Defeasible Reasoning over Facts and Norms

Emery Neufeld

TU WIEN Informatics

What are Norms?

Norms

We can represent norms **explicitly** or **implicitly**.

What are Norms?

Norms

We can represent norms **explicitly** or **implicitly**.

- Represent norms explicitly as rules.

What are Norms?

Norms

We can represent norms **explicitly** or **implicitly**.

- Represent norms explicitly as rules.
- E.g., "*p* is obligatory"

What are Norms?

Norms

We can represent norms **explicitly** or **implicitly**.

- Represent norms explicitly as rules.
- E.g., "p is obligatory"

- Represent norms implicitly as descriptions.

What are Norms?

Norms

We can represent norms **explicitly** or **implicitly**.

- Represent norms explicitly as rules.
- E.g., "*p* is obligatory"

- Represent norms implicitly as descriptions.
- E.g., "always *p*"

What are Norms?

Norms

We can represent norms **explicitly** or **implicitly**.

- Represent norms explicitly as rules.
- E.g., “*p* is obligatory”

- Represent norms implicitly as descriptions.
- E.g., “always *p*”

Norms can be...

What are Norms?

Norms

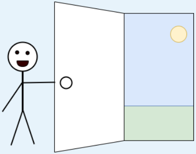
We can represent norms **explicitly** or **implicitly**.

- Represent norms explicitly as rules.
- E.g., "*p* is obligatory"

- Represent norms implicitly as descriptions.
- E.g., "always *p*"

Norms can be...

Social



What are Norms?

Norms

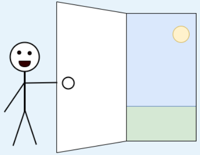
We can represent norms **explicitly** or **implicitly**.

- Represent norms explicitly as rules.
- E.g., “*p* is obligatory”

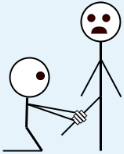
- Represent norms implicitly as descriptions.
- E.g., “always *p*”

Norms can be...

Social



Ethical



What are Norms?

Norms

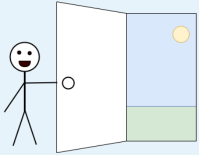
We can represent norms **explicitly** or **implicitly**.

- Represent norms explicitly as rules.
- E.g., “*p* is obligatory”

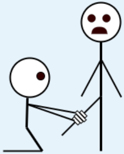
- Represent norms implicitly as descriptions.
- E.g., “always *p*”

Norms can be...

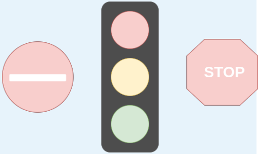
Social



Ethical



Legal



Types of Norms

Norms can be...

Types of Norms

Norms can be...

Regulative Norms

Regulative norms can be:

- Obligations $\mathbf{O}(p|q)$
  - “On Sunday, you ought to bake a cake."

Types of Norms

Norms can be...

Regulative Norms

Regulative norms can be:

- Obligations $\mathbf{O}(p|q)$
  - “On Sunday, you ought to bake a cake."
- Prohibitions $\mathbf{F}(p|q) \equiv \mathbf{O}(\neg p|q)$
  - “You are forbidden from eating cake.”

Types of Norms

Norms can be...

Regulative Norms

Regulative norms can be:

- Obligations $\mathbf{O}(p|q)$
  - “On Sunday, you ought to bake a cake."
- Prohibitions $\mathbf{F}(p|q) \equiv \mathbf{O}(\neg p|q)$
  - “You are forbidden from eating cake.”
- (Weak) Permissions $\mathbf{P}_w(p|q) \equiv \neg\mathbf{O}(\neg p|q)$
  - “On Tuesday, you are permitted to eat carrot cake.”

Types of Norms

Norms can be...

Regulative Norms

Regulative norms can be:

- Obligations $\mathbf{O}(p|q)$
  - “On Sunday, you ought to bake a cake."
- Prohibitions $\mathbf{F}(p|q) \equiv \mathbf{O}(\neg p|q)$
  - “You are forbidden from eating cake.”
- (Weak) Permissions $\mathbf{P}_w(p|q) \equiv \neg\mathbf{O}(\neg p|q)$
  - “On Tuesday, you are permitted to eat carrot cake.”
  - **Conflict!**

Types of Norms

Norms can be...

Regulative Norms

Regulative norms can be:

- Obligations $\mathbf{O}(p|q)$
  - “On Sunday, you ought to bake a cake."
- Prohibitions $\mathbf{F}(p|q) \equiv \mathbf{O}(\neg p|q)$
  - “You are forbidden from eating cake.”
- (Weak) Permissions $\mathbf{P}_w(p|q) \equiv \neg\mathbf{O}(\neg p|q)$
  - “On Tuesday, you are permitted to eat carrot cake.”
  - **Conflict!**
- (Strong) Permissions $\mathbf{P}_s(p)$
  - “On Tuesday, you are permitted to eat carrot cake.”
  - No conflict :)

Types of Norms

Norms can be...

Regulative Norms

Regulative norms can be:

- Obligations $\mathbf{O}(p|q)$
  - “On Sunday, you ought to bake a cake."
- Prohibitions $\mathbf{F}(p|q) \equiv \mathbf{O}(\neg p|q)$
  - “You are forbidden from eating cake.”
- (Weak) Permissions $\mathbf{P}_w(p|q) \equiv \neg\mathbf{O}(\neg p|q)$
  - “On Tuesday, you are permitted to eat carrot cake.”
  - **Conflict!**
- (Strong) Permissions $\mathbf{P}_s(p)$
  - “On Tuesday, you are permitted to eat carrot cake.”
  - No conflict :)

Constitutive Norms

Constitutive norms are:

- “in context C, X counts as Y"
  - $\mathbf{C}(X, Y|C)$
  - Used to define new concepts.
  - “Eating carrot cake counts as eating cake”.

