Tight Semantics for Logic Programs

Luís Moniz Pereira    Alexandre Miguel Pinto

Centre for Artificial Intelligence - CENTRIA
Universidade Nova de Lisboa

JULY 17TH, 2010 — EDINBURGH, UK
Motivation (1/3)

- NLPs are commonly used for KRR
- Stable Models fails to give semantics to all NLPs:

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

  if pass_ok then P has no SMs, though the lower (even) loop, by itself, has two:  
  \{ pass_ok \} \{ exp_pass \}
Motivation (2/3)

- SM fails to give semantics to Odd Loops Over Negation (OLONs)
- OLONs can appear in a single NLP, by combining NLPs, or by updating an NLP
- OLONs are not Integrity Constraints (ICs) — they express distinct KRR concerns. Denials model ICs, e.g., \( \bot \leftarrow \text{IC\_Body} \)
- Also, SM fails to give semantics to some Infinite Chains
  
  e.g., P: \( p(X) \leftarrow p(s(X)) \) \quad p(X) \leftarrow \text{not } p(s(X)) 

  Ground(P): \( p(0) \leftarrow p(s(0)) \) \quad p(0) \leftarrow \text{not } p(s(0)) 

  \( p(s(0)) \leftarrow p(s(s(0))) \) \quad p(s(0)) \leftarrow \text{not } p(s(s(0))) 

  ... \quad ...
Motivation (3/3)

• Need a more uniform 2-valued semantics which:

✓ Provides semantics to all kinds of loops in the same way

✓ Provides semantics to all kinds of ‘non-loop’ rule dependencies in the same way
Layering vs. Stratification (1/2)

- Stratification is atom-based
  - Puts atoms in Strata
  - Misses structural info
  - Does not cover loops

- Layering (more general) is rule-based
  ✓ Puts rules in Layers
  ✓ Captures all structural info
  ✓ Covers loops
Layering vs. Stratification (2/2)

- No Stratification

- Layering:
  - rules in loop => same layer
  - non-loop dependencies => different layers
    - e.g. rules for travel and passport_ok

| Layer 2: | beach ← not mountain |
| Odd Loop | mountain ← not travel |
|          | travel ← not beach, passport_ok |
| Layer 1: | passport_ok ← not expired_passport |
| Even Loop | expired_passport ← not passport_ok |

Saturday, July 17, 2010
Tight Models: Inductive Intuition
Tight Models: Inductive Intuition

Layer $i+1$

Layer $i$

...
Layer $i+1$

Layer $i$
Tight Models: Inductive Intuition

Layer $i+1$

Layer $i$ $\rightarrow$ Layer Tight Model $M_i$

...
Tight Models: Inductive Intuition

...
Tight Models: Inductive Intuition

...
Tight Models: Inductive Intuition

\[
\text{(Layer } i+1): M_{\leq i} \leq i \\
\text{Layer } i \rightarrow \text{Layer Tight Model } M_i
\]

Divide Layer \( i+1 \) by \( M_{\leq i} = M_{< i} \cup M_i \)

...
Tight Models:
Inductive Intuition

...
Tight Models: Inductive Intuition

... 

(Layer $i+1): M \leq_i$ 

Layer $i$ $\rightarrow$ Layer Tight Model $M_i$ 

...
Tight Models: Inductive Intuition

\[
(Layer \ i+1): M_{\leq i} \rightarrow \text{Layer Tight Model } M_{i+1}
\]

Layer \ i \rightarrow \text{Layer Tight Model } M_i

\ldots
Tight Models: Inductive Intuition

\[ M_{\leq i+1} = M_{\leq i} \cup M_{i+1} \]

(Layer \( i+1 \))\( M_{\leq i} \rightarrow \) Layer Tight Model \( M_{i+1} \)

Layer \( i \) \( \rightarrow \) Layer Tight Model \( M_i \)

...
Tight Models: Inductive Intuition

\[ M_{\leq i+1} = M_{\leq i} \cup M_{i+1} \]

\((\text{Layer } i+1): M_{\leq i} \rightarrow \text{Layer Tight Model } M_{i+1}\)

\(\text{Layer } i \rightarrow \text{Layer Tight Model } M_i\)

Divide Layer \(i+2\) by \(M_{\leq i+1}\)
Layer division — L:M

✓ Delete literals in M from bodies of rules of L
✓ Delete rules of L inconsistent with M, but source of inconsistency restricted to body literals NOT in loop
✓ Propagate consequences — compute Remainder
Symmetric Tightness: Loop Tight Models
Symmetric Tightness: Loop Tight Models

\[
\begin{align*}
\text{passport}_\text{ok} & \leftarrow \text{not expired}_\text{passport} \\
\text{expired}_\text{passport} & \leftarrow \text{not passport}_\text{ok}
\end{align*}
\]

