Simulating Negation in Positive Logic

João Marcos

LoLITA / DIMAp, UFRN, BR

Logic Colloquium 2009
Sofia, BG
Jul 31–Aug 5, 2009

Deducibility & Logical Constants

Consequence relations

$\Gamma, \gamma \vdash \delta, \Delta$ your preferred deductive formalism

$\Gamma, \gamma \vDash \delta, \Delta$ (many-valued) semantics

$\Gamma, \gamma \Vdash \delta, \Delta$ **G**eneral **A**bstract **N**onsense

Deducibility & Logical Constants

Consequence relations

$\Gamma, \gamma \vdash \delta, \Delta$ your preferred deductive formalism

$\Gamma, \gamma \vDash \delta, \Delta$ (many-valued) semantics

$\Gamma, \gamma \Vdash \delta, \Delta$ **G**eneral **A**bstract **N**onsense

Deducibility & Logical Constants

Consequence relations

$\Gamma, \gamma \vdash \delta, \Delta$ your preferred deductive formalism

$\Gamma, \gamma \vDash \delta, \Delta$ (many-valued) semantics

$\Gamma, \gamma \Vdash \delta, \Delta$ **G**eneral **A**bstract **N**onsense

Deducibility & Logical Constants

On the role of the object-language constructors

$$\frac{\Gamma \Vdash \Delta}{\Gamma, \top \Vdash \Delta} \qquad \frac{\Gamma \Vdash \Delta}{\Gamma \Vdash \bot, \Delta}$$

$$\frac{\Gamma, \alpha, \beta \Vdash \Delta}{\Gamma, \alpha \wedge \beta \Vdash \Delta} \qquad \frac{\Gamma \Vdash \alpha, \beta, \Delta}{\Gamma \Vdash \alpha \vee \beta, \Delta}$$

$$\frac{\Gamma, \alpha \Vdash \beta, \Delta}{\Gamma \Vdash \alpha \to \beta, \Delta} \qquad \frac{\Gamma, \alpha \Vdash \beta, \Delta}{\Gamma, \beta \multimap \alpha \Vdash \Delta}$$

$$\frac{\Gamma, \alpha \Vdash \Delta}{\Gamma \Vdash {\sim}\alpha, \Delta} \qquad \frac{\Gamma \Vdash \alpha, \Delta}{\Gamma, {\sim}\alpha \Vdash \Delta}$$

Consider $\sim_1 \alpha \stackrel{\text{def}}{=} \smile\alpha \stackrel{\text{def}}{=} \alpha \to \bot$.

Consider $\sim_2 \alpha \stackrel{\text{def}}{=} \frown\alpha \stackrel{\text{def}}{=} \alpha \multimap \top$.

On the role of the object-language constructors

$$\frac{\Gamma \Vdash \Delta}{\Gamma, \top \Vdash \Delta}$$

$$\frac{\Gamma, \alpha, \beta \Vdash \Delta}{\Gamma, \alpha \wedge \beta \Vdash \Delta}$$

$$\frac{\Gamma, \alpha \Vdash \beta, \Delta}{\Gamma \Vdash \alpha \to \beta, \Delta}$$

$$\frac{\Gamma, \alpha \Vdash \Delta}{\Gamma \Vdash {\sim}\alpha, \Delta}$$

Consider

$\sim_1 \alpha \stackrel{\text{def}}{=} \smile\alpha \stackrel{\text{def}}{=} \alpha \to \bot.$

$$\frac{\Gamma \Vdash \Delta}{\Gamma \Vdash \bot, \Delta}$$

$$\frac{\Gamma \Vdash \alpha, \beta, \Delta}{\Gamma \Vdash \alpha \vee \beta, \Delta}$$

$$\frac{\Gamma, \alpha \Vdash \beta, \Delta}{\Gamma, \beta \multimap \alpha \Vdash \Delta}$$

$$\frac{\Gamma \Vdash \alpha, \Delta}{\Gamma, {\sim}\alpha \Vdash \Delta}$$

Consider

$\sim_2 \alpha \stackrel{\text{def}}{=} \frown\alpha \stackrel{\text{def}}{=} \alpha \multimap \top.$

Deducibility & Logical Constants

On the role of the object-language constructors

$$\frac{\Gamma \Vdash \Delta}{\Gamma, \top \Vdash \Delta} \qquad \frac{\Gamma \Vdash \Delta}{\Gamma \Vdash \bot, \Delta}$$

$$\frac{\Gamma, \alpha, \beta \Vdash \Delta}{\Gamma, \alpha \wedge \beta \Vdash \Delta} \qquad \frac{\Gamma \Vdash \alpha, \beta, \Delta}{\Gamma \Vdash \alpha \vee \beta, \Delta}$$

$$\frac{\Gamma, \alpha \Vdash \beta, \Delta}{\Gamma \Vdash \alpha \to \beta, \Delta} \qquad \frac{\Gamma, \alpha \Vdash \beta, \Delta}{\Gamma, \beta \multimap \alpha \Vdash \Delta}$$

$$\frac{\Gamma, \alpha \Vdash \Delta}{\Gamma \Vdash \sim\alpha, \Delta} \qquad \frac{\Gamma \Vdash \alpha, \Delta}{\Gamma, \sim\alpha \Vdash \Delta}$$

Consider $\sim_1 \alpha \stackrel{\text{def}}{=} \smile\alpha \stackrel{\text{def}}{=} \alpha \to \bot$.

