Nested Sequent Calculi and Theorem Proving for Normal Conditional Logics

Nicola Olivetti\textsuperscript{1}

Gian Luca Pozzato\textsuperscript{2}

\textsuperscript{1}Aix-Marseille Université, CNRS, LSIS UMR 7296, 13397, Marseille, France
\textsuperscript{2}Dipartimento di Informatica, Università degli Studi di Torino, Italy

CILC 2013
Conditional Logics

- Extensions of classical logic by $\Rightarrow$
- Generalization of multi-modal logics
- $A \Rightarrow B$

Semantics

- Possible world semantics
- A conditional $A \Rightarrow B$ is true in a world $x$, if $B$ is true in the set of worlds where $A$ is true and that are most similar/closest/“as normal as” $x$
- Alternative semantics: from the most general selection function semantics to the stronger sphere semantics
Conditional Logics

- Extensions of classical logic by $\Rightarrow$
- Generalization of multi-modal logics
  - $A \Rightarrow B$

Semantics

- Possible world semantics
- A conditional $A \Rightarrow B$ is true in a world $x$, if $B$ is true in the set of worlds where $A$ is true and that are most similar/closest/“as normal as” $x$
- Alternative semantics: from the most general selection function semantics to the stronger sphere semantics
Conditional Logics

- Extensions of classical logic by ⇒
- Generalization of multi-modal logics
- $A \Rightarrow B$

Semantics

- Possible world semantics
- A conditional $A \Rightarrow B$ is true in a world $x$, if $B$ is true in the set of worlds where $A$ is true and that are most similar/closest/“as normal as” $x$
- Alternative semantics: from the most general selection function semantics to the stronger sphere semantics
Conditional Logics

- Extensions of classical logic by $\Rightarrow$
- Generalization of multi-modal logics
- $A \Rightarrow B$

Semantics

- Possible world semantics
  - A conditional $A \Rightarrow B$ is true in a world $x$, if $B$ is true in the set of worlds where $A$ is true and that are most similar/closest/“as normal as” $x$
  - Alternative semantics: from the most general selection function semantics to the stronger sphere semantics
Conditional Logics

- Extensions of classical logic by \( \Rightarrow \)
- Generalization of multi-modal logics
- \( A \Rightarrow B \)

Semantics

- Possible world semantics
- A conditional \( A \Rightarrow B \) is true in a world \( x \), if \( B \) is true in the set of worlds where \( A \) is true and that are most similar/closest/“as normal as” \( x \)
- Alternative semantics: from the most general selection function semantics to the stronger sphere semantics
Conditional Logics

- Extensions of classical logic by $\Rightarrow$
- Generalization of multi-modal logics
- $A \Rightarrow B$

Semantics

- Possible world semantics
- A conditional $A \Rightarrow B$ is true in a world $x$, if $B$ is true in the set of worlds where $A$ is true and that are most similar/closest/“as normal as” $x$
- Alternative semantics: from the most general selection function semantics to the stronger sphere semantics
Conditional Logics

Recent history

- a few applications in artificial intelligence and knowledge representation
  - formalization of hypothetical reasoning “if A were the case then B”
  - reason about prototypical properties [Gin86]
  - model belief change (AGM and Ramsey’s Test...) [Gra98, GGO05]
  - formalization of epistemic change in a multi-agent setting [BS08, Boa04]
  - axiomatic foundation of nonmonotonic reasoning [Bou94, KLM90] “in normal circumstances if A then B”

Proof theory

- CLs do not have however a state of the art comparable with the one of modal logics
  - *external* calculi which make use of labels and relations on them to import the semantics into the syntax [AGR02, OPS07, GGOS09]
  - *internal* calculi which stay within the language, so that a “configuration” (sequent, tableaux node...) can be directly interpreted as a formula of the language [Gen92, dS83, SPH10, PS11]
Conditional Logics

Recent history

- a few applications in artificial intelligence and knowledge representation
  - formalization of hypothetical reasoning “if $A$ were the case then $B$”
  - reason about prototypical properties [Gin86]
  - model belief change (AGM and Ramsey’s Test...) [Gra98, GGO05]
  - formalization of epistemic change in a multi-agent setting [BS08, Boa04]
  - axiomatic foundation of nonmonotonic reasoning [Bou94, KLM90] “in normal circumstances if $A$ then $B$”

Proof theory

- CLs do not have however a state of the art comparable with the one of modal logics
  - *external* calculi which make use of labels and relations on them to import the semantics into the syntax [AGR02, OPS07, GGOS09]
  - *internal* calculi which stay within the language, so that a “configuration” (sequent, tableaux node...) can be directly interpreted as a formula of the language [Gen92, dS83, SPH10, PS11]
Conditional Logics

Recent history

- a few applications in artificial intelligence and knowledge representation
- formalization of hypothetical reasoning “if A were the case then B”
- reason about prototypical properties [Gin86]
- model belief change (AGM and Ramsey’s Test...) [Gra98, GGO05]
- formalization of epistemic change in a multi-agent setting [BS08, Boa04]
- axiomatic foundation of nonmonotonic reasoning [Bou94, KLM90] “in normal circumstances if A then B”

Proof theory

- CLs do not have however a state of the art comparable with the one of modal logics
  - external calculi which make use of labels and relations on them to import the semantics into the syntax [AGR02, OPS07, GGOS09]
  - internal calculi which stay within the language, so that a “configuration” (sequent, tableaux node...) can be directly interpreted as a formula of the language [Gen92, dS83, SPH10, PS11]
Conditional Logics

Recent history

- a few applications in artificial intelligence and knowledge representation
  - formalization of hypothetical reasoning “if $A$ were the case then $B$”
  - reason about prototypical properties [Gin86]
  - model belief change (AGM and Ramsey’s Test…) [Gra98, GGO05]
  - formalization of epistemic change in a multi-agent setting [BS08, Boa04]
  - axiomatic foundation of nonmonotonic reasoning [Bou94, KLM90] “in normal circumstances if $A$ then $B$”

Proof theory

- CLs do not have however a state of the art comparable with the one of modal logics
  - *external* calculi which make use of labels and relations on them to import the semantics into the syntax [AGR02, OPS07, GGOS09]
  - *internal* calculi which stay within the language, so that a “configuration” (sequent, tableaux node…) can be directly interpreted as a formula of the language [Gen92, dS83, SPH10, PS11]
Conditional Logics

Recent history

- A few applications in artificial intelligence and knowledge representation
- Formalization of hypothetical reasoning “if A were the case then B”
- Reason about prototypical properties [Gin86]
- Model belief change (AGM and Ramsey’s Test...) [Gra98, GGO05]
- Formalization of epistemic change in a multi-agent setting [BS08, Boa04]
- Axiomatic foundation of nonmonotonic reasoning [Bou94, KLM90] “in normal circumstances if A then B”

Proof theory

- CLs do not have however a state of the art comparable with the one of modal logics
- External calculi which make use of labels and relations on them to import the semantics into the syntax [AGR02, OPS07, GGOS09]
- Internal calculi which stay within the language, so that a “configuration” (sequent, tableau node...) can be directly interpreted as a formula of the language [Gen92, dS83, SPH10, PS11]
Introduction

Our contribution

- We begin to investigate *nested sequents* calculi for conditional logics
- Nested sequents = natural generalization of ordinary sequents where sequents are allowed to occur within sequents
  - rules operating “inside a formula”, combining subformulas rather than just combining outer occurrences of formulas as in ordinary sequents
- a nested sequent always corresponds to a formula of the language
- We are able to deal with:
  - basic normal conditional logic $\mathsf{CK}$
  - extensions with ID and CEM (and MP)
  - Cumulative logic $\mathsf{C} [\mathsf{KLM90}] = \flat$ fragment of $\mathsf{CK} + \mathsf{CSO} + \mathsf{ID}$
  - In all cases, we get a PSPACE upper bound
Introduction

Our contribution

- We begin to investigate *nested sequents* calculi for conditional logics
- Nested sequents = natural generalization of ordinary sequents where sequents are allowed to occur within sequents
  - rules operating “inside a formula”, combining subformulas rather than just combining outer occurrences of formulas as in ordinary sequents
- a nested sequent always corresponds to a formula of the language
- We are able to deal with:
  - basic normal conditional logic CK
  - extensions with ID and CEM (and MP)
  - Cumulative logic C [KLM90] = flat fragment of CK+CSO+ID
  - In all cases, we get a PSPACE upper bound
Introduction

Our contribution

- We begin to investigate *nested sequents* calculi for conditional logics
- Nested sequents = natural generalization of ordinary sequents where sequents are allowed to occur within sequents
  - rules operating “inside a formula”, combining subformulas rather than just combining outer occurrences of formulas as in ordinary sequents
- a nested sequent always corresponds to a formula of the language
- We are able to deal with:
  - basic normal conditional logic CK
  - extensions with ID and CEM (and MP)
  - Cumulative logic C [KLM90] = flat fragment of CK+CSO+ID
  - In all cases, we get a PSPACE upper bound
Introduction

Our contribution

- We begin to investigate *nested sequents* calculi for conditional logics
- Nested sequents = natural generalization of ordinary sequents where sequents are allowed to occur within sequents
  - rules operating “inside a formula”, combining subformulas rather than just combining outer occurrences of formulas as in ordinary sequents
- a nested sequent always corresponds to a formula of the language
- We are able to deal with:
  - basic normal conditional logic CK
  - extensions with ID and CEM (and MP)
  - Cumulative logic $C [KLM90] = flat$ fragment of CK+CSO+ID
  - In all cases, we get a PSPACE upper bound
Our contribution

- We begin to investigate *nested sequents* calculi for conditional logics
- Nested sequents = natural generalization of ordinary sequents where sequents are allowed to occur within sequents
  - rules operating “inside a formula”, combining subformulas rather than just combining outer occurrences of formulas as in ordinary sequents
- A nested sequent always corresponds to a formula of the language
- We are able to deal with:
  - basic normal conditional logic CK
  - extensions with ID and CEM (and MP)
  - Cumulative logic $\mathbf{C}$ [KLM90] = flat fragment of $\text{CK}+\text{CSO}+\text{ID}$
- In all cases, we get a $\text{PSPACE}$ upper bound
Introduction

Our contribution

- We begin to investigate *nested sequents* calculi for conditional logics
- Nested sequents = natural generalization of ordinary sequents where sequents are allowed to occur within sequents
  - rules operating “inside a formula”, combining subformulas rather than just combining outer occurrences of formulas as in ordinary sequents
- a nested sequent always corresponds to a formula of the language
- We are able to deal with:
  - basic normal conditional logic CK
  - extensions with ID and CEM (and MP)
  - Cumulative logic \( \mathbf{C} \) [KLM90] = flat fragment of CK+CSO+ID
  - In all cases, we get a \( \text{PSPACE} \) upper bound
Language $\mathcal{L}$

**Alphabet**
- set of propositional variables $ATM$
- symbols of $false \perp$ and $true \top$
- set of connectives $\land, \lor, \neg, \rightarrow, \Rightarrow$

**Formulas**
- Generated by the following grammar:

$$A, B ::= P | T | \perp | \neg A | A \land B | A \lor B | A \rightarrow B | A \Rightarrow B$$
Language $\mathcal{L}$

**Alphabet**
- set of propositional variables $ATM$
- symbols of $false \perp$ and $true \top$
- set of connectives $\land, \lor, \neg, \rightarrow, \Rightarrow$

**Formulas**
- Generated by the following grammar:

$$A, B ::= P \mid \top \mid \perp \mid \neg A \mid A \land B \mid A \lor B \mid A \rightarrow B \mid A \Rightarrow B$$
Selection function semantics

