash \Gamma \Rightarrow \Delta$, then $M_{N4} - (\text{cut}^m) \vdash \Gamma \Rightarrow \Delta$.

**Proof** Suppose $M_{N4} \vdash \Gamma \Rightarrow \Delta$. Then, we have $M_J \vdash f(\Gamma) \Rightarrow f(\Delta)$ by Lemma 2.8 (1), and hence $M_J - (\text{cut}^m) \vdash f(\Gamma) \Rightarrow f(\Delta)$ by the cut-elimination theorem for $M_J$ [45]. By Lemma 2.8 (2), we obtain the required fact: $M_{N4} - (\text{cut}^m) \vdash \Gamma \Rightarrow \Delta$.

**Corollary 2.10** (Paraconsistency for $M_{N4}$) $M_{N4}$ is paraconsistent with respect to $\sim$.

It is remarked that the property of constructible falsity cannot be derived using $M_{N4}$.

### 2.4 $S_{N16}$

In order to define $S_{N16}$, the language used is extended with the addition of an involution connective $\star$. Roughly speaking, this connective is regarded as the logical counterpart of the involution operator which is appeared in the algebraic structures of involutive quantales. Using this connective, we can define two kinds of negation connectives by $\sim\star$ and $\star\sim$. As will be mentioned, these connectives can be regarded as a strong negation connective, and these are identified with one connective in the sense of the axiom scheme $\sim\star\alpha \leftrightarrow \star\sim\alpha$. The involution connective $\star$ is thus regarded as an intermediate connective to define such a negation connective. Hence it is difficult to explain the intuitive meaning of $\star$ because of its neutrality: $\star\alpha$ does not mean “$\alpha$ is true” or “$\alpha$ is false”.

---

4 For more information on the involution connective, see e.g., [19].

5 The use of the connective $\star$ is only as a technical device within $S_{N16}$, particularly as there are no examples given of how this connective can be used in a logic program. It seems to have a resemblance to the “structure connectives” used in display logics [48]. For example, the unary
Definition 2.11 ($S_{N4}$) The sequent system $S_{N16}$ is obtained from $S_{N4}$ by adding the initial sequents of the form:

$$\neg p \Rightarrow \neg\neg p, \quad \neg\neg p \Rightarrow \neg p$$

where $p$ is an arbitrary propositional variable \(^6\), and by adding the inference rules of the form:

\[
\begin{align*}
\Gamma \Rightarrow \alpha & \quad (\star \text{right}) \\
\Gamma \Rightarrow \star\star\alpha & \quad (\star\star \text{right}) \\
\star\alpha, \star\beta, \Gamma \Rightarrow \gamma & \quad (\star \text{\& right}) \\
\Gamma \Rightarrow \star\gamma & \quad (\star \text{\& left}) \\
\Gamma \Rightarrow \star\star\gamma & \quad (\star\star \text{\& left}) \\
\Gamma \Rightarrow \star\star\alpha \Rightarrow \star\star\gamma & \quad (\star\star \text{\& left}) \\
\end{align*}
\]

structure connectives $\star$ and $\bullet$ in display logics have the following intuitive meaning: $\star$ shifts structures from one side of an implication $\Rightarrow$ to the other, and $\bullet$ marks the structure in its scope as intentional. The inferential behaviour of $\star$ is such that Boolean negation $\neg$ can be introduced as a logical operation “declaratively identical” to $\star$. The inferential behaviour of $\bullet$ is known as suitable for formalizing introduction rules for various modal operators. In both cases, $\star$ and $\bullet$ are not just a negation connective and a modal operator, respectively, but are regarded as technical intermediate devices.

\(^6\) We can adopt the usual formula type initial sequents $\neg\alpha \Rightarrow \star\neg\alpha$ and $\star\neg\alpha \Rightarrow \neg\neg\alpha$ instead of the propositional variable type initial sequents. The use of these propositional variable type sequents is only from a technical reason: we can simplify the proof of Lemma 2.15 (1), i.e. the cases for the initial sequents. If we adopt the formula type initial sequents, then we may need a similar discussion like Proposition 2.13 in the proof of Lemma 2.15 (1).
We fix a set $\#$ from The language

Definition 2.14

By induction on $\{\ast\}$.

Proposition 2.12 (Quasi falsity) ★★α

The $\{\sim, \ast\}$-free part of $S_{N16}$ is called here $S_J$, which is a sequent system for positive intuitionistic logic.

It is noted that for any formulas $\alpha$ and $\beta$, the following conditions hold for $S_{N16}$:

1. $\ast\ast\alpha \leftrightarrow \alpha$ and $\ast(\alpha \circ \beta) \leftrightarrow \ast\alpha \circ \ast\beta$ where $\circ \in \{\land, \lor, \rightarrow\}$.

It is remarked that the following rules are derivable in $S_{N16}$: for any $\sharp \in \{\ast\sim, \ast\ast\}$,

$$\frac{\Gamma \Rightarrow \ast\alpha}{\Gamma \Rightarrow \sharp\ast\alpha} \quad \frac{\alpha, \ast\beta, \Gamma \Rightarrow \mathbf{A}}{\ast\alpha, \ast\beta, \Gamma \Rightarrow \mathbf{A}}$$

and assuming $\sharp$ as a connective, the inference rules for $\sharp$ which are displayed in Definition 2.11 are the same forms as that for $\sim$.

We then have the following proposition which was mentioned by Shramko and Wansing in [42] for some $\rightarrow$-free 16-valued logics.

Proposition 2.12 (Quasi falsity) For any formulas $\alpha$ and $\beta$, and any $\sharp \in \{\ast\sim, \ast\ast\}$, the following conditions hold for $S_{N16}$: (1) $\sharp\ast\alpha \leftrightarrow \alpha$, (2) $\sharp(\alpha \land \beta) \leftrightarrow \sharp\alpha \land \sharp\beta$, (3) $\sharp(\alpha \lor \beta) \leftrightarrow \sharp\alpha \lor \sharp\beta$, and (4) $\sharp(\alpha \rightarrow \beta) \leftrightarrow \alpha \land \sharp\beta$.

Proposition 2.12 means that $\sharp$ is a kind of strong negation connective.

It is observed that to introduce the initial sequents ($\sim\ast\mathbf{p} \Rightarrow \sim\ast\mathbf{p}$ and $\ast\sim\mathbf{p} \Rightarrow \sim\ast\mathbf{p}$) and 22 inference rules ($\ast\sim\mathbf{g}$ right) — ($\sim\ast\mathbf{g}$ left) displayed in Definition 2.11 is to characterize Proposition 2.13 below, which is called here the property of synonymous falsity. This property was introduced by Shramko and Wansing [42] as a Hilbert-style axiom scheme. An intuitive meaning of Proposition 2.13 will be explained in a later section.

Proposition 2.13 (Synonymous falsity) For any formula $\alpha$, the condition $\sim\ast\alpha \leftrightarrow \ast\sim\alpha$ holds for $S_{N16}$.

Proof By induction on $\alpha$.

Definition 2.14 We fix a set $\Phi$ of propositional variables, and define the sets $\Phi' := \{p' \mid p \in \Phi\}$, $\Phi'' := \{p'' \mid p \in \Phi\}$ and $\Phi''' := \{p''' \mid p \in \Phi\}$ of propositional variables.

The language $L_{\ast\sim}$ is defined by using $\Phi$, $\land, \lor, \sim$ and $\ast$. The language $L_{16}$ is obtained from $L_{\ast\sim}$ by adding $(\Phi', \Phi'', \Phi''')$ and by deleting both $\sim$ and $\ast$.

A mapping $f$ from $L_{\ast\sim}$ to $L_{16}$ is defined as follows.
1. $f(p) := p$, $f(\neg p) := p' \in \Phi'$, $f(*p) := p'' \in \Phi''$ and $f(\neg *p) := f(\neg p) := p''' \in \Phi'''$ for any $p \in \Phi$.
2. $f(\alpha \circ \beta) := f(\alpha) \circ f(\beta)$ where $\circ \in \{\rightarrow, \land, \lor\}$,
3. $f(\neg\neg\alpha) := f(\alpha)$,
4. $f(\neg(\alpha \lor \beta)) := f(\neg\alpha) \land f(\neg\beta)$,
5. $f(\neg(\alpha \land \beta)) := f(\neg\alpha) \lor f(\neg\beta)$,
6. $f(\neg(\alpha \rightarrow \beta)) := f(\alpha) \land f(\neg\beta)$,
7. $f(*\alpha) := f(\alpha)$,
8. $f(*\alpha \circ \beta) := f(*\alpha) \circ f(*) \beta$ where $\circ \in \{\rightarrow, \land, \lor\}$,
9. $f(\neg*\alpha) := f(\neg*\alpha) := f(\neg\alpha)$,
10. $f(*\neg(\alpha \land \beta)) := f(*\neg\alpha) \lor f(*\neg\beta)$,
11. $f(*\neg(\alpha \lor \beta)) := f(*\neg\alpha) \land f(*\neg\beta)$,
12. $f(*\neg(\alpha \rightarrow \beta)) := f(\alpha) \land f(*\neg\beta)$,
13. $f(*\neg\neg\alpha) := f(*\neg\neg\alpha) := f(*\alpha)$,
14. $f(*\neg(\alpha \land \beta)) := f(*\neg\alpha) \lor f(*\neg\beta)$,
15. $f(*\neg(\alpha \lor \beta)) := f(*\neg\alpha) \land f(*\neg\beta)$,
16. $f(*\neg(\alpha \rightarrow \beta)) := f(\alpha) \land f(*\neg\beta)$.

$f(\Gamma)$ denotes the result of replacing every occurrence of a formula $\alpha$ in $\Gamma$ by an occurrence of $f(\alpha)$.