Consider $\sim_2 \alpha \stackrel{\text{def}}{=} \frown\alpha \stackrel{\text{def}}{=} \alpha \multimap \top$.

Deducibility & Logical Constants

On the role of the object-language constructors

$$\frac{\Gamma \Vdash \Delta}{\Gamma, \top \Vdash \Delta} \qquad \frac{\Gamma \Vdash \Delta}{\Gamma \Vdash \bot, \Delta}$$

$$\frac{\Gamma, \alpha, \beta \Vdash \Delta}{\Gamma, \alpha \wedge \beta \Vdash \Delta} \qquad \frac{\Gamma \Vdash \alpha, \beta, \Delta}{\Gamma \Vdash \alpha \vee \beta, \Delta}$$

$$\frac{\Gamma, \alpha \Vdash \beta, \Delta}{\Gamma \Vdash \alpha \to \beta, \Delta} \qquad \frac{\Gamma, \alpha \Vdash \beta, \Delta}{\Gamma, \beta \multimap \alpha \Vdash \Delta}$$

$$\frac{\Gamma, \alpha \Vdash \Delta}{\Gamma \Vdash {\sim}\alpha, \Delta} \qquad \frac{\Gamma \Vdash \alpha, \Delta}{\Gamma, {\sim}\alpha \Vdash \Delta}$$

Consider $\sim_1 \alpha \stackrel{\text{def}}{=} \smile\alpha \stackrel{\text{def}}{=} \alpha \to \bot$.

Consider $\sim_2 \alpha \stackrel{\text{def}}{=} \frown\alpha \stackrel{\text{def}}{=} \alpha \multimap \top$.

Deducibility & Logical Constants

On the role of the object-language constructors

$$\frac{\Gamma \Vdash \Delta}{\Gamma, \top \Vdash \Delta} \qquad \frac{\Gamma \Vdash \Delta}{\Gamma \Vdash \bot, \Delta}$$

$$\frac{\Gamma, \alpha, \beta \Vdash \Delta}{\Gamma, \alpha \wedge \beta \Vdash \Delta} \qquad \frac{\Gamma \Vdash \alpha, \beta, \Delta}{\Gamma \Vdash \alpha \vee \beta, \Delta}$$

$$\frac{\Gamma, \alpha \Vdash \beta, \Delta}{\Gamma \Vdash \alpha \rightarrow \beta, \Delta} \qquad \frac{\Gamma, \alpha \Vdash \beta, \Delta}{\Gamma, \beta \multimap \alpha \Vdash \Delta}$$

$$\frac{\Gamma, \alpha \Vdash \Delta}{\Gamma \Vdash \sim\alpha, \Delta} \qquad \frac{\Gamma \Vdash \alpha, \Delta}{\Gamma, \sim\alpha \Vdash \Delta}$$

Consider
$\sim_1 \alpha \overset{\text{def}}{=} \smile\alpha \overset{\text{def}}{=} \alpha \rightarrow \bot$.

Consider
$\sim_2 \alpha \overset{\text{def}}{=} \frown\alpha \overset{\text{def}}{=} \alpha \multimap \top$.

On the role of the object-language constructors (contd.)

$$\frac{\Gamma, \alpha_1, \ldots, \alpha_m \Vdash \Delta}{\Gamma \Vdash \uparrow(\alpha_1, \ldots, \alpha_m), \Delta} \qquad \frac{\Gamma \Vdash \alpha_1, \ldots, \alpha_m, \Delta}{\Gamma, \downarrow(\alpha_1, \ldots, \alpha_m) \Vdash \Delta}$$

$$\frac{\dfrac{\Gamma, \alpha \dashv\Vdash \beta, \Delta}{\Gamma \Vdash \alpha \leftrightarrow \beta, \Delta}}{\Gamma, \alpha + \beta \Vdash \Delta}$$

Deducibility & Logical Constants

On the role of the object-language constructors (contd.)

$$\frac{\Gamma, \alpha_1, \ldots, \alpha_m \Vdash \Delta}{\Gamma \Vdash \uparrow(\alpha_1, \ldots, \alpha_m), \Delta} \qquad \frac{\Gamma \Vdash \alpha_1, \ldots, \alpha_m, \Delta}{\Gamma, \downarrow(\alpha_1, \ldots, \alpha_m) \Vdash \Delta}$$

$$\frac{\frac{\Gamma, \alpha \dashv\Vdash \beta, \Delta}{\Gamma \Vdash \alpha \leftrightarrow \beta, \Delta}}{\Gamma, \alpha + \beta \Vdash \Delta}$$

The Profane Approach

How do rules affect truth-tables?

Consider the simple case of a binary 2-valued connective:

©	1	0
1	1	0
0	0	0

Now, to force *the following specific restriction*...

©	1	0
1	...	...
0	0	...

... one might consider a rule such as:

$$\frac{\Gamma, \alpha \Vdash \Delta \qquad \Gamma \Vdash \beta, \Delta}{\Gamma, \alpha © \beta \Vdash \Delta}$$

The Profane Approach

How do rules affect truth-tables?