Models

- Triple $\mathcal{M} = \langle \mathcal{W}, f, [\ ] \rangle$
  - $\mathcal{W}$ is a non empty set of objects called *worlds*
  - $f$ is the *selection function* $f : \mathcal{W} \times 2^\mathcal{W} \to 2^\mathcal{W}$
  - $[\ ]$ is the *evaluation function*
    - assigns to an atom $P \in \text{ATM}$ the set of worlds where $P$ is true
    - is extended to boolean formulas as usual, whereas for conditional formulas
      $[A \Rightarrow B] = \{ w \in \mathcal{W} | f(w, [A]) \subseteq [B] \}$

Comments

- $f$ defined taking $[A]$ rather than $A$ as an argument
  - $f(w, [A])$ rather than $f(w, A)$
- equivalent to define $f$ on formulas $f(w, A)$ but imposing that if $[A] = [A']$ in the model, then $f(w, A) = f(w, A')$
  - *normality*
Selection function semantics

Models

- Triple $\mathcal{M} = \langle \mathcal{W}, f, [\ ] \rangle$
  - $\mathcal{W}$ is a non empty set of objects called worlds
  - $f$ is the selection function $f : \mathcal{W} \times 2^\mathcal{W} \rightarrow 2^\mathcal{W}$
  - $[\ ]$ is the evaluation function
    - assigns to an atom $P \in ATM$ the set of worlds where $P$ is true
    - is extended to boolean formulas as usual, whereas for conditional formulas
      \[[A \Rightarrow B] = \{w \in \mathcal{W} \mid f(w, [A]) \subseteq [B]\}\]

Comments

- $f$ defined taking $[A]$ rather than $A$ as an argument
  - $f(w, [A])$ rather than $f(w, A)$
- equivalent to define $f$ on formulas $f(w, A)$ but imposing that if $[A] = [A']$ in the model, then $f(w, A) = f(w, A')$
- normality
Selection function semantics

Models

- Triple $\mathcal{M} = \langle \mathcal{W}, f, [\,] \rangle$
  - $\mathcal{W}$ is a non empty set of objects called *worlds*
  - $f$ is the *selection function* $f : \mathcal{W} \times 2^\mathcal{W} \rightarrow 2^\mathcal{W}$
  - $[\,]$ is the *evaluation function*
    - assigns to an atom $P \in \text{ATM}$ the set of worlds where $P$ is true
    - is extended to boolean formulas as usual, whereas for conditional formulas
    - $[A \Rightarrow B] = \{ w \in \mathcal{W} \mid f(w, [A]) \subseteq [B] \}$

Comments

- $f$ defined taking $[A]$ rather than $A$ as an argument
  - $f(w, [A])$ rather than $f(w, A)$
- equivalent to define $f$ on formulas $f(w, A)$ but imposing that if $[A] = [A']$ in the model, then $f(w, A) = f(w, A')$
- normality
Selection function semantics

**Models**

- Triple $\mathcal{M} = \langle \mathcal{W}, f, [\ ] \rangle$
  - $\mathcal{W}$ is a nonempty set of objects called *worlds*
  - $f$ is the *selection function* $f : \mathcal{W} \times 2^\mathcal{W} \rightarrow 2^\mathcal{W}$
  - $[\ ]$ is the *evaluation function*
    - assigns to an atom $P \in \text{ATM}$ the set of worlds where $P$ is true
    - is extended to Boolean formulas as usual, whereas for conditional formulas
      $[A \Rightarrow B] = \{ w \in \mathcal{W} | f(w, [A]) \subseteq [B] \}$

**Comments**

- $f$ defined taking $[A]$ rather than $A$ as an argument
  - $f(w, [A])$ rather than $f(w, A)$
  - equivalent to define $f$ on formulas $f(w, A)$ but imposing that if $[A] = [A']$ in the model, then $f(w, A) = f(w, A')$
  - normality

---

Gian Luca Pozzato

Nested Sequent Calculi and Theorem Proving for Normal Conditional Logics
Selection function semantics

Models

- Triple $\mathcal{M} = \langle \mathcal{W}, f, [\cdot] \rangle$
  - $\mathcal{W}$ is a non empty set of objects called *worlds*
  - $f$ is the *selection function* $f : \mathcal{W} \times 2^\mathcal{W} \rightarrow 2^\mathcal{W}$
  - $[\cdot]$ is the *evaluation function*
    - assigns to an atom $P \in \text{ATM}$ the set of worlds where $P$ is true
    - is extended to boolean formulas as usual, whereas for conditional formulas $[A \Rightarrow B] = \{ w \in \mathcal{W} \mid f(w, [A]) \subseteq [B]\}$

Comments

- $f$ defined taking $[A]$ rather than $A$ as an argument
  - $f(w, [A])$ rather than $f(w, A)$
- equivalent to define $f$ on formulas $f(w, A)$ but imposing that if $[A] = [A']$ in the model, then $f(w, A) = f(w, A')$
Selection function semantics

Models

- Triple $\mathcal{M} = \langle \mathcal{W}, f, [\ ] \rangle$
  - $\mathcal{W}$ is a non empty set of objects called *worlds*
  - $f$ is the *selection function* $f : \mathcal{W} \times 2^\mathcal{W} \rightarrow 2^\mathcal{W}$
  - $[\ ]$ is the *evaluation function*
    - assigns to an atom $P \in \text{ATM}$ the set of worlds where $P$ is true
    - is extended to boolean formulas as usual, whereas for conditional formulas
      $[A \Rightarrow B] = \{ w \in \mathcal{W} | f(w, [A]) \subseteq [B] \}$

Comments

- $f$ defined taking $[A]$ rather than $A$ as an argument
  - $f(w, [A])$ rather than $f(w, A)$
- equivalent to define $f$ on formulas $f(w, A)$ but imposing that if $[A] = [A']$ in the model, then $f(w, A) = f(w, A')$
- normality
Introduction
Conditional Logics
Nested Sequents for Conditional Logics
A calculus for C NESCOND
Conclusions

Selection function semantics

Models

 Triple $\mathcal{M} = \langle \mathcal{W}, f, [\ ] \rangle$

- $\mathcal{W}$ is a non empty set of objects called worlds
- $f$ is the selection function $f : \mathcal{W} \times 2^\mathcal{W} \rightarrow 2^\mathcal{W}$
- $[\ ]$ is the evaluation function
  - assigns to an atom $P \in \text{ATM}$ the set of worlds where $P$ is true
  - is extended to boolean formulas as usual, whereas for conditional formulas
    $[A \Rightarrow B] = \{ w \in \mathcal{W} | f(w, [A]) \subseteq [B] \}$

Comments

- $f$ defined taking $[A]$ rather than $A$ as an argument
- $f(w, [A])$ rather than $f(w, A)$
- equivalent to define $f$ on formulas $f(w, A)$ but imposing that if $[A] = [A']$ in the model, then $f(w, A) = f(w, A')$

Gian Luca Pozzato
Nested Sequent Calculi and Theorem Proving for Normal Conditional Logics
Selection function semantics

Models

- Triple $\mathcal{M} = \langle \mathcal{W}, f, [\ ] \rangle$
  - $\mathcal{W}$ is a non empty set of objects called *worlds*
  - $f$ is the *selection function* $f: \mathcal{W} \times 2^{\mathcal{W}} \rightarrow 2^{\mathcal{W}}$
  - $[\ ]$ is the *evaluation function*
    - assigns to an atom $P \in \text{ATM}$ the set of worlds where $P$ is true
    - is extended to boolean formulas as usual, whereas for conditional formulas
      $[A \Rightarrow B] = \{ w \in \mathcal{W} | f(w, [A]) \subseteq [B] \}$

Comments

- $f$ defined taking $[A]$ rather than $A$ as an argument
  - $f(w, [A])$ rather than $f(w, A)$
- equivalent to define $f$ on formulas $f(w, A)$ but imposing that if $[A] = [A']$ in the model, then $f(w, A) = f(w, A')$
Selection function semantics

Models

- Triple $\mathcal{M} = \langle \mathcal{W}, f, [\ ] \rangle$
  - $\mathcal{W}$ is a non empty set of objects called *worlds*
  - $f$ is the *selection function* $f : \mathcal{W} \times 2^{\mathcal{W}} \rightarrow 2^{\mathcal{W}}$
  - $[\ ]$ is the *evaluation function*
    - assigns to an atom $P \in \text{ATM}$ the set of worlds where $P$ is true
    - is extended to boolean formulas as usual, whereas for conditional formulas
      $[A \Rightarrow B] = \{ w \in \mathcal{W} \mid f(w, [A]) \subseteq [B] \}$

Comments

- $f$ defined taking $[A]$ rather than $A$ as an argument
  - $f(w, [A])$ rather than $f(w, A)$
- equivalent to define $f$ on formulas $f(w, A)$ but imposing that if $[A] = [A']$ in the model, then $f(w, A) = f(w, A')$
  - *normality*
Semantics

Selection function

\[ f(x, [A]) = \{z_1, z_2, z_3\} \]
\[ f(x, [C]) = \{y_1, y_2\} \]
Semantics

Selection function

\[
\begin{align*}
\{y_1, y_2\} & \quad f(x, [A]) = \{z_1, z_2, z_3\} \\
\{y_1\} & \quad f(x, [C]) = \{y_1, y_2\}
\end{align*}
\]
Semantics

Selection function

\[ f(x, [A]) = \{z_1, z_2, z_3\} \]
\[ f(x, [C]) = \{y_1, y_2\} \]
Conditional Logics

Basic System CK

- Semantics characterizing the *basic conditional system* CK [Nut80]
- CK plays the role of K in modal logics

Axiomatization of CK

- any axiomatization of the classical propositional calculus
- $\vdash A$ and $\vdash A \rightarrow B$ implies $\vdash B$ \hspace{1cm} (Modus Ponens)
- $\vdash A \leftrightarrow B$ implies $\vdash (A \Rightarrow C) \leftrightarrow (B \Rightarrow C)$ \hspace{1cm} (RCEA)
- $\vdash (A_1 \land \cdots \land A_n) \rightarrow B$ implies $\vdash (C \Rightarrow A_1 \land \cdots \land C \Rightarrow A_n) \rightarrow (C \Rightarrow B)$ \hspace{1cm} (RCK)
Conditional Logics

Basic System CK

- Semantics characterizing the *basic conditional system* CK [Nut80]
- CK plays the role of K in modal logics

Axiomatization of CK

- any axiomatization of the classical propositional calculus
- \( \vdash A \) and \( \vdash A \rightarrow B \) implies \( \vdash B \) \hfill (Modus Ponens)
- \( \vdash A \leftrightarrow B \) implies \( \vdash (A \Rightarrow C) \leftrightarrow (B \Rightarrow C) \) \hfill (RCEA)
- \( \vdash (A_1 \land \cdots \land A_n) \rightarrow B \) implies \( \vdash (C \Rightarrow A_1 \land \cdots \land C \Rightarrow A_n) \rightarrow (C \Rightarrow B) \) \hfill (RCK)
Conditional Logics

Extensions of CK
- Obtained by assuming further properties on the selection function
- we consider some standard extensions of the basic system CK

