Each normal logic program has a 2-valued Minimal Hypotheses semantics

Alexandre Miguel Pinto and Luís Moniz Pereira

Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa

September 30, 2011
1. Introduction
2. Building a semantics
3. Minimal Hypotheses semantics
4. Conclusions and Future Work

Each NLP has a 2-valued MH semantics.
Outline

1. Introduction
2. Building a semantics
3. Minimal Hypotheses semantics
4. Conclusions and Future Work

September 30th, INAP 2011, Vienna

Each NLP has a 2-valued MH semantics
Knowledge can be written in sentences
- Sentences can be translated into logic formalisms
- Logic Programs are one such formalism for KR
- LP = Normal Logic Rules + Integrity Constraints
Knowledge can be written in sentences

Sentences can be translated into logic formalisms

Logic Programs are one such formalism for KR

LP = Normal Logic Rules + Integrity Constraints
Knowledge can be written in sentences

Sentences can be translated into logic formalisms

Logic Programs are one such formalism for KR

LP = Normal Logic Rules + Integrity Constraints
Knowledge can be written in sentences
Sentences can be translated into logic formalisms
Logic Programs are one such formalism for KR
LP = Normal Logic Rules + Integrity Constraints
Logic Rule (LR)

A LR is of the form $h \leftarrow b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m$ with $m, n \geq 0$ and finite, and where $b_i, c_j$ are ground atoms.

Notation: $\text{head}(r) = h$ and $\text{body}(r) = \{b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m\}$.

Normal Logic Program (NLP)

A NLP is a (countable) set of Normal Logic Rules (NLRs), where a NLR $r$ is s.t. $\text{head}(r)$ is a ground atom.

Integrity Constraint (IC)

A IC is a LR $r$ such that $\text{head}(r) = \bot$. 
Background (2/2)

Logic Rule (LR)
A LR is of the form $h \leftarrow b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m$ with $m, n \geq 0$ and finite, and where $b_i, c_j$ are ground atoms.

Notation: $\text{head}(r) = h$ and $\text{body}(r) = \{b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m\}$.

Normal Logic Program (NLP)
A NLP is a (countable) set of Normal Logic Rules (NLRs), where a NLR $r$ is s.t. $\text{head}(r)$ is a ground atom.

Integrity Constraint (IC)
A IC is a LR $r$ such that $\text{head}(r) = \bot$. Each NLP has a 2-valued MH semantics.
Logic Rule (LR)

A LR is of the form $h \leftarrow b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m$ with $m, n \geq 0$ and finite, and where $b_i, c_j$ are ground atoms.

Notation: $\text{head}(r) = h$ and $\text{body}(r) = \{b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m\}$.

Normal Logic Program (NLP)

A NLP is a (countable) set of Normal Logic Rules (NLRs), where a NLR $r$ is s.t. $\text{head}(r)$ is a ground atom.

Integrity Constraint (IC)

A IC is a LR $r$ such that $\text{head}(r) = \bot$. 
Ideally, KR Declarativity with LPs would allow

Full separation of concerns of rules, i.e.,
KR role of NLRs $\neq$ KR role of ICs, i.e.,

- Normal Logic Rules $\Leftrightarrow$ Define space of candidate solutions
- Integrity Constraints $\Leftrightarrow$ Prune out undesired candidates

Semantics is what determines which sets of beliefs (models) are candidates
Ideally, KR Declarativity with LPs would allow Full separation of concerns of rules, i.e., KR role of NLRs $\neq$ KR role of ICs, i.e.,

- Normal Logic Rules $\Leftrightarrow$ Define space of candidate solutions
- Integrity Constraints $\Leftrightarrow$ Prune out undesired candidates

Semantics is what determines which sets of beliefs (models) are candidates.
The Problem (1/2)

- Ideally, KR Declarativity with LPs would allow
  Full separation of concerns of rules, i.e.,
  KR role of NLRs $\neq$ KR role of ICs, i.e.,
  - Normal Logic Rules $\leftrightarrow$ Define space of candidate solutions
  - Integrity Constraints $\leftrightarrow$ Prune out undesired candidates
- Semantics is what determines which sets of beliefs (models) are candidates
The Problem (1/2)

- Ideally, KR Declarativity with LPs would allow
  Full separation of concerns of rules, i.e.,
  KR role of NLRs ≠ KR role of ICs, i.e.,
  - Normal Logic Rules ⇔ Define space of candidate solutions
  - Integrity Constraints ⇔ Prune out undesired candidates

- Semantics is what determines which sets of beliefs
  (models) are candidates
Ideally, KR Declarativity with LPs would allow
Full separation of concerns of rules, i.e.,
KR role of NLRs ≠ KR role of ICs, i.e.,
- Normal Logic Rules ⇔ Define space of candidate solutions
- Integrity Constraints ⇔ Prune out undesired candidates

Semantics is what determines which sets of beliefs
(models) are candidates
The Problem (1/2)