**Lemma 2.15** Let $\Gamma$ be a set of formulas in $L_\sim$, $\gamma$ be a formula in $L_\sim$, and $f$ be the mapping defined in Definition 2.14. (1) If $S_{N_{16}} \vdash \Gamma \Rightarrow \gamma$, then $S_{J} \vdash f(\Gamma) \Rightarrow f(\gamma)$. (2) If $S_{J} - \text{(cut)} \vdash f(\Gamma) \Rightarrow f(\gamma)$, then $S_{N_{16}} - \text{(cut)} \vdash \Gamma \Rightarrow \gamma$.

**Proof** First, we prove (1) by induction on a proof $P$ of $\Gamma \Rightarrow \gamma$ in $S_{N_{16}}$. We distinguish the cases according to the last inference of $P$. We show some cases.

Case ($*$*$p \Rightarrow *\neg p$): $P$ is of the form $*p \Rightarrow *\neg p$ where $p$ is a propositional variable. By the condition 1 of the definition of $f$, we have $f(*p) = f(*\neg p) = p'' \in \Phi''$, and hence $S_{J} \vdash p'' \Rightarrow p'''$.

Case ($*p \Rightarrow *\neg\neg\alpha$): The last inference rule of $P$ is of the form:

$$\frac{\alpha, \Sigma \Rightarrow \gamma}{*\neg\neg\alpha, \Sigma \Rightarrow \gamma} (*\neg\neg\alpha).$$

By the hypothesis of induction, we have $S_{J} \vdash f(*\alpha), f(\Sigma) \Rightarrow f(\gamma)$, and hence $S_{J} \vdash f(*\neg\neg\alpha), f(\Sigma) \Rightarrow f(\gamma)$ by the condition 13 of the definition of $f$.

Case ($*\neg\neg\alpha \Rightarrow *\neg\neg\beta$): The last inference rule of $P$ is of the form:

$$\frac{\neg\neg\alpha, \Sigma \Rightarrow \gamma \quad \neg\neg\beta, \Sigma \Rightarrow \gamma}{*\neg\neg(\alpha \land \beta), \Sigma \Rightarrow \gamma} (**\land \left).$$

By the hypothesis of induction, we have $S_{J} \vdash f(*\neg\neg\alpha), f(\Sigma) \Rightarrow f(\gamma)$, and hence $S_{J} \vdash f(*\neg\neg(\alpha \land \beta)), f(\Sigma) \Rightarrow f(\gamma)$ by the condition 13 of the definition of $f$. 
By the hypothesis of induction, we have $S_J \vdash f(\star\sim\alpha), f(\Sigma) \Rightarrow f(\gamma)$ and $S_J \vdash f(\star\sim\beta), f(\Sigma) \Rightarrow f(\gamma)$. Thus, we obtain

$$
\frac{f(\star\sim\alpha), f(\Sigma) \Rightarrow f(\gamma) \quad f(\star\sim\beta), f(\Sigma) \Rightarrow f(\gamma)}{f(\star\sim\alpha) \lor f(\star\sim\beta), f(\Sigma) \Rightarrow f(\gamma)} \quad \text{(vleft)}.
$$

Therefore we obtain the required fact $S_J \vdash f(\star\sim(\alpha \land \beta)), f(\Sigma) \Rightarrow f(\gamma)$ by the condition 14 of the definition of $f$.

Second, we prove (2) by induction on a cut-free proof $Q$ of $f(\Gamma) \Rightarrow f(\gamma)$ in $S_J$. We distinguish the cases according to the last inference of $Q$. We only show the following case. The last inference rule of $Q$ is of the form:

$$
\frac{f(\star\sim\alpha), f(\Sigma) \Rightarrow f(\gamma) \quad f(\star\sim\beta), f(\Sigma) \Rightarrow f(\gamma)}{f(\star\sim\alpha) \lor f(\star\sim\beta), f(\Sigma) \Rightarrow f(\gamma)} \quad \text{(vleft)}
$$

where $f(\star\sim\alpha) \lor f(\star\sim\beta)$ is equivalent to $f(\star\sim(\alpha \land \beta))$ by the condition 15 of the definition of $f$. By the hypothesis of induction, we have $S_{N16} - (\text{cut}) \vdash \star\sim\alpha, \Sigma \Rightarrow \gamma$ and $S_{N16} - (\text{cut}) \vdash \star\sim\beta, \Sigma \Rightarrow \gamma$. Thus, we obtain the required fact:

$$
\frac{\star\sim\alpha, \Sigma \Rightarrow \gamma \quad \star\sim\beta, \Sigma \Rightarrow \gamma}{\star\sim(\alpha \land \beta), \Sigma \Rightarrow \gamma} \quad \text{\textit{\text{(}}\star\sim \text{\textit{\text{)}} left}}.
$$

Using Lemma 2.15, we can prove:

**Theorem 2.16 (Cut-elimination theorem for $S_{N16}$)** For any sequent $\Gamma \Rightarrow \gamma$, if $S_{N16} \vdash \Gamma \Rightarrow \gamma$, then $S_{N16} - (\text{cut}) \vdash \Gamma \Rightarrow \gamma$.

Using Theorem 2.16, we obtain:

**Corollary 2.17 (Constructible falsity for $S_{N16}$)** For any formulas $\alpha$ and $\beta$, and any $\sharp \in \{\sim, \sim\star, \star\sim\}$, if $S_{N16} \vdash \Rightarrow \sharp(\alpha \land \beta)$, then either $S_{N16} \vdash \Rightarrow \sharp\alpha$ or $S_{N16} \vdash \Rightarrow \sharp\beta$.

**Corollary 2.18 (Paraconsistency for $S_{N16}$)** Let $\sharp \in \{\sim, \sim\star, \star\sim\}$. Then, $S_{N16}$ is paraconsistent with respect to $\sharp$.

It is remarked that the corresponding multiple-succedent system $M_{N16}$ and its cut-elimination result can also be obtained.

### 2.5 A simplification of $S_{N16}$

In the previous subsection, we defined two kinds of negation connectives by $\sim\star$ and $\star\sim$. In this subsection, a new connective $\sharp$ which intuitively means $\sharp = \sim\star = \star\sim$ is introduced based on Proposition 2.13 (Synonymous falsity), and then a system with this connective can be simplified. To investigate such a system named here $S_{N16\sharp}$ is motivated to simplify the syntax and program. 

---

7 The idea of simplifying $S_{N16}$ is due to an anonymous referee.
Definition 2.19 ($S_{N16}$) The sequent system $S_{N16}$ is obtained from $S_{N4}$ by adding the inference rules (in Definition 2.11): (⋆ right), (⋆ left), (⋆∧ left), (⋆∧ right), (⋆∨ right1), (⋆∨ right2), (⋆→ left), (⋆→ right), and the inference rules of the form:

\[
\begin{align*}
\Gamma \Rightarrow \star \alpha & \quad \frac{\star \alpha, \Gamma \Rightarrow \gamma}{\neg \star \alpha, \Gamma \Rightarrow \gamma} \quad (\star \neg \text{left}) \\
\Gamma \Rightarrow \neg \alpha & \quad \frac{\neg \alpha, \Gamma \Rightarrow \gamma}{\neg \star \alpha, \Gamma \Rightarrow \gamma} \quad (\star \neg \text{left}) \\
\Gamma \Rightarrow \neg \neg \alpha & \quad \frac{\neg \neg \alpha, \Gamma \Rightarrow \gamma}{\neg \star \alpha, \Gamma \Rightarrow \gamma} \quad (\star \neg \text{right}) \\
\Gamma \Rightarrow \neg \star \alpha & \quad \frac{\neg \star \alpha, \Gamma \Rightarrow \gamma}{\neg \neg \star \alpha, \Gamma \Rightarrow \gamma} \quad (\star \neg \text{right}) \\
\end{align*}
\]

It is noted that $S_{N16}$ is obtained from $S_{N16}$ by replacing $\neg \star$ and $\star \neg$ by $\neg \neg$. It is remarked that the conditions $\neg \neg \star \alpha \leftrightarrow \star \alpha$ and $\neg \neg \star \alpha \leftrightarrow \neg \star \alpha$ hold for $S_{N16}$. On the other hand, the conditions $\neg \star \alpha \leftrightarrow \star \neg \alpha$ and $\neg \neg \alpha \leftrightarrow \neg \neg \alpha$ are not always true in $S_{N16}$, and hence $S_{N16}$ and $S_{N16}$ are different logics.

Using a similar way as in the previous subsection, we can obtain the following.

Theorem 2.20 (Cut-elimination theorem for $S_{N16}$) For any sequent $\Gamma \Rightarrow \gamma$, if $S_{N16} \vdash \Gamma \Rightarrow \gamma$, then $S_{N16} - \text{(cut)} \vdash \Gamma \Rightarrow \gamma$.

Corollary 2.21 (Constructible falsity for $S_{N16}$) For any formulas $\alpha$ and $\beta$, and any $\sharp \in \{\neg, \neg\}$, if $S_{N16} \vdash \Rightarrow \sharp (\alpha \land \beta)$, then either $S_{N16} \vdash \Rightarrow \sharp \alpha$ or $S_{N16} \vdash \Rightarrow \sharp \beta$.

Corollary 2.22 (Paraconsistency for $S_{N16}$) Let $\sharp \in \{\neg, \neg\}$. Then, $S_{N16}$ is paraconsistent with respect to $\sharp$.

It is remarked that the corresponding multiple-succedent system $M_{N16}$ and its cut-elimination result can also be obtained.