Normative Systems

Regulative Norms + Constitutive norms = **Normative System**

Normative Systems

Regulative Norms + Constitutive norms = **Normative System**

Normative System Example

- You are forbidden from eating cake: $\mathbf{F}(cake|\top)$
- You are permitted to eat carrot cake on Tuesday: $\mathbf{P}(carrot|tuesday)$
- Eating carrot cake counts as eating cake: $\mathbf{C}(carrot, cake|\top)$

Normative Systems

Regulative Norms + Constitutive norms = **Normative System**

Normative System Example

- You are forbidden from eating cake: $\mathbf{F}(cake|\top)$
- You are permitted to eat carrot cake on Tuesday: $\mathbf{P}(carrot|tuesday)$
- Eating carrot cake counts as eating cake: $\mathbf{C}(carrot, cake|\top)$
- Correct conclusion: you can only eat (carrot) cake on Tuesdays.

Compliance

Factual Detachment

$$\mathbf{O}(p|q) \wedge q \implies \mathbf{O}(p)$$

Compliance to Obligations

- Compliance := not violated
- An obligation $\mathbf{O}(p)$ is violated when $\mathbf{O}(p)$ is true but p is not.

Compliance

Factual Detachment

$$\mathbf{O}(p|q) \wedge q \implies \mathbf{O}(p)$$

Compliance to Obligations

- Compliance := not violated
- An obligation $\mathbf{O}(p)$ is violated when $\mathbf{O}(p)$ is true but p is not.

Compliance to Normative Systems

- Compliance := no violations.
- Violation of normative system: for some p, $\mathbf{O}(p)$ is true but p is not.

Compliance

Factual Detachment

$$\mathbf{O}(p|q) \wedge q \implies \mathbf{O}(p)$$

Compliance to Obligations

- Compliance := not violated
- An obligation $\mathbf{O}(p)$ is violated when $\mathbf{O}(p)$ is true but p is not.

Compliance to Normative Systems

- Compliance := no violations.
- Violation of normative system: for some p, $\mathbf{O}(p)$ is true but p is not.

Permissions cannot be violated!

Challenges Associated with Reasoning about Norms

- Strong Permissions

Challenges Associated with Reasoning about Norms

- Strong Permissions
  - Act as **exceptions** to obligations or prohibitions.

Challenges Associated with Reasoning about Norms

- Strong Permissions
  - Act as **exceptions** to obligations or prohibitions.
  - Have no other function; cannot constrain, cannot be violated.

Challenges Associated with Reasoning about Norms

- Strong Permissions
  - Act as **exceptions** to obligations or prohibitions.
  - Have no other function; cannot constrain, cannot be violated.
- Conflict and Priorities

Challenges Associated with Reasoning about Norms

- Strong Permissions
  - Act as **exceptions** to obligations or prohibitions.
  - Have no other function; cannot constrain, cannot be violated.
- Conflict and Priorities
  - “When driving, you ought not swerve into another lane.”

Challenges Associated with Reasoning about Norms

- Strong Permissions
  - Act as **exceptions** to obligations or prohibitions.
  - Have no other function; cannot constrain, cannot be violated.
- Conflict and Priorities
  - “When driving, you ought not swerve into another lane.”
  - “If it is to avoid a collision, you ought to swerve into another lane.”

Challenges Associated with Reasoning about Norms

- Strong Permissions
  - Act as **exceptions** to obligations or prohibitions.
  - Have no other function; cannot constrain, cannot be violated.
- Conflict and Priorities
  - “When driving, you ought not swerve into another lane.”
  - “If it is to avoid a collision, you ought to swerve into another lane.”
- Normative Deadlock
  - What happens when compliance is not possible?

Challenges Associated with Reasoning about Norms

- Strong Permissions
  - Act as **exceptions** to obligations or prohibitions.
  - Have no other function; cannot constrain, cannot be violated.
- Conflict and Priorities
  - "When driving, you ought not swerve into another lane."
  - "If it is to avoid a collision, you ought to swerve into another lane."
- Normative Deadlock
  - What happens when compliance is not possible?
  - E.g., contrary-to-duty obligations:
    - "You ought not kill."
    - "If you kill, you ought to kill gently."

Monotonicity

Definition

$$\Gamma_1 \vdash \phi \implies \Gamma_1 \cup \Gamma_2 \vdash \phi$$

Monotonicity

Definition

$$\Gamma_1 \vdash \phi \implies \Gamma_1 \cup \Gamma_2 \vdash \phi$$

Example

- Birds can fly. ($bird \rightarrow fly$)
- Sparrows are birds. ($sparrow \rightarrow bird$)

Monotonicity

Definition

$$\Gamma_1 \vdash \phi \implies \Gamma_1 \cup \Gamma_2 \vdash \phi$$

Example

- Birds can fly. ($bird \rightarrow fly$)
- Sparrows are birds. ($sparrow \rightarrow bird$)
- $\{bird \rightarrow fly,\ sparrow \rightarrow bird\} \vdash sparrow \rightarrow fly$

Monotonicity

Definition

$$\Gamma_1 \vdash \phi \implies \Gamma_1 \cup \Gamma_2 \vdash \phi$$

Example

- Birds can fly. ($\mathit{bird} \rightarrow \mathit{fly}$)
- Sparrows are birds. ($\mathit{sparrow} \rightarrow \mathit{bird}$)
- $\{\mathit{bird} \rightarrow \mathit{fly},\ \mathit{sparrow} \rightarrow \mathit{bird}\} \vdash \mathit{sparrow} \rightarrow \mathit{fly}$
- Add: fish cannot fly.
  $\{\mathit{bird} \rightarrow \mathit{fly},\ \mathit{sparrow} \rightarrow \mathit{bird}\} \cup \{\mathit{fish} \rightarrow \neg \mathit{fly}\}$

Monotonicity

Definition

$$\Gamma_1 \vdash \phi \implies \Gamma_1 \cup \Gamma_2 \vdash \phi$$

Example

- Birds can fly. ($\mathit{bird} \rightarrow \mathit{fly}$)
- Sparrows are birds. ($\mathit{sparrow} \rightarrow \mathit{bird}$)
- $\{\mathit{bird} \rightarrow \mathit{fly},\ \mathit{sparrow} \rightarrow \mathit{bird}\} \vdash \mathit{sparrow} \rightarrow \mathit{fly}$
- Add: fish cannot fly.
  $\{\mathit{bird} \rightarrow \mathit{fly},\ \mathit{sparrow} \rightarrow \mathit{bird}\} \cup \{\mathit{fish} \rightarrow \neg \mathit{fly}\} \vdash \mathit{sparrow} \rightarrow \mathit{fly}$

Why Non-monotonicity?

