TECHNISCHE UNIVERSITÄT DRESDEN

Artificial Intelligence, Computational Logic

DEDUCTION SYSTEMS

Lecture 5 ASP Solving I *slides adapted from Torsten Schaub [Gebser et al.(2012)]

Sarah Gaggl

Dresden, 18th June 2015

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Conflict-driven ASP Solving: Overview

1. Motivation
2. Preliminaries
3. Boolean constraints
4. Nogoods from logic programs

Outline

Motivation of Conflict-driven ASP Solving

- Goal Approach to computing stable models of logic programs, based on concepts from
  - Constraint Processing (CP) and
  - Satisfiability Testing (SAT)
- Idea View inferences in ASP as unit propagation on nogoods
- Benefits:
  - A uniform constraint-based framework for different kinds of inferences in ASP
  - Advanced techniques from the areas of CP and SAT
  - Highly competitive implementation

Outline

Outline

Partial interpretations

or: 3-valued interpretations

A partial interpretation maps atoms onto truth values *true*, *false*, and *unknown*

Partial interpretations

or: 3-valued interpretations

A partial interpretation maps atoms onto truth values *true*, *false*, and *unknown*

- Representation: $\langle T, F \rangle$, where
  - T is the set of all *true* atoms and
  - F is the set of all *false* atoms
  - Truth of atoms in $\mathcal{A} \setminus (T \cup F)$ is *unknown*

Partial interpretations

or: 3-valued interpretations

A partial interpretation maps atoms onto truth values *true*, *false*, and *unknown*

- Representation: $\langle T, F \rangle$, where
  - T is the set of all *true* atoms and
  - F is the set of all *false* atoms
  - Truth of atoms in $\mathcal{A} \setminus (T \cup F)$ is *unknown*
- Properties:
  - $\langle T, F \rangle$ is conflicting if $T \cap F \neq \emptyset$
  - $\langle T, F \rangle$ is total if $T \cup F = \mathcal{A}$ and $T \cap F = \emptyset$

Partial interpretations

or: 3-valued interpretations

A partial interpretation maps atoms onto truth values *true*, *false*, and *unknown*

- Representation: $\langle T, F \rangle$, where
  - T is the set of all *true* atoms and
  - F is the set of all *false* atoms
  - Truth of atoms in $\mathcal{A} \setminus (T \cup F)$ is *unknown*
- Properties:
  - $\langle T, F \rangle$ is conflicting if $T \cap F \neq \emptyset$
  - $\langle T, F \rangle$ is total if $T \cup F = \mathcal{A}$ and $T \cap F = \emptyset$
- Definition: For $\langle T_1, F_1 \rangle$ and $\langle T_2, F_2 \rangle$, define
  - $\langle T_1, F_1 \rangle \sqsubseteq \langle T_2, F_2 \rangle$ iff $T_1 \subseteq T_2$ and $F_1 \subseteq F_2$
  - $\langle T_1, F_1 \rangle \sqcup \langle T_2, F_2 \rangle = \langle T_1 \cup T_2, F_1 \cup F_2 \rangle$

Outline

Unfounded sets

Let P be a normal logic program,
and let $\langle T, F\rangle$ be a partial interpretation

Unfounded sets

Let P be a normal logic program,
and let $\langle T, F \rangle$ be a partial interpretation

- A set $U \subseteq atom(P)$ is an unfounded set of P wrt $\langle T, F \rangle$

Unfounded sets

Let P be a normal logic program,
and let $\langle T, F \rangle$ be a partial interpretation

- A set $U \subseteq atom(P)$ is an unfounded set of P wrt $\langle T, F \rangle$
  Intuitively, $\langle T, F \rangle$ is what we already know about P

Unfounded sets

Let P be a normal logic program,
and let $\langle T, F\rangle$ be a partial interpretation

- A set $U \subseteq atom(P)$ is an unfounded set of P wrt $\langle T, F\rangle$

- if we have for each rule $r \in P$ such that $head(r) \in U$

Unfounded sets

Let P be a normal logic program,
and let $\langle T, F\rangle$ be a partial interpretation

- A set $U \subseteq atom(P)$ is an unfounded set of P wrt $\langle T, F\rangle$

- if we have for each rule $r \in P$ such that $head(r) \in U$
  either

Unfounded sets

Let P be a normal logic program,
and let $\langle T, F \rangle$ be a partial interpretation

- A set $U \subseteq atom(P)$ is an unfounded set of P wrt $\langle T, F \rangle$

- if we have for each rule $r \in P$ such that $head(r) \in U$
  either
  1. $body(r)^+ \cap F \neq \emptyset$ or $body(r)^- \cap T \neq \emptyset$ or

