Proof Methods and Theorem Proving for nonclassical logics

Gian Luca Pozzato[1]

[1]Dipartimento di Informatica, Universitá degli Studi di Torino, Italy

2nd TICAMORE meeting, Marseille 15/11/2017

Outline

- Nonclassical logics
  - Conditional logics
    - Selection function semantics
    - Lewis' spheres (Marianna in the next talk)
  - Nonmonotonic extensions of Description Logics
    - Preferential models + minimal model semantics
- Proof methods for Conditional Logics
  - external calculi
  - internal calculi
- Theorem provers
  - implementations of sequent/tableaux calculi
  - inspired by leanT^AP
  - Loop-checking + heuristics + parallel computations

Conditional Logics

Conditional Logics

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

History

- hypothetical reasoning "if A were the case then B"
- counterfactual sentences: conditionals of the form "if A were the case then B would be the case", where A is false [Lewis, 1973]

Conditional Logics

Conditional Logics

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

History

- hypothetical reasoning "if A were the case then B"
- counterfactual sentences: conditionals of the form "if A were the case then B would be the case", where A is false [Lewis, 1973]

Conditional Logics

Conditional Logics

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

History

- hypothetical reasoning "if A were the case then B"
- counterfactual sentences: conditionals of the form "if A were the case then B would be the case", where A is false [Lewis, 1973]

Conditional Logics

Conditional Logics

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

History

- knowledge update and revision [Grahne, 1998]:
  - correspondence with AGM revision [Giordano et al., 2005]
    - Ramsey test
    - $K \circ \{A\} \vdash B$ iff $K \models A \Rightarrow B$
  - axiomatic foundation of nonmonotonic reasoning [Boutilier, 1994, Kraus et al., 1990] "in normal circumstances if A then B"
  - multi-agent revision, application to game theory [Baltag and Smets, 2008, Board, 2004]

Conditional Logics

Conditional Logics

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

History

- knowledge update and revision [Grahne, 1998]:
  - correspondence with AGM revision [Giordano et al., 2005]
    - Ramsey test
    - $K \circ \{A\} \vdash B$ iff $K \models A \Rightarrow B$
  - axiomatic foundation of nonmonotonic reasoning [Boutilier, 1994, Kraus et al., 1990] "in normal circumstances if A then B"
  - multi-agent revision, application to game theory [Baltag and Smets, 2008, Board, 2004]

Conditional Logics

Conditional Logics

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

History

- knowledge update and revision [Grahne, 1998]:
  - correspondence with AGM revision [Giordano et al., 2005]
    - Ramsey test
    - $K \circ \{A\} \vdash B$ iff $K \models A \Rightarrow B$
  - axiomatic foundation of nonmonotonic reasoning [Boutilier, 1994, Kraus et al., 1990] "in normal circumstances if A then B"
  - multi-agent revision, application to game theory [Baltag and Smets, 2008, Board, 2004]

Conditional Logics

Conditional Logics

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

History

- knowledge update and revision [Grahne, 1998]:
  - correspondence with AGM revision [Giordano et al., 2005]
    - Ramsey test
    - $K \circ \{A\} \vdash B$ iff $K \models A \Rightarrow B$
  - axiomatic foundation of nonmonotonic reasoning [Boutilier, 1994, Kraus et al., 1990] "in normal circumstances if A then B"
  - multi-agent revision, application to game theory [Baltag and Smets, 2008, Board, 2004]

Conditional Logics

Conditional Logics

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

History

- knowledge update and revision [Grahne, 1998]:
  - correspondence with AGM revision [Giordano et al., 2005]
    - Ramsey test
    - $K \circ \{A\} \vdash B$ iff $K \models A \Rightarrow B$
  - axiomatic foundation of nonmonotonic reasoning [Boutilier, 1994, Kraus et al., 1990] "in normal circumstances if A then B"
  - multi-agent revision, application to game theory [Baltag and Smets, 2008, Board, 2004]

Belief Revision/Epistemic logic

Multi-agent revision [Baltag and Smets, 2008, Board, 2004]

- A model of epistemic interaction
- $A \Rightarrow_i B$ the agent i will believe that B is true if she learns A
- **Beliefs** of an agent i: $Bel_i A$ defined as $\top \Rightarrow_i A$
- **Knowledge** of an agent i: $K_i A$ defined as $\neg A \Rightarrow_i \bot$
- The logic governing $\Rightarrow_i$ is a multi-agent version of Lewis' $\mathbb{V}$

NonMonotonic Reasoning

Plausible conditionals

- Formalization of Nonmonotonic Reasoning: “typically or normally if A then B”
- KLM properties universally accepted as conservative core of nonmonotonic reasoning
  - set of postulates that any nonmonotonic reasoning system should satisfy
- axioms of KLM logics correspond to flat fragment of conditional logics

NonMonotonic Reasoning

Plausible conditionals

- Formalization of Nonmonotonic Reasoning: "typically or normally if A then B"
- KLM properties universally accepted as conservative core of nonmonotonic reasoning
  - set of postulates that any nonmonotonic reasoning system should satisfy
- axioms of KLM logics correspond to flat fragment of conditional logics

NonMonotonic Reasoning

Plausible conditionals

- Formalization of Nonmonotonic Reasoning: "typically or normally if A then B"
- KLM properties universally accepted as conservative core of nonmonotonic reasoning
  - set of postulates that any nonmonotonic reasoning system should satisfy
- axioms of KLM logics correspond to flat fragment of conditional logics

NonMonotonic Reasoning

Plausible conditionals

- Formalization of Nonmonotonic Reasoning: "typically or normally if A then B"
- KLM properties universally accepted as conservative core of nonmonotonic reasoning
  - set of postulates that any nonmonotonic reasoning system should satisfy
- axioms of KLM logics correspond to flat fragment of conditional logics

NonMonotonic Reasoning

Plausible conditionals

- Formalization of Nonmonotonic Reasoning: "typically or normally if A then B"
- KLM properties universally accepted as conservative core of nonmonotonic reasoning
  - set of postulates that any nonmonotonic reasoning system should satisfy
- axioms of KLM logics correspond to flat fragment of conditional logics

NonMonotonic Reasoning

Example

- Plausible conditionals
  - $student \Rightarrow \neg taxPayer$
  - $student \wedge worker \Rightarrow taxPayer$
  - $student \wedge worker \wedge parent \Rightarrow \neg taxPayer$
- interpreting $\Rightarrow$ as material implication we would get for instance: $\vdash \neg(student \wedge worker)$

Language $\mathcal{L}$

Alphabet

- set of propositional variables $\mathcal{V}$
- symbols of *false* $\perp$ and *true* $\top$
- set of connectives $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\Rightarrow$

Formulas

- Generated by the following grammar:

$$A, B ::= P \mid \top \mid \perp \mid \neg A \mid A \wedge B \mid A \vee B \mid A \rightarrow B \mid A \Rightarrow B$$

Language $\mathcal{L}$

Alphabet

- set of propositional variables $\mathcal{V}$
- symbols of *false* $\bot$ and *true* $\top$
- set of connectives $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\Rightarrow$

Formulas

- Generated by the following grammar:

$$A, B ::= P \mid \top \mid \bot \mid \neg A \mid A \wedge B \mid A \vee B \mid A \rightarrow B \mid A \Rightarrow B$$

Conditional Logics

Semantics

- Possible world semantics
- a conditional $A \Rightarrow B$ is true in a world w, if B is true in the set of worlds where A is true and that are most similar to/closest to/"as normal as" w
- Lack of a universally accepted semantics
- Most popular semantics (in decreasing order of generality):
  - *selection function* semantics (Stalnaker, Nute) [Nute, 1980]
  - *preferential* semantics (Lewis, Burgess) [Lewis, 1973]
  - *sphere* semantics (Lewis) [Lewis, 1973]

Conditional Logics

Semantics

- Possible world semantics
- a conditional $A \Rightarrow B$ is true in a world w, if B is true in the set of worlds where A is true and that are most similar to/closest to/ "as normal as" w
- Lack of a universally accepted semantics
- Most popular semantics (in decreasing order of generality):
    - *selection function* semantics (Stalnaker, Nute) [Nute, 1980]
    - *preferential* semantics (Lewis, Burgess) [Lewis, 1973]
    - *sphere* semantics (Lewis) [Lewis, 1973]

Conditional Logics

Semantics

- Possible world semantics
- a conditional $A \Rightarrow B$ is true in a world w, if B is true in the set of worlds where A is true and that are most similar to/closest to/"as normal as" w
- Lack of a universally accepted semantics
- Most popular semantics (in decreasing order of generality):
  - *selection function* semantics (Stalnaker, Nute) [Nute, 1980]
  - *preferential* semantics (Lewis, Burgess) [Lewis, 1973]
  - *sphere* semantics (Lewis) [Lewis, 1973]

Conditional Logics

Semantics

- Possible world semantics
- a conditional $A \Rightarrow B$ is true in a world w, if B is true in the set of worlds where A is true and that are most similar to/closest to/"as normal as" w
- Lack of a universally accepted semantics
- Most popular semantics (in decreasing order of generality):
    - *selection function* semantics (Stalnaker, Nute) [Nute, 1980]
    - *preferential* semantics (Lewis, Burgess) [Lewis, 1973]
    - *sphere* semantics (Lewis) [Lewis, 1973]

Conditional Logics

Semantics

- Possible world semantics
- a conditional $A \Rightarrow B$ is true in a world w, if B is true in the set of worlds where A is true and that are most similar to/closest to/"as normal as" w
- Lack of a universally accepted semantics
- Most popular semantics (in decreasing order of generality):
  - *selection function* semantics (Stalnaker, Nute) [Nute, 1980]
  - *preferential* semantics (Lewis, Burgess) [Lewis, 1973]
  - *sphere* semantics (Lewis) [Lewis, 1973]

Conditional Logics

Semantics

- Possible world semantics
- a conditional $A \Rightarrow B$ is true in a world w, if B is true in the set of worlds where A is true and that are most similar to/closest to/ "as normal as" w
- Lack of a universally accepted semantics
- Most popular semantics (in decreasing order of generality):
  - *selection function* semantics (Stalnaker, Nute) [Nute, 1980]
  - *preferential* semantics (Lewis, Burgess) [Lewis, 1973]
  - *sphere* semantics (Lewis) [Lewis, 1973]

Conditional Logics

Semantics

- Possible world semantics
- a conditional $A \Rightarrow B$ is true in a world w, if B is true in the set of worlds where A is true and that are most similar to/closest to/ "as normal as" w
- Lack of a universally accepted semantics
- Most popular semantics (in decreasing order of generality):
  - *selection function* semantics (Stalnaker, Nute) [Nute, 1980]
  - *preferential* semantics (Lewis, Burgess) [Lewis, 1973]
  - *sphere* semantics (Lewis) [Lewis, 1973]

Conditional Logics

Semantics

- Possible world semantics
- a conditional $A \Rightarrow B$ is true in a world w, if B is true in the set of worlds where A is true and that are most similar to/closest/"as normal as" w
- Lack of a universally accepted semantics:
- Most popular semantics (in decreasing order of generality):
  - *selection function* semantics (Stalnaker, Nute) [Nute, 1980]
  - *preferential* semantics (Lewis, Burgess) [Lewis, 1973]
  - *sphere* semantics (Lewis) [Lewis, 1973]