Example

1. Birds can fly (*bird* $\rightarrow$ *fly*)
2. Sparrows are birds (*sparrow* $\rightarrow$ *bird*)

- We can derive: sparrows can fly (*sparrow* $\rightarrow$ *fly*)

Why Non-monotonicity?

Example

1. Birds can fly ($bird \rightarrow fly$)
2. Sparrows are birds ($sparrow \rightarrow bird$)

- We can derive: sparrows can fly ($sparrow \rightarrow fly$)

1. Birds can fly. ($bird \rightarrow fly$)
2. Penguins are birds. ($penguin \rightarrow bird$)

Why Non-monotonicity?

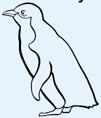
Example

1. Birds can fly (*bird* $\rightarrow$ *fly*)
2. Sparrows are birds (*sparrow* $\rightarrow$ *bird*)

- We can derive: sparrows can fly (*sparrow* $\rightarrow$ *fly*)

1. Birds can fly. (*bird* $\rightarrow$ *fly*)
2. Penguins are birds. (*penguin* $\rightarrow$ *bird*)

- We can derive: penguins can fly??? (*penguin* $\rightarrow$ *fly*)



Non-monotonicity

Example, continued...

1. Birds can fly ($bird \rightarrow fly$)
2. Penguins are birds ($penguin \rightarrow bird$)

Non-monotonicity

Example, continued...

1. Birds can fly ($bird \rightarrow fly$)
2. Penguins are birds ($penguin \rightarrow bird$)
3. **Penguins cannot fly** ($penguin \rightarrow \neg fly$)

Non-monotonicity

Example, continued...

1. Birds can fly (*bird* $\rightarrow$ *fly*)
2. Penguins are birds (*penguin* $\rightarrow$ *bird*)
3. **Penguins cannot fly** (*penguin* $\rightarrow \neg$*fly*)

- Is this necessarily a contradiction?

Defeasible Logic (DL): Rules

Literals

$AP :=$ atomic propositions

$Lit := AP \cup \{\neg p \mid p \in AP\}$

Defeasible Logic (DL): Rules

Literals

$AP :=$ atomic propositions

$Lit := AP \cup \{\neg p \mid p \in AP\}$

Rules

$$r : \; A(r) \hookrightarrow N(r)$$

Defeasible Logic (DL): Rules

Literals

$AP :=$ atomic propositions

$Lit := AP \cup \{\neg p \mid p \in AP\}$

Rules

$$r: \; A(r) \hookrightarrow N(r)$$

- $A(r) = \{a_1, ..., a_n\} \in 2^{Lit}$ is an antecedent.
- $N(r) \in Lit$ is a consequent.

Defeasible Logic (DL): Rules

Literals

$AP :=$ atomic propositions

$Lit := AP \cup \{\neg p \mid p \in AP\}$

Rules

$$r : \; A(r) \hookrightarrow N(r)$$

- $A(r) = \{a_1, ..., a_n\} \in 2^{Lit}$ is an antecedent.
- $N(r) \in Lit$ is a consequent.
- $\hookrightarrow \in \{\rightarrow$

Defeasible Logic (DL): Rules

Literals

$AP :=$ atomic propositions

$Lit := AP \cup \{\neg p \mid p \in AP\}$

Rules

$$r: \ A(r) \hookrightarrow N(r)$$

- $A(r) = \{a_1, ..., a_n\} \in 2^{Lit}$ is an antecedent.
- $N(r) \in Lit$ is a consequent.
- $\hookrightarrow \in \{\rightarrow, \Rightarrow$

Defeasible Logic (DL): Rules

Literals

$AP :=$ atomic propositions

$Lit := AP \cup \{\neg p \mid p \in AP\}$

Rules

$$r: \ A(r) \hookrightarrow N(r)$$

- $A(r) = \{a_1, \ldots, a_n\} \in 2^{Lit}$ is an antecedent.
- $N(r) \in Lit$ is a consequent.
- $\hookrightarrow \in \{\rightarrow, \Rightarrow, \leadsto\}$

Defeasible Logic (DL): Rules

Literals

$AP :=$ atomic propositions

$Lit := AP \cup \{\neg p \mid p \in AP\}$

Rules

$$r: \; A(r) \hookrightarrow N(r)$$

- $A(r) = \{a_1, ..., a_n\} \in 2^{Lit}$ is an antecedent.
- $N(r) \in Lit$ is a consequent.
- $\hookrightarrow \in \{\rightarrow, \Rightarrow, \leadsto\}$

Defeasible Theories

A defeasible theory is a tuple:

$$\langle F, R, > \rangle$$

where $F \subset Lit$ is a set of facts, R is a set of rules, and $>$ is a superiority relation over rules.

Defeasible Logic (DL): Rules

Literals

$AP :=$ atomic propositions

$Lit := AP \cup \{\neg p \mid p \in AP\}$

Rules

$$r: \; A(r) \hookrightarrow N(r)$$

- $A(r) = \{a_1, ..., a_n\} \in 2^{Lit}$ is an antecedent.
- $N(r) \in Lit$ is a consequent.
- $\hookrightarrow \in \{\rightarrow, \Rightarrow, \leadsto\}$

Defeasible Theories

A defeasible theory is a tuple:

$$\langle F, R, > \rangle$$

where $F \subset Lit$ is a set of facts, R is a set of rules, and $>$ is a superiority relation over rules.