Unfounded sets

Let P be a normal logic program,
and let $\langle T, F \rangle$ be a partial interpretation

- A set $U \subseteq atom(P)$ is an unfounded set of P wrt $\langle T, F \rangle$

- if we have for each rule $r \in P$ such that $head(r) \in U$
  either
  1. $body(r)^+ \cap F \neq \emptyset$ or $body(r)^- \cap T \neq \emptyset$ or
  2. $body(r)^+ \cap U \neq \emptyset$

Unfounded sets

Let P be a normal logic program,
and let $\langle T, F \rangle$ be a partial interpretation

- A set $U \subseteq atom(P)$ is an unfounded set of P wrt $\langle T, F \rangle$

- if we have for each rule $r \in P$ such that $head(r) \in U$
  either
  1. $body(r)^+ \cap F \neq \emptyset$ or $body(r)^- \cap T \neq \emptyset$ or
  2. $body(r)^+ \cap U \neq \emptyset$

- Rules satisfying Condition 1 are not usable for further derivations

Unfounded sets

Let P be a normal logic program,
and let $\langle T, F \rangle$ be a partial interpretation

- A set $U \subseteq atom(P)$ is an unfounded set of P wrt $\langle T, F \rangle$

- if we have for each rule $r \in P$ such that $head(r) \in U$

  either

  1. $body(r)^+ \cap F \neq \emptyset$ or $body(r)^- \cap T \neq \emptyset$ or
  2. $body(r)^+ \cap U \neq \emptyset$

- Rules satisfying Condition 1 are not usable for further derivations
- Condition 2 is the unfounded set condition treating cyclic derivations: All rules still being usable to derive an atom in U require an(other) atom in U to be true

Example

$$P = \left\{ \begin{array}{ccc} a & \leftarrow & b \\ b & \leftarrow & a \end{array} \right\}$$

Example

$$P = \left\{ \begin{array}{ccc} a & \leftarrow & b \\ b & \leftarrow & a \end{array} \right\}$$

- $\emptyset$ is an unfounded set (by definition)

Example

$$P = \left\{ \begin{array}{ccc} a & \leftarrow & b \\ b & \leftarrow & a \end{array} \right\}$$

- $\emptyset$ is an unfounded set (by definition)
- $\{a\}$ is not an unfounded set of P wrt $\langle \emptyset, \emptyset \rangle$

Example

$$P = \left\{ \begin{array}{ccc} a & \leftarrow & b \\ b & \leftarrow & a \end{array} \right\}$$

- $\emptyset$ is an unfounded set (by definition)
- $\{a\}$ is not an unfounded set of P wrt $\langle\emptyset, \emptyset\rangle$
- $\{a\}$ is an unfounded set of P wrt $\langle\emptyset, \{b\}\rangle$

Example

$$P = \left\{ \begin{array}{ccc} a & \leftarrow & b \\ b & \leftarrow & a \end{array} \right\}$$

- $\emptyset$ is an unfounded set (by definition)
- $\{a\}$ is not an unfounded set of P wrt $\langle \emptyset, \emptyset \rangle$
- $\{a\}$ is an unfounded set of P wrt $\langle \emptyset, \{b\} \rangle$
- $\{a\}$ is not an unfounded set of P wrt $\langle \{b\}, \emptyset \rangle$

Example

$$P = \left\{ \begin{array}{ccc} a & \leftarrow & b \\ b & \leftarrow & a \end{array} \right\}$$

- $\emptyset$ is an unfounded set (by definition)
- $\{a\}$ is not an unfounded set of P wrt $\langle\emptyset, \emptyset\rangle$
- $\{a\}$ is an unfounded set of P wrt $\langle\emptyset, \{b\}\rangle$
- $\{a\}$ is not an unfounded set of P wrt $\langle\{b\}, \emptyset\rangle$
- Analogously for $\{b\}$

Example

$$P = \left\{ \begin{array}{ccc} a & \leftarrow & b \\ b & \leftarrow & a \end{array} \right\}$$

- $\emptyset$ is an unfounded set (by definition)
- $\{a\}$ is not an unfounded set of P wrt $\langle \emptyset, \emptyset \rangle$
- $\{a\}$ is an unfounded set of P wrt $\langle \emptyset, \{b\} \rangle$
- $\{a\}$ is not an unfounded set of P wrt $\langle \{b\}, \emptyset \rangle$

- $\{a, b\}$ is an unfounded set of P wrt $\langle \emptyset, \emptyset \rangle$

Example

$$P = \left\{ \begin{array}{ccc} a & \leftarrow & b \\ b & \leftarrow & a \end{array} \right\}$$