- Ideally, KR Declarativity with LPs would allow
  Full separation of concerns of rules, i.e.,
  KR role of NLRs $\neq$ KR role of ICs, i.e.,
  - Normal Logic Rules $\iff$ Define space of candidate solutions
  - Integrity Constraints $\iff$ Prune out undesired candidates

- Semantics is what determines which sets of beliefs (models) are candidates
According to this Ideal KR Declarativity view... 
...since only ICs are allowed to prune out candidate models...
...LPs with no ICs (i.e., NLPs) should always allow, at least, one model...
...otherwise NLRs would be allowed to play the role of ICs too
This means: a semantics for NLPs should guarantee model existence
The Problem (2/2)

- According to this Ideal KR Declarativity view...
- ...since only ICs are allowed to prune out candidate models...
- ...LPs with no ICs (i.e., NLPs) should always allow, at least, one model...
- ...otherwise NLRs would be allowed to play the role of ICs too
- This means: a semantics for NLPs should guarantee model existence

Each NLP has a 2-valued MH semantics
According to this Ideal KR Declarativity view...

...since only ICs are allowed to prune out candidate models...

...LPs with no ICs (i.e., NLPs) should always allow, at least, one model...

...otherwise NLRs would be allowed to play the role of ICs too

This means: a semantics for NLPs should guarantee model existence
According to this Ideal KR Declarativity view. . .

. . . since only ICs are allowed to prune out candidate models. . .

. . . LPs with no ICs (i.e., NLPs) should always allow, at least, one model. . .

. . . otherwise NLRs would be allowed to play the role of ICs too

This means: a semantics for NLPs should guarantee model existence
According to this Ideal KR Declarativity view…

…since only ICs are allowed to prune out candidate models...

…LPs with no ICs (i.e., NLPs) should always allow, at least, one model...

…otherwise NLRs would be allowed to play the role of ICs too

This means: a semantics for NLPs should guarantee model existence

Each NLP has a 2-valued MH semantics
Common semantics for NLPs

- **2-valued: Stable Models**
  - Classically Supported, but...
  - Lacks guarantee of model existence — undermining liveness
cannot provide semantics to arbitrarily updated/merged programs
  - Lacks relevance
does not allow for top-down query-answering proof-procedures
  - Lacks cumulativity
does not allow use of tabling methods to speed up computations

- **3-valued: Well-Founded Semantics**
  - Classically Supported
  - Existence of Model
  - Relevance
  - Cumulativity, but...
in it is 3-valued
Common semantics for NLPs

- **2-valued: Stable Models**
  - Classically Supported, but...
  - Lacks guarantee of model existence — undermining liveness
  - cannot provide semantics to arbitrarily updated/merged programs
  - Lacks relevance
    - does not allow for top-down query-answering proof-procedures
  - Lacks cumulativity
    - does not allow use of tabling methods to speed up computations

- **3-valued: Well-Founded Semantics**
  - Classically Supported
  - Existence of Model
  - Relevance
  - Cumulativity, but...
  - it is 3-valued
Common semantics for NLPs

- **2-valued: Stable Models**
  - Classically Supported, but... 
  - Lacks guarantee of model existence — undermining liveness 
    cannot provide semantics to arbitrarily updated/merged programs 
  - Lacks relevance 
    does not allow for top-down query-answering proof-procedures 
  - Lacks cumulativity 
    does not allow use of tabling methods to speed up computations 

- **3-valued: Well-Founded Semantics**
  - Classically Supported 
  - Existence of Model 
  - Relevance 
  - Cumulativity, but... 
  - it is 3-valued
Common semantics for NLPs

- **2-valued: Stable Models**
  - Classically Supported, but...
  - Lacks guarantee of model existence — undermining liveness
can not provide semantics to arbitrarily updated/merged programs
  - Lacks relevance
does not allow for top-down query-answering proof-procedures
  - Lacks cumulativity
do not allow use of tabling methods to speed up computations

- **3-valued: Well-Founded Semantics**
  - Classically Supported
  - Existence of Model
  - Relevance
  - Cumulativity, but...
it is 3-valued
Common semantics for NLPs

2-valued: Stable Models
- Classically Supported, but...
- Lacks guarantee of model existence — undermining liveness
  cannot provide semantics to arbitrarily updated/merged programs
- Lacks relevance
  does not allow for top-down query-answering proof-procedures
- Lacks cumulativity
  does not allow use of tabling methods to speed up computations

3-valued: Well-Founded Semantics
- Classically Supported
- Existence of Model
- Relevance
- Cumulativity, but...
- it is 3-valued
Common semantics for NLPs

- **2-valued: Stable Models**
  - Classically Supported, but... 
  - Lacks guarantee of model existence — undermining liveness 
    - cannot provide semantics to arbitrarily updated/merged programs 
  - Lacks relevance 
    - does not allow for top-down query-answering proof-procedures 
  - Lacks cumulativity 
    - does not allow use of tabling methods to speed up computations