From a defeasible theory, we can derive **conclusions**.

Defeasible Logic (DL): Definite Conclusions

Derived recursively:

Definite Provability

Given a defeasible theory D, if $D \vdash +\Delta p$, then either:

1. p is a fact ($p \in F$), or
2. There is a strict rule r such that:
   1. $N(r) = p$
   2. and for every $a_i \in A(r)$, $D \vdash +\Delta a_i$.

Defeasible Logic (DL): Definite Conclusions

Derived recursively:

Definite Provability

Given a defeasible theory D, if $D \vdash +\Delta p$, then either:

1. p is a fact ($p \in F$), or
2. There is a strict rule r such that:
   1. $N(r) = p$
   2. and for every $a_i \in A(r)$, $D \vdash +\Delta a_i$.

Definite Refutability

Given a defeasible theory D, if $D \vdash -\Delta p$, then:

1. p is not a fact ($p \notin F$), and
2. For all strict rules r such that $N(r) = p$, it is the case that $\exists a_i \in A(r)$ such that $D \vdash -\Delta a_i$.

Can be computed in linear time!

Defeasible Logic (DL): Defeasible Conclusions (pt 1)

Again, derived recursively:

Defeasible Provability

Given a defeasible theory D, If $D \vdash +\partial p$, either $D \vdash +\Delta p$ or:

1. There is a strict or defeasible rule r such that:
   1. $N(r) = p$ and
   2. for every $a_i \in A(r)$, $D \vdash +\partial a_i$, and
2. $D \vdash -\Delta \neg p$, and
3. For all rules r' such that $N(r') = \neg p$, either:
   1. there is an $a_i \in A(r')$ such that $D \vdash -\partial a_i$, or
   2. There is a strict or defeasible rule r'' such that:
      1. $N(r'') = p$,
      2. for all $a_i \in A(r'')$, $D \vdash +\partial a_i$, and
      3. $r'' > r'$.

Defeasible Logic (DL): Defeasible Conclusions (pt 2)

Defeasible Refutability

Given a defeasible theory D, If $D \vdash -\partial p$, $D \vdash -\Delta p$ and:

1. For all strict and defeasible rules r such that $N(r) = p$ there is $a_i \in A(r)$ such that $D \vdash -\partial a_i$, or
2. $+\Delta\neg p$, or
3. There is a rule r' such that:
   1. $N(r') = \neg p$,
   2. For all $a_i \in A(r')$, $D \vdash +\partial a_i$, and
   3. For all strict or defeasible rules r'' such that $N(r) = p$, either
      1. there is a $a_i \in A(r'')$ such that $D \vdash -\partial a_i$, or
      2. $r'' \not> r'$.

Can be computed in linear time!

Formalizing in DL

Example: the Sparrow

1. Birds can fly. (r_1 : *bird* $\Rightarrow$ *fly*)
2. Sparrows are birds. (r_2 : *sparrow* $\rightarrow$ *bird*)

Formalizing in DL

Example: the Sparrow

1. Birds can fly. (r_1 : $\mathit{bird} \Rightarrow \mathit{fly}$)
2. Sparrows are birds. (r_2 : $\mathit{sparrow} \rightarrow \mathit{bird}$)

- Suppose we have *sparrow* as a fact. So $+\Delta \mathit{sparrow}$.

Formalizing in DL

Example: the Sparrow

1. Birds can fly. (r_1 : *bird* $\Rightarrow$ *fly*)
2. Sparrows are birds. (r_2 : *sparrow* $\rightarrow$ *bird*)

- Suppose we have *sparrow* as a fact. So $+\Delta sparrow$.
- Then we can derive $+\Delta bird$ from $+\Delta sparrow$ and r_2.

Formalizing in DL

Example: the Sparrow

1. Birds can fly. (r_1 : *bird* $\Rightarrow$ *fly*)
2. Sparrows are birds. (r_2 : *sparrow* $\rightarrow$ *bird*)

- Suppose we have *sparrow* as a fact. So $+\Delta sparrow$.
- Then we can derive $+\Delta bird$ from $+\Delta sparrow$ and r_2.
- This means we can derive $+\partial fly$ from $+\Delta bird$ and r_1.

Another attempt at Non-monotonicity

Example: the Penguin (with a defeater)

1. Birds can fly. ($r_1 : \; \mathit{bird} \Rightarrow \mathit{fly}$)

Another attempt at Non-monotonicity

Example: the Penguin (with a defeater)

1. Birds can fly. ($r_1 : \mathit{bird} \Rightarrow \mathit{fly}$)
2. Penguins are birds. ($r_3 : \mathit{penguin} \rightarrow \mathit{bird}$)

Another attempt at Non-monotonicity

Example: the Penguin (with a defeater)

1. Birds can fly. (r_1 : $\mathit{bird} \Rightarrow \mathit{fly}$)
2. Penguins are birds. (r_3 : $\mathit{penguin} \rightarrow \mathit{bird}$)
3. Penguins don't fly. (r_4 : $\mathit{penguin} \rightsquigarrow \neg \mathit{fly}$)

Another attempt at Non-monotonicity

Example: the Penguin (with a defeater)

1. Birds can fly. (r_1 : $bird \Rightarrow fly$)
2. Penguins are birds. (r_3 : $penguin \rightarrow bird$)
3. Penguins don't fly. (r_4 : $penguin \rightsquigarrow \neg fly$)

- Suppose we have *penguin* as a fact. So $+\Delta penguin$.
- Then we can derive $+\Delta bird$ from r_3 and $+\Delta penguin$

Another attempt at Non-monotonicity

Example: the Penguin (with a defeater)

1. Birds can fly. ($r_1 : \mathit{bird} \Rightarrow \mathit{fly}$)
2. Penguins are birds. ($r_3 : \mathit{penguin} \rightarrow \mathit{bird}$)
3. Penguins don't fly. ($r_4 : \mathit{penguin} \rightsquigarrow \neg \mathit{fly}$)

- Suppose we have *penguin* as a fact. So $+\Delta \mathit{penguin}$.
- Then we can derive $+\Delta \mathit{bird}$ from r_3 and $+\Delta \mathit{penguin}$
- Then we **cannot** derive $+\partial \mathit{fly}$ from r_1 and $+\Delta \mathit{bird}$; r_4 prevents this.