2.6 Non-commutative logic frameworks

In a later section, an abstract paraconsistent ordered linear logic programming language based on Wansing’s non-commutative logic COSPL (constructive sequential propositional logic) [46] will be discussed. The logic COSPL is a conservative extension of the propositional fragment of (non-modal) intuitionistic non-commutative linear logic [1]. To introduce the notion of “ordered hypothesis” in programs has already been studied by Polakow, Pfenning et al. in order to refine linear logic.
programming languages. The uniform-proof theoretic foundation for the ordered programming framework by Polakow and Pfenning [37] produces many useful applications such as a model of resource consumption with ordered resource allocation. For more information on the ordered linear logic programming, see e.g. [37, 38] and the references therein. It is remarked that the base logic by Polakow and Pfenning, which is also called an intuitionistic non-commutative linear logic, is different from that in [1]. The present framework based on COSPL is thus not an extension of Polakow and Pfenning’s one, but the spirit of the “ordered hypothesis” seems to be inherited. Although the framework using COSPL can also be adapted to linear logic programming assuming the exchange rule, such a framework is not an extension of the original framework by Hodas and Miller [17], because COSPL has no exponential modality. A Hodas and Miller-like framework for COSPL with the exchange and linear-exponential rules was presented in [18] based on a triple-context calculus, but the uniformity theorem for this calculus has not been proved yet. Also, the COSPL-based framework with the exchange and weakening rules can be considered, and this framework is regarded as a modified extension of the framework discussed by Harland et al. [16] as a contraction-less system. In this paper, we thus only discuss the COSPL-based ordered programming language. Since the COSPL-based framework can easily be extended to the $S_{\text{N4}}$-like framework with the involution connective $\star$, a discussion on such an extension is omitted here.

2.7 $S_C$

In order to define a sequent system $S_C$ for a fragment of COSPL, the language used for $S_{\text{N4}}$ is extended by adding two connectives $\star$ and $\leftarrow$. For $S_C$, Greek capital letters $\Gamma, \Delta, \ldots$ are used to represent finite (possibly empty) sequences of formulas. Parentheses for $\star$ are omitted because $\star$ is associative.

**Definition 2.23** ($S_C$ [46]) The initial sequents of $S_C$ are of the form: $\alpha \Rightarrow \alpha$.

The inference rules of $S_C$ are of the form:

\[
\Gamma \Rightarrow \alpha \quad \Delta, \alpha, \Sigma \Rightarrow \gamma \quad (\text{cut}^\star)
\]

\[
\frac{\Gamma \Rightarrow \alpha \quad \Delta, \alpha \Rightarrow \gamma}{\Delta, \alpha, \Sigma \Rightarrow \gamma} \quad (\text{left}^\star)
\]

\[
\frac{\Gamma \Rightarrow \alpha \quad \Delta, \alpha \Rightarrow \beta, \Sigma \Rightarrow \gamma}{\Delta, \alpha \Rightarrow \beta, \Sigma \Rightarrow \gamma} \quad (\rightarrow\text{left}^\star)
\]

\[
\frac{\Gamma \Rightarrow \alpha \quad \Delta, \alpha \Rightarrow \beta}{\Gamma \Rightarrow \alpha \rightarrow \beta} \quad (\rightarrow\text{right}^\star)
\]

\[
\frac{\Gamma, \alpha \Rightarrow \beta, \Delta \Rightarrow \gamma}{\Gamma, \alpha \Rightarrow \beta \rightarrow \Delta \Rightarrow \gamma} \quad (\leftarrow\text{left}^\star)
\]

\[
\frac{\Gamma, \alpha \Rightarrow \beta}{\Gamma \Rightarrow \alpha \Rightarrow \beta} \quad (\leftarrow\text{right}^\star)
\]

\[
\frac{\Gamma \Rightarrow \alpha \quad \Delta, \beta, \Sigma \Rightarrow \gamma}{\Gamma, \alpha \Rightarrow \beta, \Sigma \Rightarrow \gamma} \quad (\ast\text{left}^\star)
\]

\[
\frac{\Gamma, \alpha \Rightarrow \beta, \Delta \Rightarrow \gamma}{\Gamma \Rightarrow \alpha \Rightarrow \beta, \Delta \Rightarrow \gamma} \quad (\ast\text{right}^\star)
\]

\[
\frac{\Gamma, \alpha, \beta \Rightarrow \gamma}{\Gamma, \alpha \Rightarrow \beta \land \Delta \Rightarrow \gamma} \quad (\land\text{left1}^\star)
\]

\[
\frac{\Gamma, \alpha \Rightarrow \beta, \Delta \Rightarrow \gamma}{\Gamma, \alpha \land \Delta \Rightarrow \gamma} \quad (\land\text{right}^\star)
\]

\[
\frac{\Gamma \Rightarrow \alpha \quad \Gamma \Rightarrow \beta}{\Gamma \Rightarrow \alpha \lor \beta} \quad (\lor\text{left}^\star)
\]

\[
\frac{\Gamma, \alpha \Rightarrow \beta, \Delta \Rightarrow \gamma}{\Gamma \Rightarrow \alpha \Rightarrow \beta \lor \Delta \Rightarrow \gamma} \quad (\lor\text{right}^\star)
\]
For any sequent 

A (non-commutative) sequent system 

For any formulas 

S

Γ is a sequence of formulas, if 

⊢ ⇒ ∼

Definition 2.26 

version.

Corollary 2.25 (Constructible falsity for S

Theorem 2.24 (Cut-elimination theorem for S

The following corollary is an immediate consequence of this theorem.

Corollary 2.27 (Paraconsistency for S

The definition of paraconsistency must be modified for the non-commutative version.

Definition 2.26 A (non-commutative) sequent system L is called explosive with respect to a negation-like connective ⋆ if L ⊨ α, ⋆α ⇒ β or L ⊨ ⋆α, α ⇒ β for any formulas α and β. A (non-commutative) sequent system L is called paraconsistent with respect to ⋆ if L is not explosive with respect to ⋆.

Corollary 2.27 (Paraconsistency for S

S_C is paraconsistent with respect to ⋆.
3 Motivations and illustrative examples

In this section, firstly, it is explained that the sequent-based paradigm based on $S_{N4}$, $M_{N4}$ and $S_{N16}$ is useful to discuss some applications of medical reasoning from the point of view of paraconsistency, constructible falsity and synonymous falsity. Secondly, it is explained that the paradigm based on $S_C$ is useful to discuss bipolar preference modeling and taxonomic tree handling.

3.1 Paraconsistency

It is known that logical systems with paraconsistency can deal with inconsistency-tolerant reasoning more appropriately. Assume a large medical knowledge-base $KB$ of symptoms and diseases, such as an expert system based on the frameworks presented. It can also be assumed that $KB$ is inconsistent in the sense that there is a symptom predicate $s(x)$ such that $\sim s(x), s(x) \in KB$, where $\sim s(x)$ means “a person $x$ does not have a symptom $s$.” This assumption is very realistic, because symptom is a vague concept, which is difficult to determine by any diagnosis. For example, one doctor has said that the person $x$ has the symptom $s$, and another doctor has said that $s$ is absent in $x$. Then, $KB$ does not derive arbitrary disease $d(x)$, which means “a person $x$ suffers from a disease $d$”, since paraconsistency ensures the fact that for some formulas $s(x)$ and $d(x)$, the sequent $\sim s(x), s(x) \Rightarrow d(x)$ is not provable. A data base, which is based on a paraconsistent logic framework, is thus inconsistency-tolerant. In the classical and intuitionistic logics, the sequent $\sim s(x), s(x) \Rightarrow d(x)$ is provable for any disease $d$ (i.e. every patient has every disease), and hence the non-paraconsistent formulation based on these logics are regarded as inappropriate to the application of medical knowledge base. Another convincing example using paraconsistency, which was suggested by a referee, is robot which gets contradictory information from its sensors. Suppose that we directed a robot to proceed. Suppose also that there is a trap around the robot. An expression $\text{front}(\text{trap})$ means “a trap is in the front of the robot”, and $\sim \text{front}(\text{trap})$ means “a trap is in the back of the robot.” Then, the sequent $\text{front}(\text{trap}), \sim \text{front}(\text{trap}) \Rightarrow \text{proceed}$ can express that if the sensors of the robot obtain the contradictory information (since the trap is just one), then the robot always proceeds to the front or back. This situation is also regarded as inappropriate to the move of the robot. For more information on paraconsistency, see e.g. [11, 39].

3.2 Constructible falsity

It is known that the property of constructible falsity guarantees the constructiveness of the underlying negation connective. The disjunction connective $\lor$ of intuitionistic logic is known to be constructive, since it has the disjunction property: if $\Rightarrow \alpha \lor \beta$ is provable, then either $\Rightarrow \alpha$ or $\Rightarrow \beta$ is provable. The property of constructible falsity, which does not hold for intuitionistic logic, is regarded as the dual notion of
the disjunction property, and guarantees that the expression \( \sim \alpha \) is constructive, i.e. there is a finite piece of evidence that the information \( \alpha \) is denied [32]. The constructible falsity property produces the following inference: if \( \Rightarrow \sim (d_1(x) \land d_2(x)) \) is provable for diseases \( d_i \), then either \( \Rightarrow \sim d_1(x) \) or \( \Rightarrow \sim d_2(x) \) is provable (i.e. we have a piece of evidence that a patient \( x \) has no \( d_i \)). It is also known that logics with constructible falsity can allow to express inexact predicates [3]. An inexact predicate is an incomplete predicate in an empirical domain [44]. An example of an inexact predicate is a disease or symptom predicate such as \( \text{depression}(x) \), which means “a person \( x \) suffers from the first-stage depression or melancholia.” This predicate is incomplete in the sense that we can not determine exactly that the formula \( \sim \text{depression}(x) \lor \text{depression}(x) \) is true. This formula is an instance of the law of excluded middle which is unprovable in logics only with strong negation, but provable in classical logic. For more detailed discussions and examples, see e.g. [44, 46].