Consider the simple case of a binary 2-valued connective:

©	1	0
1	1	0
0	0	0

Now, to force *the following specific restriction*...

©	1	0
1	...	...
0	0	...

... one might consider a rule such as:

$$\frac{\Gamma, \alpha \Vdash \Delta \qquad \Gamma \Vdash \beta, \Delta}{\Gamma, \alpha \copyright \beta \Vdash \Delta}$$

The Profane Approach

How do rules affect truth-tables?

Consider the simple case of a binary 2-valued connective:

©	1	0
1	1	0
0	0	0

Now, to force *the following specific restriction*. . .

©	1	0
1	. . .	. . .
0	0	. . .

. . . one might consider a rule such as:

$$\frac{\Gamma, \alpha \Vdash \Delta \qquad \Gamma \Vdash \beta, \Delta}{\Gamma, \alpha © \beta \Vdash \Delta}$$

The Profane Approach

How do rules affect truth-tables?

Consider the simple case of a binary 2-valued connective:

©	1	0
1	1	0
0	0	0

Now, to force *the following specific restriction*...

©	1	0
1	...	...
0	0	...

... one might consider a rule such as:

$$\frac{\Gamma, \alpha \Vdash \Delta \qquad \Gamma \Vdash \beta, \Delta}{\Gamma, \alpha © \beta \Vdash \Delta}$$

The Profane Approach

How do rules affect truth-tables?

Consider the simple case of a binary 2-valued connective:

©	1	0
1	1	0
0	0	0

Now, to force *the following specific restriction*...

©	1	0
1	...	...
0	0	...

... one might consider a rule such as:

$$\frac{\Gamma, \alpha \Vdash \Delta \qquad \Gamma \Vdash \beta, \Delta}{\Gamma, \alpha © \beta \Vdash \Delta}$$

The Profane Approach

How do rules affect truth-tables?

Consider the simple case of a binary 2-valued connective:

©	1	0
1	1	0
0	0	0

On what concerns duality...

The Profane Approach

How do rules affect truth-tables?

Consider the simple case of a binary 2-valued connective:

©	1	0
1	1	0
0	0	0

On what concerns duality...

... one should invert the inputs...

The Profane Approach

How do rules affect truth-tables?

Consider the simple case of a binary 2-valued connective:

©	1	0
1	1	0
0	0	0

On what concerns duality. . .

. . . one should invert the inputs. . .

©	0	1
0	1	0
1	0	0

The Profane Approach

How do rules affect truth-tables?

Consider the simple case of a binary 2-valued connective:

©	1	0
1	1	0
0	0	0

On what concerns duality. . .

. . . and also the outputs. . .

©	0	1
0	1	0
1	0	0

The Profane Approach

How do rules affect truth-tables?

Consider the simple case of a binary 2-valued connective:

©	1	0
1	1	0
0	0	0

On what concerns duality. . .

. . . and also the outputs. . .

©	0	1
0	0	1
1	1	1

The Profane Approach

How do rules affect truth-tables?

Consider the simple case of a binary 2-valued connective:

©	1	0
1	1	0
0	0	0

On what concerns duality...

Rearranging now this table, one obtains $©^d$:

©	0	1
0	0	1
1	1	1

The Profane Approach

How do rules affect truth-tables?

Consider the simple case of a binary 2-valued connective:

©	1	0
1	1	0
0	0	0

On what concerns duality...

Rearranging now this table, one obtains $©^d$:

$©^d$	1	0
1	1	1
0	1	0

What is a ‘Negative’ Constructor?

	$\circledcirc^3_2$
1	1
1	0
0	0

	$\circledcirc^2_2$
1	1
0	1
0	0

	$\circledcirc^1_2$
1	1
0	0

kinds of affirmation

kinds of negation

	$\circledcirc^1_1$
1	0
0	1

	$\circledcirc^2_1$
1	0
0	0
0	1

	$\circledcirc^3_1$
1	0
1	1
0	1

	$\circledcirc^4_1$
1	0
1	1
0	0
0	1

Sine qua non

A **negative constructor** must be (iteratively) non-assertion-preserving and non-refutation-preserving, as well as completely antitonic.

What is a ‘Negative’ Constructor?

	$\circledcirc_2^3$
1	1
1	0
0	0

	$\circledcirc_2^2$
1	1
0	1
0	0

	$\circledcirc_2^1$
1	1
0	0

kinds of affirmation

kinds of negation

	$\circledcirc_1^1$
1	0
0	1

	$\circledcirc_1^2$
1	0
0	0
0	1

	$\circledcirc_1^3$
1	0
1	1
0	1

	$\circledcirc_1^4$
1	0
1	1
0	0
0	1

[PureRules, 2005]

A *minimally decent* negation $\sim$ is one such that:

$\Gamma, \alpha \nVdash \sim\alpha, \Delta \qquad\qquad \Gamma, \sim\alpha \nVdash \alpha, \Delta$

Sine qua non

A negative constructor must be

What is a 'Negative' Constructor?