- **3-valued: Well-Founded Semantics**
  - Classically Supported 
  - Existence of Model 
  - Relevance 
  - Cumulativity, but... 
  - it is 3-valued
2-valued: Stable Models
- Classically Supported, but...
- Lacks guarantee of model existence — undermining liveness
cannot provide semantics to arbitrarily updated/merged programs
- Lacks relevance
does not allow for top-down query-answering proof-procedures
- Lacks cumulativity
does not allow use of tabling methods to speed up computations

3-valued: Well-Founded Semantics
- Classically Supported
  - Existence of Model
  - Relevance
  - Cumulativity, but...
  - it is 3-valued
Common semantics for NLPs

- **2-valued: Stable Models**
  - Classically Supported, but...
  - Lacks guarantee of model existence — undermining liveness
    cannot provide semantics to arbitrarily updated/merged programs
  - Lacks relevance
    does not allow for top-down query-answering proof-procedures
  - Lacks cumulativity
    does not allow use of tabling methods to speed up computations

- **3-valued: Well-Founded Semantics**
  - Classically Supported
  - Existence of Model
    - Relevance
    - Cumulativity, but...
    - it is 3-valued

Each NLP has a 2-valued MH semantics
Common semantics for NLPs

- **2-valued: Stable Models**
  - Classically Supported, but...
  - Lacks guarantee of model existence — undermining liveness
  - cannot provide semantics to arbitrarily updated/merged programs
  - Lacks relevance
    - does not allow for top-down query-answering proof-procedures
  - Lacks cumulativity
    - does not allow use of tabling methods to speed up computations

- **3-valued: Well-Founded Semantics**
  - Classically Supported
  - Existence of Model
  - Relevance
    - Cumulativity, but...
  - it is 3-valued

Each NLP has a 2-valued MH semantics
Common semantics for NLPs

- **2-valued: Stable Models**
  - Classically Supported, but...
  - Lacks guarantee of model existence — undermining liveness
    cannot provide semantics to arbitrarily updated/merged programs
  - Lacks relevance
    does not allow for top-down query-answering proof-procedures
  - Lacks cumulativity
    does not allow use of tabling methods to speed up computations

- **3-valued: Well-Founded Semantics**
  - Classically Supported
  - Existence of Model
  - Relevance
  - Cumulativity, but...
  - it is 3-valued
Common semantics for NLPs

- **2-valued: Stable Models**
  - Classically Supported, but...
  - Lacks guarantee of model existence — undermining liveness
    cannot provide semantics to arbitrarily updated/merged programs
  - Lacks relevance
    does not allow for top-down query-answering proof-procedures
  - Lacks cumulativity
    does not allow use of tabling methods to speed up computations

- **3-valued: Well-Founded Semantics**
  - Classically Supported
  - Existence of Model
  - Relevance
  - Cumulativity, but...
  - it is 3-valued
Goal: Desired Semantics (1/2)

- 2-valued semantics
  - Guarantee of model existence
    (e.g., for liveness in the face of updates)
  - Relevance
    Truth of a literal is determined solely by the literals in its call-graph
  - Cumulativity
    Atoms true in all models can be safely added to program as facts
Goal: Desired Semantics (1/2)

- 2-valued semantics
- Guarantee of model existence
  (e.g., for liveness in the face of updates)
- Relevance
  Truth of a literal is determined solely by the literals in its call-graph
- Cumulativity
  Atoms true in all models can be safely added to program as facts
Goal: Desired Semantics (1/2)

- 2-valued semantics
- Guarantee of model existence
  (e.g., for liveness in the face of updates)
- Relevance
  Truth of a literal is determined solely by the literals in its call-graph
- Cumulativity
  Atoms true in all models can be safely added to program as facts
Goal: Desired Semantics (1/2)

- 2-valued semantics
- Guarantee of model existence
  (e.g., for liveness in the face of updates)
- Relevance
  Truth of a literal is determined solely by the literals in its call-graph
- Cumulativity
  Atoms true in all models can be safely added to program as facts
Outline

1. Introduction
2. Building a semantics
3. Minimal Hypotheses semantics
4. Conclusions and Future Work

Each NLP has a 2-valued MH semantics
Closed World Assumption \[\rightsquigarrow\]

Programs with stratified negation have only one model. . .
. . . which hints that. . .
. . . non-determinism (more than one model) comes from
“non-stratification” of negation
I.e., negated literals \textit{in loop} play important semantic role
Need for a different kind of Assumption for non-stratified negation
A model can be seen as set of \textit{assumed hypotheses} + their conclusions
Closed World Assumption $\rightsquigarrow$

Programs with stratified negation have only one model... 
...which hints that...
...non-determinism (more than one model) comes from “non-stratification” of negation
I.e., negated literals in loop play important semantic role
Need for a different kind of Assumption for non-stratified negation
A model can be seen as set of assumed hypotheses + their conclusions
Closed World **Assumption**

Programs with stratified negation have only one model... ...which hints that...