### 3.3 Synonymous falsity

It is seen that \( \sim \star \) and \( \star \sim \) are a kind of strong negation connective by Proposition 2.12 (Quasi falsity), and both are equivalent in the sense of Proposition 2.13 (Synonymous falsity). This fact intuitively means that the essential meanings of the negation expressions \( \sim \star \alpha \) and \( \star \sim \alpha \) are identical (i.e. “\( \sim \star \alpha \)” is a synonym for “\( \star \sim \alpha \)”), but only the expressions are different. There are a number of examples of such situations in the natural language. For example, the negation of the word “young people” is expressed as both the word “aged people”, e.g. this corresponds to the expression \( \sim \star \text{young}-\text{people} \), and the word “elderly people”, e.g. this corresponds to the expression \( \star \sim \text{young}-\text{people} \), where the words “aged people” and “elderly people” have roughly the same meaning, but only the expressions are different. It can thus be seen that the \( S_{N16} \)-based framework using Proposition 2.13 can express such situations by using \( \sim \star \) and \( \star \sim \). Thus, we call the property displayed in Proposition 2.13, synonymous falsity. The example discussed just above can be used (in a medical data-base) as the situation that a patient is an aged or elderly person. We consider another example for synonymous falsity. Suppose that a patient has the disease of the “manic-depressive psychosis”. Then, “depression” and “melancholia” can be expressed as “\( \star \sim \text{manic}-\text{state} \)” and “\( \sim \star \text{manic}-\text{state} \)”, respectively. In this case, “depression” and “melancholia” are identical as a disease. The following hypothesis is now considered: If a person is an aged person and also has depression, then the person kills oneself. Such a hypothesis is expressed formally as

\[
\text{H}: \vdash \star \sim \text{young}(x), \sim \star \text{manic}(x) \Rightarrow \text{suicide}(x).
\]

Suppose that John is an elderly person, which is expressed by \( \vdash \Rightarrow \sim \star \text{young}(\text{john}) \), and has melancholia, which is expressed by \( \vdash \Rightarrow \sim \star \text{manic}(\text{john}) \). Then, we can
conclude “\( \vdash \Rightarrow \text{suicide}(\text{john}) \)”, i.e., John kills oneself:

\[
\begin{align*}
\Rightarrow \sim\star m(j) & \quad \Rightarrow \sim\star m(j) \Rightarrow \sim\star m(j) \\
\Rightarrow \sim\star m(j) & \quad \Rightarrow \sim\star m(j) \Rightarrow \star \sim m(j) \Rightarrow s(j)
\end{align*}
\]

where \( s(j) \), \( y(j) \) and \( m(j) \) represent \( \text{suicide}(\text{john}) \), \( \text{young}(\text{john}) \) and \( \text{manic}(\text{john}) \), respectively.

### 3.4 Medical reasoning

A simple example for the following diagnosis of a disease \( d_1 \) based on three symptoms \( s_1, s_2, s_3 \), and one disease \( d_2 \) is presented below as a modified version of an example posed in [29]. These symptom and disease predicates are regarded as a kind of inexact predicates that can be expressed based on constructible falsity. The disease \( d_1 \) or \( d_2 \) can also be assumed as a kind of melancholia that can be expressed based on synonymous falsity. A part of a medical knowledge base is expressed as follows:

K1: \( s_1(x), \sim s_1(x), s_2(x), d_2(x) \Rightarrow \sim d_1(x) \)

K2: \( s_2(x), s_3(x) \Rightarrow \sim d_2(x) \)

K3: \( s_3(x), \sim d_2(x) \Rightarrow d_1(x) \)

where K1 means “if a person \( x \) has two symptoms \( s_1, s_2 \) and one disease \( d_2 \), and has no symptom \( s_1 \), then \( x \) has no disease \( d_1 \).” It is remarked that K1 is inconsistent, and such a situation is sometimes appeared despite the effort of doctors and engineers. It is then assumed that the following information is obtained by two doctors through diagnosis conducted for an individual “John”.

Doctor1: \( \vdash \Rightarrow s_1(\text{john}) \quad \vdash \Rightarrow s_2(\text{john}) \quad \vdash \Rightarrow s_3(\text{john}). \)

Doctor2: \( \vdash \Rightarrow \sim s_1(\text{john}) \quad \vdash \Rightarrow s_2(\text{john}) \quad \vdash \Rightarrow s_3(\text{john}). \)

We then ask the question: “\( \vdash \Rightarrow d_1(\text{john})? \)” (i.e. “Does John have \( d_1 \)?”), and we can obtain the answer “Yes”. In this case, if we use a framework based on intuitionistic logic, then we cannot obtain this correct answer, because K1 is always provable (i.e. John has no \( d_1 \)) in such a framework.

It is remarked that the medical knowledge base using the frameworks presented can allow more complex and realistic expressions such as

\[
\sim(\sim d_1(x) \lor d_2(x)), \sim(s_1(x) \Rightarrow d_1(x)), s_1(x), s_2(x) \Rightarrow \sim d_2(x),
\]

where the formulas in such expressions can be hereditary Harrop formulas with \( \sim \) and \( \star \).
3.5 Exception handling

It is known that logic programming with exceptions is an important issue of knowledge representation in AI [26]. An example of exceptions is the following informal rules:

\[
penguin(x) \Rightarrow \sim fly(x) \quad ostrich(x) \Rightarrow \sim fly(x).
\]

These rules are regarded as exceptions for the general rule in the real world:

\[
bird(x) \Rightarrow fly(x).
\]

The following example is a modified version of the example presented in [26], and to use the strong negation \(\sim\) in the example is posed in [2].

\[
\text{not} \sim fly(x), \ bird(x) \Rightarrow fly(x) \quad penguin(x) \Rightarrow \sim fly(x) \quad ostrich(x) \Rightarrow \sim fly(x)
\]

where \text{not} denotes Clark’s negation as failure.

We now consider an example for medical reasoning. Suppose a general rule:

\[
\text{human}(x) \land medicine(x) \Rightarrow recover(x)
\]

which means “if a person \(x\) uses a medicine, then \(x\) makes a recovery from a disease.”

The following rule is regarded as a kind of exceptions:

\[
\text{allergic-human}(x) \land medicine(x) \Rightarrow \sim recover(x)
\]

where \text{allergic-human}(x) means “a person \(x\) is allergic to the underlying medicine.”

3.6 Illustrative examples based on \(\text{S}_C\)

3.6.1 Relevance

\[8\]

It is known that logics without the weakening rules:

\[
\frac{\Gamma, \Delta \Rightarrow \gamma}{\Gamma, \alpha, \Delta \Rightarrow \gamma} \quad (\text{we-l}) \quad \frac{\Gamma \Rightarrow \alpha}{\Gamma} \quad (\text{we-r})
\]

have the property of variable sharing or relevance principle which represents that if \(\alpha \Rightarrow \beta\) is provable where \(\alpha\) and \(\beta\) are not a propositional constant, then \(\alpha\) and \(\beta\) have

\[9\]

There are a number of approaches to exception handling using certain logical frameworks. A typical approach is to use a non-monotonic (defeasible) logic which is usually based upon an extended classical (modal) logic with a non-monotonic mechanism. Such an approach is somewhat different from our approach, since the base logic and mechanism are different. A non-monotonic logic C4, which is a modal logic of consistency over \(N_4\), was studied in [49]. A comparison of various (non-monotonic) negations in logic programming was addressed in [13].

The notion of “relevance” and “resource-sensitivity” discussed below are hardly new, and are common to many logics. To consider the notion of “order” discussed later is the main aim of this subsection.
a same atomic formula in common. Logics with this property are called relevance logic or relevant logics. For example, $S_C$ is regarded as a kind of relevance logics. For more information on relevant logics, see e.g. [4]. To delete the rule (we-l) from a sequent system is to delete the redundancy of the antecedents of the sequents in such a system. For example, the following inference is allowed in logics with (we-l):

\[
\frac{\text{symptom}_1(x), \text{symptom}_2(x) \Rightarrow \text{disease}(x)}{\text{happy}(x), \text{symptom}_1(x), \text{symptom}_2(x) \Rightarrow \text{disease}(x)}
\]

where happy$(x)$ means “a person $x$ is happy.” In this example, the formula happy$(x)$ is clearly redundant (or independent) to derive the fact disease$(x)$, i.e. $x$ has a disease. It is also noted that the rule (we-r) breaks the paraconsistency.

### 3.6.2 Resource-sensitivity

It is known that logics without the contraction rule:

\[
\frac{\Gamma, \alpha, \alpha, \Delta \Rightarrow \gamma}{\Gamma, \alpha, \Delta \Rightarrow \gamma} \text{ (co)}
\]

can elegantly represent the concept of “resource consumption”. For example, we consider a sequent: coin, coin $\Rightarrow$ coffee, which means “if we consume two coins, then we can take a cup of coffee.” Then, if assuming the classical or intuitionistic logic, this sequent is logically equivalent to the sequent: coin $\Rightarrow$ coffee, because of the presence of the contraction rule. On the other hand, we desire to distinguish such two sequents in the sense of the “resource-sensitivity”, i.e. one coin and two coins have the different effect as resources. It is noted that $S_C$ is one of such resource-sensitive logics, since it has no contraction rule.

An appropriate resource consumption example is medicine consumption in medical reasoning. Consider a medicine $m$ as a resource. An expression $m(x) \Rightarrow \text{recover}(x)$ means “if a person $x$ uses a medicine $m$ to recover from a disease, then $x$ makes a recovery from the disease with the medicine.” In this case, $m(x), m(x) \Rightarrow \text{recover}(x)$ and $m(x) \Rightarrow \text{recover}(x)$ have the completely different meaning in the real world, because two medicines and one medicine have the different effect in general.