Selection function semantics

Models

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

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

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

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

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

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

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

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

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

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

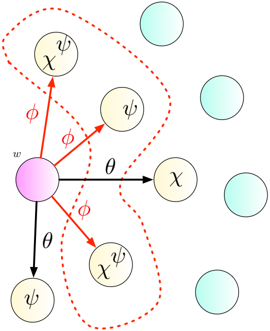
Models

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

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



Examples

$w \in [\![\phi \Rightarrow \psi]\!]$
$w \notin [\![\phi \Rightarrow \chi]\!]$
$w \notin [\![\theta \Rightarrow \chi]\!]$

The basic system CK

CK

- CK is the basic system, axiomatization:
  - any axiomatization of the classical propositional calculus
  - (Modus Ponens) $\dfrac{A \quad A \rightarrow B}{B}$
  - (R-And) $(A \Rightarrow B) \wedge (A \Rightarrow C) \rightarrow (A \Rightarrow (B \wedge C))$
  - (RCEA) $\dfrac{A \leftrightarrow B}{(A \Rightarrow C) \leftrightarrow (B \Rightarrow C)}$
  - (RCK) $\dfrac{A \rightarrow B}{(C \Rightarrow A) \rightarrow (C \Rightarrow B)}$
- A is derivable in CK iff it is valid in every selection function model

Some Extensions of CK

Basic extensions of CK

Other conditional systems are obtained by assuming further properties on the selection function, for instance:

System	Axiom	Model condition
ID	$A \Rightarrow A$	$f(w, [\![A]\!]) \subseteq [\![A]\!]$
MP	$(A \Rightarrow B) \rightarrow (A \rightarrow B)$	$w \in [\![A]\!] \rightarrow w \in f(w, [\![A]\!])$
CS	$(A \wedge B) \rightarrow (A \Rightarrow B)$	$w \in [\![A]\!] \rightarrow f(w, [\![A]\!]) \subseteq \{w\}$
CEM	$(A \Rightarrow B) \vee (A \Rightarrow \neg B)$	$\mid f(w, [\![A]\!]) \mid \leq 1$

Proof methods

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 [Artosi et al., 2002, Olivetti et al., 2007, Giordano et al., 2009]
  - *internal* calculi which stay within the language, so that a "configuration" (sequent, tableaux node...) can be directly interpreted as a formula of the language [Gent, 1992, de Swart, 1983, Schröder et al., 2010, Pattinson and Schröder, 2011]

Proof methods

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 [Artosi et al., 2002, Olivetti et al., 2007, Giordano et al., 2009]
  - *internal* calculi which stay within the language, so that a "configuration" (sequent, tableaux node...) can be directly interpreted as a formula of the language [Gent, 1992, de Swart, 1983, Schröder et al., 2010, Pattinson and Schröder, 2011]

Proof methods

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 [Artosi et al., 2002, Olivetti et al., 2007, Giordano et al., 2009]
  - *internal* calculi which stay within the language, so that a "configuration" (sequent, tableaux node...) can be directly interpreted as a formula of the language [Gent, 1992, de Swart, 1983, Schröder et al., 2010, Pattinson and Schröder, 2011]

External Calculi

SeqS for CK (with Nicola and Camilla Schwind)

- Labels used to represent possibile worlds
- Language $\mathcal{L}$ + alphabet of labels $\{x, y, z, \dots\}$
- 2 kinds of labelled formulas:
  - *world formulas* $x : A$
  - *transition formulas* $x \xrightarrow{A} y$
- representing:
  - A holds in the world x
  - $y \in f(x, [\![A]\!])$

External Calculi

SeqS for CK

- Axioms

$$\Gamma, x : \bot \vdash \Delta \qquad \Gamma, x : P \vdash \Delta, x : P$$

- Propositional rules, e.g.

$$\frac{\Gamma \vdash \Delta, x : A \quad \Gamma, x : B \vdash \Delta}{\Gamma, x : A \rightarrow B \vdash \Delta} \qquad \frac{\Gamma, x : A \vdash \Delta, x : B}{\Gamma \vdash \Delta, x : A \rightarrow B}$$

- For transition formulas

$$\frac{u : A' \vdash u : A \qquad u : A \vdash u : A'}{\Gamma, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{A'} y} \ (EQ)$$

SeqS for CK

- Conditional on the right (y new label)

$$\frac{\Gamma, x \xrightarrow{A} y \vdash \Delta, y : B}{\Gamma \vdash \Delta, x : A \Rightarrow B} \ (\Rightarrow R)$$

y new

- Conditional on the left

$$\frac{\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \qquad \Gamma, x : A \Rightarrow B, y : B \vdash \Delta}{\Gamma, x : A \Rightarrow B \vdash \Delta} \ (\Rightarrow L)$$

Extensions of CK

- ID

$$\frac{\Gamma, x \xrightarrow{A} y, y : A \vdash \Delta}{\Gamma, x \xrightarrow{A} y \vdash \Delta}\ (ID)$$

- MP

$$\frac{\Gamma \vdash \Delta, x \xrightarrow{A} x, x : A}{\Gamma \vdash \Delta, x \xrightarrow{A} x}\ (MP)$$

Theorem provers for Conditional Logics

Common ideas

- Prolog implementation of calculi (sequent/tableaux)
- inspired by leanT^AP: each axiom or rule of the calculi is implemented by a Prolog clause of the program
  - simple and compact code
- proof search provided for free by depth-first mechanism+backtracking of Prolog

Theorem provers for Conditional Logics

Common ideas

- Prolog implementation of calculi (sequent/tableaux)
- inspired by leanT^AP: each axiom or rule of the calculi is implemented by a Prolog clause of the program
  - simple and compact code
- proof search provided for free by depth-first mechanism+backtracking of Prolog

Theorem provers for Conditional Logics

Common ideas

- Prolog implementation of calculi (sequent/tableaux)
- inspired by leanT^AP: each axiom or rule of the calculi is implemented by a Prolog clause of the program
  - simple and compact code
- proof search provided for free by depth-first mechanism+backtracking of Prolog

Theorem provers for Conditional Logics

Common ideas

- Prolog implementation of calculi (sequent/tableaux)
- inspired by leanT^AP: each axiom or rule of the calculi is implemented by a Prolog clause of the program
  - simple and compact code
- proof search provided for free by depth-first mechanism+backtracking of Prolog

CondLean

CondLean (with Nicola)

- Sequents $\Gamma \vdash \Delta$ are pairs of Prolog lists `Gamma, Delta`
- `Gamma` and `Delta` are Prolog list representing multiset of formulas
- Formulas:
  - `[x,a,y]` represents $x \xrightarrow{A} y$
  - `[x,a]` represents $x : A$
- atomic formulas are represented by Prolog constants

CondLean

Predicate `prove`

- Calculi implemented by the predicate `prove`
- `prove(Cond, Gamma, Delta, Labels, Tree)` succeeds if and only if $\Gamma \vdash \Delta$ is derivable in CK (or extensions)
- `Labels` = list of labels occurring in the current branch
- if `prove` succeeds, then `Tree` matches a term representing the derivation
- `Cond` used for termination

How it works

- First, if $\Gamma \vdash \Delta$ is an axiom, then the goal will immediately succeed by using a clause of the axioms
- If it is not, then the first applicable rule will be chosen, and `prove` is recursively invoked on the premise(s) of the rule
  - Ordering of the clauses: branching and non-invertible rules are postponed

Design of CondLean

Clause implementing axiom

```
prove(_,Gamma,Delta,_,tree(ax)):-
    member([X,P],Gamma),
    member([X,P],Delta).
```

$$\Gamma, x : P \vdash \Delta, x : P$$

- No recursive calls to `prove`

Design of CondLean

Clause implementing $(\Rightarrow R)$

```
prove(Cond,Gamma,Delta,Labels,tree(condR,SubTree)):-
    select([X,A => B],Delta,NewDelta), !,
    generateNewLabel(Labels,Y),
    prove(Cond,[[X,A,Y]|Gamma],[[Y,B]|NewDelta],[Y|Labels],SubTree).
```

$$\frac{\Gamma, x \xrightarrow{A} y \vdash \Delta, y : B}{\Gamma \vdash \Delta, x : A \Rightarrow B}\ (\Rightarrow R)$$

- `generateNewLabel` introduces a new label y in the current branch;
- Invertible rule: Prolog cut `!` is used to eventually block backtracking.

Design of CondLean

Clause 1 implementing $(\Rightarrow L)$

```
prove(Cond,Gamma,Delta,Labels,tree(condL,Sub1,Sub2)):-
    member([X,A => B],Gamma),
    select([[X, A=>B],Used],Cond,NewCond),
    member([X,C,Y],Gamma),
    \+member([X,C,Y],Used), !,
    prove([[[X,A=>B],[[X,C,Y]|Used]]|NewCond],Gamma,[[X,A,Y]|Delta],Labels,Sub1),
    prove([[[X,A=>B],[[X,C,Y]|Used]]|NewCond],[[Y,B]|Gamma],Delta,Labels,Sub2).
```

$$\frac{\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \qquad \Gamma, x : A \Rightarrow B, y : B \vdash \Delta}{\Gamma, x : A \Rightarrow B \vdash \Delta} \; (\Rightarrow L)$$

- In order to ensure termination, `Cond` keeps trace of formulas $x \xrightarrow{C} y$ already used to apply $(\Rightarrow L)$ to $x : A \Rightarrow B$
- It contains pairs `[[X,A=>B],Used]`, where `Used` is the list of transitions

Design of CondLean

Clause 2 implementing $(\Rightarrow L)$

```
prove(Cond,Gamma,Delta,Labels,tree(condL,Sub1,Sub2)):-
    member([X,A => B],Gamma),
    \+member([[X, A=>B],_],Cond),
    member([X,C,Y],Gamma),
    prove([[[X,A=>B],[[X,C,Y]]]|Cond],Gamma,[[X,A,Y]|Delta],Labels,Sub1),
    prove([[[X,A=>B],[[X,C,Y]]]|Cond],[[Y,B]|Gamma],Delta,Labels,Sub2).
```

$$\frac{\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \qquad \Gamma, x : A \Rightarrow B, y : B \vdash \Delta}{\Gamma, x : A \Rightarrow B \vdash \Delta} \ (\Rightarrow L)$$

- First application of $(\Rightarrow L)$ to $x : A \Rightarrow B$

Design of CondLean

Free-variable version of $(\Rightarrow L)$

```
prove(Cond,Gamma,Delta,Labels,tree(condL,Sub1,Sub2)):-
    member([X,A => B],Gamma),
    domain([Y],1,Max),
    Y#>X,
    prove([[[X,A=>B],YDomain]|Cond],Gamma,[[X,A,Y]|Delta],Labels,Sub1,Max),
    prove([[[X,A=>B],YDomain]|Cond],[[Y,B]|Gamma],Delta,Labels,Sub2,Max).
```

$$\frac{\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \qquad \Gamma, x : A \Rightarrow B, y : B \vdash \Delta}{\Gamma, x : A \Rightarrow B \vdash \Delta} \ (\Rightarrow L)$$