Loop Tight (LT) model:
Symmetric Tightness: Loop Tight Models

\[
\text{passport}\_\text{ok} \leftarrow \text{not expired}\_\text{passport} \\
\text{expired}\_\text{passport} \leftarrow \text{not passport}\_\text{ok}
\]

Loop Tight (LT) model:
\[
\text{repeat}
\]
Symmetric Tightness: Loop Tight Models

passport_ok ← not expired_passport
expired_passport ← not passport_ok

Loop Tight (LT) model:

repeat
  1. Assume one DNL — add corresponding atom as fact
Symmetric Tightness: Loop Tight Models

\[\text{passport\_ok} \leftarrow \text{not } \text{expired\_passport}\]
\[\text{expired\_passport} \leftarrow \text{not } \text{passport\_ok}\]

Loop Tight (LT) model:

\[\text{repeat}\]
1. Assume one DNL — add corresponding atom as fact
2. Propagate consequences — compute Remainder
Symmetric Tightness:
Loop Tight Models

\[
\text{passport\_ok} \leftarrow \text{not expired\_passport}
\]
\[
\text{expired\_passport} \leftarrow \text{not passport\_ok}
\]

Loop Tight (LT) model:

\begin{verbatim}
repeat
    1. Assume one DNL — add corresponding atom as fact
    2. Propagate consequences — compute Remainder
until No more DNLs left in Loop
\end{verbatim}
Symmetric Tightness: Loop Tight Models

passport_ok ← not expired_passport
expired_passport ← not passport_ok

Loop Tight (LT) model:

repeat
  1. Assume one DNL — add corresponding atom as fact
  2. Propagate consequences — compute Remainder
until No more DNLs left in Loop
(No distinction between Even or Odd loops !)
Symmetric Tightness:
Loop Tight Models

passport_ok ← not expired_passport
expired_passport ← not passport_ok

Loop Tight (LT) model:

repeat
  1. Assume one DNL — add corresponding atom as fact
  2. Propagate consequences — compute Remainder
until No more DNLs left in Loop
(No distinction between Even or Odd loops !)

LT model is set of resulting facts
Symmetric Tightness: Loop Tight Models

passport_ok ← not expired_passport
expired_passport ← not passport_ok
passport_ok

Loop Tight (LT) model:

repeat
  1. Assume one DNL — add corresponding atom as fact
  2. Propagate consequences — compute Remainder
until No more DNLs left in Loop
(No distinction between Even or Odd loops !)

LT model is set of resulting facts
Symmetric Tightness: Loop Tight Models

\[ \text{passport}_\text{ok} \leftarrow \text{not expiredpassport} \]

\text{passport}_\text{ok}

Loop Tight (LT) model:

```
repeat
  1. Assume one DNL — add corresponding atom as fact
  2. Propagate consequences — compute Remainder
until No more DNLs left in Loop
(No distinction between Even or Odd loops !)
```

LT model is set of resulting facts
Symmetric Tightness: Loop Tight Models

passport_ok

Loop Tight (LT) model:

repeat
  1. Assume one DNL — add corresponding atom as fact
  2. Propagate consequences — compute Remainder
until No more DNLs left in Loop
(No distinction between Even or Odd loops !)

LT model is set of resulting facts
Symmetric Tightness: Loop Tight Models

Loop Tight (LT) model:

\[
\text{repeat} \\
1. \text{Assume one DNL — add corresponding atom as fact} \\
2. \text{Propagate consequences — compute Remainder} \\
\text{until} \quad \text{No more DNLs left in Loop} \\
\text{(No distinction between Even or Odd loops !)}
\]

LT model is set of resulting facts
Symmetric Tightness: Loop Tight Models

\[
\begin{align*}
\text{passport\_ok} & \leftarrow \text{not expired\_passport} \\
\text{expired\_passport} & \leftarrow \text{not passport\_ok}
\end{align*}
\]

Loop Tight (LT) model:

\[
\begin{align*}
\text{repeat} & \\
1. & \text{Assume one DNL — add corresponding atom as fact} \\
2. & \text{Propagate consequences — compute Remainder} \\
\text{until} & \text{No more DNLs left in Loop} \\
& \text{(No distinction between Even or Odd loops !)}
\end{align*}
\]

LT model is set of resulting facts
Symmetric Tightness: Loop Tight Models

passport_ok ← not expired_passport
expired_passport ← not passport_ok
expired_passport

Loop Tight (LT) model:

repeat
1. Assume one DNL — add corresponding atom as fact
2. Propagate consequences — compute Remainder
until No more DNLs left in Loop
(No distinction between Even or Odd loops !)

LT model is set of resulting facts
Symmetric Tightness:
Loop Tight Models

expired_passport ← not passport_ok
expired_passport

Loop Tight (LT) model:

repeat
  1. Assume one DNL — add corresponding atom as fact
  2. Propagate consequences — compute Remainder
until No more DNLs left in Loop
(No distinction between Even or Odd loops !)