### 3.6.3 Order

In the case of medicine consumption discussed above, it may not be sufficient to consider the effects of medicines. For example, if we consider two distinct medicines $m_1$ and $m_2$, then the meanings of the following two expressions are regarded as different: $m_1(x), m_2(x) \Rightarrow \text{recover}(x)$ and $m_2(x), m_1(x) \Rightarrow \text{recover}(x)$, because the order of using medicines change the effect of the medicines. In other words, the time order or priority of using medicines is more important in general. A more detailed example is expressed as follows. An expression meal$(x)$ means “a person $x$ have a meal.” Then, $m(x), \text{meal}(x) \Rightarrow \text{recover}(x)$ and $\text{meal}(x), m(x) \Rightarrow \text{recover}(x)$ have the different meaning, i.e. the effect of the medicine $m$ is different whether the medicine is used after or before the meal.
To express such fine-grained medical reasoning, we have to use a non-commutative logic, such as SC, because, for example, logics with the exchange rule:

\[
\frac{\Gamma, \beta, \alpha, \Delta \Rightarrow \gamma}{\Gamma, \alpha, \beta, \Delta \Rightarrow \gamma} \quad (ex)
\]
can not express the order of descending priorities of the use of medicines. It can be known that in a sequent expression

\[
\gamma_1, \gamma_2, ..., \gamma_n \Rightarrow \beta
\]
in SC, the antecedent \((\gamma_1, \gamma_2, ..., \gamma_n)\) can express the time priority of consuming the resources \(\gamma_1, \gamma_2, ..., \gamma_n\), in fact, \((\gamma_1, \gamma_2, ..., \gamma_n)\) is a sequence of formulas in SC, since SC has no exchange rule. It is remarked that two sequents \(\gamma_1, \gamma_2, ..., \gamma_n \Rightarrow \beta\) and \(\gamma_1 \ast \gamma_2 \ast \cdots \ast \gamma_n \Rightarrow \beta\) are logically equivalent in SC, and hence an expression \(\gamma_1 \ast \gamma_2\) means “first \(\gamma_1\) is consumed, next so is \(\gamma_2\).” It is also noted that in two expressions \(\alpha \rightarrow \beta\) and \(\alpha \leftarrow \beta\), the implications \(\rightarrow\) and \(\leftarrow\) represent resource consumption with priority, e.g. \(\rightarrow\) means the consumption of (subscription) descending order priority, and \(\leftarrow\) means the consumption of ascending order priority.

### 3.6.4 Bipolar preferences and taxonomic trees

Preference modeling is a basic activity for any type of decision aiding process, and traditional preference modeling uses the usual traditional mathematical language based on classical logic. On the other hand, classical logic is not always suitable to formalize real life situations, since it is unable to handle inconsistent and/or incomplete information [7]. A issue in preference modeling is to represent a preference statement of the type “I prefer \(\alpha\) to \(\beta\),” which is called a positive preference. In contrast, a negative preference is a statement of the type “I disprefer \(\alpha\) to \(\beta\).” 10 It is remarked that the negative preference is not always the converse of the positive preference, and hence both preferences are needed. Both positive and negative preferences, which are also called bipolar preferences, have been modelled by Benferhat et al. [8] using a possibilistic logic, and by Bistarell et al. [9] using soft-constraints.

In this paper, the bipolar preference statements: “I prefer \(\alpha\) to \(\beta\)” and “I disprefer \(\alpha\) to \(\beta\)” are respectively expressed formally as \(\alpha, \beta \Rightarrow \gamma\) and \(\sim \alpha, \sim \beta \Rightarrow \gamma\) where \(\gamma\) indicates a common sort of \(\alpha\) and \(\beta\). A generalization of positive preference expressions: \(\alpha_1, \alpha_2, \alpha_3, ..., \alpha_n \Rightarrow \gamma\) means a descending order \((\alpha_1, \alpha_2, \alpha_3, ..., \alpha_n)\) of positive preferences. Examples of such expressions are:

\[
\begin{align*}
\text{fruit}(x), \text{vegetable}(x) & \Rightarrow \text{food}(x) \\
\text{apple}(x), \text{orange}(x), \text{banana}(x) & \Rightarrow \text{fruit}(x) \\
\text{tomato}(x), \text{carrot}(x), \text{cucumber}(x) & \Rightarrow \text{vegetable}(x) \\
\text{Taiwan-banana}(x), \text{Philippine-banana}(x) & \Rightarrow \text{banana}(x)
\end{align*}
\]

---

10 It is explained in [9] that a positive preference is a statement of the type “I like \(\alpha\), and I like \(\beta\) even more than \(\alpha\),” and a negative preference is a statement of the type “I don’t like \(\alpha\), and I really don’t like \(\beta\).”
which represent the preference of a person $x$ for foods, and also address the following tree that represents a taxonomic hierarchy for foods:

```
food
  /    \
fruit  vegetable
    /     \
apple  orange  banana  tomato  carrot  cucumber
```

Taiwan banana  Philippine banana

An $S_C$-based proof representation of this taxonomic tree is:

$$
tom, car, cuc \Rightarrow veg  \quad fru, veg \Rightarrow food
$$

$$
tban, Pban \Rightarrow ban  \quad app, ora, ban, tom, car, cuc \Rightarrow food
$$

where the names of the foods are abbreviated.

It is remarked that the representability of $S_C$ for taxonomic tree structures for preferences is essentially depend on the non-commutativity of $S_C$.

4 Abstract paraconsistent logic programming languages

4.1 Preliminaries w.r.t. $S_{N4}$, $M_{N4}$ and $S_{N16}$

For the explanation of the notion of uniform proof and related notions, we will follow [34, 16]. The logic programming interpretation of a sequent $\Gamma \Rightarrow \Delta$ is that the antecedent $\Gamma$ represents the program, and the succedent $\Delta$ represents the goal. When searching for a proof of $\Gamma \Rightarrow \Delta$ (i.e. performing computation), the search should be “driven” by $\Delta$, and hence be goal-directed. The proof-theoretic characterization of this property is the notion of uniform proofs, which will be defined.

The syntactic variables with the corresponding general connotations are used:

$D$: A set of formulas that serve as possible definite (program) clauses of a logic programming language.

$G$: A set of formulas that serve as possible queries or goals for such a programming language.
An expression $\Gamma \vdash_U \Delta$, which means an interpreter succeeds on the goal $\Delta \subseteq G$ given the program $\Gamma \subseteq D$, represents that there is a uniform proof (w.r.t. a sequent system) for the sequent $\Gamma \Rightarrow \Delta$.

**Definition 4.1 (Abstract logic programming language)** A triple $\langle D, G, \vdash \rangle$ is called an abstract logic programming language if the following condition is satisfied: for any finite subset $\Gamma$ of $D$ and any finite subset $\Delta$ of $G$, $\Gamma \vdash L \Delta$ if and only if $\Gamma \vdash_U \Delta$.  

**4.2 Abstract paraconsistent logic programming language**

**Definition 4.2 (Uniform proof for $S_{N4}$)** A uniform proof for $S_{N4}$ is a cut-free proof in $S_{N4}$ such that, for each occurrence of a sequent $\Gamma \Rightarrow \gamma$ in it, the following conditions are satisfied.

1. If $\gamma$ is $\alpha \land \beta$, then that sequent is inferred by $(\land \text{right})$ from $\Gamma \Rightarrow \alpha$ and $\Gamma \Rightarrow \beta$.
2. If $\gamma$ is $\alpha \lor \beta$, then that sequent is inferred by $(\lor \text{right1})$ or $(\lor \text{right2})$ from $\Gamma \Rightarrow \alpha$ or $\Gamma \Rightarrow \beta$, respectively.
3. If $\gamma$ is $\alpha \rightarrow \beta$, then that sequent is inferred by $(\rightarrow \text{right})$ from $\Gamma \Rightarrow \alpha$.
4. If $\gamma$ is $\sim\sim\alpha$, then that sequent is inferred by $(\sim \text{right})$ from $\Gamma \Rightarrow \alpha$.
5. If $\gamma$ is $\sim(\alpha \land \beta)$, then that sequent is inferred by $(\sim \land \text{right1})$ or $(\sim \land \text{right2})$ from $\Gamma \Rightarrow \sim\alpha$ or $\Gamma \Rightarrow \sim\beta$, respectively.
6. If $\gamma$ is $\sim(\alpha \lor \beta)$, then that sequent is inferred by $(\sim \lor \text{right})$ from $\Gamma \Rightarrow \sim\alpha$ and $\Gamma \Rightarrow \sim\beta$.
7. If $\gamma$ is $\sim(\alpha \rightarrow \beta)$, then that sequent is inferred by $(\sim \rightarrow \text{right})$ from $\Gamma \Rightarrow \alpha$ and $\Gamma \Rightarrow \sim\beta$.

In other words, a uniform proof for $S_{N4}$ is a cut-free proof (in $S_{N4}$) in which each occurrence of a sequent whose succedent contains a non-atomic (or non-negated-atomic) formula is the lower sequent of the inference rule that introduces its outermost connective (or negated-outermost connective, e.g. $\sim(\alpha \land \beta)$).