- $\emptyset$ is an unfounded set (by definition)
- $\{a\}$ is not an unfounded set of P wrt $\langle \emptyset, \emptyset \rangle$
- $\{a\}$ is an unfounded set of P wrt $\langle \emptyset, \{b\} \rangle$
- $\{a\}$ is not an unfounded set of P wrt $\langle \{b\}, \emptyset \rangle$

- $\{a, b\}$ is an unfounded set of P wrt $\langle \emptyset, \emptyset \rangle$
- $\{a, b\}$ is an unfounded set of P wrt any partial interpretation

Outline

Assignments

- An **assignment** A over $dom(A) = atom(P) \cup body(P)$ is a sequence

$$(\sigma_1, \ldots, \sigma_n)$$

  of signed literals σ_i of form $\boldsymbol{T}v$ or $\boldsymbol{F}v$ for $v \in dom(A)$ and $1 \leq i \leq n$
- $\boldsymbol{T}v$ expresses that v is **true** and $\boldsymbol{F}v$ that it is **false**

Assignments

- An assignment A over $dom(A) = atom(P) \cup body(P)$ is a sequence

$$(\sigma_1, \ldots, \sigma_n)$$

  of signed literals σ_i of form $\boldsymbol{T}v$ or $\boldsymbol{F}v$ for $v \in dom(A)$ and $1 \leq i \leq n$
- $\boldsymbol{T}v$ expresses that v is **true** and $\boldsymbol{F}v$ that it is **false**
- The complement, $\overline{\sigma}$, of a literal σ is defined as $\overline{\boldsymbol{T}v} = \boldsymbol{F}v$ and $\overline{\boldsymbol{F}v} = \boldsymbol{T}v$

Assignments

- An assignment A over $dom(A) = atom(P) \cup body(P)$ is a sequence

$$(\sigma_1, \ldots, \sigma_n)$$

  of signed literals σ_i of form $\boldsymbol{T}v$ or $\boldsymbol{F}v$ for $v \in dom(A)$ and $1 \leq i \leq n$
- $\boldsymbol{T}v$ expresses that v is **true** and $\boldsymbol{F}v$ that it is **false**
- The complement, $\overline{\sigma}$, of a literal σ is defined as $\overline{\boldsymbol{T}v} = \boldsymbol{F}v$ and $\overline{\boldsymbol{F}v} = \boldsymbol{T}v$
- $A \circ \sigma$ stands for the result of appending σ to A

Assignments

- An assignment A over $dom(A) = atom(P) \cup body(P)$ is a sequence

$$(\sigma_1, \ldots, \sigma_n)$$

  of signed literals σ_i of form $\boldsymbol{T}v$ or $\boldsymbol{F}v$ for $v \in dom(A)$ and $1 \leq i \leq n$
- $\boldsymbol{T}v$ expresses that v is **true** and $\boldsymbol{F}v$ that it is **false**
- The complement, $\overline{\sigma}$, of a literal σ is defined as $\overline{\boldsymbol{T}v} = \boldsymbol{F}v$ and $\overline{\boldsymbol{F}v} = \boldsymbol{T}v$
- $A \circ \sigma$ stands for the result of appending σ to A
- Given $A = (\sigma_1, \ldots, \sigma_{k-1}, \sigma_k, \ldots, \sigma_n)$, we let $A[\sigma_k] = (\sigma_1, \ldots, \sigma_{k-1})$

Assignments

- An assignment A over $dom(A) = atom(P) \cup body(P)$ is a sequence

$$(\sigma_1, \ldots, \sigma_n)$$

  of signed literals σ_i of form $\boldsymbol{T}v$ or $\boldsymbol{F}v$ for $v \in dom(A)$ and $1 \leq i \leq n$
- $\boldsymbol{T}v$ expresses that v is **true** and $\boldsymbol{F}v$ that it is **false**
- The complement, $\overline{\sigma}$, of a literal σ is defined as $\overline{\boldsymbol{T}v} = \boldsymbol{F}v$ and $\overline{\boldsymbol{F}v} = \boldsymbol{T}v$
- $A \circ \sigma$ stands for the result of appending σ to A
- Given $A = (\sigma_1, \ldots, \sigma_{k-1}, \sigma_k, \ldots, \sigma_n)$, we let $A[\sigma_k] = (\sigma_1, \ldots, \sigma_{k-1})$
- We sometimes identify an assignment with the set of its literals