Axiomatization and semantic conditions

<table>
<thead>
<tr>
<th>System</th>
<th>Axiom</th>
<th>Model condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>$A \Rightarrow A$</td>
<td>$f(w, [A]) \subseteq [A]$</td>
</tr>
<tr>
<td>CEM</td>
<td>$(A \Rightarrow B) \lor (A \Rightarrow \neg B)$</td>
<td>$</td>
</tr>
<tr>
<td>CSO</td>
<td>$(A \Rightarrow B) \land (B \Rightarrow A) \Rightarrow ((A \Rightarrow C) \rightarrow (B \Rightarrow C))$</td>
<td>$f(w, [A]) \subseteq [B]$ and $f(w, [B]) \subseteq [A]$ implies $f(w, [A]) = f(w, [B])$</td>
</tr>
</tbody>
</table>
Nested Sequents

Nested Sequent $\Gamma$
- Defined inductively as follows:
  - A finite multiset of formulas of $\mathcal{L}$ is a nested sequent
  - If $A$ is a formula and $\Gamma$ is a nested sequent, then $[A : \Gamma]$ is a nested sequent
  - A finite multiset of nested sequents is a nested sequent

Nested Sequents (1)
- Can be displayed as $A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n]$
  - $n, m \geq 0$
  - $A_1, \ldots, A_m, B_1, \ldots, B_n$ are formulas
  - $\Gamma_1, \ldots, \Gamma_n$ are nested sequents
Nested Sequents

Nested Sequent $\Gamma$

- Defined inductively as follows:
  - A finite multiset of formulas of $\mathcal{L}$ is a nested sequent
  - If $A$ is a formula and $\Gamma$ is a nested sequent, then $[A : \Gamma]$ is a nested sequent
  - A finite multiset of nested sequents is a nested sequent

Nested Sequents (1)

- Can be displayed as $A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n],$
  - $n, m \geq 0$
  - $A_1, \ldots, A_m, B_1, \ldots, B_n$ are formulas
  - $\Gamma_1, \ldots, \Gamma_n$ are nested sequents
Nested Sequents

**Nested Sequent \( \Gamma \)**

- Defined inductively as follows:
  - A finite multiset of formulas of \( L \) is a nested sequent
  - If \( A \) is a formula and \( \Gamma \) is a nested sequent, then \([A : \Gamma]\) is a nested sequent
  - A finite multiset of nested sequents is a nested sequent

**Nested Sequents (1)**

- Can be displayed as \( A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n] \),
  - \( n, m \geq 0 \)
  - \( A_1, \ldots, A_m, B_1, \ldots, B_n \) are formulas
  - \( \Gamma_1, \ldots, \Gamma_n \) are nested sequents
Nested Sequents

**Nested Sequent** $\Gamma$

- Defined inductively as follows:
  - A finite multiset of formulas of $L$ is a nested sequent
  - If $A$ is a formula and $\Gamma$ is a nested sequent, then $[A : \Gamma]$ is a nested sequent
  - A finite multiset of nested sequents is a nested sequent

**Nested Sequents (1)**

- Can be displayed as $A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n],$
  - $n, m \geq 0$
  - $A_1, \ldots, A_m, B_1, \ldots, B_n$ are formulas
  - $\Gamma_1, \ldots, \Gamma_n$ are nested sequents
Nested Sequents

Nested Sequent $\Gamma$

- Defined inductively as follows:
  - A finite multiset of formulas of $\mathcal{L}$ is a nested sequent
  - If $A$ is a formula and $\Gamma$ is a nested sequent, then $[A : \Gamma]$ is a nested sequent
  - A finite multiset of nested sequents is a nested sequent

Nested Sequents (1)

- Can be displayed as $A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n]$,
  - $n, m \geq 0$
  - $A_1, \ldots, A_m, B_1, \ldots, B_n$ are formulas
  - $\Gamma_1, \ldots, \Gamma_n$ are nested sequents
Nested Sequents

Nested Sequent $\Gamma$
- Defined inductively as follows:
  - A finite multiset of formulas of $\mathcal{L}$ is a nested sequent
  - If $A$ is a formula and $\Gamma$ is a nested sequent, then $[A : \Gamma]$ is a nested sequent
  - A finite multiset of nested sequents is a nested sequent

Nested Sequents (1)
- Can be displayed as $A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n]$
  - $n, m \geq 0$
  - $A_1, \ldots, A_m, B_1, \ldots, B_n$ are formulas
  - $\Gamma_1, \ldots, \Gamma_n$ are nested sequents
**Nested Sequents**

**Nested Sequent \( \Gamma \)**
- Defined inductively as follows:
  - A finite multiset of formulas of \( \mathcal{L} \) is a nested sequent
  - If \( A \) is a formula and \( \Gamma \) is a nested sequent, then \([A : \Gamma]\) is a nested sequent
  - A finite multiset of nested sequents is a nested sequent

**Nested Sequents (1)**
- Can be displayed as \( A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n] \),
  - \( n, m \geq 0 \)
  - \( A_1, \ldots, A_m, B_1, \ldots, B_n \) are formulas
  - \( \Gamma_1, \ldots, \Gamma_n \) are nested sequents
Nested Sequent Calculi and Theorem Proving for Normal Conditional Logics

Nested Sequent (2)

- Can be directly interpreted as a formula
  - replace “,” by $\lor$ and “:” by $\Rightarrow$
  - interpretation of $\Gamma = A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n]$ inductively defined by
    \[
    \mathcal{F}(\Gamma) = A_1 \lor \ldots \lor A_m \lor (B_1 \Rightarrow \mathcal{F}(\Gamma_1)) \lor \ldots \lor (B_n \Rightarrow \mathcal{F}(\Gamma_n))
    \]
- Example: $A, B, [A : C, [B : E, F]], [A : D]$ denotes $A \lor B \lor (A \Rightarrow C \lor (B \Rightarrow (E \lor F))) \lor (A \Rightarrow D)$
Nested Sequent (2)

- Can be directly interpreted as a formula
  - replace “,” by $\lor$ and “:” by $\Rightarrow$
  - interpretation of $\Gamma = A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n]$ inductively defined by

$$F(\Gamma) = A_1 \lor \ldots \lor A_m \lor (B_1 \Rightarrow F(\Gamma_1)) \lor \ldots \lor (B_n \Rightarrow F(\Gamma_n))$$

- $A \lor B \lor (A \Rightarrow C \lor (B \Rightarrow (E \lor F))) \lor (A \Rightarrow D)$
Nested Sequent (2)

- Can be directly interpreted as a formula
  - replace “,” by ∨ and “:” by ⇒
  - interpretation of \( \Gamma = A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n] \) inductively defined by
    \[
    \mathcal{F}(\Gamma) = A_1 \lor \ldots \lor A_m \lor (B_1 \Rightarrow \mathcal{F}(\Gamma_1)) \lor \ldots \lor (B_n \Rightarrow \mathcal{F}(\Gamma_n))
    \]
  - Example: \( A, B, [A : C, [B : E, F]], [A : D] \) denotes
    \[
    A \lor B \lor (A \Rightarrow C \lor (B \Rightarrow (E \lor F))) \lor (A \Rightarrow D)
    \]
Nested Sequent (2)

- Can be directly interpreted as a formula
  - replace "," by $\lor$ and ":" by $\Rightarrow$
  - interpretation of $\Gamma = A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n]$ inductively defined by
    $$\mathcal{F}(\Gamma) = A_1 \lor \ldots \lor A_m \lor (B_1 \Rightarrow \mathcal{F}(\Gamma_1)) \lor \ldots \lor (B_n \Rightarrow \mathcal{F}(\Gamma_n))$$

  $$A \lor B \lor (A \Rightarrow C \lor (B \Rightarrow (E \lor F))) \lor (A \Rightarrow D)$$
Nested Sequents for Conditional Logics

Contexts

- A context: a unique empty position, within a sequent that can be filled by a formula/sequent
- ( ) denotes the empty context
- Defined inductively as follows:
  - If $\Delta$ is a nested sequent, $\Gamma(\ ) = \Delta$, ( ) is a context
  - If $\Delta$ is a nested sequent and $\Sigma(\ )$ is a context, $\Gamma(\ ) = \Delta, [A : \Sigma(\ )]$ is a context

Filling a context by a sequent

- Context $\Gamma(\ )$ and sequent $\Lambda$
- Sequent $\Gamma(\Lambda)$ obtained by filling the context by $\Lambda$:
  - If $\Gamma(\ ) = \Delta, (\ )$, then $\Gamma(\Lambda) = \Delta, \Lambda$
  - If $\Gamma(\ ) = \Delta, [A : \Sigma(\ )]$, then $\Gamma(\Lambda) = \Delta, [A : \Sigma(\Lambda)]$
Nested Sequent Calculi and Theorem Proving for Normal Conditional Logics

**Nested Sequents for Conditional Logics**

**Contexts**

- A *context*: a *unique* empty position, within a sequent that can be filled by a formula/sequent
- ( ) denotes the empty context
- Defined inductively as follows:
  - If ∆ is a nested sequent, Γ( ) = ∆, ( ) is a context
  - If ∆ is a nested sequent and Σ( ) is a context, Γ( ) = ∆, [A : Σ( )] is a context

**Filling a context by a sequent**

- Context Γ( ) and sequent Λ
- Sequent Γ(Λ) obtained by filling the context by Λ:
  - If Γ( ) = ∆, ( ), then Γ(Λ) = ∆, Λ
  - If Γ( ) = ∆, [A : Σ( )], then Γ(Λ) = ∆, [A : Σ(Λ)﹍]
Nested Sequents for Conditional Logics

Contexts

- A context: a unique empty position, within a sequent that can be filled by a formula/sequent
- ( ) denotes the empty context
- Defined inductively as follows:
  - If \( \Delta \) is a nested sequent, \( \Gamma(\ ) = \Delta, (\ ) \) is a context
  - If \( \Delta \) is a nested sequent and \( \Sigma(\ ) \) is a context, \( \Gamma(\ ) = \Delta, [A : \Sigma(\ )] \) is a context

Filling a context by a sequent

- Context \( \Gamma(\ ) \) and sequent \( \Lambda \)
- Sequent \( \Gamma(\Lambda) \) obtained by filling the context by \( \Lambda \):
  - If \( \Gamma(\ ) = \Delta, (\ ) \), then \( \Gamma(\Lambda) = \Delta, \Lambda \)
  - If \( \Gamma(\ ) = \Delta, [A : \Sigma(\ )] \), then \( \Gamma(\Lambda) = \Delta, [A : \Sigma(\Lambda)] \)
Nested Sequents for Conditional Logics

Contexts

A context: a unique empty position, within a sequent that can be filled by a formula/sequent

( ) denotes the empty context

Defined inductively as follows:

- If $\Delta$ is a nested sequent, $\Gamma(\ ) = \Delta$, ( ) is a context
- If $\Delta$ is a nested sequent and $\Sigma(\ )$ is a context, $\Gamma(\ ) = \Delta$, $[A : \Sigma(\ )]$ is a context

Filling a context by a sequent

- Context $\Gamma(\ )$ and sequent $\Lambda$
- Sequent $\Gamma(\Lambda)$ obtained by filling the context by $\Lambda$:
  - If $\Gamma(\ ) = \Delta$, ( ), then $\Gamma(\Lambda) = \Delta, \Lambda$
  - If $\Gamma(\ ) = \Delta$, $[A : \Sigma(\ )]$, then $\Gamma(\Lambda) = \Delta, [A : \Sigma(\Lambda)]$
Nested Sequents for Conditional Logics

**Contexts**

- A *context*: a *unique* empty position, within a sequent that can be filled by a formula/sequent
- ( ) denotes the empty context
- Defined inductively as follows:
  - If $\Delta$ is a nested sequent, $\Gamma(\ ) = \Delta$, ( ) is a context
  - If $\Delta$ is a nested sequent and $\Sigma(\ )$ is a context, $\Gamma(\ ) = \Delta, [A : \Sigma(\ )]$ is a context