- y is not fixed, but it is a free-variable
- labels are integer, and `Max` is the highest value in the current branch
- Library clpfd is used to manage free-variables domains

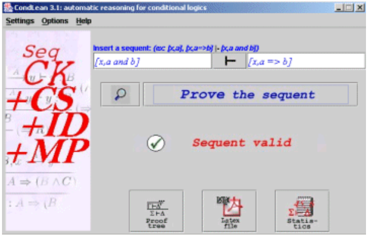
Design of CondLean

Heuristic version

$$\frac{\Gamma \vdash \Delta, x \xrightarrow{A} y \qquad \Gamma, y : B \vdash \Delta}{\Gamma, x : A \Rightarrow B \vdash \Delta} \ (\Rightarrow L)$$

- Not-invertible version of $(\Rightarrow L)$
- Heuristic:
  - Phase 1: not complete calculus with not-invertible $(\Rightarrow L)$
  - Phase 2: in case of a failure, free-variable version is executed

CondLean


CondLean 3.1: automatic reasoning for conditional logics
Settings
Options
Help
Seq
CK
+CS
+ID
+MP
Insert a sequent: (ex: [x,a], [x,a=>b] |- [x,a and b])
[x,a and b]
⊢
[x,a => b]
Prove the sequent
Sequent valid
Proof tree
Latex file
Statistics

Internal Calculi

Nested Sequents (1) (with Nicola and Régis Alenda)

- Nested sequents = generalization of ordinary sequents where sequents may occur within sequents
- A special case of deep-inference calculi (Guglielmi and co-workers)
- $A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n]$
  - $n, m \geq 0$
  - $A_1, \ldots, A_m, B_1, \ldots, B_n$ are formulas
  - $\Gamma_1, \ldots, \Gamma_n$ are nested sequents

Nested Sequents (2) [Alenda et al., 2013]

- Internal calculi: a nested sequent corresponds to a formula of the language
  - replace "," by $\vee$ and ":" by $\Rightarrow$
  - interpretation of $\Gamma = A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n]$ inductively defined by

$$\mathcal{F}(\Gamma) = A_1 \vee \ldots \vee A_m \vee (B_1 \Rightarrow \mathcal{F}(\Gamma_1)) \vee \ldots \vee (B_n \Rightarrow \mathcal{F}(\Gamma_n))$$

Internal Calculi

Nested Sequents (1) (with Nicola and Régis Alenda)

- Nested sequents = generalization of ordinary sequents where sequents may occur within sequents
- A special case of deep-inference calculi (Guglielmi and co-workers)
- $A_1, \dots, A_m, [B_1 : \Gamma_1], \dots, [B_n : \Gamma_n]$
  - $n, m \geq 0$
  - $A_1, \dots, A_m, B_1, \dots, B_n$ are formulas
  - $\Gamma_1, \dots, \Gamma_n$ are nested sequents

Nested Sequents (2) [Alenda et al., 2013]

- Internal calculi: a nested sequent corresponds to a formula of the language
  - replace "," by $\vee$ and ":" by $\Rightarrow$
  - interpretation of $\Gamma = A_1, \dots, A_m, [B_1 : \Gamma_1], \dots, [B_n : \Gamma_n]$ inductively defined by

$$\mathcal{F}(\Gamma) = A_1 \vee \dots \vee A_m \vee (B_1 \Rightarrow \mathcal{F}(\Gamma_1)) \vee \dots \vee (B_n \Rightarrow \mathcal{F}(\Gamma_n))$$

Internal Calculi

Nested Sequents (1) (with Nicola and Régis Alenda)

- Nested sequents = generalization of ordinary sequents where sequents may occur within sequents
- A special case of deep-inference calculi (Guglielmi and co-workers)
- $A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n]$
  - $n, m \geq 0$
  - $A_1, \ldots, A_m, B_1, \ldots, B_n$ are formulas
  - $\Gamma_1, \ldots, \Gamma_n$ are nested sequents

Nested Sequents (2) [Alenda et al., 2013]

- Internal calculi: a nested sequent corresponds to a formula of the language
  - replace "," by $\vee$ and ":" by $\Rightarrow$
  - interpretation of $\Gamma = A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n]$ inductively defined by

$$\mathcal{F}(\Gamma) = A_1 \vee \ldots \vee A_m \vee (B_1 \Rightarrow \mathcal{F}(\Gamma_1)) \vee \ldots \vee (B_n \Rightarrow \mathcal{F}(\Gamma_n))$$

Internal Calculi

Nested Sequents (1) (with Nicola and Régis Alenda)

- Nested sequents = generalization of ordinary sequents where sequents may occur within sequents
- A special case of deep-inference calculi (Guglielmi and co-workers)
- $A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n]$
  - $n, m \geq 0$
  - $A_1, \ldots, A_m, B_1, \ldots, B_n$ are formulas
  - $\Gamma_1, \ldots, \Gamma_n$ are nested sequents

Nested Sequents (2) [Alenda et al., 2013]

- Internal calculi: a nested sequent corresponds to a formula of the language
  - replace "," by $\vee$ and ":" by $\Rightarrow$
  - interpretation of $\Gamma = A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n]$ inductively defined by

$$\mathcal{F}(\Gamma) = A_1 \vee \ldots \vee A_m \vee (B_1 \Rightarrow \mathcal{F}(\Gamma_1)) \vee \ldots \vee (B_n \Rightarrow \mathcal{F}(\Gamma_n))$$

Nested Sequents [Alenda et al., 2013]

Rules of Nested Sequents [Alenda et al., 2013]

$$\Gamma(P, \neg P) \quad (AX)_{P \in ATM} \qquad \Gamma(\top) \quad (AX_\top) \qquad \Gamma(\neg\bot) \quad (AX_\bot)$$

$$\frac{\Gamma(A)}{\Gamma(\neg\neg A)}\,(\neg) \qquad \frac{\Gamma(A) \qquad \Gamma(B)}{\Gamma(A \wedge B)}\,(\wedge^+) \qquad \frac{\Gamma(\neg A, \neg B)}{\Gamma(\neg(A \wedge B))}\,(\wedge^-)$$

$$\frac{\Gamma(A, B)}{\Gamma(A \vee B)}\,(\vee^+) \qquad \frac{\Gamma(\neg A) \qquad \Gamma(\neg B)}{\Gamma(\neg(A \vee B))}\,(\vee^-) \qquad \frac{\Gamma(\neg A, B)}{\Gamma(A \rightarrow B)}\,(\rightarrow^+) \qquad \frac{\Gamma(A) \qquad \Gamma(\neg B)}{\Gamma(\neg(A \rightarrow B))}\,(\rightarrow^-)$$

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

$$\frac{\Gamma(\neg(A \Rightarrow B), A) \qquad \Gamma(\neg(A \Rightarrow B), \neg B)}{\Gamma(\neg(A \Rightarrow B))}\,(MP) \qquad \frac{\Gamma([A : \Delta, \Sigma], [B : \Sigma]) \qquad A, \neg B \qquad B, \neg A}{\Gamma([A : \Delta], [B : \Sigma])}\,(CEM)$$

NESCOND

Implementation of nested sequent calculi (with Nicola and Régis Alenda)

- Prolog list

```
[F_1, F_2, ..., F_m,[[A_1,Gamma_1],AppliedConditionals_1],
[[A_2,Gamma_2],AppliedConditionals_2], ..., [[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,...,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)

NESCOND

Auxiliary predicates

- 3 predicates to manipulate formulas "inside" a sequent:
  - `deepMember(+Formulas,+NS)` succeeds if and only if either (i) `NS` contains all the fomulas 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`

NESCOND

Auxiliary predicates

- 3 predicates to manipulate formulas "inside" a sequent:
  - `deepMember(+Formulas,+NS)` succeeds if and only if either (i) `NS` contains all the fomulas 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`

NESCOND

Auxiliary predicates

- 3 predicates to manipulate formulas "inside" a sequent:
  - `deepMember(+Formulas,+NS)` succeeds if and only if either (i) `NS` contains all the fomulas 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`

NESCOND

Auxiliary predicates

- 3 predicates to manipulate formulas “inside” a sequent:
  - `deepMember(+Formulas,+NS)` succeeds if and only if either (i) `NS` contains all the fomulas 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`

NESCOND

Main predicate

- Calculi $\mathcal{N}S$ implemented by the predicate

```
prove(NS,ProofTree).
```

- success in case list NS is derivable
- if succeeds, the output term ProofTree matches with a representation of the derivation found by the prover, used in order to display the proof tree

NESCOND

Main predicate

- Calculi $\mathcal{N}S$ implemented by the predicate

```
prove(NS,ProofTree).
```

- success in case list `NS` is derivable
- if succeeds, the output term `ProofTree` matches with a representation of the derivation found by the prover, used in order to display the proof tree

NESCOND

Main predicate

- Calculi $\mathcal{N}S$ implemented by the predicate

```
prove(NS,ProofTree).
```

- success in case list `NS` is derivable
- if succeeds, the output term `ProofTree` matches with a representation of the derivation found by the prover, used in order to display the proof tree

NESCOND

Example

- Check the validity of $(A \Rightarrow (B \wedge C)) \rightarrow (A \Rightarrow B)$
- Query NESCOND with the goal `prove([(a => b ^ c) -> (a => 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),!.
```

NESCOND

Example

- Check the validity of $(A \Rightarrow (B \wedge C)) \rightarrow (A \Rightarrow B)$
- Query NESCOND with the goal `prove([(a => b ^ c) -> (a => 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),!.
```

NESCOND

Example

- Check the validity of $(A \Rightarrow (B \wedge C)) \rightarrow (A \Rightarrow B)$
- Query NESCOND with the goal `prove([(a => b ^ c) -> (a => 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),!.
```

NESCOND

The whole procedure

- To search a derivation of a nested sequent Γ, NESCOND proceeds as follows:
  - first of all, if Γ is an axiom, the goal will succeed immediately by using one of the clauses for the axioms
  - if Γ 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.

NESCOND

The whole procedure

- To search a derivation of a nested sequent Γ, NESCOND proceeds as follows:
  - first of all, if Γ is an axiom, the goal will succeed immediately by using one of the clauses for the axioms
  - if Γ 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.

NESCOND

The whole procedure

- To search a derivation of a nested sequent Γ, NESCOND proceeds as follows:
  - first of all, if Γ is an axiom, the goal will succeed immediately by using one of the clauses for the axioms
  - if Γ 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.