Assignments

- An assignment A over $dom(A) = atom(P) \cup body(P)$ is a sequence

$$(\sigma_1, \ldots, \sigma_n)$$

  of signed literals σ_i of form $\boldsymbol{T}v$ or $\boldsymbol{F}v$ for $v \in dom(A)$ and $1 \leq i \leq n$
- $\boldsymbol{T}v$ expresses that v is **true** and $\boldsymbol{F}v$ that it is **false**
- The complement, $\overline{\sigma}$, of a literal σ is defined as $\overline{\boldsymbol{T}v} = \boldsymbol{F}v$ and $\overline{\boldsymbol{F}v} = \boldsymbol{T}v$
- $A \circ \sigma$ stands for the result of appending σ to A
- Given $A = (\sigma_1, \ldots, \sigma_{k-1}, \sigma_k, \ldots, \sigma_n)$, we let $A[\sigma_k] = (\sigma_1, \ldots, \sigma_{k-1})$
- We sometimes identify an assignment with the set of its literals
- Given this, we access **true** and **false** propositions in A via

$$A^{\boldsymbol{T}} = \{v \in dom(A) \mid \boldsymbol{T}v \in A\} \text{ and } A^{\boldsymbol{F}} = \{v \in dom(A) \mid \boldsymbol{F}v \in A\}$$

Nogoods, solutions, and unit propagation

- A nogood is a set $\{\sigma_1, \ldots, \sigma_n\}$ of signed literals, expressing a constraint violated by any assignment containing $\sigma_1, \ldots, \sigma_n$

Nogoods, solutions, and unit propagation

- A nogood is a set $\{\sigma_1, \ldots, \sigma_n\}$ of signed literals, expressing a constraint violated by any assignment containing $\sigma_1, \ldots, \sigma_n$
- An assignment A such that $A^{\boldsymbol{T}} \cup A^{\boldsymbol{F}} = dom(A)$ and $A^{\boldsymbol{T}} \cap A^{\boldsymbol{F}} = \emptyset$ is a solution for a set Δ of nogoods, if $\delta \not\subseteq A$ for all $\delta \in \Delta$

Nogoods, solutions, and unit propagation

- A nogood is a set $\{\sigma_1, \ldots, \sigma_n\}$ of signed literals, expressing a constraint violated by any assignment containing $\sigma_1, \ldots, \sigma_n$
- An assignment A such that $A^{\boldsymbol{T}} \cup A^{\boldsymbol{F}} = dom(A)$ and $A^{\boldsymbol{T}} \cap A^{\boldsymbol{F}} = \emptyset$ is a solution for a set Δ of nogoods, if $\delta \not\subseteq A$ for all $\delta \in \Delta$
- For a nogood δ, a literal $\sigma \in \delta$, and an assignment A, we say that $\overline{\sigma}$ is unit-resulting for δ wrt A, if
  1. $\delta \setminus A = \{\sigma\}$ and
  2. $\overline{\sigma} \notin A$

Nogoods, solutions, and unit propagation

- A nogood is a set $\{\sigma_1, \ldots, \sigma_n\}$ of signed literals, expressing a constraint violated by any assignment containing $\sigma_1, \ldots, \sigma_n$
- An assignment A such that $A^{\boldsymbol{T}} \cup A^{\boldsymbol{F}} = dom(A)$ and $A^{\boldsymbol{T}} \cap A^{\boldsymbol{F}} = \emptyset$ is a solution for a set Δ of nogoods, if $\delta \not\subseteq A$ for all $\delta \in \Delta$
- For a nogood δ, a literal $\sigma \in \delta$, and an assignment A, we say that $\overline{\sigma}$ is unit-resulting for δ wrt A, if
  1. $\delta \setminus A = \{\sigma\}$ and
  2. $\overline{\sigma} \notin A$
- For a set Δ of nogoods and an assignment A, unit propagation is the iterated process of extending A with unit-resulting literals until no further literal is unit-resulting for any nogood in Δ

Outline

Outline

Nogoods from logic programs
via program completion

The completion of a logic program P can be defined as follows:

$$
\begin{aligned}
& \{v_B \leftrightarrow a_1 \wedge \cdots \wedge a_m \wedge \neg a_{m+1} \wedge \cdots \wedge \neg a_n \mid \\
& \qquad B \in body(P), B = \{a_1, \ldots, a_m, not\ a_{m+1}, \ldots, not\ a_n\}\} \\
\cup \quad & \{a \leftrightarrow v_{B_1} \vee \cdots \vee v_{B_k} \mid a \in atom(P), body(a) = \{B_1, \ldots, B_k\}\} \ ,
\end{aligned}
$$

where $body(a) = \{body(r) \mid r \in P, head(r) = a\}$

Nogoods from logic programs
via program completion

- The (body-oriented) equivalence

$$v_B \leftrightarrow a_1 \wedge \cdots \wedge a_m \wedge \neg a_{m+1} \wedge \cdots \wedge \neg a_n$$

  can be decomposed into two implications:

Nogoods from logic programs
via program completion

- The (body-oriented) equivalence

$$v_B \leftrightarrow a_1 \wedge \cdots \wedge a_m \wedge \neg a_{m+1} \wedge \cdots \wedge \neg a_n$$

can be decomposed into two implications:

1. $v_B \rightarrow a_1 \wedge \cdots \wedge a_m \wedge \neg a_{m+1} \wedge \cdots \wedge \neg a_n$
   is equivalent to the conjunction of

$$\neg v_B \vee a_1 , \ \ldots , \ \neg v_B \vee a_m , \ \neg v_B \vee \neg a_{m+1} , \ \ldots , \ \neg v_B \vee \neg a_n$$

   and induces the set of nogoods

$$\Delta(B) = \{ \{\boldsymbol{T}B, \boldsymbol{F}a_1\}, \ldots, \{\boldsymbol{T}B, \boldsymbol{F}a_m\}, \{\boldsymbol{T}B, \boldsymbol{T}a_{m+1}\}, \ldots, \{\boldsymbol{T}B, \boldsymbol{T}a_n\} \}$$

Nogoods from logic programs
via program completion

- The (body-oriented) equivalence

$$v_B \leftrightarrow a_1 \wedge \cdots \wedge a_m \wedge \neg a_{m+1} \wedge \cdots \wedge \neg a_n$$

can be decomposed into two implications:

2. $a_1 \wedge \cdots \wedge a_m \wedge \neg a_{m+1} \wedge \cdots \wedge \neg a_n \rightarrow v_B$
   gives rise to the nogood

$$\delta(B) = \{\boldsymbol{F}B, \boldsymbol{T}a_1, \ldots, \boldsymbol{T}a_m, \boldsymbol{F}a_{m+1}, \ldots, \boldsymbol{F}a_n\}$$

Nogoods from logic programs
via program completion

- Analogously, the (atom-oriented) equivalence

$$a \leftrightarrow v_{B_1} \vee \cdots \vee v_{B_k}$$

yields the nogoods

1. $\Delta(a) = \{ \{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \ldots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\} \}$ and
2. $\delta(a) = \{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\}$

Nogoods from logic programs
atom-oriented nogoods

- For an atom a where $body(a) = \{B_1, \ldots, B_k\}$, we get

$$\{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\} \quad \text{and} \quad \{\,\{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \ldots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\}\,\}$$

Nogoods from logic programs

atom-oriented nogoods

- For an atom a where $body(a) = \{B_1, \dots, B_k\}$, we get

$$\{\boldsymbol{T}a, \boldsymbol{F}B_1, \dots, \boldsymbol{F}B_k\} \quad \text{and} \quad \{\,\{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \dots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\}\,\}$$

- Example Given Atom x with $body(x) = \{\{y\}, \{not\ z\}\}$, we obtain

x	$\leftarrow$	y
x	$\leftarrow$	$not\ z$

$$\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$$
$$\{\,\{\boldsymbol{F}x, \boldsymbol{T}\{y\}\}, \{\boldsymbol{F}x, \boldsymbol{T}\{not\ z\}\}\,\}$$

Nogoods from logic programs
atom-oriented nogoods

- For an atom a where $body(a) = \{B_1, \ldots, B_k\}$, we get

$$\{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\} \quad \text{and} \quad \{\,\{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \ldots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\}\,\}$$

- Example Given Atom x with $body(x) = \{\{y\}, \{not\ z\}\}$, we obtain

x	$\leftarrow$	y
x	$\leftarrow$	$not\ z$

$$\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$$
$$\{\,\{\boldsymbol{F}x, \boldsymbol{T}\{y\}\}, \{\boldsymbol{F}x, \boldsymbol{T}\{not\ z\}\}\,\}$$

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal

Nogoods from logic programs
atom-oriented nogoods

- For an atom a where $body(a) = \{B_1, \ldots, B_k\}$, we get

$$\{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\} \quad \text{and} \quad \{\, \{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \ldots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\} \,\}$$

- Example Given Atom x with $body(x) = \{\{y\}, \{not\ z\}\}$, we obtain

x	$\leftarrow$	y
x	$\leftarrow$	$not\ z$

$$\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$$
$$\{\, \{\boldsymbol{F}x, \boldsymbol{T}\{y\}\}, \{\boldsymbol{F}x, \boldsymbol{T}\{not\ z\}\} \,\}$$

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal

Nogoods from logic programs
atom-oriented nogoods

- For an atom a where $body(a) = \{B_1, \ldots, B_k\}$, we get

$$\{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\} \quad \text{and} \quad \{\,\{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \ldots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\}\,\}$$