**Filling a context by a sequent**

- Context $\Gamma(\ )$ and sequent $\Lambda$
- Sequent $\Gamma(\Lambda)$ obtained by filling the context by $\Lambda$:
  - If $\Gamma(\ ) = \Delta, (\ )$, then $\Gamma(\Lambda) = \Delta, \Lambda$
  - If $\Gamma(\ ) = \Delta, [A : \Sigma(\ )]$, then $\Gamma(\Lambda) = \Delta, [A : \Sigma(\Lambda)]$
Nested Sequents for Conditional Logics

**Contexts**

- A *context*: a *unique* empty position, within a sequent that can be filled by a formula/sequent
- ( ) denotes the empty context
- Defined inductively as follows:
  - If $\Delta$ is a nested sequent, $\Gamma(\ ) = \Delta$, ( ) is a context
  - If $\Delta$ is a nested sequent and $\Sigma(\ )$ is a context, $\Gamma(\ ) = \Delta, [A : \Sigma(\ )]$ is a context

**Filling a context by a sequent**

- Context $\Gamma(\ )$ and sequent $\Lambda$
- Sequent $\Gamma(\Lambda)$ obtained by filling the context by $\Lambda$:
  - If $\Gamma(\ ) = \Delta, (\ )$, then $\Gamma(\Lambda) = \Delta, \Lambda$
  - If $\Gamma(\ ) = \Delta, [A : \Sigma(\ )]$, then $\Gamma(\Lambda) = \Delta, [A : \Sigma(\Lambda)]$
Nested Sequent Calculi $\mathcal{N}S$ (1)

**Rules of $\mathcal{N}S$**

- $\Gamma(P, \neg P) \quad (AX) \quad \Gamma(\top) \quad (AX\top)$
- $\Gamma\left(\neg \frac{\Gamma(A)}{\Gamma(\neg A)}\right)$
- $\Gamma\left(\frac{\Gamma(A) \quad \Gamma(B)}{\Gamma(A \land B)}\right) \quad (^+)$
- $\Gamma\left(\frac{\Gamma(\neg A) \quad \Gamma(\neg B)}{\Gamma(\neg(A \lor B))}\right) \quad (\lor^+)$
- $\Gamma\left(\frac{\Gamma(\rightarrow A) \quad \Gamma(B)}{\Gamma(\neg(A \rightarrow B))}\right)$
- $\Gamma\left(\frac{\Gamma\left(\rightarrow A, B\right)}{\Gamma(\neg(A \rightarrow B))}\right) \quad (\rightarrow^+)$
- $\Gamma\left(\frac{\Gamma\left(\rightarrow\right)}{\Gamma(\neg(A \rightarrow B))}\right) \quad (\rightarrow^-)$

$P \in ATM$

$\Gamma(\top) \quad (AX\top)$

$\Gamma(\neg \top) \quad (AX\neg \top)$
Nested Sequent Calculi $N^\Sigma$ (1)

Rules of $N^\Sigma$

\[
\frac{\Gamma([A : B])}{\Gamma(A \Rightarrow B)} (\Rightarrow^+)
\]

\[
\frac{\Gamma([A : \Delta, \neg A])}{\Gamma([A : \Delta])} (ID)
\]

\[
\frac{\Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B])}{\Gamma(\neg(A \Rightarrow B), [A' : \Delta])} (\Rightarrow^-)
\]

\[
\frac{\Gamma([A : \Delta, \Sigma], [B : \Sigma])}{\Gamma([A : \Delta, [B : \Sigma]])} (CEM)
\]
Nested Sequent Calculi $\mathcal{NS}$ (2)

**Usual definitions**

- $\Gamma$ is *derivable* in $\mathcal{NS}$ if it admits a *derivation*.
- Derivation of $\Gamma = \text{closed upward tree}$
  - the root is $\Gamma$
  - a leaf is an instance of an Axiom (closure)
  - a non-leaf node is (an instance of) the conclusion of a rule having (an instance of) the premises of the rule as parents.
Nested Sequent Calculi $\mathcal{NS}$ (3)

Example in CK+ID

\[(\top \Rightarrow Q) \rightarrow ((P \Rightarrow P) \Rightarrow Q)\]
Nested Sequent Calculi $\mathcal{NS}$ (3)

Example in CK+ID

$\neg (\top \Rightarrow Q), (P \Rightarrow P) \Rightarrow Q$

$(\top \Rightarrow Q) \rightarrow ((P \Rightarrow P) \Rightarrow Q) \quad (\rightarrow ^{+})$
Nested Sequent Calculi $\mathcal{NS}$ (3)

Example in CK+ID

\[
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q] \\
\neg (\top \Rightarrow Q), (P \Rightarrow P) \Rightarrow Q \\
(\top \Rightarrow Q) \rightarrow ((P \Rightarrow P) \Rightarrow Q)
\]
Nested Sequent Calculi $\mathcal{NS}$ (3)

Example in CK+ID

\[
\begin{align*}
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q] & \quad (\Rightarrow^{-}) \\
\neg (\top \Rightarrow Q), (P \Rightarrow P) \Rightarrow Q & \quad (\Rightarrow^{+}) \\
(\top \Rightarrow Q) \rightarrow ((P \Rightarrow P) \Rightarrow Q) & \quad (\rightarrow^{+})
\end{align*}
\]
Nested Sequent Calculi $\mathcal{N}S$ (3)

Example in CK+ID

\[
\begin{align*}
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q, \neg Q] & \\
\neg (\top \Rightarrow Q), [P \Rightarrow P] & \\
\neg (\top \Rightarrow Q), (P \Rightarrow P) \Rightarrow Q & \\
(\top \Rightarrow Q) \rightarrow ((P \Rightarrow P) \Rightarrow Q) & \\
\end{align*}
\]
Nested Sequent Calculi $\mathcal{NS}$ (3)

Example in CK+ID

\[
\begin{align*}
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q, \neg Q] & \quad \top, \neg (P \Rightarrow P) \\
\quad \frac{\neg (\top \Rightarrow Q), [P \Rightarrow P : Q]}{\neg (\top \Rightarrow Q), (P \Rightarrow P) \Rightarrow Q} & \quad (\Rightarrow^+) \\
\frac{\neg (\top \Rightarrow Q), (P \Rightarrow P) \Rightarrow Q}{(\top \Rightarrow Q) \rightarrow ((P \Rightarrow P) \Rightarrow Q)} & \quad (\rightarrow^+) \\
\end{align*}
\]
Nested Sequent Calculi $\mathcal{NS}$ (3)

Example in CK+ID

\[
\begin{align*}
\neg(\top \Rightarrow Q), [P \Rightarrow P : Q, \neg Q] & \quad \top, \neg(P \Rightarrow P) & \quad \neg\top, P \Rightarrow P \quad (\Rightarrow^-) \\
\neg(\top \Rightarrow Q), [P \Rightarrow P : Q] & \quad \neg(\top \Rightarrow Q), (P \Rightarrow P) \Rightarrow Q \quad (\Rightarrow^+) \\
(\top \Rightarrow Q) \rightarrow ((P \Rightarrow P) \Rightarrow Q) & \quad (\rightarrow^+) 
\end{align*}
\]
Nested Sequent Calculi $\mathcal{NS}$ (3)

Example in CK+ID

\[\neg (\top \Rightarrow Q), [P \Rightarrow P : Q, \neg Q] \quad \top, \neg (P \Rightarrow P) \quad \neg \top, P \Rightarrow P \]

\[\vdash (\Rightarrow^{-})\]

\[\neg (\top \Rightarrow Q), [P \Rightarrow P : Q] \quad \vdash (\Rightarrow^{+})\]

\[\vdash (\top \Rightarrow Q), (P \Rightarrow P) \Rightarrow Q \quad (\rightarrow^{+})\]

\[\vdash (\top \Rightarrow Q) \rightarrow ((P \Rightarrow P) \Rightarrow Q) \quad (\rightarrow^{+})\]
Nested Sequent Calculi $\mathcal{NS}$ (3)

Example in CK+ID

\[
\begin{align*}
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q, \neg Q] & \quad \text{(AX)} \\
\top, \neg (P \Rightarrow P) & \\
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q] & \quad \text{($\Rightarrow^+$)} \\
\neg (\top \Rightarrow Q), (P \Rightarrow P) \Rightarrow Q & \\
(\top \Rightarrow Q) \Rightarrow ((P \Rightarrow P) \Rightarrow Q) & \quad \text{($\rightarrow^+$)}
\end{align*}
\]
Nested Sequent Calculi $\mathcal{NS} (3)$

**Example in CK+ID**

\[
\begin{align*}
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q, \neg Q] & \quad (AX) \\
\top, \neg (P \Rightarrow P) & \quad (AX_{\top}) \\
\neg \top, P \Rightarrow P & \quad (\Rightarrow^-) \\
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q] & \quad (\Rightarrow^+) \\
\neg (\top \Rightarrow Q), (P \Rightarrow P) \Rightarrow Q & \quad (\rightarrow^+) \\
(\top \Rightarrow Q) & \rightarrow ((P \Rightarrow P) \Rightarrow Q) & (\rightarrow^+)
\end{align*}
\]
Nested Sequent Calculi $\mathcal{NS}$ (3)

Example in CK+ID

\[
\begin{align*}
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q, \neg Q] & \quad (AX) \\
\top, \neg (P \Rightarrow P) & \quad (AX_\top) \\
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q] & \quad (\Rightarrow^+) \\

\neg (\top \Rightarrow Q), (P \Rightarrow P) \Rightarrow Q & \quad (\Rightarrow^+) \\

(\top \Rightarrow Q) \Rightarrow ((P \Rightarrow P) \Rightarrow Q) & \quad (\Rightarrow^+) \\
\end{align*}
\]
Nested Sequent Calculi $\mathcal{NS}$ (3)

Example in CK+ID

\[
\begin{align*}
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q, \neg Q] & \quad (AX) \\
\top, \neg (P \Rightarrow P) & \quad (AX_\top) \\
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q] & \quad (\Rightarrow^+) \\
\neg (\top \Rightarrow Q), (P \Rightarrow P) & \Rightarrow Q \\
\top \Rightarrow Q \Rightarrow ((P \Rightarrow P) \Rightarrow Q) & \quad (\Rightarrow^+) \\
\end{align*}
\]
Nested Sequent Calculi $\mathcal{NS}$ (3)

Example in CK+ID

\[
\begin{array}{c}
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q, \neg Q] \\
\hline
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q, \neg Q] & (AX) & \neg (\top \Rightarrow Q), [P \Rightarrow P : Q, \neg Q] & \hline
\top, \neg (P \Rightarrow P) & (AX_T) & \top, \neg (P \Rightarrow P) & \hline
\neg (\top \Rightarrow Q), [P \Rightarrow P : Q] & (\Rightarrow^+) & \neg (\top \Rightarrow Q), [P \Rightarrow P : Q] & (\Rightarrow^+) \\
\hline
\neg (\top \Rightarrow Q), (P \Rightarrow P) \Rightarrow Q & (\Rightarrow^-) & \neg (\top \Rightarrow Q), (P \Rightarrow P) \Rightarrow Q & (\Rightarrow^-) \\
\hline
\top \Rightarrow Q) \rightarrow ((P \Rightarrow P) \Rightarrow Q) & (\rightarrow^+) & \top \Rightarrow Q) \rightarrow ((P \Rightarrow P) \Rightarrow Q) & (\rightarrow^+) \\
\end{array}
\]