**Definition 4.3 (Hereditary Harrop formula with $\sim$)** Hereditary Harrop formulas with strong negation are defined by the following grammar using mutual induction, assuming $p$ represents atomic formulas:

$$
\begin{align*}
\gamma &::= p \mid \gamma \land \gamma \mid \gamma \lor \gamma \mid \delta \rightarrow \gamma \mid \sim p \mid \sim \sim \gamma \mid \sim(\gamma \land \gamma) \mid \sim(\gamma \lor \gamma) \mid \sim(\gamma \rightarrow \gamma) \\
\delta &::= p \mid \delta \land \delta \mid \gamma \rightarrow \delta \mid \sim p \mid \sim \delta \mid \sim(\delta \lor \delta) \mid \sim(\delta \rightarrow \delta).
\end{align*}
$$

---

11 In the single-succedent case, $\Delta$ becomes a single formula $\gamma$.  

23
The formulas $\gamma$ which are defined above are called $\gamma$-formulas or goal formulas, and the formulas $\delta$ which are defined above are called $\delta$-formulas or program (definite) formulas.

**Theorem 4.4 (Uniformity in $S_{N4}$)** Let $D_1$ be the set of $\delta$-formulas and $G_1$ be the set of $\gamma$-formulas. Then, the triple $\langle D_1, G_1, \vdash_{S_{N4}} \rangle$ is an abstract logic programming language.

**Proof** We only give a sketch of the proof. The theorem for the $\sim$-free part is proved in [34], and a similar theorem is also proved in [33]. Let $\Gamma_1$ be a finite subset of $D_1$ and $\gamma$ be a member of $G$. It is obvious that $\Gamma \vdash_U \gamma$ implies $\Gamma \vdash_{S_{N4}} \gamma$, since any uniform proof is also a cut-free proof in $S_{N4}$. Thus, we show that $\Gamma \vdash_{S_{N4}} \gamma$ implies $\Gamma \vdash_U \gamma$. Suppose $\Gamma \vdash_{S_{N4}} \gamma$. Then we have $\Gamma \vdash_{S_{N4}-(cut)} \gamma$ by Theorem 2.3. Let $P$ be a cut-free proof of $\Gamma \Rightarrow \gamma$ in $S_{N4}$. In order to obtain a uniform proof, we now consider to lift-up the left-inference rules and (we) in $P$ over the right rules. By Theorem 2.3, the subformula property-like property holds for $S_{N4}-(cut)$, and hence every formula occurring in any sequent in $P$ is a subformula or negated-subformula of some formulas occurring in $\Gamma \Rightarrow \gamma$. By this fact and Definition 4.3, there is no application of the rule ($\lor$left) or ($\sim \land$left) in $P$. 12 Using the facts in Appendix for inference permutability [25, 14, 16], we then have the fact that the remained (left-)inference rules ($\land$left), ($\rightarrow$left), (we), ($\sim$left), ($\sim \lor$left) and ($\sim \rightarrow$left) appearing in $P$ can move the upper direction in $P$ as lift-up as possible, i.e. all of these possible (left-)rules permute upward as possible. Then we can eliminate non-uniform inferences by permuting left-rules up. A resulting proof $P'$ by lifting-up these inference rules appropriately is a uniform proof of $\Gamma \Rightarrow \gamma$, and hence $\Gamma \vdash_U \gamma$.

### 4.3 Abstract paraconsistent disjunctive logic programming language

**Definition 4.5 (Uniform proof for $M_{N4}$)** A uniform proof for $M_{N4}$ is a cut-free proof in $M_{N4}$ such that, every sequent (in the proof) which contains a non-atomic (or non-negated-atomic) formula in the succedent is the conclusion of a right rule. 13

The following definition is an extension of the definition of the formulas proposed in [16, 31], a definition which gives a proof-theoretic characterization of the notion of disjunctive logic programming [27]. Here we call such formulas hereditary disjunctive formulas. The following definition gives a proof-theoretic characterization of paraconsistent disjunctive logic programming.

---

12 The rules ($\lor$left) and ($\sim \land$left) fail to permute upward.
13 Although we can give a precise definition which is similar to Definition 4.2, we omit such a definition.
Definition 4.6 (Hereditary disjunctive formula with \( \sim \)) Hereditary disjunctive formulas with strong negation are defined by the following grammar using mutual induction, assuming \( p \) represents atomic formulas:

\[
\begin{align*}
\gamma &::= p \mid \gamma \land \gamma \mid \gamma \lor \gamma \mid \sim p \mid \sim \sim \gamma \mid \sim(\gamma \land \gamma) \mid \sim(\gamma \lor \gamma) \mid \sim(\gamma \rightarrow \gamma) \\
\delta &::= p \mid \delta \land \delta \mid \delta \lor \delta \mid \gamma \rightarrow \delta \mid \sim p \mid \sim \sim \delta \mid \sim(\delta \land \delta) \mid \sim(\delta \lor \delta) \mid \sim(\delta \rightarrow \delta).
\end{align*}
\]

The formulas \( \gamma \) which are defined above are called disjunctive \( \gamma \)-formulas, and the formulas \( \delta \) which are defined above are called disjunctive \( \delta \)-formulas.

In this definition, an important point is that the program clauses allow us to use disjunctive formulas, i.e. disjunctive (indefinite) information can be represented in the assumption set or program. \(^{14}\) This is also different from the framework using \( SN_4 \). The ability to express indefinite information and case constructions appearing in real life are known to be desirable in some applications: reasoning about declarative specifications, reasoning about actions and diagnosis in medicine \([28]\). Proof-theoretic mechanisms to handle such specifications are thus considered to be important.

Theorem 4.7 (Uniformity in \( MN_4 \)) Let \( D_2 \) be the set of disjunctive \( \delta \)-formulas and \( G_2 \) be the set of disjunctive \( \gamma \)-formulas. Then, the triple \( \langle D_2, G_2, \vdash_{MN_4} \rangle \) is an abstract logic programming language.

Proof The proof of this theorem is similar to the proof in \([16]\), using inference permutabilities. Also, such a proof is performed in the same manner of that of Theorem 4.4. The existence of \( \sim \) is not essential in the proof. \( \square \)

4.4 Abstract extended paraconsistent logic programming language

The framework below is regarded as an extension of the framework based on \( SN_4 \), and hence this is more expressive than that for \( SN_4 \). This framework is motivated to obtain a proof-theoretic programming interpretation for trilattice based 16-valued reasoning. The uniformity theorem for \( SN_{16} \) is regarded as a starting point of such an approach.

Definition 4.8 (Uniform proof for \( SN_{16} \)) A uniform proof for \( SN_{16} \) is a cut-free proof in \( SN_{16} \) such that, every sequent (in the proof) that does not have a formula of the form \( p, \sim p, \star p, \sim \star p, \) or \( \star \sim p \) (\( p \) is an atomic formula) as the succedent is the conclusion of a right rule.

\(^{14}\) We can use the following formula in program clauses:

\[ lh\text{-}\text{broken}(\text{John}) \lor rh\text{-}\text{broken}(\text{John}) \]

which means that John with a broken left arm or a broken right arm, but we do not remember which. For such an expression, see e.g. \([28]\).
Definition 4.9 (Hereditary Harrop formula with \( \sim \) and \( \ast \)) Hereditary Harrop formulas with strong negation and involution are defined by the following grammar using mutual induction, assuming \( p \) represents atomic formulas:

\[
\gamma ::= p \mid \gamma \land \gamma \mid \gamma \lor \gamma \mid \delta \to \gamma \mid \sim p \mid \sim \sim \gamma \mid \sim (\gamma \land \gamma) \mid \sim (\gamma \lor \gamma) \mid \sim (\gamma \to \gamma) \\
\ast p \mid \ast \ast \gamma \mid \ast (\gamma \land \gamma) \mid \ast (\gamma \lor \gamma) \mid \ast (\delta \to \gamma) \\
\sim \ast p \mid \sim \ast \sim \gamma \mid \sim \ast (\gamma \land \gamma) \mid \sim \ast (\gamma \lor \gamma) \mid \sim \ast (\delta \to \gamma) \\
\ast \sim p \mid \ast \ast \sim \gamma \mid \ast \ast \ast \gamma \mid \ast \ast (\gamma \land \gamma) \mid \ast \ast (\gamma \lor \gamma) \mid \ast \ast (\delta \to \gamma)
\]

The formulas \( \gamma \) which are defined above are called synonymous \( \gamma \)-formulas, and the formulas \( \delta \) which are defined above are called synonymous \( \delta \)-formulas. \(^{15}\)

Theorem 4.10 (Uniformity in \( S_{N16} \)) Let \( D_3 \) be the set of synonymous \( \delta \)-formulas and \( G_3 \) be the set of synonymous \( \gamma \)-formulas. Then, the triple \((D_3, G_3, \vdash_{S_{N16}})\) is an abstract logic programming language.

Proof (Sketch) The proof of this theorem is similar to that of Theorem 4.4, using some facts in Appendix.

The uniformity theorem for \( S_{N16} \) can also be obtained.

Definition 4.11 (Uniform proof for \( S_{N16} \)) A uniform proof for \( S_{N16} \) is a cut-free proof in \( S_{N16} \) such that, every sequent (in the proof) that does not have a formula of the form \( p, \sim p, \ast p \) or \( \sharp p \) (\( p \) is an atomic formula) as the succedent is the conclusion of a right rule.