- Example Given Atom x with $body(x) = \{\{y\}, \{not\ z\}\}$, we obtain

$$\begin{array}{|lcl|}\hline x & \leftarrow & y \\ x & \leftarrow & not\ z \\ \hline\end{array}$$

$$\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$$
$$\{\,\{\boldsymbol{F}x, \boldsymbol{T}\{y\}\}, \{\boldsymbol{F}x, \boldsymbol{T}\{not\ z\}\}\,\}$$

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal

- $\boldsymbol{F}x$ is unit-resulting wrt assignment $(\boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\})$ and

Nogoods from logic programs
atom-oriented nogoods

- For an atom a where $body(a) = \{B_1, \ldots, B_k\}$, we get

$$\{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\} \quad \text{and} \quad \{\,\{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \ldots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\}\,\}$$

- Example Given Atom x with $body(x) = \{\{y\}, \{not\ z\}\}$, we obtain

x	$\leftarrow$	y
x	$\leftarrow$	$not\ z$

$$\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$$
$$\{\,\{\boldsymbol{F}x, \boldsymbol{T}\{y\}\}, \{\boldsymbol{F}x, \boldsymbol{T}\{not\ z\}\}\,\}$$

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal

- $\boldsymbol{F}x$ is unit-resulting wrt assignment $(\boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\})$ and

Nogoods from logic programs
atom-oriented nogoods

- For an atom a where $body(a) = \{B_1, \ldots, B_k\}$, we get

$$\{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\} \quad \text{and} \quad \{\, \{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \ldots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\}\, \}$$

- Example Given Atom x with $body(x) = \{\{y\}, \{not\ z\}\}$, we obtain

$$\begin{array}{|lcl|} \hline x & \leftarrow & y \\ x & \leftarrow & not\ z \\ \hline \end{array}$$

$$\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$$
$$\{\, \{\boldsymbol{F}x, \boldsymbol{T}\{y\}\}, \{\boldsymbol{F}x, \boldsymbol{T}\{not\ z\}\}\, \}$$

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal

- $\boldsymbol{F}x$ is unit-resulting wrt assignment $(\boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\})$ and

Nogoods from logic programs
atom-oriented nogoods

- For an atom a where $body(a) = \{B_1, \ldots, B_k\}$, we get

$$\{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\} \quad \text{and} \quad \{\,\{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \ldots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\}\,\}$$

- Example Given Atom x with $body(x) = \{\{y\}, \{not\ z\}\}$, we obtain

x	$\leftarrow$	y
x	$\leftarrow$	$not\ z$

$$\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$$
$$\{\,\{\boldsymbol{F}x, \boldsymbol{T}\{y\}\}, \{\boldsymbol{F}x, \boldsymbol{T}\{not\ z\}\}\,\}$$

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal

- $\boldsymbol{F}x$ is unit-resulting wrt assignment $(\boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\})$ and

Nogoods from logic programs
atom-oriented nogoods

- For an atom a where $body(a) = \{B_1, \ldots, B_k\}$, we get

$$\{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\} \quad \text{and} \quad \{\, \{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \ldots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\}\,\}$$

- Example Given Atom x with $body(x) = \{\{y\}, \{not\ z\}\}$, we obtain

$$\begin{array}{lcl} x & \leftarrow & y \\ x & \leftarrow & not\ z \end{array}$$

$$\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$$
$$\{\, \{\boldsymbol{F}x, \boldsymbol{T}\{y\}\}, \{\boldsymbol{F}x, \boldsymbol{T}\{not\ z\}\}\,\}$$

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal

- $\boldsymbol{F}x$ is unit-resulting wrt assignment $(\boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\})$ and

Nogoods from logic programs
atom-oriented nogoods

- For an atom a where $body(a) = \{B_1, \dots, B_k\}$, we get

$$\{\boldsymbol{T}a, \boldsymbol{F}B_1, \dots, \boldsymbol{F}B_k\} \quad \text{and} \quad \{\,\{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \dots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\}\,\}$$

- Example Given Atom x with $body(x) = \{\{y\}, \{not\ z\}\}$, we obtain

x	$\leftarrow$	y
x	$\leftarrow$	$not\ z$

$$\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$$
$$\{\,\{\boldsymbol{F}x, \boldsymbol{T}\{y\}\}, \{\boldsymbol{F}x, \boldsymbol{T}\{not\ z\}\}\,\}$$

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal

  - $\boldsymbol{T}\{not\ z\}$ is unit-resulting wrt assignment $(\boldsymbol{T}x, \boldsymbol{F}\{y\})$