...non-determinism (more than one model) comes from “non-stratification” of negation
I.e., negated literals *in loop* play important semantic role

Need for a different kind of **Assumption** for non-stratified negation

A model can be seen as set of *assumed hypotheses* + their conclusions
Closed World Assumption $\leadsto$

Programs with stratified negation have only one model...

... which hints that...

... non-determinism (more than one model) comes from “non-stratification” of negation

I.e., negated literals in loop play important semantic role

Need for a different kind of Assumption for non-stratified negation

A model can be seen as set of assumed hypotheses + their conclusions
Closed World **Assumption**  
Programs with stratified negation have only one model. . .
. . . which hints that. . .
. . . non-determinism (more than one model) comes from “non-stratification” of negation
I.e., negated literals *in loop* play important semantic role
Need for a different kind of **Assumption** for non-stratified negation
A model can be seen as set of *assumed hypotheses* + their conclusions
Closed World **Assumption** ⾃✂
Programs with stratified negation have only one model. . .
. . . which hints that. . .
. . . non-determinism (more than one model) comes from “non-stratification” of negation
I.e., negated literals *in loop* play important semantic role
Need for a different kind of **Assumption** for non-stratified negation

A model can be seen as set of *assumed hypotheses* + their conclusions
Closed World **Assumption**  
Programs with stratified negation have only one model...  
...which hints that...  
...non-determinism (more than one model) comes from  
“non-stratification” of negation  
I.e., negated literals *in loop* play important semantic role  
Need for a different kind of **Assumption** for non-stratified negation  
A model can be seen as set of *assumed hypotheses* + their conclusions
Maximal skepticism $\rightarrow$ atomic beliefs must have support

CWA is particular case

- **Support**
  - Classical
    - belief in atom requires a rule for it (head) with *all* body literals true
    - OR
  - (new) Layered (Stratified)
    - belief in atom requires a rule for it (head) with *non-loop* body literals true

Classical Support $\Rightarrow$ Layered Support

Maximal skepticism $\rightarrow$ minimal atomic beliefs

- **Minimality**
  - Of *all* atoms true in the model
    - OR
  - Of *hypotheses assumed* true in the model
Maximal skepticism $\rightarrow$ atomic beliefs must have **support**

CWA is particular case

- **Support**
  - **Classical**
    belief in atom requires a rule for it (head) with *all* body literals true
  - (new) **Layered (Stratified)**
    belief in atom requires a rule for it (head) with *non-loop* body literals true

Classical Support $\Rightarrow$ Layered Support

Maximal skepticism $\rightarrow$ **minimal** atomic beliefs

- **Minimality**
  - Of *all* atoms true in the model
  - Of *hypotheses assumed* true in the model
Maximal skepticism $\rightarrow$ atomic beliefs must have **support**

CWA is particular case

- **Support**
  - **Classical**
    - belief in atom requires a rule for it (head) with *all* body literals true
    - OR
  - **(new) Layered (Stratified)**
    - belief in atom requires a rule for it (head) with *non-loop* body literals true

Classical Support $\Rightarrow$ Layered Support

Maximal skepticism $\rightarrow$ **minimal** atomic beliefs

- **Minimality**
  - Of *all* atoms true in the model
  - OR
  - Of *hypotheses assumed* true in the model
Maximal skepticism $\rightarrow$ atomic beliefs must have **support**

CWA is particular case

- **Support**
  - Classical
    - belief in atom requires a rule for it (head) with *all* body literals true
    - OR
  - (new) Layered (Stratified)
    - belief in atom requires a rule for it (head) with *non-loop* body literals true
    - Classical Support $\Rightarrow$ Layered Support

Maximal skepticism $\rightarrow$ **minimal** atomic beliefs

- **Minimality**
  - Of *all* atoms true in the model
    - OR
  - Of *hypotheses assumed* true in the model
Maximal skepticism → atomic beliefs must have support
CWA is particular case

- Support
  - Classical
    belief in atom requires a rule for it (head) with all body literals true
  - (new) Layered (Stratified)
    belief in atom requires a rule for it (head) with non-loop body literals true

Classical Support ⇒ Layered Support

Maximal skepticism → minimal atomic beliefs

- Minimality
  - Of all atoms true in the model
  - OR
  - Of hypotheses assumed true in the model
Maximal skepticism $\rightarrow$ atomic beliefs must have **support**

CWA is particular case

- **Support**
  - Classical
    - belief in atom requires a rule for it (head) with *all* body literals true
  - (new) Layered (Stratified)
    - belief in atom requires a rule for it (head) with *non-loop* body literals true
  
  Classical Support $\Rightarrow$ Layered Support

Maximal skepticism $\rightarrow$ **minimal** atomic beliefs

- **Minimality**
  - Of *all* atoms true in the model
  - Of *hypotheses assumed* true in the model
Positive hypotheses or negative hypotheses?
Traditionally, maximum negative hypotheses, but…

Problem with negative hypotheses

\[ a \leftarrow \neg a \]

Assuming neg. hyp. \( \neg a \) leads to contradiction: \( a \)
Assuming pos. hyp. \( a \) does \textit{not} lead to contradiction!

