Tight Semantics for Logic Programs

Luís Moniz Pereira    Alexandre Pinto
Centre for Artificial Intelligence - CENTRIA
Universidade Nova de Lisboa

at Maurice Bruynooghe’s 60th birthday symposium
7 July 2010, Leuven, Belgium
Summary Motivation

• Gelfond-Lifschitz operator (Γ) fixed-point requirement, by Stable Models (SMs), induces an asymmetry in dealing with Even Loops and Odd Loops Over Negation (ELONs, OLONs)

• Also, when the syntactical dependency order of rules in a program is not well-founded, there is no guarantee of existence of SMs

• We introduce a 2-valued semantics for all Normal Logic Programs (NLPs) — the Tight Semantics (TS) — that generalizes SM semantics by:
  - Dealing uniformly with ELONs and OLONs
  - Dealing with all syntactic dependencies of rules (well-founded or not)

• TS conservatively extends the SM semantics (all SMs are Tight Models), enjoys Relevance and Cumulativity, guarantees model Existence, and respects the Well-Founded Model (WFM)
Motivation
Layering vs. Stratification
Tightness — Symmetries and Asymmetries
Properties of Tight semantics
Conclusions and Future Work
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
- But OLONs can appear by themselves, or when combining NLPs, or when updating an NLP
- OLONs are not Integrity Constraints (ICs) — they express distinct KRR concerns. Denials model ICs.
- Also, SM fails to give semantics to some Infinite Chains say to P: \( p(X) \leftarrow p(s(X)) \quad p(X) \leftarrow \text{not } p(s(X)) \)
Motivation (3/3)

Need for a more uniform 2-valued semantics which:

- Treats all kinds of loops the same way
- Treats all kinds of non-loops the same way
Layering vs. Stratification (1/3)

- Classical notion of Stratification is atom-based => loses structural info
- Stratification does not cover loops
- More general notion — Layering — is rule-based => puts each rule in a Layer
Layering vs. Stratification (2/3)

- This previous program has no Stratification
- Layering covers loops: all rules of a loop are in the same layer \(\Rightarrow\) no distinction between Even or Odd loops
- Non-loop dependencies are mapped into different layers — e.g. the rules for \textit{travel} and \textit{passport_ok}
Layering vs. Stratification (3/3)

Another example

\[
\begin{align*}
c & \leftarrow \text{not } d, \text{not } y, \text{not } a & \text{Layer 3} \\
d & \leftarrow \text{not } c \\
y & \leftarrow \text{not } x & b & \leftarrow \text{not } x & \text{Layer 2} \\
x & \leftarrow \text{not } x & b & \text{Layer 1}
\end{align*}
\]

- Layering is rule-based: captures all structural info
- Rules for same head — e.g. rules for \( b \) — may belong in different layers
**Tightness**

Loops $\Rightarrow$ Syntactic & Semantic Symmetry

$\neq$ Layers $\Rightarrow$ Syntactic & Semantic Asymmetry

Which semantic symmetry/asymmetry is desired?

- Default Negated Literals (DNLs) are assumable hypotheses
- **Semantic Symmetry**: truth-value of any DNL of a loop can be assumed — then propagate consequences throughout loop
- **Semantic Asymmetry**: assume truth-value of DNLs in lower layers — then propagate consequences to DNLs of next layer

Assuming DNL truth-value $+$ Propagation of consequences $=$ Tightening
Tight Models: Inductive Intuition
Tight Models: Inductive Intuition

Layer \( i+1 \)

Layer \( i \)

...
Tight Models: Inductive Intuition

Layer $i+1$

Layer $i$

...
Tight Models: Inductive Intuition

Layer $i$

Layer $i+1$
Tight Models: Inductive Intuition

...
Tight Models: Inductive Intuition

...
Tight Models: Inductive Intuition

\((\text{Layer } i+1): M_i\) (includes computing Remainder)

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

Divide Layer \(i+1\) by \(M_{\leq i}\)

...
Tight Models: Inductive Intuition

(Layer $i+1$):$M_i$
(includes computing Remainder)

Layer $i$ → Layer Tight Model $M_i$
Tight Models: Inductive Intuition

...
Tight Models: Inductive Intuition

(Layer \( i+1 \)) \( : M_i \) → Layer Tight Model \( M_{i+1} \)

(includes computing Remainder)

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

...
Tight Models: Inductive Intuition

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

(Layer \( i+1 \)): \( M_i \) -> Layer Tight Model \( M_{i+1} \)
(includes computing Remainder)

Layer \( i \) -> Layer Tight Model \( M_i \)