NESCOND

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).
```

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

NESCOND

Web Application

`http://www.di.unito.it/~pozzato/nescond/index.html`

Internal Calculi for Lewis CLs (with Nicola, Marianna, Bjoern)

Lewis's CLs

- counterfactual reasoning
- Comparative plausibility operator (primitive): $A \preccurlyeq B$, "A is at least as plausible as B"
- the two connectives $\preccurlyeq$ and $\Rightarrow$ are interdefinable:
  - $A \Rightarrow B \equiv (\bot \preccurlyeq A) \vee \neg(A \wedge \neg B \preccurlyeq A)$
  - $A \preccurlyeq B \equiv ((A \vee B) \Rightarrow \bot) \vee \neg((A \vee B) \Rightarrow \neg A)$

Axiomatization

- classical axioms and rules
- if $B \rightarrow (A_1 \vee \ldots \vee A_n)$ then $(A_1 \preccurlyeq B) \vee \ldots \vee (A_n \preccurlyeq B)$
- $(A \preccurlyeq B) \vee (B \preccurlyeq A)$
- $(A \preccurlyeq B) \wedge (B \preccurlyeq C) \rightarrow (A \preccurlyeq C)$
- $A \Rightarrow B \equiv (\bot \preccurlyeq A) \vee \neg(A \wedge \neg B \preccurlyeq A)$

Internal Calculi for Lewis CLs (with Nicola, Marianna, Bjoern)

Lewis's CLs

- counterfactual reasoning
- Comparative plausibility operator (primitive): $A \preccurlyeq B$, "A is at least as plausible as B"
- the two connectives $\preccurlyeq$ and $\Rightarrow$ are interdefinable:
  - $A \Rightarrow B \equiv (\bot \preccurlyeq A) \vee \neg(A \wedge \neg B \preccurlyeq A)$
  - $A \preccurlyeq B \equiv ((A \vee B) \Rightarrow \bot) \vee \neg((A \vee B) \Rightarrow \neg A)$

Axiomatization

- classical axioms and rules
- if $B \rightarrow (A_1 \vee \ldots \vee A_n)$ then $(A_1 \preccurlyeq B) \vee \ldots \vee (A_n \preccurlyeq B)$
- $(A \preccurlyeq B) \vee (B \preccurlyeq A)$
- $(A \preccurlyeq B) \wedge (B \preccurlyeq C) \rightarrow (A \preccurlyeq C)$
- $A \Rightarrow B \equiv (\bot \preccurlyeq A) \vee \neg(A \wedge \neg B \preccurlyeq A)$

Internal calculi $\mathcal{I}_{\mathbb{V}}$ for comparative plausibility

Internal calculus for **V**

- **Idea**: extend the language by another "connective" which encodes *several* $\preccurlyeq$-formulas into one
- Basic constituent of sequents are **blocks** representing disjunctions of $\preccurlyeq$-formulas:

$$[A_1, \ldots, A_m \lhd A]$$

$$(A_1 \preccurlyeq A) \vee (A_2 \preccurlyeq A) \vee \cdots \vee (A_m \preccurlyeq A)$$

Blocks

- Compact encoding:

$$\Gamma \vdash \Delta', [\Sigma_1 \lhd A_1], \ldots, [\Sigma_n \lhd A_n] := \bigwedge \Gamma \to \bigvee \Delta' \vee \bigvee_{1 \leq i \leq n} \bigvee_{B \in \Sigma_i} (B \preccurlyeq A_i)$$

The invertible calculi $\mathcal{I}^{\mathsf{i}}_{\mathcal{L}}$

Rules of $\mathcal{I}^{\mathsf{i}}_{\mathbb{V}}$

- Axioms, rules for the implication:

$$\frac{}{\Gamma, \bot \vdash \Delta} \; \bot_L \qquad \frac{}{\Gamma, p \vdash \Delta, p} \; \text{init} \qquad \frac{\Gamma, B \vdash \Delta \qquad \Gamma \vdash \Delta, A}{\Gamma, A \to B \vdash \Delta} \; \to_L \qquad \frac{\Gamma, A \vdash \Delta, B}{\Gamma \vdash \Delta, A \to B} \; \to_R$$

- Rules for the comparative plausibility operator:

$$\frac{\Gamma \vdash \Delta, [A \lhd B]}{\Gamma \vdash \Delta, A \preccurlyeq B} \; \preccurlyeq_R$$

$$\frac{\Gamma, A \preccurlyeq B \vdash \Delta, [B, \Sigma \lhd C] \quad \Gamma, A \preccurlyeq B \vdash \Delta, [\Sigma \lhd A], [\Sigma \lhd C]}{\Gamma, A \preccurlyeq B \vdash \Delta, [\Sigma \lhd C]} \; \preccurlyeq^{\mathsf{i}}_L$$

- Rules for blocks:

$$\frac{\Gamma \vdash \Delta, [\Sigma_1, \Sigma_2 \lhd A], [\Sigma_2 \lhd B] \quad \Gamma \vdash \Delta, [\Sigma_1 \lhd A], [\Sigma_1, \Sigma_2 \lhd B]}{\Gamma \vdash \Delta, [\Sigma_1 \lhd A], [\Sigma_2 \lhd B]} \; \mathsf{com}^{\mathsf{i}}$$

$$\frac{A \vdash \Sigma}{\Gamma \vdash \Delta, [\Sigma \lhd A]} \; \mathsf{jump}$$

The invertible calculi $\mathcal{I}^{\mathsf{i}}_{\mathcal{L}}$

Rules for extensions of $\mathbb{V}$

$$\frac{\Gamma \vdash \Delta, [\bot \lhd \top]}{\Gamma \vdash \Delta}\ \mathsf{N}$$

$$\frac{\Gamma, A \preccurlyeq B \vdash \Delta, B}{\Gamma, A \preccurlyeq B \vdash \Delta}\ \mathsf{T^i} \qquad \frac{\Gamma \vdash \Delta, [\Sigma \lhd A], \Sigma}{\Gamma \vdash \Delta, [\Sigma \lhd A]}\ \mathsf{W^i}$$

$$\frac{\Gamma, A \preccurlyeq B \vdash \Delta, B \quad \Gamma, A \preccurlyeq B, A \vdash \Delta}{\Gamma, A \preccurlyeq B \vdash \Delta}\ \mathsf{C^i} \qquad \frac{\Gamma^{\preccurlyeq}, B \vdash \Delta^{\preccurlyeq}, [\Sigma \lhd B], \Sigma}{\Gamma \vdash \Delta, [\Sigma \lhd B]}\ \mathsf{A^i}$$

$\Gamma^{\preccurlyeq} \vdash \Delta^{\preccurlyeq} = \Gamma \vdash \Delta$ restricted to formulas of the form $C \preccurlyeq D$ and blocks

$$\mathcal{I}^{\mathsf{i}}_{\mathbb{VN}} := \mathcal{I}^{\mathsf{i}}_{\mathbb{V}} \cup \{\mathsf{N}\} \qquad \mathcal{I}^{\mathsf{i}}_{\mathbb{VW}} := \mathcal{I}^{\mathsf{i}}_{\mathbb{V}} \cup \{\mathsf{N}, \mathsf{T^i}, \mathsf{W^i}\} \qquad \mathcal{I}^{\mathsf{i}}_{\mathbb{VA}} := \mathcal{I}^{\mathsf{i}}_{\mathbb{V}} \cup \{\mathsf{A^i}\}$$

$$\mathcal{I}^{\mathsf{i}}_{\mathbb{VT}} := \mathcal{I}^{\mathsf{i}}_{\mathbb{V}} \cup \{\mathsf{N}, \mathsf{T^i}\} \qquad \mathcal{I}^{\mathsf{i}}_{\mathbb{VC}} := \mathcal{I}^{\mathsf{i}}_{\mathbb{V}} \cup \{\mathsf{N}, \mathsf{T^i}, \mathsf{W^i}, \mathsf{C^i}\} \qquad \mathcal{I}^{\mathsf{i}}_{\mathbb{VNA}} := \mathcal{I}^{\mathsf{i}}_{\mathbb{V}} \cup \{\mathsf{N}, \mathsf{A^i}\}$$

Design of VINTE

Program VINTE (with Nicola, Marianna, Bjoern, Vitalis Quentin

- Sequent $\Gamma \vdash \Delta$ represented with a pair of Prolog lists `[Gamma,Delta]`
- Elements of `Gamma` are formulas
- Elements of `Delta`: either formulas or pairs `[Sigma,A]` where `Sigma` is a Prolog list
- $\top$ and $\bot$ represented by constants `true` and `false`
- connectives $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\preccurlyeq$, and $>$ represented by `-`, `^`, `?`, `->`, `<`, and `=>`
- Propositional variables are represented by Prolog atoms

Example

```
[ [-(p?q), p, p -> q, p < r], [q, p => (q ^ r), [ [true, p, q], r] ] ]
```

represents the sequent

$$\neg(P \vee Q), P, P \to Q, P \preccurlyeq R \vdash Q, P \Rightarrow (Q \wedge R), [\top, P, Q \lhd R]$$

Design of VINTE

The calculi $\mathcal{I}^{i}_{\mathcal{L}}$ are implemented by the predicate

```
prove([Gamma,Delta],ProofTree).
```

The predicate `prove`

- Succeeds if and only if $\Gamma \vdash \Delta$ represented by `[Gamma,Delta]` is derivable;
- When it succeeds, the output term `ProofTree` matches with a representation of the derivation found by the prover.

Example

- Is $(A \preccurlyeq B) \vee (B \preccurlyeq A)$ valid in $\mathbb{V}$?
- Query VINTE with the goal:

```
prove([[],[(a<b)?(b<a)]],ProofTree).
```

Design of VINTE

To search a derivation (1)

- Each clause of `prove` implements an axiom or rule of $\mathcal{I}^{\mathrm{i}}_{\mathcal{L}}$
- if $\Gamma \vdash \Delta$ is an instance of either $\bot_L$ or $\top_R$ or init, the goal will succeed immediately by using one of the clauses for the axioms;

Clauses for the axioms

```
prove([Gamma,Delta],tree(axb):-member(false,Gamma),!.
prove([Gamma,Delta],tree(axt)):-member(true,Delta),!.
prove([Gamma,Delta],tree(init)):-member(P,Gamma),member(P,Delta),!.
```

$$\frac{}{\Gamma, \bot \vdash \Delta}\ \bot_L \qquad \frac{}{\Gamma \vdash \Delta, \top}\ \top_R \qquad \frac{}{\Gamma, p \vdash \Delta, p}\ \text{init}$$

Design of VINTE

To search a derivation (2)

- If $\Gamma \Rightarrow \Delta$ is not an axiom, the first **applicable** rule will be chosen;
- VINTE will be recursively invoked on premise(s) of selected rule;
- Ordering of the clauses: the application of the branching rules is postponed as much as possible (exception: jump).

Design of VINTE

Clause implementing $\preccurlyeq_R$

```
prove([Gamma,Delta],tree(precR,[Gamma,Delta],SubTree,no)):-
    select(A<B,Delta,NewDelta),\+memberOrdSet([[A],B],Delta),!,
    prove([Gamma,[[[A],B]|NewDelta]],SubTree).
```

$$\frac{\Gamma \vdash \Delta, [A \lhd B]}{\Gamma \vdash \Delta, A \preccurlyeq B} \preccurlyeq_R$$

- `memberOrdSet([Sigma, A], Delta)` succeeds iff `Delta` contains a block `[Psi, A]` such that `Sigma` $\subseteq$ `Psi`;
- Invertible rule: Prolog cut `!` is used to eventually block backtracking.

Design of VINTE

Clause implementing $\preccurlyeq^{i}_{L}$