No explicit \( \neg a \) can be derived since we are using NLPs

Minimum positive hypotheses it is, then!
In the example above \( a \) has no Classical Support… but it has Layered Support!
Positive hypotheses or negative hypotheses?
Traditionally, maximum negative hypotheses, but... 

Problem with negative hypotheses

\[ a \leftarrow \neg a \]

Assuming neg. hyp. \( \neg a \) leads to contradiction: \( a \)
Assuming pos. hyp. \( a \) does \textit{not} lead to contradiction!

No explicit \( \neg a \) can be derived since we are using NLPs

Minimum positive hypotheses it is, then!
In the example above \( a \) has no Classical Support... but it has
Layered Support!
Positive hypotheses or negative hypotheses?
Traditionally, maximum negative hypotheses, but...
Positive hypotheses or negative hypotheses?
Traditionally, maximum negative hypotheses, but... 

Problem with negative hypotheses

\[ a \leftarrow \neg a \]

Assuming neg. hyp. \( not \ a \) leads to contradiction: \( a \)
Assuming pos. hyp. \( a \) does \textit{not} lead to contradiction!

No explicit \( \neg a \) can be derived since we are using NLPs

Minimum positive hypotheses it is, then!
In the example above \( a \) has no Classical Support... but it has Layered Support!
Positive hypotheses or negative hypotheses? Traditionally, maximum negative hypotheses, but... 

**Problem with negative hypotheses**

\[ a \leftarrow \neg a \]

Assuming neg. hyp. \textit{not} \(a\) leads to contradiction: \(a\)
Assuming pos. hyp. \(a\) does \textit{not} lead to contradiction!

No explicit \(\neg a\) can be derived since we are using NLPs

Minimum positive hypotheses it is, then!
In the example above \(a\) has no Classical Support... but it has
Layered Support!
Positive hypotheses or negative hypotheses? Traditionally, maximum negative hypotheses, but... 

Problem with negative hypotheses

\[ a \leftarrow \neg a \]

Assuming neg. hyp. \( \neg a \) leads to contradiction: \( a \)
Assuming pos. hyp. \( a \) does \textbf{not} lead to contradiction!
No explicit \( \neg a \) can be derived since we are using NLPs

Minimum positive hypotheses it is, then!
In the example above \( a \) has no Classical Support... but it has Layered Support!
Each NLP has a 2-valued MH semantics
Assumable hypotheses of program:
\( \text{Hyps} = \) (positive) \textbf{atoms} of negative literals in loop

Select a minimal subset \( H \) of \( \text{Hyps} \) assumed true such that

\( H \) is enough to propagate 2-valuedness to all literals

If so, an MH-model is the \( M = \) consequences of \( H \)

Propagation of truth-values via deterministic polynomial-time Remainder operator (generalization of \( T_P \) operator)
Assumable hypotheses of program: 
\(\text{Hyps} = \text{(positive) atoms}\) of negative literals in loop 

Select a minimal subset \(H\) of \(\text{Hyps}\) assumed true such that 

- \(H\) is enough to propagate 2-valuedness to all literals 
- If so, an MH-model is the \(M = \text{consequences of } H\) 

Propagation of truth-values via deterministic polynomial-time Remainder operator (generalization of \(T_P\) operator)
Minimal Hypotheses semantics — Intuitive definition

- Assumable hypotheses of program: $\text{Hyps} = \text{(positive) atoms}$ of negative literals in loop
- Select a minimal subset $H$ of $\text{Hyps}$ assumed true such that
- $H$ is enough to propagate 2-valuedness to all literals
- If so, an MH-model is the $M =$ consequences of $H$

- Propagation of truth-values via deterministic polynomial-time Remainder operator (generalization of $T_P$ operator)
Assumable hypotheses of program: $Hyps = (\text{positive}) \text{ atoms}$ of negative literals in loop
Select a minimal subset $H$ of $Hyps$ assumed true such that $H$ is enough to propagate 2-valuedness to all literals
If so, an MH-model is the $M = \text{consequences of } H$
Propagation of truth-values via deterministic polynomial-time Remainder operator (generalization of $T_P$ operator)
Vacation example

beach ← not mountain
mountain ← not travel
travel ← not beach, passport_ok
passport_ok ← not expired_passport
expired_passport ← not passport_ok
Vacation example

\[
\begin{align*}
&\text{beach} \leftarrow \text{not mountain} \\
&\text{mountain} \leftarrow \text{not travel} \\
&\text{travel} \leftarrow \text{not beach, passport\_ok} \\
&\text{passport\_ok} \leftarrow \text{not expired\_passport} \\
&\text{expired\_passport} \leftarrow \text{not passport\_ok} \\
&\text{expired\_passport}
\end{align*}
\]
Vacation example