Definition 4.12 (Hereditary Harrop formula with \( \sim \), \( \ast \) and \( \sharp \)) Hereditary Harrop formulas with \( \sim \), \( \ast \) and \( \sharp \) are defined by the following grammar using mutual induction, assuming \( p \) represents atomic formulas:

\[
\gamma ::= p \mid \gamma \land \gamma \mid \gamma \lor \gamma \mid \delta \to \gamma \mid \sim p \mid \sim \sim \gamma \mid \sim (\gamma \land \gamma) \mid \sim (\gamma \lor \gamma) \mid \sim (\gamma \to \gamma) \\
\ast p \mid \ast \ast \gamma \mid \ast (\gamma \land \gamma) \mid \ast (\gamma \lor \gamma) \mid \ast (\delta \to \gamma) \\
\sharp p \mid \sharp \sim \gamma \mid \sharp \ast \gamma \mid \sharp (\gamma \land \gamma) \mid \sharp (\gamma \lor \gamma) \mid \sharp (\gamma \to \gamma)
\]

\[
\delta ::= p \mid \delta \land \delta \mid \gamma \to \delta \mid \sim p \mid \sim \sim \delta \mid \sim (\delta \lor \delta) \mid \sim (\delta \to \delta) \\
\ast p \mid \ast \ast \delta \mid \ast (\delta \land \delta) \mid \ast (\gamma \to \delta) \\
\sharp p \mid \sharp \sim \delta \mid \sharp \ast \delta \mid \sharp (\delta \lor \delta) \mid \sharp (\delta \to \delta)
\]

\(^{15}\) Although this definition seems not very intuitive, reading \( \sim \ast \) and \( \ast \sim \) as a negation connective, we may (partially) understand the intuitive meaning of this definition.
The formulas $\gamma$ which are defined above are called simplified synonymous $\gamma$-formulas, and the formulas $\delta$ which are defined above are called simplified synonymous $\delta$-formulas.

**Theorem 4.13 (Uniformity in $S_{N16}^{\#}$)** Let $D_3'$ be the set of simplified synonymous $\delta$-formulas and $G_3'$ be the set of simplified synonymous $\gamma$-formulas. Then, the triple $\langle D_3', G_3', \vdash_{S_{N16}^{\#}} \rangle$ is an abstract logic programming language.

We can consider the framework for the multiple-succedent versions $M_{N16}$ and $M_{N16}^{\#}$ in a similar way.

### 4.5 Abstract paraconsistent ordered linear logic programming language

The framework presented below are motivated to obtain a proof-theoretic account of ordered programming languages. The main difference for the other approaches in this paper is that the program clauses are treated as a sequence, which can be interpreted as an order of programs. Thus, we have to change the definition of abstract programming language.

The syntactic variables with the corresponding general connotations are used:

- $D$: A sequence of formulas that serve as possible ordered definite (program) clauses of a logic programming language.
- $G$: A set of formulas that serve as possible queries or goals for such a programming language.

An expression $\Gamma \vdash_U \gamma$, which means an interpreter succeeds on the goal $\gamma \in G$ given the ordered program $\Gamma$ (a subsequence of $D$), represents that there is a uniform proof (w.r.t a sequent system) for the sequent $\Gamma \Rightarrow \gamma$.

**Definition 4.14 (Abstract ordered linear logic programming language)** A triple $\langle D, G, \vdash_L \rangle$ is called an abstract ordered linear logic programming language if the following condition is satisfied: for any finite subsequence $\Gamma$ of $D$ and any formula $\gamma$ in $G$, $\Gamma \vdash_L \gamma$ if and only if $\Gamma \vdash_U \gamma$.

**Definition 4.15 (Uniform proof for $S_{C}$)** A uniform proof for $S_{C}$ is a cut-free proof in $S_{C}$ such that, every sequent (in the proof) which contains a non-atomic (or non-negated-atomic) formula in the succedent is the conclusion of a right rule.

**Definition 4.16 (Hereditary Harrop formula with $\sim$, $*$ and $\leftarrow$)** Hereditary Harrop formulas with $\sim$, $*$ and $\leftarrow$ are defined by the following grammar using mutual induction, assuming $p$ represents atomic formulas:
\[\gamma ::= p \mid \gamma \land \gamma \mid \gamma \lor \gamma \mid \gamma \ast \gamma \mid \delta \rightarrow \gamma \mid \delta \leftarrow \gamma \mid \sim p \mid \sim \sim \gamma \mid \sim (\gamma \land \gamma) \mid \sim (\gamma \lor \gamma) \mid \sim (\gamma \ast \gamma) \mid \sim (\gamma \rightarrow \gamma) \mid \sim (\gamma \leftarrow \gamma)\]

\[\delta ::= p \mid \delta \land \delta \mid \gamma \rightarrow \delta \mid \gamma \leftarrow \delta \mid \sim p \mid \sim \sim \delta \mid \sim (\delta \lor \delta)\]

The formulas \(\gamma\) which are defined above are called \emph{ordered \(\gamma\)-formulas} or \emph{ordered goal formulas}, and the formulas \(\delta\) which are defined above are called \emph{ordered \(\delta\)-formulas} or \emph{ordered program (definite) formulas}.

**Theorem 4.17 (Uniformity in \(S_C\))** Let \(D_4\) be the sequence of ordered \(\delta\)-formulas and \(G_4\) be the set of ordered \(\gamma\)-formulas. Then, the triple \((D_4, G_4, \vdash_{S_C})\) is an abstract ordered linear logic programming language.

**Proof** (Sketch) This theorem can also be proved by the same way as that for \(S_{N4}\), i.e. the permutability properties of the inference rules of \(S_C\) are used in such a proof. Some facts used to prove this theorem are presented in Appendix.

5 Conclusions

The main results of this paper are the uniformity theorems (Theorems 4.4, 4.7, 4.10, 4.13, and 4.17) for the sequent systems \(S_{N4}\), \(M_{N4}\), \(S_{N16}\), \(S_{N16}\) and \(S_{C}\). The systems \(S_{N4}\) and \(M_{N4}\) are sequent systems for Nelson’s paraconsistent logic \(N4\). The system \(S_{N16}\) is a sequent system for a variant of Shramko and Wansing’s 16-valued logic, which is an extension of \(N4\). The system \(S_{N16}\) is a simplification of \(S_{N16}\). The system \(S_{C}\) is a sequent system for the constant-free fragment of Wansing’s non-commutative logic COSPL, which is regarded as a non-commutative version of \(N4\). Although a system \(M_{N16}\) for the multiple succedent formulation like \(M_{N4}\) and a non-commutative system with \(\ast\) like \(S_{N16}\) are not discussed in this paper, such systems and the corresponding uniformity results can be obtained. The uniformity results presented give us concrete proof-theoretic characterizations of: (1) a paraconsistent logic programming language that can express inconsistency-tolerant reasoning, (2) a paraconsistent disjunctive logic programming language that can allow to express disjunctive (indefinite) information in the program clauses, (3) an extended paraconsistent logic programming language that can also express a certain kind of synonymous information, and (4) a paraconsistent ordered linear logic programming language that can represent both ordered and hierarchical information.

The advantages of the approach presented are as follows. (1) The compatibility between the previous established works and the present one is guaranteed. The

---

\(^{16}\) The permutability properties of the inference rules in classical (one-sided) linear logic were proved by Galmiche and Perrier [14]. The method used in [14] can apply the systems \(S_C\) and \(S_C +\) (ex), by putting some appropriate modifications.

\(^{17}\) These systems and frameworks may be useful, but a concrete application example for them has not been found yet.
present approach is indeed a natural extension of the previous approaches by Miller et al. [34] and by Harland et al. [16]. (2) By the virtue of the uniform proof formalism, the reasoning procedure of the proposed framework can be implemented as an automated reasoning tool, because it is a pure syntactic framework. Thus, it can be used as a base tool for paraconsistent logic programming, which has almost been studied semantically. (3) Paraconsistency and constructible falsity are subsumed in the frameworks, and hence inconsistency-tolerant reasoning and inexact information can suitably be expressed. (4) A property of synonymous falsity is subsumed in the framework using $S_{N16}$, and hence certain synonymous information can be expressed. (5) The notion of ordered hypothesis in programs, which has been studied by Polokow and Pfenning, is partially subsumed in the framework using $S_C$, and using this notion, some bipolar (positive and negative) preferences and taxonomic trees can be described naturally. In conclusion, this approach sheds a new light upon the area of negation handling in logic programming.

Acknowledgments. I would like to thank Prof. Dale Miller and anonymous referees for their valuable comments and kind consideration.

References


6 Appendix: Inference permutability

6.1 Preliminaries

In order to prove the uniformity theorems for the underlying systems, cut-elimination theorems and permutability properties are needed. In Section 2, some cut-elimination theorems were proved simply using some versions of Rautenberg’s embedding. In the present section, the permutability properties (for the underlying systems), which are straightforward extensions of those of the negation (involution)-free fragments, are shortly discussed.

In order to obtain a uniform proof, which is defined as a cut-free goal-directed proof, we have to consider that

(a) the left (right) introduction rules are permuted upward (downward) as possible in a given cut-free proof,

(b) the formulas we can deal with are restricted to a version of hereditary Harrop formulas.

It is known that in some cut-free proofs, some inference rules such as $(\lor\text{left})$ in LJ cannot be permuted upward (over some right introduction rules), i.e. the fact (a) does not hold for a usual (unrestricted) formulas in general. Thus we have to impose the condition (b) in order to delete such left introduction rules in the cut-free proof.

As mentioned above, the order of the inference rules in a proof of a sequent system can often be permuted upward or downward. This property is called the permutabilities of inference rules [14, 16, 25]. In the following explanation of the permutability, we will follow [16].

In an inference rule, the formulas which are present in the premise(s) but not in the conclusion is called the active formulas, and the formula which is present in the conclusion but not in the premise(s) is called the principal formulas. For example, in the following rule:

\[
\frac{\Gamma \Rightarrow \alpha \quad \Gamma \Rightarrow \sim \beta}{\Gamma \Rightarrow \sim (\alpha \rightarrow \beta)} \quad (\sim \rightarrow \text{right}),
\]

$\sim (\alpha \rightarrow \beta)$ is the principal formula of $(\sim \rightarrow \text{right})$, and both $\alpha$ and $\sim \beta$ are active formulas of $(\sim \rightarrow \text{right})$. It is remarked that the rule (we) has a principal formula but has no active formula.