Another attempt at Non-monotonicity

Example: the Penguin (with a defeater)

1. Birds can fly. (r_1 : $bird \Rightarrow fly$)
2. Penguins are birds. (r_3 : $penguin \rightarrow bird$)
3. Penguins don't fly. (r_4 : $penguin \rightsquigarrow \neg fly$)

- Suppose we have *penguin* as a fact. So $+\Delta penguin$.
- Then we can derive $+\Delta bird$ from r_3 and $+\Delta penguin$
- Then we **cannot** derive $+\partial fly$ from r_1 and $+\Delta bird$; r_4 prevents this.

Example: the Penguin (with a superiority relation)

1. Birds can fly. (r_1 : $bird \Rightarrow fly$)
2. Penguins are birds. (r_3 : $penguin \rightarrow bird$)

Another attempt at Non-monotonicity

Example: the Penguin (with a defeater)

1. Birds can fly. (r_1 : $bird \Rightarrow fly$)
2. Penguins are birds. (r_3 : $penguin \rightarrow bird$)
3. Penguins don't fly. (r_4 : $penguin \rightsquigarrow \neg fly$)

- Suppose we have *penguin* as a fact. So $+\Delta penguin$.
- Then we can derive $+\Delta bird$ from r_3 and $+\Delta penguin$
- Then we **cannot** derive $+\partial fly$ from r_1 and $+\Delta bird$; r_4 prevents this.

Example: the Penguin (with a superiority relation)

1. Birds can fly. (r_1 : $bird \Rightarrow fly$)
2. Penguins are birds. (r_3 : $penguin \rightarrow bird$)
3. Penguins don't fly (r_5 : $penguin \Rightarrow \neg fly$)
4. $r_5 > r_1$

Another attempt at Non-monotonicity

Example: the Penguin (with a defeater)

1. Birds can fly. (r_1 : *bird* $\Rightarrow$ *fly*)
2. Penguins are birds. (r_3 : *penguin* $\rightarrow$ *bird*)
3. Penguins don't fly. (r_4 : *penguin* $\leadsto \neg$*fly*)

- Suppose we have *penguin* as a fact. So $+\Delta$*penguin*.
- Then we can derive $+\Delta$*bird* from r_3 and $+\Delta$*penguin*
- Then we **cannot** derive $+\partial$*fly* from r_1 and $+\Delta$*bird*; r_4 prevents this.

Example: the Penguin (with a superiority relation)

1. Birds can fly. (r_1 : *bird* $\Rightarrow$ *fly*)
2. Penguins are birds. (r_3 : *penguin* $\rightarrow$ *bird*)
3. Penguins don't fly (r_5 : *penguin* $\Rightarrow \neg$*fly*)
4. $r_5 > r_1$

- Suppose we have *penguin* as a fact. So $+\Delta$*penguin*.
- Then we can derive $+\Delta$*bird* from r_3 and $+\Delta$*penguin*
- Then we derive $+\partial\neg$*fly* from r_5 and $+\Delta$*penguin*; r_1 conflicts, but is defeated.

An Extension: Defeasible Deontic Logic

We can extend DL with deontic operators!

New Syntax

- Introduce modal literals: $ModLit = \{\mathbf{O}(lit) | lit \in Lit\}$
- Introduce rule modalities: $\hookrightarrow_* \in \{\rightarrow_*, \Rightarrow_*, \rightsquigarrow_*\}, * \in \{C, O\}$
  - For any $r: \; A(r) \hookrightarrow_O N(r), N(r) \in ModLit$

We can extend DL with deontic operators!

New Syntax

- Introduce modal literals: $ModLit = \{\mathbf{O}(lit) | lit \in Lit\}$
- Introduce rule modalities: $\hookrightarrow_* \in \{\rightarrow_*, \Rightarrow_*, \rightsquigarrow_*\}, * \in \{C, O\}$
  - For any $r: \ A(r) \hookrightarrow_O N(r), N(r) \in ModLit$

Translating Norms in DDL

- $\mathbf{O}(p|q)$ translates to $q \Rightarrow_O p$

An Extension: Defeasible Deontic Logic

We can extend DL with deontic operators!

New Syntax

- Introduce modal literals: $ModLit = \{\mathbf{O}(lit) | lit \in Lit\}$
- Introduce rule modalities: $\hookrightarrow_* \in \{\rightarrow_*, \Rightarrow_*, \leadsto_*\}$, $* \in \{C, O\}$
  - For any $r: \ A(r) \hookrightarrow_O N(r), N(r) \in ModLit$

Translating Norms in DDL

- $\mathbf{O}(p|q)$ translates to $q \Rightarrow_O p$
- $\mathbf{F}(p|q)$ translates to $q \Rightarrow_O \neg p$

An Extension: Defeasible Deontic Logic

We can extend DL with deontic operators!

New Syntax

- Introduce modal literals: $ModLit = \{\mathbf{O}(lit) | lit \in Lit\}$
- Introduce rule modalities: $\hookrightarrow_* \in \{\rightarrow_*, \Rightarrow_*, \rightsquigarrow_*\}$, $* \in \{C, O\}$
  - For any r : $A(r) \hookrightarrow_O N(r)$, $N(r) \in ModLit$

Translating Norms in DDL

- $\mathbf{O}(p|q)$ translates to $q \Rightarrow_O p$
- $\mathbf{F}(p|q)$ translates to $q \Rightarrow_O \neg p$
- $\mathbf{P}_s(p|q)$ translates to $q \rightsquigarrow_O p$

An Extension: Defeasible Deontic Logic

We can extend DL with deontic operators!