	$\circledcirc_2^3$
1	1
1	0
0	0

	$\circledcirc_2^2$
1	1
0	1
0	0

	$\circledcirc_2^1$
1	1
0	0

kinds of affirmation

kinds of negation

	$\circledcirc_1^1$
1	0
0	1

	$\circledcirc_1^2$
1	0
0	0
0	1

	$\circledcirc_1^3$
1	0
1	1
0	1

	$\circledcirc_1^4$
1	0
1	1
0	0
0	1

[PureRules, 2005]

A *minimally decent* negation $\sim$ is one such that:

$\Gamma, \alpha \not\Vdash \sim\alpha, \Delta$ $\qquad$ $\Gamma, \sim\alpha \not\Vdash \alpha, \Delta$

In particular, given *weakening*:

$\Gamma \not\Vdash \sim\alpha, \Delta$ $\qquad$ $\Gamma, \sim\alpha \not\Vdash \Delta$

What is a ‘Negative’ Constructor?

[PureRules, 2005]

An *iteratively minimally decent* negation $\sim$ is one such that, for each n:

$\Gamma, \sim^n\alpha \nVdash \sim^{n+1}\alpha, \Delta$ $\Gamma, \sim^{n+1}\alpha \nVdash \sim^n\alpha, \Delta$

Sine qua non

A **negative constructor** must be
(iteratively) non-assertion-preserving and non-refutation-preserving,
as well as completely antitonic.

What is a ‘Negative’ Constructor?

Some properties that a negative constructor should fail to have

Let ⓒ be an *m*-ary connective.

Sine qua non

A **negative constructor** must be
(iteratively) non-assertion-preserving and non-refutation-preserving,
as well as completely antitonic.

What is a 'Negative' Constructor?

Some properties that a negative constructor should fail to have

Let © be an m-ary connective.

Say that © is *assertion-preserving* in case:

$v(p_1) = \ldots = v(p_m) = 1 \Rightarrow v(©(p_1, \ldots, p_m)) = 1$

Examples: $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$

Say that © is *refutation-preserving* in case:

$v(p_1) = \ldots = v(p_m) = 0 \Rightarrow v(©(p_1, \ldots, p_m)) = 0$

Examples: $\wedge$, $\vee$, $\multimap$ and $+$

Sine qua non

A **negative constructor** must be
(iteratively) non-assertion-preserving and non-refutation-preserving,
as well as completely antitonic.

What is a ‘Negative’ Constructor?

Some properties that a negative constructor should fail to have

Let ⓒ be an m-ary connective.

Say that ⓒ is *assertion-preserving* in case:

$v(p_1) = \ldots = v(p_m) = 1 \quad \Rightarrow \quad v(ⓒ(p_1, \ldots, p_m)) = 1$

Examples: $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$

Say that ⓒ is *refutation-preserving* in case:

$v(p_1) = \ldots = v(p_m) = 0 \quad \Rightarrow \quad v(ⓒ(p_1, \ldots, p_m)) = 0$

Examples: $\wedge$, $\vee$, $\multimap$ and $+$

Sine qua non

A **negative constructor** must be (iteratively) non-assertion-preserving and non-refutation-preserving, as well as completely antitonic.

What is a 'Negative' Constructor?

Some properties that a negative constructor should fail to have

Let © be an m-ary connective.

Say that © is *assertion-preserving* in case:

$v(p_1) = \ldots = v(p_m) = 1 \quad \Rightarrow \quad v(©(p_1, \ldots, p_m)) = 1$

Examples: $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$

Say that © is *refutation-preserving* in case:

$v(p_1) = \ldots = v(p_m) = 0 \quad \Rightarrow \quad v(©(p_1, \ldots, p_m)) = 0$

Examples: $\wedge$, $\vee$, $\multimap$ and $+$

Sine qua non

A **negative constructor** must be
(iteratively) non-assertion-preserving and non-refutation-preserving,
as well as completely antitonic.

What is a ‘Negative’ Constructor?

Some properties that a negative constructor should fail to have

Let ⓒ be an m-ary connective.

Say that ⓒ is *assertion-preserving* in case:

$v(p_1) = \ldots = v(p_m) = 1 \quad \Rightarrow \quad v(ⓒ(p_1, \ldots, p_m)) = 1$

Examples: $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$

Say that ⓒ is *refutation-preserving* in case:

$v(p_1) = \ldots = v(p_m) = 0 \quad \Rightarrow \quad v(ⓒ(p_1, \ldots, p_m)) = 0$

Examples: $\wedge$, $\vee$, $\multimap$ and $+$

Sine qua non

A **negative constructor** must be
(iteratively) non-assertion-preserving and non-refutation-preserving,
as well as completely antitonic.

What is a 'Negative' Constructor?

Some properties that a negative constructor should fail to have

Let © be an *m*-ary connective.

Say now that © is *monotonic* over its *i*-th argument if:

$v(p_i) \leq v(q_i) \;\Rightarrow\; v(©(\ldots)) \leq v(©(\ldots)[p_i \mapsto q_i])$

Examples:

$\wedge$ and $\vee$ are monotonic over both arguments

$\to$ and $\multimap$ are monotonic only over the 2nd argument

A constructor will be called *completely antitonic* if it is non-monotonic over each of its arguments.

Examples:

$\sim$ (both $\smile$ and $\frown$), $\uparrow$ and $\downarrow$

Sine qua non

A **negative constructor** must be (iteratively) non-assertion-preserving and non-refutation-preserving, as well as completely antitonic.