- beach $\leftarrow$ not mountain
- mountain $\leftarrow$ not travel
- travel $\leftarrow$ not beach, passport_ok
- expired_passport
MH semantics — Example

Vacation example

- beach $\leftarrow$ not mountain
- mountain $\leftarrow$ not travel
- expired_passport
Each NLP has a 2-valued MH semantics
Vacation example

mountain
expired_passport
### Vacation example

<table>
<thead>
<tr>
<th>Condition</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>beach</td>
<td>$\leftarrow$ not mountain</td>
</tr>
<tr>
<td>mountain</td>
<td>$\leftarrow$ not travel</td>
</tr>
<tr>
<td>travel</td>
<td>$\leftarrow$ not beach, passport_ok</td>
</tr>
<tr>
<td>passport_ok</td>
<td>$\leftarrow$ not expired_passport</td>
</tr>
<tr>
<td>expired_passport</td>
<td>$\leftarrow$ not passport_ok</td>
</tr>
<tr>
<td>passport_ok</td>
<td>$\leftarrow$ not passport_ok</td>
</tr>
</tbody>
</table>
MH semantics — Example

Vacation example

beach ← not mountain
mountain ← not travel
travel ← not beach
passport_ok
Vacation example

\[
\begin{align*}
\text{beach} & \iff \neg \text{mountain} \\
\text{mountain} & \iff \neg \text{travel} \\
\text{travel} & \iff \neg \text{beach} \\
\text{passport\_ok} & \iff \text{travel}
\end{align*}
\]
MH semantics — Example

Vacation example

\[
\begin{align*}
\text{beach} & \iff \neg \text{not mountain} \\
\text{passport\_ok} & \\
\text{travel} &
\end{align*}
\]
MH semantics — Example

Vacation example

```
beach
passport_ok
travel
```
Reasoning as Query Answering

In the MH 2-v semantics, models are the deductive consequences of a specialized abduction for NLPs.

Existential query answering (a.k.a. brave reasoning)
“is there a model (min. hyps+conseqs.) where query is true?”

More efficient with Relevant semantics, e.g. MH (not SM)

1. Get relevant part of $P$ for query; get Hyps for relevant part
2. Get MH sub-model satisfying query

Relevance guarantees sub-model is extendible to a full one; no need to compute full models
Reasoning as Query Answering

- In the MH 2-v semantics, models are the deductive consequences of a specialized abduction for NLPs.
- Existential query answering (a.k.a. brave reasoning): “is there a model (min. hyps+conseqs.) where \textit{query} is true?”

More efficient with Relevant semantics, e.g. MH (not SM):

1. Get relevant part of $P$ for query; get Hyps for relevant part.
2. Get MH sub-model satisfying query.

Relevance guarantees sub-model is extendible to a full one; no need to compute full models.
Reasoning as Query Answering

- In the MH 2-v semantics, models are the deductive consequences of a specialized abduction for NLPs
- Existential query answering (a.k.a. brave reasoning) “is there a model (min. hyps+conseqs.) where \textit{query} is true?”

More efficient with Relevant semantics, e.g. MH (not SM)

1. Get relevant part of $P$ for query; get Hyps for relevant part
2. Get MH sub-model satisfying query

Relevance guarantees sub-model is extendible to a full one; no need to compute full models
Reasoning as Query Answering

- In the MH 2-v semantics, models are the deductive consequences of a specialized abduction for NLPs.
- Existential query answering (a.k.a. brave reasoning)
  "is there a model (min. hyps+conseqs.) where query is true?"

More efficient with Relevant semantics, e.g. MH (not SM)

1. Get relevant part of $P$ for query; get Hyps for relevant part
2. Get MH sub-model satisfying query
   Relevance guarantees sub-model is extendible to a full one; no need to compute full models
Outline

1. Introduction
2. Building a semantics
3. Minimal Hypotheses semantics
4. Conclusions and Future Work
Conclusions

- NLR $\neq$ ICs
- NLPs must have model
- Minimum positive hypotheses: generalization of maximum negative hypotheses
- Minimum positive hypotheses $\rightarrow$ Layered Support (generalization of Classical Support)
- Minimal Hypotheses semantics:
  - 2-valued semantics
  - relevance, cumulativity, existence (liveness in face of updates)
  - generalizes SMs (all SMs are MH models)
Conclusions

- NLR $\neq$ ICs
- NLPs must have model
  - Minimum positive hypotheses: generalization of maximum negative hypotheses
  - Minimum positive hypotheses $\rightarrow$ Layered Support (generalization of Classical Support)
- Minimal Hypotheses semantics:
  - 2-valued semantics
  - relevance, cumulativity, existence (liveness in face of updates)
  - generalizes SMs (all SMs are MH models)
Conclusions

- NLR $\neq$ ICs
- NLPs must have model

**Minimum positive hypotheses**: generalization of maximum negative hypotheses

- Minimum positive hypotheses $\rightarrow$ Layered Support (generalization of Classical Support)