New Syntax

- Introduce modal literals: $ModLit = \{\mathbf{O}(lit)|lit \in Lit\}$
- Introduce rule modalities: $\hookrightarrow_* \in \{\rightarrow_*, \Rightarrow_*, \rightsquigarrow_*\}, * \in \{C, O\}$
  - For any $r: \; A(r) \hookrightarrow_O N(r), N(r) \in ModLit$

Translating Norms in DDL

- $\mathbf{O}(p|q)$ translates to $q \Rightarrow_O p$
- $\mathbf{F}(p|q)$ translates to $q \Rightarrow_O \neg p$
- $\mathbf{P}_s(p|q)$ translates to $q \rightsquigarrow_O p$
- $\mathbf{C}(x, y|c)$ translates to $c, x \rightarrow_C y$

Reframing Compliance

DDL theory = Facts + Normative System

Reframing Compliance

DDL theory = Facts + Normative System

Interpreting Conclusions

- $+\partial_O p$ means p is obligatory.

Reframing Compliance

DDL theory = Facts + Normative System

Interpreting Conclusions

- $+\partial_O p$ means p is obligatory.
- $+\partial_O \neg p$ means p is forbidden.

Reframing Compliance

DDL theory = Facts + Normative System

Interpreting Conclusions

- $+\partial_O p$ means p is obligatory.
- $+\partial_O \neg p$ means p is forbidden.
- $-\partial_O \neg p$ means p is (weakly) permissible.

Reframing Compliance

DDL theory = Facts + Normative System

Interpreting Conclusions

- $+\partial_O p$ means p is obligatory.
- $+\partial_O \neg p$ means p is forbidden.
- $-\partial_O \neg p$ means p is (weakly) permissible.
- $+\partial_C p$ means we can prove p is true.

Reframing Compliance

DDL theory = Facts + Normative System

Interpreting Conclusions

- $+\partial_O p$ means p is obligatory.
- $+\partial_O \neg p$ means p is forbidden.
- $-\partial_O \neg p$ means p is (weakly) permissible.
- $+\partial_C p$ means we can prove p is true.
- $-\partial_C p$ means we cannot prove p is true.

Reframing Compliance

DDL theory = Facts + Normative System

Interpreting Conclusions

- $+\partial_O p$ means p is obligatory.
- $+\partial_O \neg p$ means p is forbidden.
- $-\partial_O \neg p$ means p is (weakly) permissible.
- $+\partial_C p$ means we can prove p is true.
- $-\partial_C p$ means we cannot prove p is true.

Violation in DDL

Suppose we have a DDL theory D representing a set of facts F and a normative system. Then a violation is a literal *lit* such that:

$$D \vdash +\partial_O lit,\ -\partial_C lit$$

Example: a Normative System in DDL (pt 1)

A simple normative system...and a violation

- You are forbidden from eating cake. ($r_1 : \Rightarrow_O \neg cake$)
- Eating carrot cake counts as eating cake. ($r_2 :\ carrot \rightarrow_C cake$)

Example: a Normative System in DDL (pt 1)

A simple normative system...and a violation

- You are forbidden from eating cake. ($r_1 : \Rightarrow_O \neg cake$)
- Eating carrot cake counts as eating cake. ($r_2 : \; carrot \rightarrow_C cake$)
- Suppose it is a fact that you eat carrot cake, so we have $+\Delta_C carrot$.

Example: a Normative System in DDL (pt 1)

A simple normative system...and a violation

- You are forbidden from eating cake. ($r_1 : \Rightarrow_O \neg cake$)
- Eating carrot cake counts as eating cake. ($r_2 : \ carrot \rightarrow_C cake$)
- Suppose it is a fact that you eat carrot cake, so we have $+\Delta_C carrot$.
- With r_2 we can derive $+\Delta_C cake$, from which we get $+\partial_C cake$.

Example: a Normative System in DDL (pt 1)

A simple normative system...and a violation

- You are forbidden from eating cake. ($r_1 : \Rightarrow_O \neg cake$)
- Eating carrot cake counts as eating cake. ($r_2 :\ carrot \rightarrow_C cake$)
- Suppose it is a fact that you eat carrot cake, so we have $+\Delta_C carrot$.
- With r_2 we can derive $+\Delta_C cake$, from which we get $+\partial_C cake$.
- If we have $+\partial_C cake$, we cannot have $+\partial_C \neg cake$; instead, we get $-\partial_C \neg cake$.

Example: a Normative System in DDL (pt 1)

A simple normative system...and a violation

- You are forbidden from eating cake. ($r_1 : \Rightarrow_O \neg cake$)
- Eating carrot cake counts as eating cake. ($r_2 : \ carrot \rightarrow_C cake$)
- Suppose it is a fact that you eat carrot cake, so we have $+\Delta_C carrot$.
- With r_2 we can derive $+\Delta_C cake$, from which we get $+\partial_C cake$.
- If we have $+\partial_C cake$, we cannot have $+\partial_C \neg cake$; instead, we get $-\partial_C \neg cake$.
- However, from r_1 we get $+\partial_O \neg cake$.

Example: a Normative System in DDL (pt 1)

A simple normative system...and a violation

- You are forbidden from eating cake. ($r_1 : \Rightarrow_O \neg cake$)
- Eating carrot cake counts as eating cake. ($r_2 : carrot \rightarrow_C cake$)
- Suppose it is a fact that you eat carrot cake, so we have $+\Delta_C carrot$.
- With r_2 we can derive $+\Delta_C cake$, from which we get $+\partial_C cake$.
- If we have $+\partial_C cake$, we cannot have $+\partial_C \neg cake$; instead, we get $-\partial_C \neg cake$.
- However, from r_1 we get $+\partial_O \neg cake$.
- There is a violation!

Example: a Normative System in DDL (pt 2)

Adding permission

- Take r_1 and r_2 as above.
- On Tuesdays, you are permitted to eat cake. ($r_3 : \; \mathit{tuesday} \leadsto_O \mathit{cake}$)

Example: a Normative System in DDL (pt 2)

Adding permission

- Take r_1 and r_2 as above.
- On Tuesdays, you are permitted to eat cake. ($r_3 :\ tuesday \rightsquigarrow_O cake$)
- As above, if we take *carrot* as a fact, we can derive $-\partial_C \neg cake$.