```
prove([Gamma,Delta],tree(precL,Sub1,Sub2)):-
    member(A < B,Gamma),
    select([Sigma,C],Delta,NewDelta),
    remove_duplicates([B|Sigma],NewSigma),
    \+memberOrdSet([NewSigma,C],Delta),
    \+memberOrdSet([Sigma,A],Delta), !,
    prove([Gamma,[[NewSigma,C]|NewDelta]],Sub1),
    prove([Gamma,[[Sigma,A]|Delta]],Sub2).
```

$$\frac{\Gamma, A \preccurlyeq B \vdash \Delta, [B, \Sigma \lhd C] \quad \Gamma, A \preccurlyeq B \vdash \Delta, [\Sigma \lhd A], [\Sigma \lhd C]}{\Gamma, A \preccurlyeq B \vdash \Delta, [\Sigma \lhd C]} \preccurlyeq^{i}_{L}$$

- `remove_duplicates([B|Sigma], NewSigma)` invoked to remove duplicated formulas in the list B, Σ

Design of VINTE

Clause implementing jump

```
prove([Gamma,Delta],tree(jump,SubTree)):-
    member([Sigma,A],Delta),
    prove([[A],Sigma],SubTree).
```

$$\frac{A \vdash \Sigma}{\Gamma \vdash \Delta, [\Sigma \lhd A]} \text{ jump}$$

- Prolog cut ! is missing, since jump is the only non-invertible rule.

VINTE

Web Application

http://193.51.60.97:8000/vinte/

Description Logics

Description Logics

- Important formalisms of knowledge representation
- Two key advantages:
  - well-defined semantics based on first-order logic
  - good trade-off between expressivity and complexity
- at the base of languages for the semantic (e.g. OWL)

Knowledge bases

- Two components:
  - TBox = inclusion relations among concepts
  - ABox = instances of concepts and roles = properties and relations among individuals

Description Logics

Description Logics

- Important formalisms of knowledge representation
- Two key advantages:
  - well-defined semantics based on first-order logic
  - good trade-off between expressivity and complexity
- at the base of languages for the semantic (e.g. OWL)

Knowledge bases

- Two components:
  - TBox=inclusion relations among concepts
    - $Footballer \sqsubseteq Athlete$
  - ABox= instances of concepts and roles = properties and relations among individuals
    - $Footballer(paul)$

Description Logics

Description Logics

- Important formalisms of knowledge representation
- Two key advantages:
  - well-defined semantics based on first-order logic
  - good trade-off between expressivity and complexity
- at the base of languages for the semantic (e.g. OWL)

Knowledge bases

- Two components:
  - TBox=inclusion relations among concepts
    - $Footballer \sqsubseteq Athlete$
  - ABox= instances of concepts and roles = properties and relations among individuals
    - $Footballer(paul)$

Description Logics

Description Logics

- Important formalisms of knowledge representation
- Two key advantages:
  - well-defined semantics based on first-order logic
  - good trade-off between expressivity and complexity
- at the base of languages for the semantic (e.g. OWL)

Knowledge bases

- Two components:
  - TBox=inclusion relations among concepts
    - $Footballer \sqsubseteq Athlete$
  - ABox= instances of concepts and roles = properties and relations among individuals
    - $Footballer(paul)$

Description Logics

Reasoning

- TBox = taxonomy of concepts
- need of representing prototypical properties and of reasoning about defeasible inheritance
- integration with nonmonotonic reasoning mechanism to handle defeasible inheritance (default rules, circumscription, MKNF)
- all these methods present some difficulties

Our solution (with Nicola, Laura Giordano, Valentina Gliozzi)

- DLs + typicality operator $\mathbf{T}$ for defeasible reasoning in DLs
- meaning of $\mathbf{T}$: (for any concept C) $\mathbf{T}(C)$ singles out the "typical" instances of C
- semantics of $\mathbf{T}$ defined by a set of postulates that are a restatement of Kraus-Lehmann-Magidor axioms of preferential or rational logic

Description Logics

Reasoning

- TBox = taxonomy of concepts
- need of representing prototypical properties and of reasoning about defeasible inheritance
- integration with nonmonotonic reasoning mechanism to handle defeasible inheritance (default rules, circumscription, MKNF)
- all these methods present some difficulties

Our solution (with Nicola, Laura Giordano, Valentina Gliozzi)

- DLs + typicality operator **T** for defeasible reasoning in DLs
- meaning of **T**: (for any concept C) $\mathbf{T}(C)$ singles out the "typical" instances of C
- semantics of **T** defined by a set of postulates that are a restatement of Kraus-Lehmann-Magidor axioms of preferential or rational logic

The logic $\mathcal{ALC} + \mathbf{T}_{\mathsf{R}}^{\mathsf{RaCl}}$

Basic notions

- A KB comprises assertions $\mathbf{T}(C) \sqsubseteq D$
- $\mathbf{T}(\mathit{Student}) \sqsubseteq \mathit{FacebookUsers}$ means "normally, students use Facebook"
- $\mathbf{T}$ is nonmonotonic
  - $C \sqsubseteq D$ does not imply $\mathbf{T}(C) \sqsubseteq \mathbf{T}(D)$

The logic $\mathcal{ALC} + \mathbf{T}_{\mathbf{R}}^{\mathbf{RaCl}}$

Basic notions

- A KB comprises assertions $\mathbf{T}(C) \sqsubseteq D$
- $\mathbf{T}(Student) \sqsubseteq FacebookUsers$ means "normally, students use Facebook"
- $\mathbf{T}$ is nonmonotonic
  - $C \sqsubseteq D$ does not imply $\mathbf{T}(C) \sqsubseteq \mathbf{T}(D)$

The logic $\mathcal{ALC} + \mathbf{T}_{\mathbf{R}}^{\mathsf{RaCl}}$

Example

$$SumoWrestler \sqsubseteq Athlete$$
$$\mathbf{T}(Athlete) \sqsubseteq \neg Fat$$
$$\mathbf{T}(SumoWrestler) \sqsubseteq Fat$$

Reasoning

- ABox:

  $$Athlete(paul)$$

- Expected conclusions:

  $$\neg Fat(paul)$$

The logic $\mathcal{ALC} + \mathbf{T}_{\mathsf{R}}^{\mathsf{RaCl}}$

Example

$SumoWrestler \sqsubseteq Athlete$
$\mathbf{T}(Athlete) \sqsubseteq \neg Fat$
$\mathbf{T}(SumoWrestler) \sqsubseteq Fat$

Reasoning

- ABox:
  - $Athlete(paul)$
- Expected conclusions:
  - $\neg Fat(paul)$

The logic $\mathcal{ALC} + \mathbf{T}_{\mathsf{R}}^{\mathsf{RaCl}}$

Example

$SumoWrestler \sqsubseteq Athlete$
$\mathbf{T}(Athlete) \sqsubseteq \neg Fat$
$\mathbf{T}(SumoWrestler) \sqsubseteq Fat$

Reasoning

- ABox:
  - $Athlete(paul)$
- Expected conclusions:
  - $\neg Fat(paul)$

The logic $\mathcal{ALC} + \mathbf{T}_{\mathsf{R}}^{\mathsf{RaCl}}$

Example

$SumoWrestler \sqsubseteq Athlete$
$\mathbf{T}(Athlete) \sqsubseteq \neg Fat$
$\mathbf{T}(SumoWrestler) \sqsubseteq Fat$

Reasoning

- ABox:
  - $Athlete(paul), SumoWrestler(paul)$
- Expected conclusions:
  - $Fat(paul)$

The logic $\mathcal{ALC} + \mathbf{T}_{\mathbf{R}}^{\mathbf{RaCl}}$

Example

$SumoWrestler \sqsubseteq Athlete$
$\mathbf{T}(Athlete) \sqsubseteq \neg Fat$
$\mathbf{T}(SumoWrestler) \sqsubseteq Fat$

Reasoning

- ABox:
  - $Athlete(paul), SumoWrestler(paul)$
- Expected conclusions:
  - $Fat(paul)$

The logic $\mathcal{ALC} + \mathbf{T}$

Semantics

- $\mathcal{M} = \langle \Delta, <, .^{\mathcal{I}} \rangle$
  - Δ is the domain
  - for each concept C, $C^{\mathcal{I}} \subseteq \Delta$, and for each role R $R^{\mathcal{I}} \subseteq \Delta \times \Delta$
  - $<$ is an irreflexive, transitive and well-founded relation over Δ:
    - for all $S \subseteq \Delta$, for all $x \in S$, either $x \in Min_<(S)$ or $\exists y \in Min_<(S)$ such that $y < x$
    - $Min_<(S) = \{u : u \in S \text{ and } \nexists z \in S \text{ s.t. } z < u\}$
  - $.^{\mathcal{I}}$ extended to complex concepts, e.g. $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
  - Semantics of the $\mathbf{T}$ operator: $(\mathbf{T}(C))^{\mathcal{I}} = Min_<(C^{\mathcal{I}})$