What is a 'Negative' Constructor?

Some properties that a negative constructor should fail to have

Let © be an m-ary connective.

Say now that © is *monotonic* over its i-th argument if:

$v(p_i) \leq v(q_i) \;\; \Rightarrow \;\; v(©(\ldots)) \leq v(©(\ldots)[p_i \mapsto q_i])$

Examples:

$\wedge$ and $\vee$ are monotonic over both arguments

$\rightarrow$ and $\multimap$ are monotonic only over the 2nd argument

A constructor will be called *completely antitonic*
if it is non-monotonic over each of its arguments.

Examples:

$\sim$ (both $\smile$ and $\frown$), $\uparrow$ and $\downarrow$

Sine qua non

A **negative constructor** must be
(iteratively) non-assertion-preserving and non-refutation-preserving,
as well as completely antitonic.

What is a 'Negative' Constructor?

Some properties that a negative constructor should fail to have

Let © be an *m*-ary connective.

Say now that © is *monotonic* over its *i*-th argument if:

$v(p_i) \leq v(q_i) \quad \Rightarrow \quad v(©(\ldots)) \leq v(©(\ldots)[p_i \mapsto q_i])$

Examples:

$\wedge$ and $\vee$ are monotonic over both arguments

$\rightarrow$ and $\multimap$ are monotonic only over the 2nd argument

A constructor will be called *completely antitonic*

if it is non-monotonic over each of its arguments.

Examples:

$\sim$ (both $\smile$ and $\frown$), $\uparrow$ and $\downarrow$

Sine qua non

A **negative constructor** must be

(iteratively) non-assertion-preserving and non-refutation-preserving,

as well as completely antitonic.

What is a ‘Negative’ Constructor?

Some properties that a negative constructor should fail to have

Let $\copyright$ be an m-ary connective.

Say now that $\copyright$ is *monotonic* over its i-th argument if:
$v(p_i) \leq v(q_i) \;\Rightarrow\; v(\copyright(\ldots)) \leq v(\copyright(\ldots)[p_i \mapsto q_i])$
Examples:
$\wedge$ and $\vee$ are monotonic over both arguments
$\rightarrow$ and $\multimap$ are monotonic only over the 2nd argument

A constructor will be called *completely antitonic*
if it is non-monotonic over each of its arguments.
Examples:
$\sim$ (both $\smile$ and $\frown$), $\uparrow$ and $\downarrow$

Sine qua non

A **negative constructor** must be
(iteratively) non-assertion-preserving and non-refutation-preserving,
as well as completely antitonic.

What is a 'Negative' Constructor?

Disclosing some further lessons about negation

From an abstract perspective:

© is *assertion-preserving* in case $\Gamma, \alpha_1, \ldots, \alpha_m \Vdash ©(\alpha_1, \ldots, \alpha_m), \Delta$

© is *refutation-preserving* in case $\Gamma, ©(\alpha_1, \ldots, \alpha_m) \Vdash \alpha_1, \ldots, \alpha_m, \Delta$

© is *monotonic over its i-th argument* if

$\Gamma, \alpha \Vdash \beta, \Delta \;\Rightarrow\; \Gamma, ©(\ldots, p_i, \ldots)[p_i \mapsto \alpha] \Vdash ©(\ldots, p_i, \ldots)[p_i \mapsto \beta], \Delta$

Sine qua non

A **negative constructor** must be

(iteratively) non-assertion-preserving and non-refutation-preserving,

as well as completely antitonic.

What is a 'Negative' Constructor?

Disclosing some further lessons about negation

From an abstract perspective:

© is *assertion-preserving* in case $\Gamma, \alpha_1, \ldots, \alpha_m \Vdash ©(\alpha_1, \ldots, \alpha_m), \Delta$

© is *refutation-preserving* in case $\Gamma, ©(\alpha_1, \ldots, \alpha_m) \Vdash \alpha_1, \ldots, \alpha_m, \Delta$

© is *monotonic over its i-th argument* if

$\Gamma, \alpha \Vdash \beta, \Delta \;\Rightarrow\; \Gamma, ©(\ldots, p_i, \ldots)[p_i \mapsto \alpha] \Vdash ©(\ldots, p_i, \ldots)[p_i \mapsto \beta], \Delta$

Sine qua non

A **negative constructor** must be

(iteratively) non-assertion-preserving and non-refutation-preserving,

as well as completely antitonic.

What is a 'Negative' Constructor?

Disclosing some further lessons about negation

From an abstract perspective:

Ⓒ is *assertion-preserving* in case $\Gamma, \alpha_1, \ldots, \alpha_m \Vdash Ⓒ(\alpha_1, \ldots, \alpha_m), \Delta$

Ⓒ is *refutation-preserving* in case $\Gamma, Ⓒ(\alpha_1, \ldots, \alpha_m) \Vdash \alpha_1, \ldots, \alpha_m, \Delta$

Ⓒ is *monotonic over its i-th argument* if

$\Gamma, \alpha \Vdash \beta, \Delta \;\Rightarrow\; \Gamma, Ⓒ(\ldots, p_i, \ldots)[p_i \mapsto \alpha] \Vdash Ⓒ(\ldots, p_i, \ldots)[p_i \mapsto \beta], \Delta$

Sine qua non

A **negative constructor** must be
(iteratively) non-assertion-preserving and non-refutation-preserving,
as well as completely antitonic.