Example: a Normative System in DDL (pt 2)

Adding permission

- Take r_1 and r_2 as above.
- On Tuesdays, you are permitted to eat cake. ($r_3 : \mathit{tuesday} \leadsto_O \mathit{cake}$)
- As above, if we take *carrot* as a fact, we can derive $-\partial_C \neg \mathit{cake}$.
- However, if we also have the fact *tuesday*, r_3 prevents us from deriving $+\partial_O \neg \mathit{cake}$ from r_1.
- No violation!

Example: a Normative System in DDL (pt 2)

Adding permission

- Take r_1 and r_2 as above.
- On Tuesdays, you are permitted to eat cake. (r_3 : $\mathit{tuesday} \leadsto_O \mathit{cake}$)
- As above, if we take *carrot* as a fact, we can derive $-\partial_C \neg \mathit{cake}$.
- However, if we also have the fact *tuesday*, r_3 prevents us from deriving $+\partial_O \neg \mathit{cake}$ from r_1.
- No violation!

Conflicting rules

- Take r_1 and r_2 as above.
- If it is a gift, you ought to eat the cake. (r_4 : $\mathit{gift} \Rightarrow_O \mathit{cake}$)

Example: a Normative System in DDL (pt 2)

Adding permission

- Take r_1 and r_2 as above.
- On Tuesdays, you are permitted to eat cake. ($r_3 :\ tuesday \leadsto_O cake$)
- As above, if we take *carrot* as a fact, we can derive $-\partial_C \neg cake$.
- However, if we also have the fact *tuesday*, r_3 prevents us from deriving $+\partial_O \neg cake$ from r_1.
- No violation!

Conflicting rules

- Take r_1 and r_2 as above.
- If it is a gift, you ought to eat the cake. ($r_4 :\ gift \Rightarrow_O cake$)
- Suppose *gift* is a fact. If we take $r_4 > r_1$, then we derive $+\partial_O cake$ instead of $+\partial_O \neg cake$.

Example: a Normative System in DDL (pt 2)

Adding permission

- Take r_1 and r_2 as above.
- On Tuesdays, you are permitted to eat cake. (r_3 : $tuesday \leadsto_O cake$)
- As above, if we take *carrot* as a fact, we can derive $-\partial_C \neg cake$.
- However, if we also have the fact *tuesday*, r_3 prevents us from deriving $+\partial_O \neg cake$ from r_1.
- No violation!

Conflicting rules

- Take r_1 and r_2 as above.
- If it is a gift, you ought to eat the cake. (r_4 : $gift \Rightarrow_O cake$)
- Suppose *gift* is a fact. If we take $r_4 > r_1$, then we derive $+\partial_O cake$ instead of $+\partial_O \neg cake$.
- If we have *carrot* as a fact, we can derive $+\Delta_C cake$ so we cannot derive $-\partial_C cake$.
- No violation!

German

Für das gesamte Plangebiet wird bestimmt: Sofern nichts anderes bestimmt ist, sind Flachdächer von Gebäuden ab einer bebauten Fläche von 30 m^2, soweit sie nicht als begehbare Terrassen ausgebildet werden, nach dem Stand der technischen Wissenschaften zu begrünen.

English

The following is stipulated for the entire plan area: Unless otherwise stipulated, flat roofs of buildings with a built-up area of 30 m^2 or more, unless they are designed as accessible terraces, are to be greened in accordance with the state of the art.

BRISE: Regulation 6817e_5_0

Extracted Concepts

1. BegruenungDach [content] [greened roof]
2. Dachart(Flachdach) [condition] [roof type: flat roof]
3. Dachart(begehbare Terrasse) [conditionException] [roof type: accessible terrace]
4. GesamtePlangebiet [condition] [entire plan area]
5. BebauteFlaecheMin(30 m2) [condition] [built-up area minimum]

BRISE: Regulation 6817e_5_0

Extracted Concepts

1. BegruenungDach [content] [greened roof]
2. Dachart(Flachdach) [condition] [roof type: flat roof]
3. Dachart(begehbare Terrasse) [conditionException] [roof type: accessible terrace]
4. GesamtePlangebiet [condition] [entire plan area]
5. BebauteFlaecheMin(30 m2) [condition] [built-up area minimum]

DDL Formalization

$$r_1 : \mathit{GesamtePlangebiet}, \mathit{BebauteFlaecheMin}(30m2), \mathit{Dachart}(\mathit{Flachdach}) \Rightarrow_O \mathit{BegruenungDach}$$

BRISE: Regulation 6817e_5_0

Extracted Concepts

1. BegruenungDach [content] [greened roof]
2. Dachart(Flachdach) [condition] [roof type: flat roof]
3. Dachart(begehbare Terrasse) [conditionException] [roof type: accessible terrace]
4. GesamtePlangebiet [condition] [entire plan area]
5. BebauteFlaecheMin(30 m2) [condition] [built-up area minimum]

DDL Formalization

$$r_1 : \ \mathit{GesamtePlangebiet}, \mathit{BebauteFlaecheMin}(30m2), \mathit{Dachart}(\mathit{Flachdach}) \Rightarrow_O \mathit{BegruenungDach}$$

$$r_2 : \ \mathit{GesamtePlangebiet}, \mathit{BebauteFlaecheMin}(30m2), \mathit{Dachart}(\mathit{begehbareTerrasse}) \rightsquigarrow_O \neg \mathit{BegruenungDach}$$

BRISE: Regulation 6817e_5_0

Extracted Concepts

1. BegruenungDach [content] [greened roof]
2. Dachart(Flachdach) [condition] [roof type: flat roof]
3. Dachart(begehbare Terrasse) [conditionException] [roof type: accessible terrace]
4. GesamtePlangebiet [condition] [entire plan area]
5. BebauteFlaecheMin(30 m2) [condition] [built-up area minimum]