$\neg \top, [P : P, \neg P]$ (ID)

$\neg \top, [P : P]$ ($\Rightarrow^+$)

$\neg \top, P \Rightarrow P$ ($\Rightarrow^-$)

$\neg \top, [P : P]$ ($\Rightarrow^+$)

$\neg \top, [P : P]$ ($\Rightarrow^+$)
Nested Sequent Calculi $\mathcal{NS}$ (3)

Example in CK+ID

\[
\frac{\neg (T \Rightarrow Q), [P \Rightarrow P : Q, \neg Q]}{(AX)} \quad \frac{T, \neg (P \Rightarrow P)}{(AX_{\top})}
\]

\[
\frac{\neg (T \Rightarrow Q), [P \Rightarrow P : Q]}{(\Rightarrow^+)}
\]

\[
\frac{\neg (T \Rightarrow Q), (P \Rightarrow P) \Rightarrow Q}{(\neg^-)}
\]

\[
\frac{T \Rightarrow Q) \to ((P \Rightarrow P) \Rightarrow Q)}{()\Rightarrow^+}
\]

\[
\frac{\neg \top, [P : P, \neg P]}{(AX)} \quad \frac{\neg \top, [P : P]}{(ID)\Rightarrow'^+)}
\]

\[
\frac{\neg \top, P \Rightarrow P}{(\Rightarrow^-)}
\]
Nested Sequent Calculi $\mathcal{NS}$ (4)

Soundness and completeness

- Nested sequent calculi $\mathcal{NS}$ are sound and complete for the respective conditional logics
- A formula $F \in \mathcal{L}$ is valid if and only if the sequent $F$ is derivable

Completeness and cut

- Completeness is an easy consequence of the admissibility of the cut rule:

\[ \frac{\Gamma(F) \quad \Gamma(\neg F)}{\Gamma(\emptyset)} \quad (cut) \]
Admissibility of cut

- The standard proof of admissibility of cut proceeds by a double induction over the complexity of $F$ and the sum of the heights of the derivations of the two premises of $(cut)$
  - we replace one cut by one or several cuts on formulas of smaller complexity, or on sequents derived by shorter derivations
- However, in $\mathcal{NS}$ the standard proof does not work in the following case:

$$
\begin{align*}
(1) & \quad \Gamma([A : B], [A' : \Delta]) & \quad (\Rightarrow^+) \\
(2) & \quad \Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B]) & \quad A, \neg A' & \quad A', \neg A \\
(3) & \quad \Gamma(A \Rightarrow B, [A' : \Delta]) & \quad \Gamma(\neg(A \Rightarrow B), [A' : \Delta]) \\
& \quad \Gamma([A' : \Delta]) & \quad (cut)
\end{align*}
$$
Admissibility of cut

- The standard proof of admissibility of cut proceeds by a double induction over the complexity of $F$ and the sum of the heights of the derivations of the two premises of (cut).

  - we replace one cut by one or several cuts on formulas of smaller complexity, or on sequents derived by shorter derivations.

- However, in $\mathcal{NS}$ the standard proof does not work in the following case:

\[
\begin{align*}
(1) \quad & \Gamma([A : B], [A' : \Delta]) \\
& \quad \Rightarrow^+ \\
(2) \quad & \Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B]) \quad A, \neg A' \quad A', \neg A \\
& \quad \Rightarrow^- \\
(3) \quad & \Gamma(\neg(A \Rightarrow B), [A' : \Delta]) \\
& \quad \Rightarrow_{\Delta} \\
\end{align*}
\]

\[
\Gamma([A' : \Delta]) 
\]
Nested Sequent Calculi \( \mathcal{NS} (6) \)

Completions via cut elimination

- We “split” the admissibility of cut in two (mutually dependent) statements:
  - If \( \Gamma(F) \) and \( \Gamma(\neg F) \) are derivable, so is \( \Gamma(\emptyset) \)
  - If \( \Gamma([A : \Delta]), A, \neg A' \) and \( A', \neg A \) are derivable, so is \( \Gamma([A' : \Delta]) \)

- In the critical case:

\[
\begin{align*}
(1) \quad & \Gamma([A : B], [A' : \Delta]) \\
\frac{\quad \Gamma(A \Rightarrow B, [A' : \Delta])}{(\Rightarrow^+)}
\end{align*}
\]

\[
\begin{align*}
(2) \quad & \Gamma(\neg (A \Rightarrow B), [A' : \Delta, \neg B]) \\
\quad & A, \neg A' \\
\frac{\quad A', \neg A}{(\Rightarrow^-)}
\end{align*}
\]

\[
\begin{align*}
(3) \quad & \Gamma(\neg (A \Rightarrow B), [A' : \Delta]) \\
\frac{\quad \Gamma([A' : \Delta])}{(cut)}
\end{align*}
\]

\[
\begin{align*}
(1') \quad & \Gamma([A' : \Delta, B], [A' : \Delta]) \\
\frac{\quad \Gamma([A' : \Delta, B, [A' : \Delta])}{(cut)}
\end{align*}
\]

\[
\begin{align*}
(2') \quad & \Gamma(\neg (A \Rightarrow B), [A' : \Delta, \neg B]) \\
\quad & (3') \quad \Gamma(A \Rightarrow B, [A' : \Delta, \neg B])
\end{align*}
\]

\[
\begin{align*}
\frac{\quad \Gamma([A' : \Delta, \neg B])}{(cut)}
\end{align*}
\]

\[
\begin{align*}
\frac{\quad \Gamma([A' : \Delta, \neg B])}{(cut)}
\end{align*}
\]

- and we conclude since contraction is admissible

Gian Luca Pozzato

Nested Sequent Calculi and Theorem Proving for Normal Conditional Logics
Nested Sequent Calculi $\mathcal{NS}$ (6)

Completeness via cut elimination

We “split” the admissibility of cut in two (mutually dependent) statements:

- If $\Gamma(F)$ and $\Gamma(\neg F)$ are derivable, so is $\Gamma(\emptyset)$
- if $\Gamma([A : \Delta]), A, \neg A'$ and $A', \neg A$ are derivable, so is $\Gamma([A' : \Delta])$

In the critical case:

1. $\Gamma([A : B], [A' : \Delta])$  \hspace{1cm}  (2) $\Gamma(-(A \Rightarrow B), [A' : \Delta, \neg B])$  \hspace{1cm} $A, \neg A', A', \neg A$
2. $\Gamma(-((A \Rightarrow B), [A' : \Delta]))$  \hspace{1cm} $\Gamma(-(A \Rightarrow B), [A' : \Delta])$  \hspace{1cm} (⇒)  \hspace{1cm} (⇒−)  \hspace{1cm} (cut)

\[ \Gamma([A' : \Delta]) \]

\[ \Gamma([-((A \Rightarrow B), [A' : \Delta])]) \]

(1') $\Gamma([A' : \Delta, B], [A' : \Delta])$  \hspace{1cm} $\Gamma([A' : \Delta, \neg B])$  \hspace{1cm} (cut)
2. $\Gamma(-(A \Rightarrow B), [A' : \Delta, \neg B])$  \hspace{1cm} (3') $\Gamma(A \Rightarrow B, [A' : \Delta, \neg B])$

\[ \Gamma([A' : \Delta], [A' : \Delta]) \]

and we conclude since contraction is admissible.
Nested Sequent Calculi $\mathcal{NS}$ (6)

Completeness via cut elimination

- We “split” the admissibility of cut in two (mutually dependent) statements:
  - If $\Gamma(F)$ and $\Gamma(\neg F)$ are derivable, so is $\Gamma(\emptyset)$
  - If $\Gamma([A \colon \Delta])$, $A$, $\neg A'$ and $A'$, $\neg A$ are derivable, so is $\Gamma([A' \colon \Delta])$
- In the critical case:

\[
\begin{align*}
(1) & \quad \Gamma([A \colon B], [A' \colon \Delta]) \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad
Nested Sequent Calculi $\mathcal{NS}$ (6)

Completeness via cut elimination

- We “split” the admissibility of cut in two (mutually dependent) statements:
  - If $\Gamma(F)$ and $\Gamma(\neg F)$ are derivable, so is $\Gamma(\emptyset)$
  - If $\Gamma([A : \Delta]), A, \neg A'$ and $A', \neg A$ are derivable, so is $\Gamma([A' : \Delta])$

- In the critical case:
  
  $$(1) \quad \Gamma([A : B], [A' : \Delta]) \quad (\Rightarrow^+)$$
  
  $$(2) \quad \Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B]) \quad A, \neg A' \quad A', \neg A \quad (\Rightarrow^-)$$
  
  $$(3) \quad \Gamma(A \Rightarrow B, [A' : \Delta]) \quad \Gamma(\neg(A \Rightarrow B), [A' : \Delta]) \quad (\text{cut})$$

  $\Gamma([A' : \Delta])$

  $$(2') \quad \Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B])$$

  $$(3') \quad \Gamma(A \Rightarrow B, [A' : \Delta, \neg B]) \quad (\text{cut})$$

  $$(1') \quad \Gamma([A' : \Delta, B], [A' : \Delta])$$

  $\Gamma([A' : \Delta, \neg B]) \quad (\text{cut})$

  $\Gamma([A' : \Delta], [A' : \Delta])$
Nested Sequent Calculi $\mathcal{NS}$ (6)

Completeness via cut elimination

- We “split” the admissibility of cut in two (mutually dependent) statements:
  - If $\Gamma(F)$ and $\Gamma(\neg F)$ are derivable, so is $\Gamma(\emptyset)$
  - if $\Gamma([A : \Delta])$, $A, \neg A'$ and $A', \neg A$ are derivable, so is $\Gamma([A' : \Delta])$

- In the critical case:

  $$(1) \quad \Gamma([A : B], [A' : \Delta]) \quad \quad (2) \quad \Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B]) \quad \quad \Gamma(\neg(A \Rightarrow B), [A' : \Delta])$$

  $$(\Rightarrow^+) \quad \quad \quad \quad \quad \quad \quad \quad \quad (\Rightarrow^-) \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad (cut)$$

  $$(3) \quad \Gamma(A \Rightarrow B, [A' : \Delta]) \quad \quad \quad \quad (cut)$$

$$(1') \quad \Gamma([A' : \Delta, B], [A' : \Delta]) \quad \quad \quad \quad \quad \Gamma([A' : \Delta, \neg B])$$

$$(3') \quad \Gamma(A \Rightarrow B, [A' : \Delta, \neg B])$$

$$(cut)$$

and we conclude since contraction is admissible.
Termination and Complexity

Terminating calculi

- Some restrictions to ensure termination:
  - $(\Rightarrow \neg)$ is applied only once to each formula $\neg (A \Rightarrow B)$ with a context $[A' : \Delta]$ in each branch.
  - $(ID)$ is applied only once to each context $[A : \Delta]$ in each branch.
  - For the systems allowing CEM, we need a more sophisticated machinery (not enough time to introduce it now...).

- The calculi $\mathcal{NS}$ with the termination restrictions is sound and complete for their respective logics.