What is a 'Negative' Constructor?

[PureRules, 2005]

An *iteratively minimally decent* negation $\sim$ is one such that, for each n:

$$\Gamma, \sim^n\alpha \nVdash \sim^{n+1}\alpha, \Delta \qquad \Gamma, \sim^{n+1}\alpha \nVdash \sim^n\alpha, \Delta$$

Disclosing some further lessons about negation

From an abstract perspective:

© is *assertion-preserving* in case $\Gamma, \alpha_1, \ldots, \alpha_m \Vdash ©(\alpha_1, \ldots, \alpha_m), \Delta$

© is *refutation-preserving* in case $\Gamma, ©(\alpha_1, \ldots, \alpha_m) \Vdash \alpha_1, \ldots, \alpha_m, \Delta$

© is *monotonic over its i-th argument* if

$$\Gamma, \alpha \Vdash \beta, \Delta \;\Rightarrow\; \Gamma, ©(\ldots, p_i, \ldots)[p_i \mapsto \alpha] \Vdash ©(\ldots, p_i, \ldots)[p_i \mapsto \beta], \Delta$$

Sine qua non

A **negative constructor** must be
(iteratively) non-assertion-preserving and non-refutation-preserving,
as well as completely antitonic.

What is a 'Negative' Constructor?

[PureRules, 2005]

An *iteratively minimally decent* negation $\sim$ is one such that, for each n:

$\Gamma, \sim^n\alpha \nVdash \sim^{n+1}\alpha, \Delta$ $\qquad\qquad$ $\Gamma, \sim^{n+1}\alpha \nVdash \sim^n\alpha, \Delta$

Disclosing some further lessons about negation

From an abstract perspective:

© is *assertion-preserving* in case $\Gamma, \alpha_1, \ldots, \alpha_m \Vdash ©(\alpha_1, \ldots, \alpha_m), \Delta$

© is *refutation-preserving* in case $\Gamma, ©(\alpha_1, \ldots, \alpha_m) \Vdash \alpha_1, \ldots, \alpha_m, \Delta$

© is *monotonic over its i-th argument* if

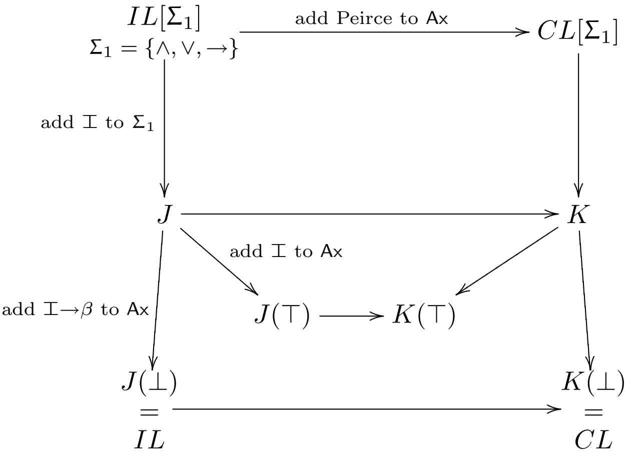
$\Gamma, \alpha \Vdash \beta, \Delta \;\Rightarrow\; \Gamma, ©(\ldots, p_i, \ldots)[p_i \mapsto \alpha] \Vdash ©(\ldots, p_i, \ldots)[p_i \mapsto \beta], \Delta$

Sine qua non

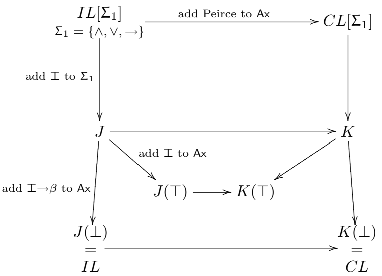
A **negative constructor** must be
(iteratively) non-assertion-preserving and non-refutation-preserving,
as well as completely antitonic.

Negation as You Like It [NTF, 2009]

Consider the following systems:



Here are some remarkable valid inferences:



Assume $\sim\alpha \stackrel{\text{def}}{=} \smile\alpha \stackrel{\text{def}}{=} \alpha \to$ ⌶.

in J:

$\alpha, \sim\alpha \Vdash \sim\beta$

$\alpha \to \beta, \alpha \to \sim\beta \Vdash \sim\alpha$

$\alpha \Vdash \sim\sim\alpha$

in $J(\bot)$:

$\alpha, \sim\alpha \Vdash \beta$

in K:

$\alpha \to \sim\alpha \Vdash \sim\alpha$

$\alpha \to \beta, \sim\alpha \to \beta \Vdash \beta$

$\Vdash \alpha, \sim\alpha$

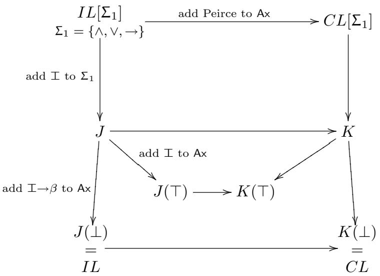
in $K(\bot)$:

$\sim\alpha \to \alpha \Vdash \alpha$

$\sim\sim\alpha \Vdash \alpha$

Negation as You Like It [NTF, 2009]

Here are some remarkable valid inferences:



Assume $\sim\alpha \overset{\text{def}}{=} \smile\alpha \overset{\text{def}}{=} \alpha \to ⌶$.

in J:

$\alpha, \sim\alpha \Vdash \sim\beta$

$\alpha \to \beta, \alpha \to \sim\beta \Vdash \sim\alpha$

$\alpha \Vdash \sim\sim\alpha$

in $J(\bot)$:

$\alpha, \sim\alpha \Vdash \beta$

in K:

$\alpha \to \sim\alpha \Vdash \sim\alpha$

$\alpha \to \beta, \sim\alpha \to \beta \Vdash \beta$

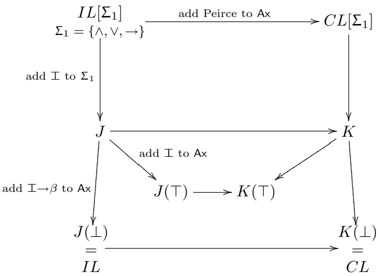
$\Vdash \alpha, \sim\alpha$

in $K(\bot)$:

$\sim\alpha \to \alpha \Vdash \alpha$

$\sim\sim\alpha \Vdash \alpha$

Here are some remarkable valid inferences:



Assume $\sim \alpha \overset{\text{def}}{=} \smile\alpha \overset{\text{def}}{=} \alpha \to \text{⌶}$.

in J:

$\alpha, \sim\alpha \Vdash \sim\beta$

$\alpha \to \beta, \alpha \to \sim\beta \Vdash \sim\alpha$

$\alpha \Vdash \sim\sim\alpha$

in $J(\bot)$:

$\alpha, \sim\alpha \Vdash \beta$

in K:

$\alpha \to \sim\alpha \Vdash \sim\alpha$

$\alpha \to \beta, \sim\alpha \to \beta \Vdash \beta$

$\Vdash \alpha, \sim\alpha$

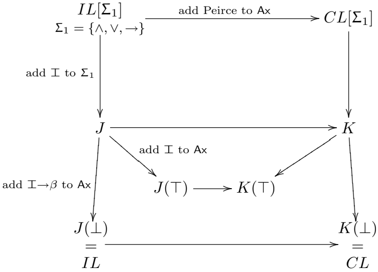
in $K(\bot)$:

$\sim\alpha \to \alpha \Vdash \alpha$

$\sim\sim\alpha \Vdash \alpha$

Negation as You Like It [NTF, 2009]

Here are some remarkable valid inferences:



Assume $\sim \alpha \overset{\text{def}}{=} \smile\alpha \overset{\text{def}}{=} \alpha \rightarrow$ ⌶.

in J:

$\alpha, \sim\alpha \Vdash \sim\beta$

$\alpha \rightarrow \beta, \alpha \rightarrow \sim\beta \Vdash \sim\alpha$

$\alpha \Vdash \sim\sim\alpha$

in $J(\bot)$:

$\alpha, \sim\alpha \Vdash \beta$

in K:

$\alpha \rightarrow \sim\alpha \Vdash \sim\alpha$

$\alpha \rightarrow \beta, \sim\alpha \rightarrow \beta \Vdash \beta$

$\Vdash \alpha, \sim\alpha$

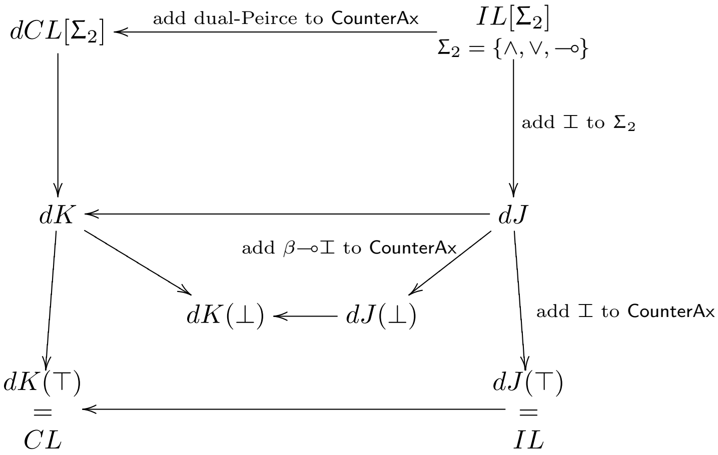
in $K(\bot)$:

$\sim\alpha \rightarrow \alpha \Vdash \alpha$

$\sim\sim\alpha \Vdash \alpha$

Negation as You Might Imagine It

Consider next the following dual systems:



Assume $\sim \alpha \stackrel{\text{def}}{=} \frown\alpha \stackrel{\text{def}}{=} \alpha \multimap \top$.

A Non-deterministic Approach

On truth-tables

Let © be an m-ary constructor, and v a valuation.