**Minimal Hypotheses semantics**:
- 2-valued semantics
- relevance, cumulativity, existence (liveness in face of updates)
- generalizes SMs (all SMs are MH models)
Conclusions

- NLR $\neq$ ICs
- NLPs must have model
- **Minimum positive hypotheses**: generalization of maximum negative hypotheses
- Minimum positive hypotheses $\rightarrow$ **Layered Support** (generalization of Classical Support)
- **Minimal Hypotheses semantics**:
  - 2-valued semantics
  - relevance, cumulativity, existence (liveness in face of updates)
  - generalizes SMs (all SMs are MH models)

Each NLP has a 2-valued MH semantics
Conclusions

- NLR $\neq$ ICs
- NLPs must have model
- **Minimum positive hypotheses**: generalization of maximum negative hypotheses
- Minimum positive hypotheses $\rightarrow$ **Layered Support** (generalization of Classical Support)
- **Minimal Hypotheses semantics**:
  - 2-valued semantics
  - relevance, cumulativity, existence (liveness in face of updates)
  - generalizes SMs (all SMs are MH models)
Conclusions

- NLR ≠ ICs
- NLPs must have model
- **Minimum positive hypotheses**: generalization of maximum negative hypotheses
- Minimum positive hypotheses $\rightarrow$ **Layered Support** (generalization of Classical Support)
- **Minimal Hypotheses semantics**:
  - 2-valued semantics
  - relevance, cumulativity, existence (liveness in face of updates)
  - generalizes SMs (all SMs are MH models)
Conclusions

- NLR ≠ ICs
- NLPs must have model
- **Minimum positive hypotheses**: generalization of maximum negative hypotheses
- Minimum positive hypotheses → Layered Support (generalization of Classical Support)
- **Minimal Hypotheses semantics**:
  - 2-valued semantics
  - relevance, cumulativity, existence (liveness in face of updates)
  - generalizes SMs (all SMs are MH models)
Conclusions

- NLR \neq ICs
- NLPs must have model
- **Minimum positive hypotheses**: generalization of maximum negative hypotheses
- Minimum positive hypotheses \rightarrow **Layered Support**
  (generalization of Classical Support)
- **Minimal Hypotheses semantics**:
  - 2-valued semantics
  - relevance, cumulativity, existence (liveness in face of updates)
  - generalizes SMs (all SMs are MH models)
Future Work

- Implementation of MH semantics-based system (ongoing)
- Further comparisons: MH vs RSMs, MH vs PStable, others
- Generalize the MH semantics to Extended Logic Programs
- Applications: Semantic Web, KR (adding regular abduction), combination of 2-v with 3-v
Future Work

- Implementation of MH semantics-based system (ongoing)
- Further comparisons: MH vs RSMs, MH vs PStable, others
- Generalize the MH semantics to Extended Logic Programs
- Applications: Semantic Web, KR (adding regular abduction), combination of 2-v with 3-v
Future Work

- Implementation of MH semantics-based system (ongoing)
- Further comparisons: MH vs RSMs, MH vs PStable, others
- Generalize the MH semantics to Extended Logic Programs
- Applications: Semantic Web, KR (adding regular abduction), combination of 2-v with 3-v
Future Work

- Implementation of MH semantics-based system (ongoing)
- Further comparisons: MH vs RSMs, MH vs PStable, others
- Generalize the MH semantics to Extended Logic Programs
- Applications: Semantic Web, KR (adding regular abduction), combination of 2-v with 3-v
Thank you
Disjunctive Logic Programs (1/2)

- Disjunctive LPs (DisjLPs): rules are of the form

\[ h_1 \lor \ldots \lor h_q \leftarrow b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m \]

with \( q \geq 1 \), \( m, n \geq 0 \) and finite, and where \( h_k, b_i, c_j \) are atoms

- Can be transformed into NLPs via Shifting Rule

\[
\begin{align*}
  a \lor b & \leftarrow \text{Body} \\
  a & \leftarrow \text{not } b, \text{Body} \\
  b & \leftarrow \text{not } a, \text{Body}
\end{align*}
\]

- Shifting Rule produces loops over default negation
- Can be problematic to SMs
- MH solves \textit{any} loops by assuming minimal positive hypotheses

Each NLP has a 2-valued MH semantics
Disjunctive Logic Programs (1/2)

- Disjunctive LPs (DisjLPs): rules are of the form

  \[ h_1 \lor \ldots \lor h_q \leftarrow b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m \]

  with \( q \geq 1, m, n \geq 0 \) and finite, and where \( h_k, b_i, c_j \) are atoms

- Can be transformed into NLPs via Shifting Rule

  \[
  a \lor b \leftarrow \text{Body} \quad \leadsto \\
  a \leftarrow \text{not } b, \text{Body} \\
  b \leftarrow \text{not } a, \text{Body}
  \]

  Shifting Rule produces loops over default negation
  Can be problematic to SMs

- MH solves any loops by assuming minimal positive hypotheses

Each NLP has a 2-valued MH semantics
Disjunctive Logic Programs (1/2)