Complexity

- The above restrictions ensure a terminating proof search.
- The calculi $\mathcal{NS}$ with the termination restrictions give a $\text{PSPACE}$ decision procedure for their respective logics.
Termination and Complexity

Terminating calculi

- Some restrictions to ensure termination:
  - $(\Rightarrow \neg)$ is applied only once to each formula $\neg(A \Rightarrow B)$ with a context $[A' : \Delta]$ in each branch
  - $(ID)$ is applied only once to each context $[A : \Delta]$ in each branch
  - For the systems allowing CEM, we need a more sophisticated machinery (not enough time to introduce it now...)
  - The calculi $\mathcal{N}S$ with the termination restrictions is sound and complete for their respective logics

Complexity

- The above restrictions ensure a terminating proof search
- The calculi $\mathcal{N}S$ with the termination restrictions give a $\mathbf{PSPACE}$ decision procedure for their respective logics
Termination and Complexity

Terminating calculi

- Some restrictions to ensure termination:
  - $(\Rightarrow \neg)$ is applied only once to each formula $\neg(A \Rightarrow B)$ with a context $[A' : \Delta]$ in each branch
  - $(ID)$ is applied only once to each context $[A : \Delta]$ in each branch
  - For the systems allowing CEM, we need a more sophisticated machinery (not enough time to introduce it now...)

- The calculi $\mathcal{NS}$ with the termination restrictions is sound and complete for their respective logics

Complexity

- The above restrictions ensure a terminating proof search
- The calculi $\mathcal{NS}$ with the termination restrictions give a $\text{PSPACE}$ decision procedure for their respective logics
Termination and Complexity

Terminating calculi

- Some restrictions to ensure termination:
  - $(\Rightarrow \neg)$ is applied only once to each formula $\neg (A \Rightarrow B)$ with a context $[A' : \Delta]$ in each branch
  - $(ID)$ is applied only once to each context $[A : \Delta]$ in each branch
  - For the systems allowing CEM, we need a more sophisticated machinery (not enough time to introduce it now...)

- The calculi $\mathcal{N}S$ with the termination restrictions is sound and complete for their respective logics

Complexity

- The above restrictions ensure a terminating proof search
- The calculi $\mathcal{N}S$ with the termination restrictions give a $\text{PSPACE}$ decision procedure for their respective logics
Termination and Complexity

Terminating calculi

- Some restrictions to ensure termination:
  - $\rightarrow\neg$ is applied only once to each formula $\neg(A \Rightarrow B)$ with a context $[A' : \Delta]$ in each branch
  - $ID$ is applied only once to each context $[A : \Delta]$ in each branch
  - For the systems allowing CEM, we need a more sophisticated machinery (not enough time to introduce it now...)
- The calculi $\mathcal{N}\mathcal{S}$ with the termination restrictions is sound and complete for their respective logics

Complexity

- The above restrictions ensure a terminating proof search
- The calculi $\mathcal{N}\mathcal{S}$ with the termination restrictions give a PSPACE decision procedure for their respective logics
Termination and Complexity

**Terminating calculi**

- Some restrictions to ensure termination:
  - $(\Rightarrow \neg)$ is applied only once to each formula $\neg (A \Rightarrow B)$ with a context $[A' : \Delta]$ in each branch
  - $(ID)$ is applied only once to each context $[A : \Delta]$ in each branch
  - For the systems allowing CEM, we need a more sophisticated machinery (not enough time to introduce it now...)

- The calculi $\mathcal{NS}$ with the termination restrictions is sound and complete for their respective logics

**Complexity**

- The above restrictions ensure a terminating proof search
- The calculi $\mathcal{NS}$ with the termination restrictions give a $\text{PSPACE}$ decision procedure for their respective logics
Cumulative logic **C**

- **Cumulative Logic C** is the weakest system in the family of KLM logics [KLM90]
- C is the flat fragment (i.e. without nested ⇒) of CK+CSO+ID
- Formulas are boolean combinations of propositional formulas and conditionals A ⇒ B where A and B are propositionals
- No “simple” and internal proof system is known for this logic ([SPH10], [GGOP09])
Nested Sequents for C (1)

Cumulative logic C

- *Cumulative Logic C* is the weakest system in the family of KLM logics [KLM90]
- C is the flat fragment (i.e. without nested ⇒) of CK+CSO+ID
- Formulas are boolean combinations of propositional formulas and conditionals $A \Rightarrow B$ where $A$ and $B$ are propositionals
- No “simple” and internal proof system is known for this logic ([SPH10], [GGOP09])
Nested Sequents for C (1)

Cumulative logic C

- *Cumulative Logic C* is the weakest system in the family of KLM logics [KLM90]
- C is the flat fragment (i.e. without nested \( \Rightarrow \)) of CK+CSO+ID
- Formulas are boolean combinations of propositional formulas and conditionals \( A \Rightarrow B \) where A and B are propositionals
- No “simple” and internal proof system is known for this logic ([SPH10], [GGOP09])
Cumulative logic $\mathbf{C}$

- *Cumulative Logic $\mathbf{C}$* is the weakest system in the family of KLM logics [KLM90]
- $\mathbf{C}$ is the flat fragment (i.e. without nested $\Rightarrow$) of CK+CSO+ID
- Formulas are boolean combinations of propositional formulas and conditionals $A \Rightarrow B$ where $A$ and $B$ are propositionals
- No “simple” and internal proof system is known for this logic ([SPH10], [GGOP09])
Nested Sequents for C (1)

The calculus $\mathcal{NC}K + CSO + ID$

- Sequents have the form
  \[ A_1, \ldots, A_m, [B_1 : \Delta_1], \ldots, [B_m : \Delta_m] \]
  where $B_i$ and $\Delta_i$ are propositional
- The rules are those ones of $\mathcal{NC}K + ID$ (restricted to the flat fragment) expect the rule $(\Rightarrow \neg)$
- The rule $(\Rightarrow \neg)$ is replaced by the following rule (CSO):
  \[
  \frac{
  \Gamma, \neg(C \Rightarrow D), [A : \Delta, \neg D] \\
  \Gamma, \neg(C \Rightarrow D), [A : C] \\
  \Gamma, \neg(C \Rightarrow D), [C : A]
  }{
  \Gamma, \neg(C \Rightarrow D), [A : \Delta]
  }
  \]
Nested Sequents for C (1)

The calculus $\mathcal{N}CK + CSO + ID$

- Sequents have the form

$$A_1, \ldots, A_m, [B_1 : \Delta_1], \ldots, [B_m : \Delta_m]$$

where $B_i$ and $\Delta_i$ are propositional

- The rules are those ones of $\mathcal{N}CK + ID$ (restricted to the flat fragment)

  expect the rule ($\Rightarrow -$)

- The rule ($\Rightarrow -$) is replaced by the following rule (CSO):

$$\Gamma, \neg(C \Rightarrow D), [A : \Delta, \neg D] \quad \Gamma, \neg(C \Rightarrow D), [A : C] \quad \Gamma, \neg(C \Rightarrow D), [C : A]$$

$$\Gamma, \neg(C \Rightarrow D), [A : \Delta]$$
Nested Sequents for C (1)

The calculus $N^\mathcal{CK} + CSO + ID$

- Sequents have the form

$$A_1, \ldots, A_m, [B_1 : \Delta_1], \ldots, [B_m : \Delta_m]$$

where $B_i$ and $\Delta_i$ are propositional

- The rules are those ones of $N^\mathcal{CK} + ID$ (restricted to the flat fragment) expect the rule $(\Rightarrow^-)$

- The rule $(\Rightarrow^-)$ is replaced by the following rule (CSO):

$$\Gamma, \neg (C \Rightarrow D), [A : \Delta, \neg D] \quad \Gamma, \neg (C \Rightarrow D), [A : C] \quad \Gamma, \neg (C \Rightarrow D), [C : A]$$

$$\Gamma, \neg (C \Rightarrow D), [A : \Delta]$$
Nested Sequents for C (2)

Example: a derivation of the cumulative axiom

\[( (A \Rightarrow B) \land (A \Rightarrow C)) \rightarrow (A \land B) \Rightarrow C \]

\[
\begin{array}{l}
\Sigma, [A \land B : A, \neg A, \neg B)] \\
\hline
\Sigma, [A \land B : A, \neg (A \land B)] \quad (\land^2) \\
\hline
\neg (A \Rightarrow B), \neg (A \Rightarrow C), [A \land B : C, \neg C] \quad (ID) \\
\hline
\neg (A \Rightarrow B), \neg (A \Rightarrow C), [A \land B : A] \\
\hline
\neg (A \Rightarrow B), \neg (A \Rightarrow C), [A : A \land B] \quad (CSO) \\
\hline
\neg (A \Rightarrow B), \neg (A \Rightarrow C), [A \land B : C] \\
\hline
\neg (A \Rightarrow B), (A \Rightarrow C) \Rightarrow C \quad (\Rightarrow^+) \\
\hline
\end{array}
\]

where \( \Pi_1 \) is the following derivation:

\[
\begin{array}{l}
\Sigma, [A : A, \neg A] \\
\hline
\Sigma, [A : A, \neg A] \quad (ID) \\
\hline
\Sigma, [A : B, \neg B] \\
\hline
\Sigma, [A : A] \quad (ID) \\
\hline
\Sigma, [A : A, \neg A] \\
\hline
\Sigma, [A : A] \quad (CSO) \\
\hline
\Sigma, [A : A, \neg A] \\
\hline
\Sigma, [A : A, \neg A] \quad (ID) \\
\hline
\Sigma, [A : B, \neg B] \\
\hline
\Sigma, [A : A] \\
\hline
\Sigma, [A : A, \neg A] \quad (CSO) \\
\hline
\neg (A \Rightarrow B), \neg (A \Rightarrow C), [A : A \land B] \quad (\land^+) \\
\hline
\end{array}
\]
Results

- The calculus $\mathcal{N}CK + CSO + ID$ is sound and complete for $\mathbf{C}$
- Completeness from admissibility of cut, proved as for $\mathcal{N}S$
- Restriction on (CSO) to ensure termination:
  - (CSO) is applied only once to each formula $\neg (A \Rightarrow B)$ with a context $[A' : \Delta]$ in each branch
- The calculus $\mathcal{N}CK + CSO + ID$ with the termination restrictions give a $\text{PSPACE}$ decision procedure for $\mathbf{C}$
**Theorem prover for Conditional Logics**

- **Prolog implementation of nested sequent calculi**
  - inspired by lean $T^A_P$: each axiom or rule of the nested sequent calculi is implemented by a Prolog clause of the program
    - simple and compact code
    - implementation for CK: 6 predicates, 24 clauses, 34 lines of code
  
- **PSPACE decision procedure for the respective logics**

- available at [http://www.di.unito.it/~thicksim$pozzato/nescond/](http://www.di.unito.it/~thicksim$pozzato/nescond/) (when our server is up...)
Theorem prover for Conditional Logics

- Prolog implementation of nested sequent calculi
- inspired by lean $TAP$: each axiom or rule of the nested sequent calculi is implemented by a Prolog clause of the program
  - simple and compact code
  - implementation for CK: 6 predicates, 24 clauses, 34 lines of code
- PSPACE decision procedure for the respective logics
- available at http://www.di.unito.it/$\thicksim$pozzato/nescond/ (when our server is up...)
Theorem prover for Conditional Logics

- Prolog implementation of nested sequent calculi
- inspired by lean $TAP$: each axiom or rule of the nested sequent calculi is implemented by a Prolog clause of the program
  - simple and compact code
  - implementation for CK: 6 predicates, 24 clauses, 34 lines of code
- $\mathbf{PSPACE}$ decision procedure for the respective logics
- available at http://www.di.unito.it/$\thicksim$pozzato/nescond/ (when our server is up...)
Theorem prover for Conditional Logics

- Prolog implementation of nested sequent calculi
- inspired by lean $T^A_P$: each axiom or rule of the nested sequent calculi is implemented by a Prolog clause of the program
  - simple and compact code
  - implementation for CK: 6 predicates, 24 clauses, 34 lines of code
- $\text{PSPACE}$ decision procedure for the respective logics
- available at http://www.di.unito.it/$\text{\textbackslash thicksim}$pozzato/nescond/ (when our server is up...)
Theorem prover for Conditional Logics