The logic $\mathcal{ALC} + \mathbf{T}$

Semantics

- $\mathcal{M} = \langle \Delta, <, .^{\mathcal{I}} \rangle$
  - Δ is the domain
  - for each concept C, $C^{\mathcal{I}} \subseteq \Delta$, and for each role R $R^{\mathcal{I}} \subseteq \Delta \times \Delta$
  - $<$ is an irreflexive, transitive and well-founded relation over Δ:
    - for all $S \subseteq \Delta$, for all $x \in S$, either $x \in Min_<(S)$ or $\exists y \in Min_<(S)$ such that $y < x$
    - $Min_<(S) = \{u : u \in S \text{ and } \nexists z \in S \text{ s.t. } z < u\}$
  - $.^{\mathcal{I}}$ extended to complex concepts, e.g. $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
  - Semantics of the $\mathbf{T}$ operator: $(\mathbf{T}(C))^{\mathcal{I}} = Min_<(C^{\mathcal{I}})$

The logic $\mathcal{ALC} + \mathbf{T}$

Semantics

- $\mathcal{M} = \langle \Delta, <, .^{\mathcal{I}} \rangle$
  - Δ is the domain
  - for each concept C, $C^{\mathcal{I}} \subseteq \Delta$, and for each role R $R^{\mathcal{I}} \subseteq \Delta \times \Delta$
  - $<$ is an irreflexive, transitive and well-founded relation over Δ:
    - for all $S \subseteq \Delta$, for all $x \in S$, either $x \in Min_{<}(S)$ or $\exists y \in Min_{<}(S)$ such that $y < x$
    - $Min_{<}(S) = \{u : u \in S \text{ and } \nexists z \in S \text{ s.t. } z < u\}$
  - $.^{\mathcal{I}}$ extended to complex concepts, e.g. $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
  - Semantics of the $\mathbf{T}$ operator: $(\mathbf{T}(C))^{\mathcal{I}} = Min_{<}(C^{\mathcal{I}})$

The logic $\mathcal{ALC} + \mathbf{T}$

Semantics

- $\mathcal{M} = \langle \Delta, <, .^{\mathcal{I}} \rangle$
  - Δ is the domain
  - for each concept C, $C^{\mathcal{I}} \subseteq \Delta$, and for each role R $R^{\mathcal{I}} \subseteq \Delta \times \Delta$
  - $<$ is an irreflexive, transitive and well-founded relation over Δ:
    - for all $S \subseteq \Delta$, for all $x \in S$, either $x \in Min_<(S)$ or $\exists y \in Min_<(S)$ such that $y < x$
    - $Min_<(S) = \{u : u \in S \text{ and } \nexists z \in S \text{ s.t. } z < u\}$
  - $.^{\mathcal{I}}$ extended to complex concepts, e.g. $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
  - Semantics of the $\mathbf{T}$ operator: $(\mathbf{T}(C))^{\mathcal{I}} = Min_<(C^{\mathcal{I}})$

The logic $\mathcal{ALC} + \mathbf{T}$

Semantics

- $\mathcal{M} = \langle \Delta, <, .^{\mathcal{I}} \rangle$
  - Δ is the domain
  - for each concept C, $C^{\mathcal{I}} \subseteq \Delta$, and for each role R $R^{\mathcal{I}} \subseteq \Delta \times \Delta$
  - $<$ is an irreflexive, transitive and well-founded relation over Δ:
    - for all $S \subseteq \Delta$, for all $x \in S$, either $x \in Min_<(S)$ or $\exists y \in Min_<(S)$ such that $y < x$
    - $Min_<(S) = \{u : u \in S \text{ and } \nexists z \in S \text{ s.t. } z < u\}$
  - $.^{\mathcal{I}}$ extended to complex concepts, e.g. $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
  - Semantics of the $\mathbf{T}$ operator: $(\mathbf{T}(C))^{\mathcal{I}} = Min_<(C^{\mathcal{I}})$

Weakness of monotonic semantics

Logic $\mathcal{ALC} + \mathbf{T}$

- The operator $\mathbf{T}$ is nonmonotonic, but...
- The logic is monotonic
  - If KB $\models F$, then KB' $\models F$ for all KB' $\supseteq$ KB

Example

- in the KB of the previous slides:
  - if $Athlete(paul) \in$ ABox, we are not able to:

Weakness of monotonic semantics

Logic $\mathcal{ALC} + \mathbf{T}$

- The operator $\mathbf{T}$ is nonmonotonic, but...
- The logic is monotonic
  - If KB $\models F$, then KB' $\models F$ for all KB' $\supseteq$ KB

Example

- in the KB of the previous slides:
  - if $Athlete(paul) \in$ ABox, we are not able to:
    - assume that $\mathbf{T}(Athlete)(paul)$
    - infer that $\neg Fat(paul)$

Weakness of monotonic semantics

Logic $\mathcal{ALC} + \mathbf{T}$

- The operator **T** is nonmonotonic, but...
- The logic is monotonic
  - If KB $\models F$, then KB' $\models F$ for all KB' $\supseteq$ KB

Example

- in the KB of the previous slides:
  - if $Athlete(paul) \in$ ABox, we are not able to:
    - assume that $\mathbf{T}(Athlete)(paul)$
    - infer that $\neg Fat(paul)$

Weakness of monotonic semantics

Logic $\mathcal{ALC} + \mathbf{T}$

- The operator $\mathbf{T}$ is nonmonotonic, but...
- The logic is monotonic
  - If KB $\models F$, then KB' $\models F$ for all KB' $\supseteq$ KB

Example

- in the KB of the previous slides:
  - if $Athlete(paul) \in$ ABox, we are not able to:
    - assume that $\mathbf{T}(Athlete)(paul)$
    - infer that $\neg Fat(paul)$

The nonmonotonic logic $\mathcal{ALC} + \mathbf{T}_{\mathsf{R}}^{\mathsf{RaCl}}$

Minimal entailment

- Preference relation among models of a KB
  - $\mathcal{M}_1 < \mathcal{M}_2$ if $\mathcal{M}_1$ contains less exceptional (not minimal) elements
  - $\mathcal{M}$ minimal model of KB if there is no $\mathcal{M}'$ model of KB such that $\mathcal{M}' < \mathcal{M}$
- Minimal entailment
  - KB $\models_{min} F$ if F holds in all *minimal* models of KB
- Nonmonotonic logic
  - KB $\models_{min} F$ does not imply KB' $\models_{min} F$ with KB' $\supset$ KB

The nonmonotonic logic $\mathcal{ALC} + \mathbf{T}_{\mathbf{R}}^{\mathbf{RaCl}}$

Minimal entailment

- Preference relation among models of a KB
  - $\mathcal{M}_1 < \mathcal{M}_2$ if $\mathcal{M}_1$ contains less exceptional (not minimal) elements
  - $\mathcal{M}$ minimal model of KB if there is no $\mathcal{M}'$ model of KB such that $\mathcal{M}' < \mathcal{M}$
- Minimal entailment
  - KB $\models_{min}$ F if F holds in all *minimal* models of KB
- Nonmonotonic logic
  - KB $\models_{min}$ F does not imply KB' $\models_{min}$ F with KB' $\supset$ KB

The nonmonotonic logic $\mathcal{ALC} + \mathbf{T}_{\mathbf{R}}^{\mathbf{RaCl}}$

Minimal entailment

- Preference relation among models of a KB
  - $\mathcal{M}_1 < \mathcal{M}_2$ if $\mathcal{M}_1$ contains less exceptional (not minimal) elements
  - $\mathcal{M}$ minimal model of KB if there is no $\mathcal{M}'$ model of KB such that $\mathcal{M}' < \mathcal{M}$
- Minimal entailment
  - KB $\models_{min} F$ if F holds in all *minimal* models of KB
- Nonmonotonic logic
  - KB $\models_{min} F$ does not imply KB' $\models_{min} F$ with KB' $\supset$ KB

The nonmonotonic logic $\mathcal{ALC} + \mathbf{T}_{\mathsf{R}}^{\mathsf{RaCl}}$

Minimal entailment

- Satisfiability of a KB $\rightarrow$ satisfiability of a constraint system $\langle S \mid U \rangle$
  - $S = \{a : C \mid C(a) \in \text{ABox}\} \cup \{a \xrightarrow{R} b \mid R(a, b) \in \text{ABox}\}$
  - $U = \{C \sqsubseteq D^{\emptyset} \mid C \sqsubseteq D \in \text{TBox}\}$
- In order to check whether F is entailed from KB:
  - step 1: check the satisfiability of KB $\cup \{\neg F\}$
  - step 2: check whether each open branch B built by step 1 represents a minimal model of the KB

The nonmonotonic logic $\mathcal{ALC} + \mathbf{T}_{\mathbf{R}}^{\mathsf{RaCl}}$

Minimal entailment

- Satisfiability of a KB $\rightarrow$ satisfiability of a constraint system $\langle S \mid U \rangle$
  - $S = \{a : C \mid C(a) \in \text{ABox}\} \cup \{a \xrightarrow{R} b \mid R(a, b) \in \text{ABox}\}$
  - $U = \{C \sqsubseteq D^{\emptyset} \mid C \sqsubseteq D \in \text{TBox}\}$
- In order to check whether F is entailed from KB:
  - step 1: check the satisfiability of KB $\cup \{\neg F\}$