...
Tight Models: Inductive Intuition

Divide Layer $i+2$ by $M_{\leq i+1}$

$M_{\leq i+1} = M_{\leq i} \cup M_{i+1}$

(Layer $i+1$): $M_i$ → Layer Tight Model $M_{i+1}$

(includes computing Remainder)

Layer $i$ → Layer Tight Model $M_i$
Symmetric Tightness: Loop Tight Models
Symmetric Tightness: Loop Tight Models

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

- How to symmetrically “Tighten” a single Loop?
Symmetric Tightness: Loop Tight Models

passport_ok ← not expired_passport
expired_passport ← not passport_ok

- How to symmetrically “Tighten” a single Loop?
  repeat
Symmetric Tightness: Loop Tight Models

passport_ok ← not expired_passport
expired_passport ← not passport_ok

• How to symmetrically “Tighten” a single Loop?

  repeat
    1. Assume one DNL in Loop as false — e.g. ‘not passport_ok’ is false — and add corresponding atom ‘passport_ok’ as fact
Symmetric Tightness: Loop Tight Models

passport_ok ← not expired_passport
expired_passport ← not passport_ok

- How to symmetrically “Tighten” a single Loop?
  repeat
    1. Assume one DNL in Loop as false — e.g. ‘not passport_ok’ is false — and add corresponding atom ‘passport_ok’ as fact
    2. Compute Remainder — propagate consequences
Symmetric Tightness: Loop Tight Models

passport_ok ← not expired_passport
expired_passport ← not passport_ok

- How to symmetrically “Tighten” a single Loop?
  repeat
   1. Assume one DNL in Loop as false — e.g. ‘not passport_ok’ is false — and add corresponding atom ‘passport_ok’ as fact
   2. Compute Remainder — propagate consequences
  until No more DNLs left in Loop
Symmetric Tightness: Loop Tight Models

passport_ok ← not expired_passport
expired_passport ← not passport_ok

• How to symmetrically “Tighten” a single Loop?
  repeat
    1. Assume one DNL in Loop as false — e.g. ‘not passport_ok’ is false — and add corresponding atom ‘passport_ok’ as fact
    2. Compute Remainder — propagate consequences
  until No more DNLs left in Loop
• Loop Tight (LT) models are: resulting facts + original DNLs still true
Symmetric Tightness: Loop Tight Models

passport_ok ← not expired_passport
expired_passport ← not passport_ok
passport_ok

• How to symmetrically “Tighten” a single Loop?

  repeat
    1. Assume one DNL in Loop as false — e.g. ‘not passport_ok’ is false — and add corresponding atom ‘passport_ok’ as fact
    2. Compute Remainder — propagate consequences
  until No more DNLs left in Loop

• Loop Tight (LT) models are: resulting facts + original DNLs still true
Symmetric Tightness: Loop Tight Models

passport_ok ← not expired_passport

- How to symmetrically “Tighten” a single Loop?

  repeat
  1. Assume one DNL in Loop as false — e.g. ‘not passport_ok’ is false — and add corresponding atom ‘passport_ok’ as fact
  2. Compute Remainder — propagate consequences
  until No more DNLs left in Loop

- Loop Tight (LT) models are: resulting facts + original DNLs still true
Symmetric Tightness: Loop Tight Models

- How to symmetrically “Tighten” a single Loop?
  
  repeat
  
  1. Assume one DNL in Loop as false — e.g. ‘not passport_ok’ is false — and add corresponding atom ‘passport_ok’ as fact
  
  2. Compute Remainder — propagate consequences
  
  until  No more DNLs left in Loop

- Loop Tight (LT) models are: resulting facts + original DNLs still true

passport_ok
Symmetric Tightness: Loop Tight Models

- How to symmetrically “Tighten” a single Loop?
  
  repeat
  1. Assume one DNL in Loop as false — e.g. ‘not passport_ok’ is false — and add corresponding atom ‘passport_ok’ as fact
  2. Compute Remainder — propagate consequences
  until No more DNLs left in Loop

- Loop Tight (LT) models are: resulting facts + original DNLs still true
Symmetric Tightness: Loop Tight Models

- How to symmetrically “Tighten” a single Loop?
  repeat
    1. Assume one DNL in Loop as false — e.g. ‘not passport_ok’ is false — and add corresponding atom ‘passport_ok’ as fact
    2. Compute Remainder — propagate consequences
  until No more DNLs left in Loop
- Loop Tight (LT) models are: resulting facts + original DNLs still true

expired_passport
Symmetric Tightness: Loop Tight Models

- How to symmetrically “Tighten” a single Loop?