- Prolog implementation of nested sequent calculi
- inspired by lean $TAP$: each axiom or rule of the nested sequent calculi is implemented by a Prolog clause of the program
  - simple and compact code
  - implementation for CK: 6 predicates, 24 clauses, 34 lines of code
- $\mathbb{PSpace}$ decision procedure for the respective logics
- available at http://www.di.unito.it/$\text{\~{}pozzato}/nescond/
  (when our server is up...)
Nested sequent

- **Prolog list**
  
  \[
  [F_1, F_2, \ldots, F_m, [[A_1, \Gamma_1], \text{AppliedConditionals}_1],
  [[A_2, \Gamma_2], \text{AppliedConditionals}_2], \ldots, [[A_n, \Gamma_n], \text{AppliedConditionals}_n]]
  \]

- **List items are either formulas or contexts**

- **Context**: pair \([\text{Context}, \text{AppliedConditionals}]\)
  
  - Context is also a pair \([F, \Gamma]\) (\(F\) formula and \(\Gamma\) is a nested sequent)
  
  - AppliedConditionals is a Prolog list \([A_1 \Rightarrow B_1, A_2 \Rightarrow B_2, \ldots, A_k \Rightarrow B_k]\), keeping track of the negated conditionals to which the rule \((\Rightarrow \neg)\) has been already applied by using Context in the current branch (to implement the restriction on \((\Rightarrow \neg)\) for termination)
Introduction
Conditional Logics
Nested Sequents for Conditional Logics
A calculus for C
NESCOND
Conclusions

NESCOND

Nested sequent

- Prolog list

  \[ [F_1, F_2, \ldots, F_m, [[A_1, Gamma_1], AppliedConditionals_1],
  [[A_2, Gamma_2], AppliedConditionals_2], \ldots, [[A_n, Gamma_n], AppliedConditionals_n] ] \]

- List items are either formulas or contexts

- Context: pair \([\text{Context}, \text{AppliedConditionals}]\)
  - Context is also a pair \([F, Gamma]\) (F formula and Gamma is a nested sequent)
  - AppliedConditionals is a Prolog list \([A_1 \Rightarrow B_1, A_2 \Rightarrow B_2, \ldots, A_k \Rightarrow B_k]\), keeping track of the negated conditionals to which the rule \((\Rightarrow \neg)\) has been already applied by using Context in the current branch (to implement the restriction on \((\Rightarrow \neg)\) for termination)
Nested sequent

- Prolog list

  \[ [F_1, F_2, \ldots, F_m, [[A_1, \Gamma_1], \text{AppliedConditionals}_1],
  [[A_2, \Gamma_2], \text{AppliedConditionals}_2], \ldots, [[A_n, \Gamma_n], \text{AppliedConditionals}_n]] \]

- List items are either formulas or contexts

- Context: pair \([\text{Context}, \text{AppliedConditionals}]\)
  - Context is also a pair \([F, \Gamma]\) (\(F\) formula and \(\Gamma\) is a nested sequent)
  - AppliedConditionals is a Prolog list \([A_1 \Rightarrow B_1, A_2 \Rightarrow B_2, \ldots, A_k \Rightarrow B_k]\),
    keeping track of the negated conditionals to which the rule \((\Rightarrow \neg)\) has been already applied by using Context in the current branch (to implement the restriction on \((\Rightarrow \neg)\) for termination)
### NESCOND

#### Nested sequent

- **Prolog list**

  
  \[
  [F_1, F_2, \ldots, F_m, [[A_1, Gamma_1], AppliedConditionals_1],
  [[A_2, Gamma_2], AppliedConditionals_2], \ldots, [[A_n, Gamma_n], AppliedConditionals_n]]
  \]

- **List items are either formulas or contexts**

- **Context: pair** [Context, AppliedConditionals]
  
  - **Context is also a pair** [F, Gamma] (F formula and Gamma is a nested sequent)
  
  - **AppliedConditionals** is a Prolog list \([A_1 \Rightarrow B_1, A_2 \Rightarrow B_2, \ldots, A_k \Rightarrow B_k]\), keeping track of the negated conditionals to which the rule \((\Rightarrow^-)\) has been already applied by using Context in the current branch (to implement the restriction on \((\Rightarrow^-)\) for termination)
Nested sequent

- Prolog list
  
  \[
  [F_1, F_2, \ldots, F_m, [[A_1, Gamma_1], AppliedConditionals_1],
  [[A_2, Gamma_2], AppliedConditionals_2], \ldots, [[A_n, Gamma_n], AppliedConditionals_n]]
  \]

- List items are either formulas or contexts

- Context: pair \([Context, AppliedConditionals]\)
  
  - Context is also a pair \([F, Gamma]\) (F formula and Gamma is a nested sequent)
  - AppliedConditionals is a Prolog list \([A_1=>B_1, A_2=>B_2, \ldots, A_k=>B_k]\), keeping track of the negated conditionals to which the rule \((\Rightarrow^{-})\) has been already applied by using Context in the current branch (to implement the restriction on \((\Rightarrow^{-})\) for termination)
Auxiliary predicates

- **3 predicates to manipulate formulas “inside” a sequent:**
  - `deepMember(+Formulas,+NS)` succeeds if and only if either (i) `NS` contains all the formulas in `Formulas` or (ii) there exists a context `[[A,Delta],AppliedConditionals]` in `NS` such that `deepMember(Formulas,Delta)` succeeds.
  - `deepSelect(+Formulas,+NS,-NewNS)` (as `deepMember`, but replacing formulas in `NS` with a placeholder hole).
  - `fillTheHole(+NS,+Formulas,-NewNS)` replaces hole in `NS` with the formulas in `Formulas`. 

Gian Luca Pozzato

Nested Sequent Calculi and Theorem Proving for Normal Conditional Logics
Auxiliary predicates

- 3 predicates to manipulate formulas “inside” a sequent:
  - `deepMember(+Formulas,+NS)` succeeds if and only if either (i) NS contains all the formulas in Formulas or (ii) there exists a context `[[A,Delta],AppliedConditionals]` in NS such that `deepMember(Formulas,Delta)` succeeds
  - `deepSelect(+Formulas,+NS,-NewNS)` (as `deepMember`, but replacing formulas in NS with a placeholder hole)
  - `fillTheHole(+NS,+Formulas,-NewNS)` replaces hole in NS with the formulas in Formulas
Auxiliary predicates

- 3 predicates to manipulate formulas “inside” a sequent:
  - `deepMember(+Formulas,+NS)` succeeds if and only if either (i) `NS` contains all the formulas in `Formulas` or (ii) there exists a context `[[A,D],AppliedConditionals]` in `NS` such that `deepMember(Formulas,D)` succeeds
  - `deepSelect(+Formulas,+NS,-NewNS)` (as `deepMember`, but replacing formulas in `NS` with a placeholder `hole`)
  - `fillTheHole(+NS,+Formulas,-NewNS)` replaces `hole` in `NS` with the formulas in `Formulas`
Auxiliary predicates

- 3 predicates to manipulate formulas “inside” a sequent:
  - `deepMember(+Formulas,+NS)` succeeds if and only if either (i) `NS` contains all the formulas in `Formulas` or (ii) there exists a context `[[A,Delta],AppliedConditionals]` in `NS` such that `deepMember(Formulas,Delta)` succeeds
  - `deepSelect(+Formulas,+NS,-NewNS)` (as `deepMember`, but replacing formulas in `NS` with a placeholder `hole`)
  - `fillTheHole(+NS,+Formulas,-NewNS)` replaces `hole` in `NS` with the formulas in `Formulas`
Main predicate

- Calculi \( \mathcal{NS} \) implemented by the predicate

\[
\text{prove}(+\mathcal{NS},-\text{ProofTree}).
\]

- success in case list \( \mathcal{NS} \) is derivable

- if succeeds, the output term \( \text{ProofTree} \) matches with a representation of the derivation found by the prover, used in order to display the proof tree
**Main predicate**

- Calculi $\mathcal{N}S$ implemented by the predicate
  
  $$\text{prove}(+\text{NS},-\text{ProofTree}).$$

- success in case list $\text{NS}$ is derivable

- if succeeds, the output term $\text{ProofTree}$ matches with a representation of the derivation found by the prover, used in order to display the proof tree
NESCOND

Main predicate

- Calculi $\mathcal{N} \mathcal{S}$ implemented by the predicate
  
  $$\text{prove}(+\text{NS},-\text{ProofTree}).$$

- success in case list $\text{NS}$ is derivable

- if succeeds, the output term $\text{ProofTree}$ matches with a representation of the derivation found by the prover, used in order to display the proof tree
Example

- Check the validity of \((A \Rightarrow (B \land C)) \rightarrow (A \Rightarrow B)\)
- Query NESCOND with the goal `prove([\((a \Rightarrow b \land c) \rightarrow (a \Rightarrow b)\)],ProofTree)`.

Clauses for the axioms

- `prove(NS,tree(ax)):-deepMember([P,!P],NS),!.`
- `prove(NS,tree(axt)):-deepMember([top],NS),!.`
- `prove(NS,tree(axb)):-deepMember([!bot],NS),!.`
Example

- Check the validity of \((A \Rightarrow (B \land C)) \rightarrow (A \Rightarrow B)\)
- Query NESCOND with the goal `prove([\((a \Rightarrow b \land c) \rightarrow (a \Rightarrow b)\)],ProofTree)`.

Clauses for the axioms

- `prove(NS,tree(ax)):-deepMember([P,!P],NS),!.`
- `prove(NS,tree(axt)):-deepMember([top],NS),!.`
- `prove(NS,tree(axb)):-deepMember([!bot],NS),!.`
Example

- Check the validity of \((A \Rightarrow (B \land C)) \rightarrow (A \Rightarrow B)\)
- Query NESCOND with the goal \texttt{prove([[(a \Rightarrow b \land c) \rightarrow (a \Rightarrow b)],ProofTree])}.