Nogoods from logic programs
atom-oriented nogoods

- For an atom a where $body(a) = \{B_1, \ldots, B_k\}$, we get

$$\{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\} \quad \text{and} \quad \{\,\{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \ldots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\}\,\}$$

- Example Given Atom x with $body(x) = \{\{y\}, \{not\ z\}\}$, we obtain

x	$\leftarrow$	y
x	$\leftarrow$	$not\ z$

$$\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$$
$$\{\,\{\boldsymbol{F}x, \boldsymbol{T}\{y\}\}, \{\boldsymbol{F}x, \boldsymbol{T}\{not\ z\}\}\,\}$$

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal

  - $\boldsymbol{T}\{not\ z\}$ is unit-resulting wrt assignment $(\boldsymbol{T}x, \boldsymbol{F}\{y\})$

Nogoods from logic programs
atom-oriented nogoods

- For an atom a where $body(a) = \{B_1, \dots, B_k\}$, we get

$$\{\boldsymbol{T}a, \boldsymbol{F}B_1, \dots, \boldsymbol{F}B_k\} \quad \text{and} \quad \{\,\{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \dots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\}\,\}$$

- Example Given Atom x with $body(x) = \{\{y\}, \{not\ z\}\}$, we obtain

x	$\leftarrow$	y
x	$\leftarrow$	$not\ z$

$$\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$$
$$\{\,\{\boldsymbol{F}x, \boldsymbol{T}\{y\}\}, \{\boldsymbol{F}x, \boldsymbol{T}\{not\ z\}\}\,\}$$

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal

  - $\boldsymbol{T}\{not\ z\}$ is unit-resulting wrt assignment $(\boldsymbol{T}x, \boldsymbol{F}\{y\})$

Nogoods from logic programs
atom-oriented nogoods

- For an atom a where $body(a) = \{B_1, \ldots, B_k\}$, we get

$$\{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\} \quad \text{and} \quad \{\,\{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \ldots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\}\,\}$$

- Example Given Atom x with $body(x) = \{\{y\}, \{not\ z\}\}$, we obtain

$$x \leftarrow y$$
$$x \leftarrow not\ z$$

$$\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$$
$$\{\,\{\boldsymbol{F}x, \boldsymbol{T}\{y\}\}, \{\boldsymbol{F}x, \boldsymbol{T}\{not\ z\}\}\,\}$$

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal

   – $\boldsymbol{T}\{not\ z\}$ is unit-resulting wrt assignment $(\boldsymbol{T}x, \boldsymbol{F}\{y\})$

Nogoods from logic programs
atom-oriented nogoods

- For an atom a where $body(a) = \{B_1, \dots, B_k\}$, we get

$$\{\boldsymbol{T}a, \boldsymbol{F}B_1, \dots, \boldsymbol{F}B_k\} \quad \text{and} \quad \{\,\{\boldsymbol{F}a, \boldsymbol{T}B_1\}, \dots, \{\boldsymbol{F}a, \boldsymbol{T}B_k\}\,\}$$

- Example Given Atom x with $body(x) = \{\{y\}, \{not\ z\}\}$, we obtain

$$\begin{array}{lcl} x & \leftarrow & y \\ x & \leftarrow & not\ z \end{array}$$

$$\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$$
$$\{\,\{\boldsymbol{F}x, \boldsymbol{T}\{y\}\}, \{\boldsymbol{F}x, \boldsymbol{T}\{not\ z\}\}\,\}$$

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal

- $\boldsymbol{T}\{not\ z\}$ is unit-resulting wrt assignment $(\boldsymbol{T}x, \boldsymbol{F}\{y\})$

Nogoods from logic programs
body-oriented nogoods

- For a body $B = \{a_1, \ldots, a_m, not\ a_{m+1}, \ldots, not\ a_n\}$, we get

$$\{\boldsymbol{F}B, \boldsymbol{T}a_1, \ldots, \boldsymbol{T}a_m, \boldsymbol{F}a_{m+1}, \ldots, \boldsymbol{F}a_n\}$$
$$\{\ \{\boldsymbol{T}B, \boldsymbol{F}a_1\}, \ldots, \{\boldsymbol{T}B, \boldsymbol{F}a_m\}, \{\boldsymbol{T}B, \boldsymbol{T}a_{m+1}\}, \ldots, \{\boldsymbol{T}B, \boldsymbol{T}a_n\}\ \}$$