Deterministic approach:

(D1) $© : \mathcal{V}^m \longrightarrow \mathcal{V}$ is a total mapping s.t.:

(D2) $v(©(\alpha_1, \ldots, \alpha_m)) = ©(v(\alpha_1), \ldots, v(\alpha_m))$

Non-deterministic approach:

(ND1) $© : \mathcal{V}^m \longrightarrow \mathrm{Pow}(\mathcal{V}) \setminus \varnothing$ is a total mapping s.t.:

(ND2) $v(©(\alpha_1, \ldots, \alpha_m)) \in ©(v(\alpha_1), \ldots, v(\alpha_m))$

Example (On negation)

Paraconsistent:

α	$\smile \alpha$
0	$\{1\}$
1	$\{0, 1\}$

Paracomplete:

α	$\frown \alpha$
0	$\{0, 1\}$
1	$\{0\}$

A Non-deterministic Approach

On truth-tables

Let © be an m-ary constructor, and v a valuation.

Deterministic approach:
(D1) $© : \mathcal{V}^m \longrightarrow \mathcal{V}$ is a total mapping s.t.:
(D2) $v(©(\alpha_1, \ldots, \alpha_m)) = ©(v(\alpha_1), \ldots, v(\alpha_m))$

Non-deterministic approach:
(ND1) $© : \mathcal{V}^m \longrightarrow \mathrm{Pow}(\mathcal{V}) \setminus \varnothing$ is a total mapping s.t.:
(ND2) $v(©(\alpha_1, \ldots, \alpha_m)) \in ©(v(\alpha_1), \ldots, v(\alpha_m))$

Example (On negation)

Paraconsistent:

α	$\smile \alpha$
0	$\{1\}$
1	$\{0, 1\}$

Paracomplete:

α	$\frown \alpha$
0	$\{0, 1\}$
1	$\{0\}$

A Non-deterministic Approach

On truth-tables

Let ⓒ be an m-ary constructor, and v a valuation.

Deterministic approach:
(D1) ⓒ : $\mathcal{V}^m \longrightarrow \mathcal{V}$ is a total mapping s.t.:
(D2) $v(ⓒ(\alpha_1, \ldots, \alpha_m)) = ⓒ(v(\alpha_1), \ldots, v(\alpha_m))$

Non-deterministic approach:
(ND1) ⓒ : $\mathcal{V}^m \longrightarrow \text{Pow}(\mathcal{V}) \setminus \varnothing$ is a total mapping s.t.:
(ND2) $v(ⓒ(\alpha_1, \ldots, \alpha_m)) \in ⓒ(v(\alpha_1), \ldots, v(\alpha_m))$

Example (On negation)

Paraconsistent:

α	$\smile \alpha$
0	$\{1\}$
1	$\{0, 1\}$

Paracomplete:

α	$\frown \alpha$
0	$\{0, 1\}$
1	$\{0\}$

A Non-deterministic Approach

Example (On negation)

Paraconsistent:

α	$\smile\alpha$
0	$\{1\}$
1	$\{0,1\}$

Paracomplete:

α	$\frown\alpha$
0	$\{0,1\}$
1	$\{0\}$

Interpretations for K and dK (adaptable for J and dJ)

Assume the classical interpretations of $\{\wedge, \vee, \rightarrow, \multimap\}$ over $\{0,1\}$.

Interpret $\bot$ non-deterministically by setting
$\bot : \varnothing \longrightarrow \{0,1\}$, i.e., allow $v(\bot) \in \{0,1\}$.
You may now in fact *define*:

$\smile\alpha \stackrel{\text{def}}{=} \alpha \rightarrow \bot$

$\frown\alpha \stackrel{\text{def}}{=} \alpha \multimap \bot$

A Non-deterministic Approach

Example (On negation)

Paraconsistent:

α	$\smile \alpha$
0	$\{1\}$
1	$\{0, 1\}$

Paracomplete:

α	$\frown \alpha$
0	$\{0, 1\}$
1	$\{0\}$

Interpretations for K and dK (adaptable for J and dJ)

Assume the classical interpretations of $\{\wedge, \vee, \rightarrow, \multimap\}$ over $\{0, 1\}$.

Interpret $\top\!\!\!\bot$ non-deterministically by setting $\top\!\!\!\bot : \varnothing \longrightarrow \{0, 1\}$, i.e., allow $v(\top\!\!\!\bot) \in \{0, 1\}$.

You may now in fact *define*:

$\smile\alpha \stackrel{\text{def}}{=} \alpha \rightarrow \top\!\!\!\bot$

$\frown\alpha \stackrel{\text{def}}{=} \alpha \multimap \top\!\!\!\bot$

A Non-deterministic Approach

Example (On negation)

Paraconsistent:

α	$\smile\alpha$
0	$\{1\}$
1	$\{0,1\}$

Paracomplete:

α	$\frown\alpha$
0	$\{0,1\}$
1	$\{0\}$

Interpretations for K and dK (adaptable for J and dJ)

Assume the classical interpretations of $\{\wedge, \vee, \rightarrow, \multimap\}$ over $\{0,1\}$.

Interpret I non-deterministically by setting
$\mathrm{I} : \varnothing \longrightarrow \{0,1\}$, i.e., allow $v(\mathrm{I}) \in \{0,1\}$.
You may now in fact *define*:

$\smile\alpha \stackrel{\text{def}}{=} \alpha \rightarrow \mathrm{I}$

$\frown\alpha \stackrel{\text{def}}{=} \alpha \multimap \mathrm{I}$