Clauses for the axioms

- \texttt{prove(NS,tree(ax)):-deepMember([P,!P],NS),!}.
- \texttt{prove(NS,tree(axt)):-deepMember([top],NS),!}.
- \texttt{prove(NS,tree(axb)):-deepMember([!bot],NS),!}.
Clause for the rule \(\Rightarrow \neg\)

- \texttt{prove(NS,tree(condn,A,B,Sub1,Sub2,Sub3)):-}
  - \texttt{deepSelect([!(A \Rightarrow B),[[C,Delta],AppliedConditionals]],NS,NewNS)},
  - \texttt{\texttt{\neg}member(!(A \Rightarrow B),AppliedConditionals),!},
  - \texttt{prove([A,!C],Sub2)},
  - \texttt{prove([C,!A],Sub3)},
  - \texttt{fillTheHole(NewNS,[!(A \Rightarrow B),[[C,![B|Delta]],![A \Rightarrow B]|[AppliedConditionals]],DefNS)},
  - \texttt{prove(DefNS,Sub1)}. 
Clause for the rule (⇒−)

- prove(NS,tree(condn,A,B,Sub1,Sub2,Sub3)):-
  - deepSelect([!(A => B),[[C,Delta],AppliedConditionals]],NS,NewNS),
  - member(!(A => B),AppliedConditionals),!,
  - prove([A,!C],Sub2),
  - prove([C,!A],Sub3),
  - fillTheHole(NewNS,[(A => B), [[C,![B|Delta]],!(A => B)|AppliedConditionals]],DefNS),
  - prove(DefNS,Sub1).
Clause for the rule ($\Rightarrow -$)

- `prove(NS,tree(condn,A,B,Sub1,Sub2,Sub3)):-`
- `deepSelect([!(A => B),[[C,Delta],AppliedConditionals]],NS,NewNS),`
- `\+member(!(A => B),AppliedConditionals),!`,
- `prove([A,!C],Sub2),`
- `prove([C,!A],Sub3),`
- `fillTheHole(NewNS,[(A => B), [[C,=!B|Delta]],[!(A => B)|AppliedConditionals]],DefNS),`
- `prove(DefNS,Sub1).`
Clause for the rule ($\Rightarrow^\_\_\_\_\_\_\_\_\_$)

- `prove(NS, tree(condn, A, B, Sub1, Sub2, Sub3)):-`
- `deepSelect([\!(A \Rightarrow B), [[C, Delta], AppliedConditionals]], NS, NewNS),`
- `\! member([A \Rightarrow B], AppliedConditionals),!`
- `prove([A, !C], Sub2),`
- `prove([C, !A], Sub3),`
- `fillTheHole(NewNS, ![A \Rightarrow B], [[C, ![B|Delta]], ![A => B]|AppliedConditionals]], DefNS),`
- `prove(DefNS, Sub1).`
Introduction
Conditional Logics
Nested Sequent Calculi for Conditional Logics
A calculus for CESCOND
NESCOND
Conclusions

NESCOND

Clause for the rule (⇒−)

prove(NS,tree(condn,A,B,Sub1,Sub2,Sub3)):-
depthSelect([!(A ⇒ B),[[C,Delta],AppliedConditionals]], NS,NewNS),
\+member(!(A ⇒ B),AppliedConditionals),!,
prove([A,!C],Sub2),
prove([C,!A],Sub3),
fillTheHole(NewNS,![A ⇒ B],[[C,[!B|Delta]],!(A ⇒ B)|AppliedConditionals]],DefNS),
prove(DefNS,Sub1).
Clause for the rule ($\Rightarrow -$)

- prove(NS, tree(condn, A, B, Sub1, Sub2, Sub3)) :-
- deepSelect([! (A $\Rightarrow$ B), [[C, Delta], AppliedConditionals]], NS, NewNS),
- \+member(! (A $\Rightarrow$ B), AppliedConditionals), !,
- prove([A, !C], Sub2),
- prove([C, !A], Sub3),
- fillTheHole(NewNS, ![A $\Rightarrow$ B], [[C, ![B|Delta]], ![A $\Rightarrow$ B]|AppliedConditionals]], DefNS),
- prove(DefNS, Sub1).
The whole procedure

- **each clause of the** *prove* **predicate implements an axiom or rule of** $\mathcal{NS}$
- **To search a derivation of a nested sequent** $\Gamma$, **NESCOND** **proceeds as follows:**
  - first of all, if $\Gamma$ is an axiom, the goal will succeed immediately by using one of the clauses for the axioms
  - if $\Gamma$ is not an instance of the axioms, then the first applicable rule will be chosen, and **NESCOND** will be recursively invoked on its premises. The ordering of the clauses is such that the application of the branching rules is postponed as much as possible.
The whole procedure

- each clause of the `prove` predicate implements an axiom or rule of $\mathcal{NS}$
- To search a derivation of a nested sequent $\Gamma$, NESCOND proceeds as follows:
  - first of all, if $\Gamma$ is an axiom, the goal will succeed immediately by using one of the clauses for the axioms
  - if $\Gamma$ is not an instance of the axioms, then the first applicable rule will be chosen, and NESCOND will be recursively invoked on its premises. The ordering of the clauses is such that the application of the branching rules is postponed as much as possible.
The whole procedure

- each clause of the `prove` predicate implements an axiom or rule of \( \mathcal{N}S \)
- To search a derivation of a nested sequent \( \Gamma \), NESCOND proceeds as follows:
  - first of all, if \( \Gamma \) is an axiom, the goal will succeed immediately by using one of the clauses for the axioms
  - if \( \Gamma \) is not an instance of the axioms, then the first applicable rule will be chosen, and NESCOND will be recursively invoked on its premises. The ordering of the clauses is such that the application of the branching rules is postponed as much as possible.
The whole procedure

- each clause of the `prove` predicate implements an axiom or rule of $\mathcal{NS}$
- To search a derivation of a nested sequent $\Gamma$, NESCOND proceeds as follows:
  - first of all, if $\Gamma$ is an axiom, the goal will succeed immediately by using one of the clauses for the axioms
  - if $\Gamma$ is not an instance of the axioms, then the first applicable rule will be chosen, and NESCOND will be recursively invoked on its premises. The ordering of the clauses is such that the application of the branching rules is postponed as much as possible.
Performances of NESCOND

CK over 88 valid sequence

<table>
<thead>
<tr>
<th>Time limit (ms)</th>
<th>1</th>
<th>100</th>
<th>1000</th>
<th>5000</th>
<th>30000</th>
</tr>
</thead>
<tbody>
<tr>
<td>NESCOND</td>
<td>16</td>
<td>13</td>
<td>10</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>CondLean</td>
<td>22</td>
<td>16</td>
<td>14</td>
<td>10</td>
<td>9</td>
</tr>
</tbody>
</table>

CK over random formulas

<table>
<thead>
<tr>
<th>number of formulas: 30</th>
<th>Prop. vars: 5</th>
<th>Depth: 3</th>
<th>Timeout: 1 ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>73,00%</td>
<td>21,00%</td>
<td>6,00%</td>
</tr>
<tr>
<td>no</td>
<td>74,00%</td>
<td>0,00%</td>
<td>26,00%</td>
</tr>
<tr>
<td>Nested sequents</td>
<td>74,00%</td>
<td>26,00%</td>
<td>0,00%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>number of formulas: 10</th>
<th>Prop. vars: 3</th>
<th>Depth: 15</th>
<th>Timeout: 1ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>55,00%</td>
<td>4,00%</td>
<td>41,00%</td>
</tr>
<tr>
<td>no</td>
<td>69,00%</td>
<td>0,00%</td>
<td>31,00%</td>
</tr>
<tr>
<td>Nested sequents</td>
<td>69,00%</td>
<td>31,00%</td>
<td>0,00%</td>
</tr>
</tbody>
</table>
Conclusions and Future Works

Our contribution

- Nested sequent calculi are **natural** for conditional logics
- calculi for the basic CK and some extensions
- calculus for KLM cumulative logic \( C \)
- Nested sequent calculi:
  - “Simple” (finite number of rules with a fixed number of premises)
  - analytic
  - completeness established via cut-elimination
  - provide a decision procedure, in some cases of optimal complexity
Conclusions and Future Works

Our contribution

- Nested sequent calculi are natural for conditional logics
- Calculi for the basic CK and some extensions
- Calculus for KLM cumulative logic C
- Nested sequent calculi:
  - “Simple” (finite number of rules with a fixed number of premises)
  - Analytic
  - Completeness established via cut-elimination
  - Provide a decision procedure, in some cases of optimal complexity
Conclusions and Future Works

Our contribution

- Nested sequent calculi are **natural** for conditional logics
- calculi for the basic CK and some extensions
- calculus for KLM cumulative logic $C$
- Nested sequent calculi:
  - “Simple” (finite number of rules with a fixed number of premises)
  - analytic
  - completeness established via cut-elimination
  - provide a decision procedure, in some cases of optimal complexity
Conclusions and Future Works

Our contribution

- Nested sequent calculi are natural for conditional logics
- calculi for the basic CK and some extensions
- calculus for KLM cumulative logic $C$
- Nested sequent calculi:
  - “Simple” (finite number of rules with a fixed number of premises)
  - analytic
  - completeness established via cut-elimination
  - provide a decision procedure, in some cases of optimal complexity
Conclusions and Future Works

Our contribution

- Nested sequent calculi are natural for conditional logics
- calculi for the basic CK and some extensions
- calculus for KLM cumulative logic $\mathcal{C}$
- Nested sequent calculi:
  - “Simple” (finite number of rules with a fixed number of premises)
  - analytic
  - completeness established via cut-elimination
  - provide a decision procedure, in some cases of optimal complexity
Conclusions and Future Works

Our contribution

- Nested sequent calculi are **natural** for conditional logics
- calculi for the basic CK and some extensions
- calculus for KLM cumulative logic **C**
- Nested sequent calculi:
  - “Simple” (finite number of rules with a fixed number of premises)
  - analytic
  - completeness established via cut-elimination
  - provide a decision procedure, in some cases of optimal complexity
Conclusions and Future Works

Our contribution

- Nested sequent calculi are **natural** for conditional logics
- calculi for the basic CK and some extensions
- calculus for KLM cumulative logic $\mathbf{C}$
- Nested sequent calculi:
  - “Simple” (finite number of rules with a fixed number of premises)
  - analytic
  - completeness established via cut-elimination
  - provide a decision procedure, in some cases of optimal complexity
Conclusions and Future Works

Our contribution

- Nested sequent calculi are **natural** for conditional logics
- calculi for the basic CK and some extensions
- calculus for KLM cumulative logic $\mathcal{C}$
- Nested sequent calculi:
  - “Simple” (finite number of rules with a fixed number of premises)
  - analytic
  - completeness established via cut-elimination
  - provide a decision procedure, in some cases of optimal complexity
Conclusions and Future Works

Our contribution

- Nested sequent calculi are **natural** for conditional logics
- calculi for the basic CK and some extensions
- calculus for KLM cumulative logic $C$
- Nested sequent calculi:
  - “Simple” (finite number of rules with a fixed number of premises)
  - analytic
  - completeness established via cut-elimination
  - provide a decision procedure, in some cases of optimal complexity
Conclusions and Future Works

Open issues

- Extensions of nested sequents to a wider spectrum of conditional logics (e.g. CS, etc.)
- Improvements of the calculi towards efficiency, based on a better control of duplication (e.g. CEM)
- Take advantage of the calculi to study logical properties of the corresponding systems (disjunction property, interpolation) in a constructive way
Conclusions and Future Works

Open issues

- Extentions of nested sequents to a wider spectrum of conditional logics (e.g. CS, etc.)
- Improvements of the calculi towards efficiency, based on a better control of duplication (e.g. CEM)
- Take advantage of the calculi to study logical properties of the corresponding systems (disjunction property, interpolation) in a constructive way
Conclusions and Future Works

Open issues

- Extentions of nested sequents to a wider spectrum of conditional logics (e.g. CS, etc.)
- Improvements of the calculi towards efficiency, based on a better control of duplication (e.g. CEM)
- Take advantage of the calculi to study logical properties of the corresponding systems (disjunction property, interpolation) in a constructive way
Conclusions and Future Works

Open issues

- Extentions of nested sequents to a wider spectrum of conditional logics (e.g. CS, etc.)
- Improvements of the calculi towards efficiency, based on a better control of duplication (e.g. CEM)
- Take advantage of the calculi to study logical properties of the corresponding systems (disjunction property, interpolation) in a constructive way
Proof methods for conditional logics

A few references


Thank you!!!!!
References

A. Artosi, G. Governatori, and A. Rotolo.
Labelled tableaux for non-monotonic reasoning: Cumulative consequence relations.

Oliver Board.
Dynamic interactive epistemology.

C. Boutilier.
Conditional logics of normality: a modal approach.

Alexandru Baltag and Sonja Smets.
The logic of conditional doxastic actions.
*Texts in Logic and Games, Special Issue on New Perspectives on Games and Interaction*, 4:9–31, 2008.