When looking to permute the order of two inference rules in a proof, it is necessary to check that the principal formula of the upper inference rule is not an active formula of the lower one; otherwise, no permutation is possible. When this fact occurs, the two inference rules are said to be in permutation position. For example, consider the proof of the form:

\[
\frac{\vdots}{\Gamma \Rightarrow \alpha, \sim \beta, \sim \alpha \rightarrow \gamma} \quad (\rightarrow \text{right})
\]

\[
\frac{\sim \alpha, \sim \beta, \Gamma \Rightarrow \delta \rightarrow \gamma}{\sim (\alpha \lor \beta), \Gamma \Rightarrow \delta \rightarrow \gamma} \quad (\sim \lor \text{left})
\]
where the active formulas of (→right) are δ and γ, the principal formula of (→right) is δ → γ, the active formulas of (¬∨ left) are ¬α and ¬β, and the principal formula of (¬∨ left) is ¬(α ∨ β). In this case, the rules (→right) and (¬∨ left) are in permutation position, and then (¬∨ left) can be permuted upward:

\[\frac{\delta, \sim \alpha, \sim \beta, \Gamma \Rightarrow \gamma}{\delta, \sim (\alpha \lor \beta), \Gamma \Rightarrow \gamma} (\sim \lor \text{left})\]

\[\frac{\sim (\alpha \lor \beta), \Gamma \Rightarrow \delta \rightarrow \gamma}{\sim (\alpha \lor \beta), \Gamma \Rightarrow \delta \rightarrow \gamma} (\rightarrow \text{right}).\]

### 6.2 Case for S\textsubscript{N4}

**Proposition 6.1** For a cut-free proof \(P\) of S\textsubscript{N4}, an inference rule \(I \in \{(\rightarrow \text{left}), (\land \text{left}), (\sim \rightarrow \text{left}), (\sim \lor \text{left})\}\) appearing in \(P\) can be permuted upward as possible, i.e. up to initial sequent(s) or a position that the active formula(s) of \(I\) is(are) introduced by other applications of inference rules.

**Proof** (Sketch) It can be proved this proposition in the same way as that for LJ. The result for LJ is well-known, and hence the negation-less part of this proposition is true.

Before to prove this proposition, the following facts are remarked. The form (except forms of the formulas) of the rules (¬left), (¬→left) and (¬∨ left) is similar to that of (∧left). The form of the rules (¬→right) and (¬∨ right) is similar to that of (∧right). The form of the rule (¬∧ left) is similar to that of (∨left). The form of the rules (¬right), (¬∧ right\textsubscript{1}) and (¬∧ right\textsubscript{2}) is similar to that of (∨right\textsubscript{1}) or (∨right\textsubscript{2}). These form similarity or symmetry for ¬-rules derives the same permutability properties as the corresponding ¬-less similar rules.

It is known that the inference rules (we), (∧left) and (¬left) in LJ can be permuted upward in an arbitrary cut-free proof in LJ. Since the form of inference rules (¬→left) and (¬∨ left) is similar to that of (∧left), it can be obtained that the permutability properties of (¬→left) and (¬∨ left) are the same as that of (∧left), i.e. these rules can be permuted upward. Similarly, it can be shown that the permutability property of the inference rule (¬left) is the same as that of (∧left).

We then consider only the case for (¬left), since other cases can also be shown in a similar way. In this case, we have to show that for every inference rule \(I\) just over (¬left) in a given cut-free proof, (¬left) can be permuted over \(I\). Here we consider only the following case:

\[\frac{\alpha, \Gamma \Rightarrow \delta \quad \pi, \alpha, \Gamma \Rightarrow \gamma}{\alpha, \delta \rightarrow \pi, \Gamma \Rightarrow \gamma} (\rightarrow \text{left})\]

\[\frac{\sim \alpha, \delta \rightarrow \pi, \Gamma \Rightarrow \gamma}{\sim \alpha, \delta \rightarrow \pi, \Gamma \Rightarrow \gamma} (\sim \text{left}).\]
This can be transformed into the proof:

\[
\frac{\alpha, \Gamma \Rightarrow \delta}{\sim \sim \alpha, \Gamma \Rightarrow \delta \ (\sim \text{left})} \quad \frac{\pi, \alpha, \Gamma \Rightarrow \gamma}{\pi, \sim \sim \alpha, \Gamma \Rightarrow \gamma \ (\sim \text{left})} \quad \frac{\sim \sim \alpha, \delta \rightarrow \pi, \Gamma \Rightarrow \gamma}{\sim \sim \alpha, \delta \rightarrow \pi, \Gamma \Rightarrow \gamma \ (\sim \text{left}).}
\]

### 6.3 Case for \( S_{N16} \)

**Proposition 6.2** For a cut-free proof \( P \) of \( S_{N16} \), an inference rule \( I \in \{ (\rightarrow \text{left}), (\& \text{left}), (\ast \& \text{left}), (\ast \rightarrow \text{left}), (\sim \sim \rightarrow \text{left}), (\sim \& \text{left}), (\sim \ast \& \text{left}), (\sim \ast \rightarrow \text{left}), (\sim \& \& \text{left}), (\sim \ast \& \& \text{left}), (\sim \& \ast \text{left}), (\sim \ast \& \text{left}), (\sim \& \& \ast \text{left}), (\sim \ast \& \ast \text{left}), (\sim \& \& \& \text{left}) \} \) appearing in \( P \) can be permuted upward as possible.

**Proof** (Sketch) The proof of this proposition can be shown in a similar way as that for \( S_{N4} \). Thus the precise proof is omitted here, but only the following facts are remarked. The form (except forms of formulas) of the rules \((\ast \& \& \text{right}), (\ast \& \& \ast \text{right}), (\sim \& \& \& \text{right})\) is similar to that of \((\sim \& \& \text{right})\). The form of the rules \((\sim \& \& \& \text{left}), (\sim \& \& \& \& \text{left}), (\sim \& \& \& \text{left})\) is similar to that of \((\sim \& \& \& \text{left})\). Using these facts, we can consider the following facts. The proof of the cases of \((\ast \& \& \& \text{left}), (\sim \& \& \& \& \text{left}), (\sim \& \& \& \text{left})\) and \((\sim \& \& \& \& \& \text{left})\) is the same proof as those of \((\sim \& \& \& \text{left})\) in \( S_{N4} \), respectively. The proof of the cases of \((\sim \& \& \& \text{left}), (\sim \& \& \& \text{left}), (\sim \& \& \& \text{left})\) and \((\sim \& \& \& \text{left})\) is considered to be the same proof as those of \((\sim \& \& \& \text{left})\) in \( S_{N4} \). The proof of the cases of \((\sim \& \& \& \& \text{left})\), \((\sim \& \& \& \& \& \text{left})\) and \((\sim \& \& \& \& \& \text{left})\) is the same as those of \((\& \text{left})\) in \( S_{N4} \).

### 6.4 Case for \( S_C \)

**Proposition 6.3** For a cut-free proof \( P \) of \( S_C \), an inference rule \( I \in \{ (\rightarrow \text{left}^c), (\rightarrow \text{left}^c), (\& \text{left}^1), (\& \text{left}^2), (\sim \& \& \text{left}^1), (\sim \& \& \& \text{left}^2) \} \) appearing in \( P \) can be permuted upward as possible.

**Proof** (Sketch) This proposition can be shown in a similar way as that for \( S_{N4} \). The result of the \( \sim \)-free \( S_C \) with the exchange rule, i.e. non-modal intuitionistic linear logic is known as folklore, and in this result, the role of the exchange rule is not essential in considering permutability. Moreover, the cases for the rules \((\rightarrow \text{left}^c), (\sim \& \& \text{left}^1), (\sim \& \& \& \text{left}^2)\) can also be handled in a similar way as those for \((\rightarrow \text{left}^c), (\& \text{left}^1)\) and \((\& \text{left}^2)\), using the form similarity like the cases for \( S_{N4} \).
Similarly, it is remarked that the rules (\(\land^{\mathsf{right}}\)), (\(\rightarrow^{\mathsf{right}}\)), (\(\leftarrow^{\mathsf{right}}\)) and (\(\neg\land^{\mathsf{right}}\)) can be permuted downward as possible.

It is noted in Proposition 6.3 that the rule (\(\ast^{\mathsf{left}}\)) is not included in \(I\). An example for non-permutable cases for (\(\ast^{\mathsf{left}}\)) is

\[
\vdots
\frac{\alpha \Rightarrow \gamma \quad \beta \Gamma \Rightarrow \delta}{\alpha, \beta, \Gamma \Rightarrow \gamma \ast \delta} \quad (\ast^{\mathsf{right}})
\]

\[
\frac{\alpha \ast \beta, \Gamma \Rightarrow \gamma \ast \delta}{\alpha \ast \beta, \Gamma \Rightarrow \gamma \ast \delta} \quad (\ast^{\mathsf{left}})
\]

where \(\Gamma \neq \emptyset\) and \(\alpha \notin \Gamma\).

On the other hand, the following proposition holds.

**Proposition 6.4** For a cut-free proof \(P\) of \(SC\), an inference rule \(I \in \{\ast^{\mathsf{left}}, \neg\rightarrow^{\mathsf{left}}, \neg\leftarrow^{\mathsf{left}}, \neg\ast^{\mathsf{left}}\}\) appearing in \(P\) can be permuted downward as possible.

Similarly, it is remarked that the rules (\(\ast^{\mathsf{right}}\)), (\(\neg\rightarrow^{\mathsf{right}}\)), (\(\neg\leftarrow^{\mathsf{right}}\)) and (\(\neg\ast^{\mathsf{right}}\)) can be permuted upward as possible.