  - step 2: check whether each open branch **B** built by step 1 represents a minimal model of the KB

The nonmonotonic logic $\mathcal{ALC} + \mathbf{T}_{\mathbf{R}}^{\mathbf{RaCl}}$

Minimal entailment

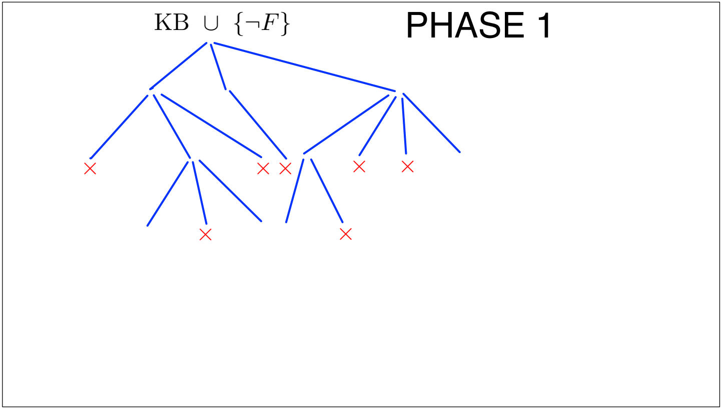
- Satisfiability of a KB $\rightarrow$ satisfiability of a constraint system $\langle S \mid U \rangle$
  - $S = \{a : C \mid C(a) \in \text{ABox}\} \cup \{a \xrightarrow{R} b \mid R(a, b) \in \text{ABox}\}$
  - $U = \{C \sqsubseteq D^{\emptyset} \mid C \sqsubseteq D \in \text{TBox}\}$
- In order to check whether F is entailed from KB:
  - step 1: check the satisfiability of KB $\cup \{\neg F\}$
  - step 2: check whether each open branch **B** built by step 1 represents a minimal model of the KB

The nonmonotonic logic $\mathcal{ALC} + \mathbf{T}_{\mathbf{R}}^{\mathsf{RaCl}}$

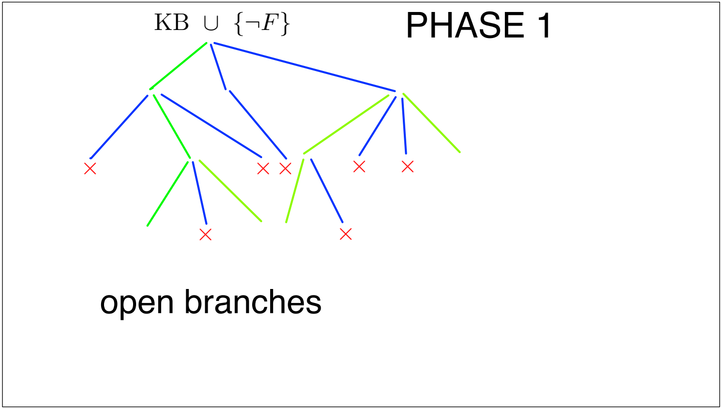
Minimal entailment

- Satisfiability of a KB $\rightarrow$ satisfiability of a constraint system $\langle S \mid U \rangle$
  - $S = \{a : C \mid C(a) \in \text{ABox}\} \cup \{a \xrightarrow{R} b \mid R(a,b) \in \text{ABox}\}$
  - $U = \{C \sqsubseteq D^{\emptyset} \mid C \sqsubseteq D \in \text{TBox}\}$
- In order to check whether F is entailed from KB:
  - step 1: check the satisfiability of KB $\cup \{\neg F\}$
  - step 2: check whether each open branch **B** built by step 1 represents a minimal model of the KB

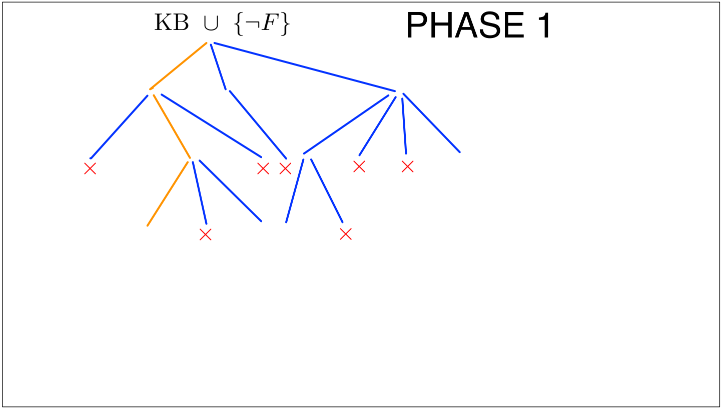
The calculus



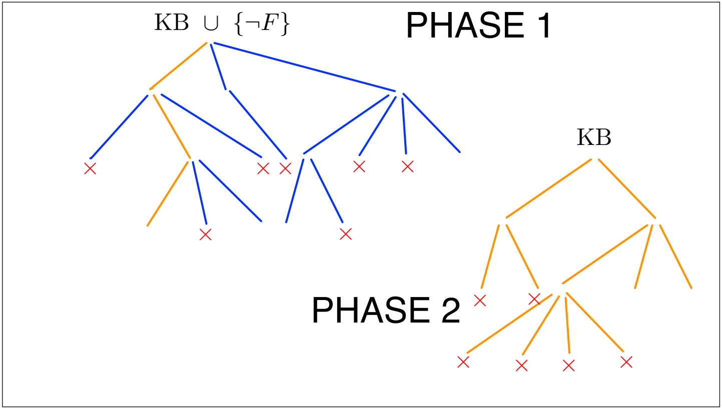
The calculus



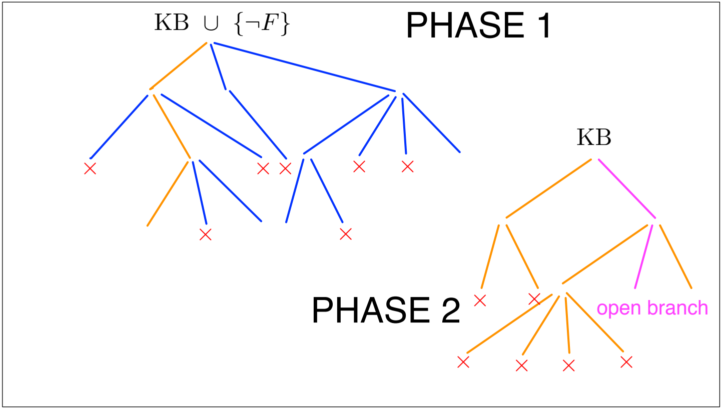
The calculus



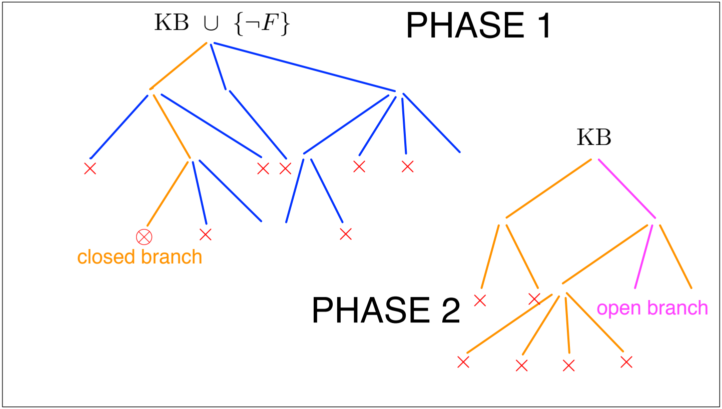
The calculus



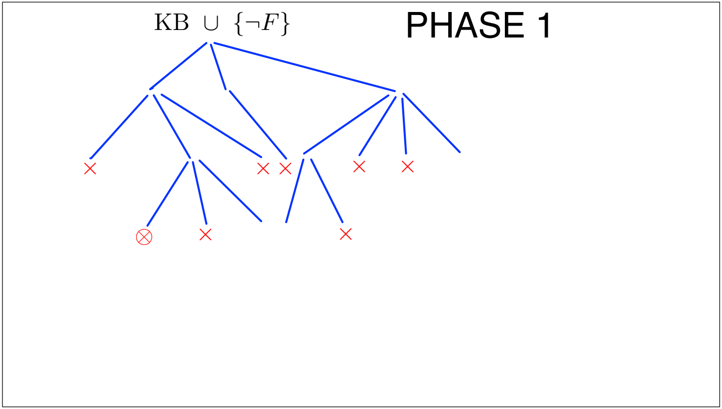
The calculus



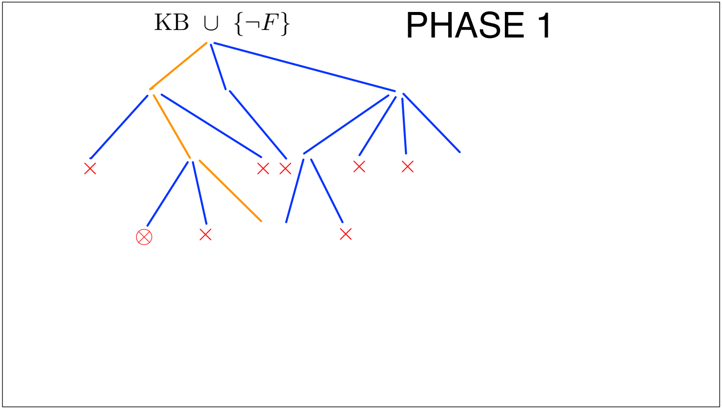
The calculus



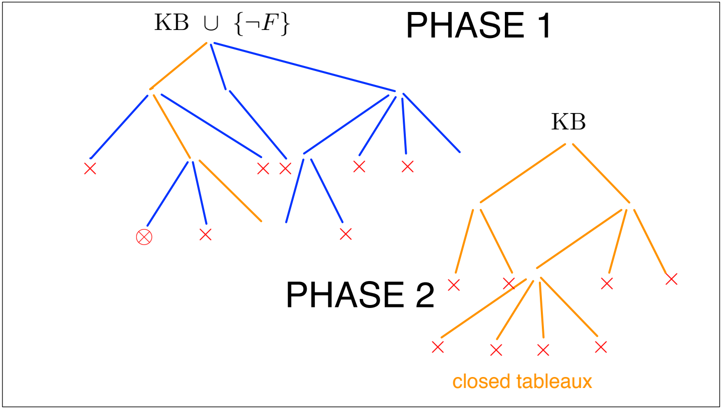
The calculus



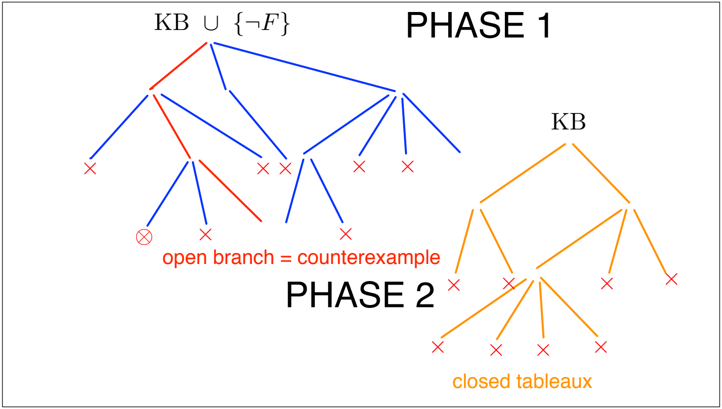
The calculus



The calculus



The calculus



The calculus

Box formulas

- Idea: semantics of $\mathbf{T}$ specified by modal logic
  - interpretation of $\mathbf{T}$ split into two parts $= x \in (\mathbf{T}(C))^{\mathcal{I}}$:
    1. $x \in C^{\mathcal{I}}$
    2. there is no $y \in C^{\mathcal{I}}$ such that $y < x$
- Condition 2 can be represented by an additional modality $\Box$, whose semantics is given by the preference relation $<$ interpreted as an accessibility relation: $(\Box C)^{\mathcal{I}} = \{x \in \Delta \mid \text{for every } y \in \Delta, \text{ if } y < x \text{ then } y \in C^{\mathcal{I}}\}$
- we get $x \in (\mathbf{T}(C))^{\mathcal{I}}$ if and only if $x \in (C \sqcap \Box\neg C)^{\mathcal{I}}$

The calculus: step 1

$$\langle S, x : C, x : \neg C \mid U \rangle \ \text{(Clash)} \qquad \langle S, x : \neg\top \mid U \rangle \ \text{(Clash)}_\top \qquad \langle S, x : \bot \mid U \rangle \ \text{(Clash)}_\bot$$

$$\frac{\langle S, x : \neg(C \sqcap D) \mid U \rangle}{\langle S, x : \neg(C \sqcap D), x : \neg C \mid U \rangle \qquad \langle S, x : \neg(C \sqcap D), x : \neg D \mid U \rangle}\ (\sqcap^-) \quad \text{if } x : \neg C \notin S \text{ and } x : \neg D \notin S$$

$$\frac{\langle S, x : C \sqcap D \mid U \rangle}{\langle S, x : C \sqcap D, x : C, x : D \mid U \rangle}\ (\sqcap^+) \quad \text{if } \{x : C, x : D\} \not\subseteq S$$

$$\frac{\langle S, x : C \sqcup D \mid U \rangle}{\langle S, x : C \sqcup D, x : C \mid U \rangle \qquad \langle S, x : C \sqcup D, x : D \mid U \rangle}\ (\sqcup^+) \quad \text{if } x : C \notin S \text{ and } x : D \notin S$$

$$\frac{\langle S, x : \neg(C \sqcup D) \mid U \rangle}{\langle S, x : \neg(C \sqcup D), x : \neg C, x : \neg D \mid U \rangle}\ (\sqcup^-) \quad \text{if } \{x : \neg C, x : \neg D\} \not\subseteq S$$

$$\frac{\langle S, x : \neg\neg C \mid U \rangle}{\langle S, x : \neg\neg C, x : C \mid U \rangle}\ (\neg) \quad \text{if } x : C \notin S$$

$$\frac{\langle S, x : \mathbf{T}(C) \mid U \rangle}{\langle S, x : \mathbf{T}(C), x : C, x : \Box\neg C \mid U \rangle}\ (\mathbf{T}^+) \quad \text{if } \{x : C, x : \Box\neg C\} \not\subseteq S$$

$$\frac{\langle S, x : \neg\mathbf{T}(C) \mid U \rangle}{\langle S, x : \neg\mathbf{T}(C), x : \neg C \mid U \rangle \qquad \langle S, x : \neg\mathbf{T}(C), x : \neg\Box\neg C \mid U \rangle}\ (\mathbf{T}^-) \quad \text{if } x : \neg C \notin S \text{ and } x : \neg\Box\neg C \notin S$$

$$\frac{\langle S \mid U \rangle}{\langle S, x : \Box\neg C \mid U \rangle \qquad \langle S, x : \neg\Box\neg C \mid U \rangle}\ (cut) \quad \begin{array}{r}\text{if } x : \neg\Box\neg C \notin S \text{ and } x : \Box\neg C \notin S \\ C \in \mathcal{L}_{\mathbf{T}} \\ x \text{ occurs in } S\end{array}$$

$$\frac{\langle S \mid U, C \sqsubseteq D^L \rangle}{\langle S, x : \neg C \sqcup D \mid U, C \sqsubseteq D^{L,x} \rangle}\ (\sqsubseteq) \quad \text{if } x \text{ occurs in } S \text{ and } x \notin L$$

$$\frac{\langle S, x : \forall R.C, x \xrightarrow{R} y \mid U \rangle}{\langle S, x : \forall R.C, x \xrightarrow{R} y, y : C \mid U \rangle}\ (\forall^+) \quad \text{if } y : C \notin S$$

$$\frac{\langle S, x : \exists R.C \mid U \rangle}{\langle S, x : \exists R.C, x \xrightarrow{R} y, y : C \mid U \rangle \quad \langle S, x : \exists R.C, x \xrightarrow{R} v_1, v_1 : C \mid U \rangle \quad \langle S, x : \exists R.C, x \xrightarrow{R} v_2, v_2 : C \mid U \rangle \ \cdots \ \langle S, x : \exists R.C, x \xrightarrow{R} v_n, v_n : C \mid U \rangle}\ (\exists^+)$$

$$\begin{array}{r} y \text{ new} \\ \text{if } \nexists z \prec x \text{ s.t. } z \equiv_{S, x : \exists R.C} x \text{ and } \nexists u \text{ s.t. } x \xrightarrow{R} u \in S \text{ and } u : C \in S \\ \forall v_i \text{ occurring in } S \end{array}$$

$$\frac{\langle S, x : \neg\Box\neg C \mid U \rangle}{\langle S, x : \neg\Box\neg C, y < x, y : C, y : \Box\neg C, S^M_{x \to y} \mid U \rangle \ \langle S, x : \neg\Box\neg C, v_1 < x, v_1 : C, v_1 : \Box\neg C, S^M_{x \to v_1} \mid U \rangle \ \cdots \ \langle S, x : \neg\Box\neg C, v_n < x, v_n : C, v_n : \Box\neg C, S^M_{x \to v_n} \mid U \rangle}\ (\Box^-)$$

$$\begin{array}{r} y \text{ new} \\ \text{if } \nexists z \prec x \text{ s.t. } z \equiv_{S, x : \neg\Box\neg C} x \text{ and } \nexists u \text{ s.t. } \{u < x, u : C, u : \Box\neg C, S^M_{x \to u}\} \subseteq S \\ \forall v_i \text{ occurring in } S, x \neq v_i \end{array}$$

The calculus: step 2

$$\langle S, x : C, x : \neg C \mid U \mid K\rangle \ (\text{Clash}) \qquad \langle S, x : \bot \mid U \mid K\rangle \ (\text{Clash})_{\bot} \qquad \langle S, x : \neg\top \mid U \mid K\rangle \ (\text{Clash})_{\top} \qquad \langle S \mid U \mid \emptyset\rangle \ (\text{Clash})_{\emptyset}$$

$$\langle S, x : \neg\Box\neg C \mid U \mid K\rangle \ (\text{Clash})_{\Box^-} \quad \text{if } x : \neg\Box\neg C \notin \mathbf{B}^{\Box^-}$$

$$\frac{\langle S, x : \mathbf{T}(C) \mid U \mid K\rangle}{\langle S, x : C, x : \Box\neg C \mid U \mid K\rangle}\ (\mathbf{T}^+)$$

$$\frac{\langle S, x : \neg\mathbf{T}(C) \mid U \mid K\rangle}{\langle S, x : \neg C \mid U \mid K\rangle \qquad \langle S, x : \neg\Box\neg C \mid U \mid K\rangle}\ (\mathbf{T}^-)$$

$$\frac{\langle S \mid U, C \sqsubseteq D^L \mid K\rangle}{\langle S, x : \neg C \sqcup D \mid U, C \sqsubseteq D^{L,x} \mid K\rangle}\ (\sqsubseteq) \quad x \in \mathcal{D}(\mathbf{B}) \text{ and } x \notin L$$