Nogoods from logic programs
body-oriented nogoods

- For a body $B = \{a_1, \ldots, a_m, not\ a_{m+1}, \ldots, not\ a_n\}$, we get

$$\{\boldsymbol{F}B, \boldsymbol{T}a_1, \ldots, \boldsymbol{T}a_m, \boldsymbol{F}a_{m+1}, \ldots, \boldsymbol{F}a_n\}$$
$$\{\ \{\boldsymbol{T}B, \boldsymbol{F}a_1\}, \ldots, \{\boldsymbol{T}B, \boldsymbol{F}a_m\}, \{\boldsymbol{T}B, \boldsymbol{T}a_{m+1}\}, \ldots, \{\boldsymbol{T}B, \boldsymbol{T}a_n\}\ \}$$

- Example Given Body $\{x, not\ y\}$, we obtain

$$\begin{array}{|c|}\hline \ldots \leftarrow x, not\ y \\ \vdots \\ \ldots \leftarrow x, not\ y \\ \hline\end{array}$$

$$\{\boldsymbol{F}\{x, not\ y\}, \boldsymbol{T}x, \boldsymbol{F}y\}$$
$$\{\ \{\boldsymbol{T}\{x, not\ y\}, \boldsymbol{F}x\}, \{\boldsymbol{T}\{x, not\ y\}, \boldsymbol{T}y\}\ \}$$

Nogoods from logic programs
body-oriented nogoods

- For a body $B = \{a_1, \dots, a_m, not\ a_{m+1}, \dots, not\ a_n\}$, we get

$$\{\boldsymbol{F}B, \boldsymbol{T}a_1, \dots, \boldsymbol{T}a_m, \boldsymbol{F}a_{m+1}, \dots, \boldsymbol{F}a_n\}$$
$$\{\ \{\boldsymbol{T}B, \boldsymbol{F}a_1\}, \dots, \{\boldsymbol{T}B, \boldsymbol{F}a_m\}, \{\boldsymbol{T}B, \boldsymbol{T}a_{m+1}\}, \dots, \{\boldsymbol{T}B, \boldsymbol{T}a_n\}\ \}$$

- Example Given Body $\{x, not\ y\}$, we obtain

$$\begin{array}{|c|}\hline \dots \leftarrow x, not\ y \\ \vdots \\ \dots \leftarrow x, not\ y \\ \hline\end{array}$$

$$\{\boldsymbol{F}\{x, not\ y\}, \boldsymbol{T}x, \boldsymbol{F}y\}$$
$$\{\ \{\boldsymbol{T}\{x, not\ y\}, \boldsymbol{F}x\}, \{\boldsymbol{T}\{x, not\ y\}, \boldsymbol{T}y\}\ \}$$

For nogood $\delta(\{x, not\ y\}) = \{\boldsymbol{F}\{x, not\ y\}, \boldsymbol{T}x, \boldsymbol{F}y\}$, the signed literal

- $\boldsymbol{T}\{x, not\ y\}$ is unit-resulting wrt assignment $(\boldsymbol{T}x, \boldsymbol{F}y)$ and
- $\boldsymbol{T}y$ is unit-resulting wrt assignment $(\boldsymbol{F}\{x, not\ y\}, \boldsymbol{T}x)$

Characterization of stable models

for tight logic programs, ie. free of positive recursion

Let P be a logic program and

$$\begin{aligned}\Delta_P \quad &= \quad \{\delta(a) \mid a \in atom(P)\} \cup \{\delta \in \Delta(a) \mid a \in atom(P)\} \\ &\cup \quad \{\delta(B) \mid B \in body(P)\} \cup \{\delta \in \Delta(B) \mid B \in body(P)\}\end{aligned}$$

Characterization of stable models

for tight logic programs, ie. free of positive recursion

Let P be a logic program and

$$\begin{aligned}\Delta_P \quad &= \quad \{\delta(a) \mid a \in atom(P)\} \cup \{\delta \in \Delta(a) \mid a \in atom(P)\} \\ &\cup \quad \{\delta(B) \mid B \in body(P)\} \cup \{\delta \in \Delta(B) \mid B \in body(P)\}\end{aligned}$$

Theorem

Let P be a tight logic program. Then,
$X \subseteq atom(P)$ is a stable model of P iff
$X = A^{\boldsymbol{T}} \cap atom(P)$ for a (unique) solution A for Δ_P

Summary

- Partial assignments
- Unfounded sets
- Unit resulting literals
- Unit propagation
- Nogoods via program completion
- Characterization of stable models of tight programs in terms of nogoods.

References

Martin Gebser, Benjamin Kaufmann Roland Kaminski, and Torsten Schaub.
Answer Set Solving in Practice.
Synthesis Lectures on Artificial Intelligence and Machine Learning.
Morgan and Claypool Publishers, 2012.
doi=10.2200/S00457ED1V01Y201211AIM019.

- See also: `http://potassco.sourceforge.net`