DDL Formalization

$$r_1 : \mathit{GesamtePlangebiet}, \mathit{BebauteFlaecheMin}(30m2), \mathit{Dachart}(\mathit{Flachdach}) \Rightarrow_O \mathit{BegruenungDach}$$

$$r_2 : \mathit{GesamtePlangebiet}, \mathit{BebauteFlaecheMin}(30m2), \mathit{Dachart}(\mathit{begehbareTerrasse}) \leadsto_O \neg \mathit{BegruenungDach}$$

$$r_3 : \mathit{Dachart}(\mathit{begehbareTerrasse}) \rightarrow_C \mathit{Dachart}(\mathit{Flachdach})$$

BRISE: Regulation 6817e_5_0

Example 1: Facts

- GesamtePlangebiet
- BebauteFlaecheMin(30 m2)
- Dachart(Flachdach)
- BegruenungDach

Example 1: Facts

- GesamtePlangebiet
- BebauteFlaecheMin(30 m2)
- Dachart(Flachdach)
- BegruenungDach

Example 1: Conclusions

- Given the above facts, we get $+\Delta_C GesamtePlangebiet$, $+\Delta_C BebauteFlaecheMin(30m2)$, $+\Delta_C Dachart(Flachdach)$, $+\Delta_C BegruenungDach$.

Example 1: Facts

- GesamtePlangebiet
- BebauteFlaecheMin(30 m2)
- Dachart(Flachdach)
- BegruenungDach

Example 1: Conclusions

- Given the above facts, we get $+\Delta_C GesamtePlangebiet$, $+\Delta_C BebauteFlaecheMin(30m2)$, $+\Delta_C Dachart(Flachdach)$, $+\Delta_C BegruenungDach$.
- Then from r_1 we can derive $+\partial_O BegruenungDach$.
- Since we have $+\Delta_C BegruenungDach$, there is no violation.

Example 2: Facts

- GesamtePlangebiet
- BebauteFlaecheMin(30 m2)
- Dachart(begehbare Terrasse)

Example 2: Facts

- GesamtePlangebiet
- BebauteFlaecheMin(30 m2)
- Dachart(begehbare Terrasse)

Example 2: Conclusions

- Given the above facts, we get $+\Delta_C GesamtePlangebiet$, $+\Delta_C BebauteFlaecheMin(30m2)$, $+\Delta_C Dachart(begehbareTerrasse)$.

Example 2: Facts

- GesamtePlangebiet
- BebauteFlaecheMin(30 m2)
- Dachart(begehbare Terrasse)

Example 2: Conclusions

- Given the above facts, we get $+\Delta_C GesamtePlangebiet$, $+\Delta_C BebauteFlaecheMin(30m2)$, $+\Delta_C Dachart(begehbareTerrasse)$.
- From r_3 and $+\Delta_C Dachart(begehbareTerrasse)$, we can derive $+\Delta_C Dachart(Flachdach)$

Example 2: Facts

- GesamtePlangebiet
- BebauteFlaecheMin(30 m2)
- Dachart(begehbare Terrasse)

Example 2: Conclusions

- Given the above facts, we get $+\Delta_C GesamtePlangebiet$, $+\Delta_C BebauteFlaecheMin(30m2)$, $+\Delta_C Dachart(begehbareTerrasse)$.
- From r_3 and $+\Delta_C Dachart(begehbareTerrasse)$, we can derive $+\Delta_C Dachart(Flachdach)$
- So both r_1 and r_2 are triggered; r_2 defeats r_1 and we cannot derive $+\partial_O BegruenungDach$.

Example 2: Facts

- GesamtePlangebiet
- BebauteFlaecheMin(30 m2)
- Dachart(begehbare Terrasse)

Example 2: Conclusions

- Given the above facts, we get $+\Delta_C GesamtePlangebiet$, $+\Delta_C BebauteFlaecheMin(30m2)$, $+\Delta_C Dachart(begehbareTerrasse)$.
- From r_3 and $+\Delta_C Dachart(begehbareTerrasse)$, we can derive $+\Delta_C Dachart(Flachdach)$
- So both r_1 and r_2 are triggered; r_2 defeats r_1 and we cannot derive $+\partial_O BegruenungDach$.
- There are no obligations to violate.

To Take Home with You

Why logic?

To Take Home with You

Why logic?

- Logic allows us to model arbitrarily complex constraints.

To Take Home with You

Why logic?

- Logic allows us to model arbitrarily complex constraints.
- When we use a theorem prover to check compliance, it gives us a sort of “certificate” of the derived results.

To Take Home with You

Why logic?

- Logic allows us to model arbitrarily complex constraints.
- When we use a theorem prover to check compliance, it gives us a sort of "certificate" of the derived results.
- From a set of conclusions, facts, and rules we can always reconstruct the reasoning that led to those conclusions.

To Take Home with You

Why logic?

- Logic allows us to model arbitrarily complex constraints.
- When we use a theorem prover to check compliance, it gives us a sort of “certificate” of the derived results.
- From a set of conclusions, facts, and rules we can always reconstruct the reasoning that led to those conclusions.
- Representing ideal behaviour through rules helps with explainability and transparency.

Literature I

References

[1] D. Nute, "Defeasible logic," in *Handbook of Logic in Artificial Intelligence and Logic Programming: Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning*, vol. 3, Oxford University Press, 1993.

[2] M. J. Maher, A. Rock, G. Antoniou, D. Billington, and T. Miller, "Efficient defeasible reasoning systems," *International Journal on Artificial Intelligence Tools*, vol. 10, no. 04, pp. 483–501, 2001.

[3] G. Governatori and A. Rotolo, "BIO logical agents: Norms, beliefs, intentions in defeasible logic," *Journal of Autonomous Agents and Multi Agent Systems*, vol. 17, no. 1, pp. 36–69, 2008.

[4] G. Governatori, "Practical normative reasoning with defeasible deontic logic," in *Reasoning Web International Summer School*, Springer, 2018, pp. 1–25.