  repeat
  1. Assume one DNL in Loop as false — e.g. ‘not passport_ok’ is false — and add corresponding atom ‘passport_ok’ as fact
  2. Compute Remainder — propagate consequences
  until No more DNLs left in Loop

- Loop Tight (LT) models are: resulting facts + original DNLs still true
Symmetric Tightness: Loop Tight Models

Loop Tight models:

\{\text{passport\_ok, not expired\_passport}\}
\{\text{expired\_passport, not passport\_ok}\}

- How to symmetrically “Tighten” a single Loop?
  
  repeat
  1. Assume one DNL in Loop as false — e.g. ‘not passport\_ok’ is false — and add corresponding atom ‘passport\_ok’ as fact
  2. Compute Remainder — propagate consequences
  until No more DNLs left in Loop

- Loop Tight (LT) models are: resulting facts + original DNLs still true
Symmetric Tightness: SCC Tight Models

How to symmetrically “Tighten” several mutually dependent Loops (i.e. a Strongly Connected Component — SCC)?

SCC Tight models are minimal SCC models, obtained by unions of the atoms of individual, single Loop Tight models.

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

The Tight models of a single layer — Layer Tight Models — are:

- minimal models of the layer
- that contain one SCC Tight model atoms set
- per each SCC in the layer
Asymmetric Tightness: Layer Division

• How to asymmetrically “Tighten” across Layers?
• Divide Layer $L_{i+1}$ by $M_{\leq i}$ — layer division engenders Asymmetrical Tightness
• Layer division — $L:M$
  • Delete from bodies of rules in $L$ literals in $M$
  • Delete rules with bodies false in $M \setminus \text{heads}(L)$
  • Compute Remainder — propagate consequences
Intuitive operational intuition:

1. Find the (unique) Layering of P:
   \[ O(|\text{Nodes}| + |\text{Edges}|) \]  
   (Nodes and Edges refer to Rule Graph)

2. \( \text{TM} := \emptyset \)

3. For a layer, repeat non-deterministically (starting from the bottom layer):
   3.1. Compute any one Layer Tight Model LTM for the layer
   3.2. \( \text{TM} := \text{TM} \cup \text{LTM} \)
   3.3. Divide next Layer above by TM
   3.4. Go on to the Layer above

4. \( \text{TM} \) is a Tight Model of P
**Tight Models**

If there is a bottom Layer, let $M_0 = \emptyset$ be the unique Tight Model of Layer $\cup_{\text{Layers} \leq 0} = \emptyset$

For $\forall \alpha > 0$

$M_{\leq \alpha}$ is Tight Model of $\cup_{\text{Layers} \leq \alpha}$ iff

$\exists M_{< \alpha}$ Tight Model of $\cup_{\text{Layers} < \alpha}$ such that

$M_{\leq \alpha} = M_\alpha \cup M_{< \alpha}$

where $M_\alpha$ is a Layer Tight model of Layer $\alpha$: $M_{< \alpha}$
Infinite Chains

Tight semantics also copes with unbound term-length programs
Semantics definition based on ordinal $\alpha$ (can be either successor or limit)

ex:

\[
\text{p}(X) \leftarrow \text{p}(s(X)) \quad \text{p}(X) \leftarrow \text{not} \ \text{p}(s(X))
\]

Ground version:

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

- This OLON-free program has no Stable Models
- It has unique Tight model = \{ p(0), p(s(0)), p(s(s(0))), \ldots \}
Properties of Tight Semantics

• Conservative **Extension** of Stable Models — every Stable model is a Tight model

• Guarantee of model **Existence** — all NLPs have 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 add Tight model atoms as facts; not enjoyed by SM, as neither Relevance nor Existence

• Tight models **Respect** the Well-Founded Model
Complexity of Tight Semantics

- Model existence? YES – constant-time
- Brave reasoning — query is true in one model:
  \( \Sigma^2_p \)-complete on the part Relevant for the query
- Cautious reasoning — query is true in all models:
  \( \Pi^2_p \)-complete on the part Relevant for the query
- **The good news**: Answers can be found incrementally — with incremental grounding — and each LT model for a loop speeds remaining computation
  - If each SCC has only one Loop  \( \Rightarrow \)
    Brave reasoning & Cautious reasoning are NP-complete
  - For stratified programs, then unique Tight Model = WFM  \( \Rightarrow \)
    Brave reasoning & Cautious reasoning are Polynomial-time
Conclusions

- TS properties:
  - generalization of Stable Models;
  - model existence; relevance;
  - cumulativity;
  - respects WFM
- Suitable for abductive queries ‘by need’
- Applications afforded by TS:
  - all those of SM plus those where OLONs are employed 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?