$$\frac{\langle S, x : \forall R.C, x \xrightarrow{R} y \mid U \mid K\rangle}{\langle S, x : \forall R.C, x \xrightarrow{R} y, y : C \mid U \mid K\rangle}\ (\forall^+) \quad \text{if } y : C \notin S$$

$$\frac{\langle S, x : \exists R.C \mid U \mid K\rangle}{\langle S, x \xrightarrow{R} v_1, v_1 : C \mid U \mid K\rangle \quad \langle S, x \xrightarrow{R} v_2, v_2 : C \mid U \mid K\rangle \cdots \langle S, x \xrightarrow{R} v_n, v_n : C \mid U \mid K\rangle}\ (\exists^+) \quad \forall v_i \in \mathcal{D}(\mathbf{B})$$

$$\frac{\langle S \mid U \mid K\rangle}{\langle S, x : \Box\neg C \mid U \mid K\rangle \qquad \langle S, x : \neg\Box\neg C \mid U \mid K\rangle}\ (cut) \quad \text{if } x : \neg\Box\neg C \notin S \text{ and } x : \Box\neg C \notin S \quad x \in \mathcal{D}(\mathbf{B}) \quad C \in \mathcal{L}_{\mathbf{T}}$$

$$\frac{\langle S, x : \neg\Box\neg C \mid U \mid K, x : \neg\Box\neg C\rangle}{\langle S, v_1 : C, v_1 : \Box\neg C, S^M_{x\to v_1}, x : \neg\Box\neg C \mid U \mid K\rangle \quad \langle S, v_2 : C, v_2 : \Box\neg C, S^M_{x\to v_2}, x : \neg\Box\neg C \mid U \mid K\rangle \quad \cdots \quad \langle S, v_n : C, v_n : \Box\neg C, S^M_{x\to v_n}, x : \neg\Box\neg C \mid U \mid K\rangle}\ (\Box^-)$$

$$\text{if } \nexists u \text{ s.t. } \{u : C, u : \Box\neg C, S^M_{x\to u}\} \subseteq S \qquad \forall v_i \in \mathcal{D}(\mathbf{B}), x \neq v_i$$

DysToPic (with Luca Violanti)

Minimal entailment

- multi-engine theorem prover for reasoning in $\mathcal{ALC} + \mathbf{T}_{min}$
- SICStus Prolog implementation of the two-steps tableaux calculi wrapped by a Java interface which relies on the Java RMI APIs for the distribution of the computation
- "worker/employer" paradigm: the computational burden for the "employer" can be spread among an arbitrarily high number of "workers" which operate in complete autonomy, so that they can be either deployed on a single machine or on a computer grid

DysToPic

Minimal entailment

- Basic idea: no need for step 1 to wait for the result of one elaboration of step 2 on an open branch, before generating another candidate branch
    - step 1 can be executed on a machine
    - every time that a branch remains open after step 1, the execution of step 2 for this branch is performed in parallel, on a different machine
    - Meanwhile, the worker can carry on with the computation of step 1 potentially generating other branches
- if a branch remains open after step 2, then F is not minimally entailed from KB, so the computation process can be interrupted early

DysToPic

Minimal entailment

- Basic idea: no need for step 1 to wait for the result of one elaboration of step 2 on an open branch, before generating another candidate branch
  - step 1 can be executed on a machine
  - every time that a branch remains open after step 1, the execution of step 2 for this branch is performed in parallel, on a different machine
  - Meanwhile, the worker can carry on with the computation of step 1 potentially generating other branches
- if a branch remains open after step 2, then F is not minimally entailed from KB, so the computation process can be interrupted early

DysToPic

Minimal entailment

- Basic idea: no need for step 1 to wait for the result of one elaboration of step 2 on an open branch, before generating another candidate branch
  - step 1 can be executed on a machine
  - every time that a branch remains open after step 1, the execution of step 2 for this branch is performed in parallel, on a different machine
  - Meanwhile, the worker can carry on with the computation of step 1 potentially generating other branches
- if a branch remains open after step 2, then F is not minimally entailed from KB, so the computation process can be interrupted early

Conclusions

Future issues

- Currently working on the implementation of hypersequent calculi for other logics of the Lewis' family (with Nicola, Marianna, Bjoern)
- Currently fixing a Prótégé plugin for reasoning in nonmonotonic DLs
- Alternative semantics for nonmonotonic extensions of DLs
  - Need of implementing reasoners for these extensions

Conclusions

Future issues

- Currently working on the implementation of hypersequent calculi for other logics of the Lewis' family (with Nicola, Marianna, Bjoern)
- Currently fixing a Prótégé plugin for reasoning in nonmonotonic DLs
- Alternative semantics for nonmonotonic extensions of DLs
  - Need of implementing reasoners for these extensions

Conclusions

Future issues

- Currently working on the implementation of hypersequent calculi for other logics of the Lewis' family (with Nicola, Marianna, Bjoern)
- Currently fixing a Prótégé plugin for reasoning in nonmonotonic DLs
- Alternative semantics for nonmonotonic extensions of DLs
  - Need of implementing reasoners for these extensions

Conditional logics

References (1)

- A. Artosi, G. Governatori, and A. Rotolo (2002), Labelled tableaux for non-monotonic reasoning: Cumulative consequence relations. *Journal of Logic and Computation*, 12, 1027–1060.
- R. Alenda, N. Olivetti, and G. L. Pozzato. (2013), Nested Sequents Calculi for Normal Conditional Logics. *Journal of Logic and Computation*, to appear.
- O. Board (2004). Dynamic interactive epistemology. *Games and Economic Behavior*, 49(1):49–80.
- C. Boutilier (1994). Conditional logics of normality: a modal approach. *Artificial Intelligence*, 68(1):87–154.
- A. Baltag and S. Smets (2008). The logic of conditional doxastic actions. *Texts in Logic and Games, Special Issue on New Perspectives on Games and Interaction*, 4:9–31.
- H. C. M. de Swart (1983), A gentzen- or beth-type system, a practical decision procedure and a constructive completeness proof for the counterfactual logics $\mathbb{VC}$ and $\mathbb{VCS}$. *Journal of Symbolic Logic*, 48, 1–20.
- I. P. Gent (1992),A sequent or tableaux-style system for lewis's counterfactual logic $\mathbb{VC}$. *Notre Dame Journal of Formal Logic*, 33(3):369–382.
- L. Giordano, V. Gliozzi, N. Olivetti, and C.B. Schwind (2005), Weak AGM postulates and strong ramsey test: A logical formalization. *Artificial Intelligence*, 168(1-2):1–37.
- L. Giordano, V. Gliozzi, N. Olivetti, and C.B. Schwind (2009), Tableau calculus for preference-based conditional logics: PCL and its extensions. *ACM Trans. Comput. Logic*, 10.
- M. L. Ginsberg (1986), Counterfactuals. *Artificial Intelligence*, 30(1):35–79.

Conditional logics

References (2)

- G. Grahne (1998), Updates and counterfactuals. *Journal of Logic and Computation*, 8(1):87–117.
- S. Kraus, D. Lehmann, and M. Magidor (1990), Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1-2):167–207.
- D. Lewis (1973), Counterfactuals. *Basil Blackwell Ltd*.
- D. Nute (1980), Topics in conditional logic. *Reidel, Dordrecht*.
- N. Olivetti and G. L. Pozzato (2014), Nescond: An implementation of nested sequent calculi for conditional logics. *Proceedings of IJCAR 2014*, LNAI 8562, pages 511–518.
- N. Olivetti, G. L. Pozzato, and C. B. Schwind (2007), A Sequent Calculus and a Theorem Prover for Standard Conditional Logics. *ACM ToCL*, 8(4).
- D. Pattinson and L. Schröder (2011), Generic modal cut elimination applied to conditional logics. *Logical Methods in Computer Science*, 7.
- L. Schröder, D. Pattinson, and D. Hausmann (2010), Optimal tableaux for conditional logics with cautious monotonicity. In *ECAI*, 707–712.

Thank you!!!!!