LT model is set of resulting facts
Symmetric Tightness: Loop Tight Models

expired_passport

Loop Tight (LT) model:

repeat

1. Assume one DNL — add corresponding atom as fact
2. Propagate consequences — compute Remainder

until No more DNLs left in Loop

(No distinction between Even or Odd loops !)

LT model is set of resulting facts
Symmetric Tightness: Loop Tight Models

Loop Tight (LT) model:

repeat
1. Assume one DNL — add corresponding atom as fact
2. Propagate consequences — compute Remainder
until No more DNLs left in Loop
(No distinction between Even or Odd loops!)

LT model is set of resulting facts
Symmetric Tightness: Loop Tight Models

Loop Tight (LT) model:

repeat
1. Assume one DNL — add corresponding atom as fact
2. Propagate consequences — compute Remainder
until No more DNLs left in Loop
(No distinction between Even or Odd loops !)

LT model is set of resulting facts

passport_ok ← not expired_passport
expired_passport ← not passport_ok
Symmetric Tightness: Loop Tight Models

\[ \text{passport\_ok} \leftarrow \text{not expired\_passport} \]
\[ \text{expired\_passport} \leftarrow \text{not passport\_ok} \]

Loop Tight (LT) model:

\begin{verbatim}
repeat
  1. Assume one DNL — add corresponding atom as fact
  2. Propagate consequences — compute Remainder
until No more DNLs left in Loop
(No distinction between Even or Odd loops !)
LT model is set of resulting facts
\end{verbatim}
**Symmetric Tightness:**

**Layer Tight Models**

<table>
<thead>
<tr>
<th>Loop Tight Models:</th>
<th>Loop Tight Models:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a \leftarrow \text{not } b) ({A})</td>
<td>(a \leftarrow \text{not } a) ({A})</td>
</tr>
<tr>
<td>(b \leftarrow \text{not } a) ({B})</td>
<td></td>
</tr>
</tbody>
</table>

**Layer Tight models:**

\[
\{A\} \cup \{A\} \cup \text{Remaining DNLs} = \{A\} \cup \{\text{not } B\} = \{A, \text{not } B\}
\]

\[
\{B\} \cup \{A\} \cup \text{Remaining DNLs} = \{A,B\} \cup \{\} = \{A,B\}
\]

**Layer Tight model:**

✓ minimal model of Layer
✓ combination of individual Loop Tight models
✓ remaining DNLs

If \(b \leftarrow \text{not } b\) is added, then single Layer Tight model = \(\{A,B\}\)
Tightness

loops $\Rightarrow$ syntactic $\&$ semantic symmetry

$\not\Rightarrow$ layers $\Rightarrow$ syntactic $\&$ semantic asymmetry

- Default Negated Literals (DNLs) in loops: assumable hypotheses
- Semantic symmetry (loops): assume one DNL as true, then propagate consequences throughout loop
- Semantic asymmetry (non-loops): assume model for lower layers, then propagate consequences of model to next layer

Assuming DNL truth-values $+$ Propagating consequences

$=$ Tightening
**Example Models**

$$
\text{beach} \leftarrow \text{not mountain}
$$

$$
\text{mountain} \leftarrow \text{not travel}
$$

$$
\text{travel} \leftarrow \text{not beach}, \quad \text{passport\_ok}
$$

$$
\text{passport\_ok} \leftarrow \text{not expired\_passport}
$$

$$
\text{expired\_passport} \leftarrow \text{not passport\_ok}
$$

Tight Models:

$$\{\text{expired\_passport, not passport\_ok, not travel, mountain, not beach}\} - \text{SM}$$

$$\{\text{not expired\_passport, passport\_ok, not travel, mountain, beach}\}$$

$$\{\text{not expired\_passport, passport\_ok, travel, mountain, not beach}\}$$

$$\{\text{not expired\_passport, passport\_ok, travel, not mountain, beach}\}$$
Properties of Tight Semantics

- **Generalization** of Stable Models — every Stable model is a Tight model
- **Model Existence** — all NLPs have Tight models
- **Relevance**
  - Strict call-graph top-down querying is sound — no need to compute whole models
  - Grounding by need — by relevant call-graph only
- **Cumulativity** — can use Tabling techniques
- **Tight models respect the Well-Founded Model**
Conclusions

- OLONs are not Integrity Constraints
- Layering: generalization of Stratification
- Tight Semantics:
  - generalization of Stable Models
  - suitable for top-down queries: e.g., abduction ‘by need’
- Applications afforded by TS:
  all those of SM plus those where OLONs can be used for problem representation
Future Work

- Further analysis of properties, complexity, comparisons
- Extensions to ELPs and GLPs
- Abduction, updates, constructive negation
- Implementations, applications
Thank you for your attention!

Questions?