- Disjunctive LPs (DisjLPs): rules are of the form

\[ h_1 \lor \ldots \lor h_q \leftarrow b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m \]

with \( q \geq 1, m, n \geq 0 \) and finite, and where \( h_k, b_i, c_j \) are atoms

- Can be transformed into NLPs via Shifting Rule

\[
\begin{align*}
  a \lor b & \leftarrow \text{Body} \\
  a & \leftarrow \text{not } b, \text{Body} \\
  b & \leftarrow \text{not } a, \text{Body}
\end{align*}
\]

- Shifting Rule produces loops over default negation
  Can be problematic to SMs

- MH solves any loops by assuming minimal positive hypotheses

Each NLP has a 2-valued MH semantics
Disjunctive Logic Programs (1/2)

- Disjunctive LPs (DisjLPs): rules are of the form

  \[ h_1 \lor \ldots \lor h_q \leftarrow b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m \]

  with \( q \geq 1, m, n \geq 0 \) and finite, and where \( h_k, b_i, c_j \) are atoms

- Can be transformed into NLPs via Shifting Rule

  \[
  a \lor b \leftarrow \text{Body} \quad \sim \quad \\
  a \leftarrow \text{not } b, \text{Body} \quad \\
  b \leftarrow \text{not } a, \text{Body}
  \]

- Shifting Rule produces loops over default negation
  Can be problematic to SMs

- MH solves \textit{any} loops by assuming minimal positive hypotheses

Each NLP has a 2-valued MH semantics
Disjunctive Logic Programs (2/2)

Disjunctions / Shifting Rule / Loops

\[
\begin{align*}
  a \lor b & \quad a \leftarrow \neg b & \quad b \leftarrow \neg a \\
  b \lor c & \quad b \leftarrow \neg c & \quad c \leftarrow \neg b \\
  c \lor a & \quad c \leftarrow \neg a & \quad a \leftarrow \neg c
\end{align*}
\]

Basis of MH intuition = conceptually reverse the Shifting Rule:
Loops (any kind of loops) encode disjunction of hypotheses
Disjunctive Logic Programs (2/2)

Disjunctions / Shifting Rule / Loops

\[
\begin{align*}
a \lor b & \quad a \leftarrow \neg b \quad b \leftarrow \neg a \\
b \lor c & \quad b \leftarrow \neg c \quad c \leftarrow \neg b \\
c \lor a & \quad c \leftarrow \neg a \quad a \leftarrow \neg c
\end{align*}
\]

Basis of MH intuition = conceptually reverse the Shifting Rule:

Loops (any kind of loops) encode disjunction of hypotheses

Each NLP has a 2-valued MH semantics
Disjunctive Logic Programs (2/2)

Disjunctions / Shifting Rule / Loops

\[ a \lor b \quad a \leftarrow \neg b \quad b \leftarrow \neg a \]
\[ b \lor c \quad b \leftarrow \neg c \quad c \leftarrow \neg b \]
\[ c \lor a \quad c \leftarrow \neg a \quad a \leftarrow \neg c \]

Basis of MH intuition = conceptually reverse the Shifting Rule: Loops (any kind of loops) encode disjunction of hypotheses

September 30th, INAP 2011, Vienna

Each NLP has a 2-valued MH semantics
Motivation for the Layered Negative Reduction

Variation of the vacation example

\[
\text{beach} \leftrightarrow \text{not mountain} \\
\text{mountain} \leftrightarrow \text{not travel} \\
\text{travel} \leftrightarrow \text{not beach}
\]

Three MHs: \{beach, travel, not mountain\}, \{beach, not travel, mountain\}, \{not beach, travel, mountain\}.

Variation of the vacation example with fourth friend

\[
\text{beach} \leftrightarrow \text{not mountain} \\
\text{mountain} \leftrightarrow \text{not travel} \\
\text{travel} \leftrightarrow \text{not beach} \\
\text{beach}
\]

Two MHs: \{beach, travel, not mountain\}, \{beach, not travel, mountain\}.

Each NLP has a 2-valued MH semantics.
Motivation for the Layered Negative Reduction

Variation of the vacation example

\[
\begin{align*}
\text{beach} & \leftrightarrow \text{not mountain} \\
\text{mountain} & \leftrightarrow \text{not travel} \\
\text{travel} & \leftrightarrow \text{not beach}
\end{align*}
\]

Three MHs: \{beach, travel, not mountain\}, \{beach, not travel, mountain\}, \{not beach, travel, mountain\}.

Variation of the vacation example with fourth friend

\[
\begin{align*}
\text{beach} & \leftrightarrow \text{not mountain} \\
\text{mountain} & \leftrightarrow \text{not travel} \\
\text{travel} & \leftrightarrow \text{not beach} \\
\text{beach}
\end{align*}
\]

Two MHs: \{beach, travel, not mountain\}, \{beach, not travel, mountain\}. 

Each NLP has a 2-valued MH semantics.