DEDUCTION SYSTEMS

Answer-Set Programming II
* slides adapted from Torsten Schaub [Gebser et al.(2012)]

Sarah Gaggl

Dresden, 22nd May 2013
Agenda

Solving procedure for ASP programs

- Partial Interpretations
- Unfounded sets
- Assignments
- Nogoods
- Unit-propagation
- Conflict-driven nogood learning algorithm (CDNL)
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
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 \textit{true}, \textit{false}, and \textit{unknown}

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

- Properties:
  - \( \langle T, F \rangle \) is conflicting if \( T \cap F \neq \emptyset \)
  - \( \langle T, F \rangle \) is total if \( T \cup F = A \) 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 \( F_1 \subseteq F_2 \)
  - \( \langle T_1, F_1 \rangle \sqcup \langle T_2, F_2 \rangle = \langle T_1 \cup T_2, F_1 \cup F_2 \rangle \)
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 $U \subseteq \text{atom}(P)$ 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 $U \subseteq \text{atom}(P)$ is an unfounded set of $P$ wrt $\langle T, F \rangle$.
  Intuitively, $\langle T, F \rangle$ is what we already know about $P$.

  1. Rules satisfying Condition 1 are not usable for further derivations.
  2. 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.
Unfounded sets

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

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

- If we have for each rule $r \in P$ such that $\text{head}(r) \in U$.

$\text{body}(r) + \cap F \neq \emptyset$ or $\text{body}(r) - \cap T \neq \emptyset$ or $\text{body}(r) + \cap U \neq \emptyset$.
Unfounded sets

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

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

- if we have for each rule $r \in P$ such that $\text{head}(r) \in U$
  either
Unfounded sets

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

- A set \( U \subseteq \text{atom}(P) \) is an **unfounded set** of \( P \) wrt \( \langle T, F \rangle \).

- If we have for each rule \( r \in P \) such that \( \text{head}(r) \in U \), either
  
  1. \( \text{body}(r)^+ \cap F \neq \emptyset \) or \( \text{body}(r)^- \cap T \neq \emptyset \) or
Unfounded sets

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

- A set \( U \subseteq \text{atom}(P) \) is an unfounded set of \( P \) wrt \( \langle T, F \rangle \)
- if we have for each rule \( r \in P \) such that \( \text{head}(r) \in U \)
  - either
    1. \( \text{body}(r)^+ \cap F \neq \emptyset \) or \( \text{body}(r)^- \cap T \neq \emptyset \) or
    2. \( \text{body}(r)^+ \cap U \neq \emptyset \)
Unfounded sets

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

- A set \( U \subseteq \text{atom}(P) \) is an **unfounded set** of \( P \) wrt \( \langle T, F \rangle \).

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

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

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

- 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

\[ P = \begin{cases} \ a & \leftrightarrow \ b \\ \ b & \leftrightarrow \ a \end{cases} \]
Example

\[ P = \{ a \leftarrow b, b \leftarrow a \} \]

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

\[ P = \{ \begin{array}{c@{\quad \leftarrow \quad} c} a & b \\ b & a \end{array} \} \]

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

\[ P = \{ \begin{array}{cc} a & \leftarrow b \\ b & \leftarrow a \end{array} \} \]

- \( \emptyset \) 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

\[ P = \left\{ \begin{array}{c} a \leftarrow b \\ b \leftarrow a \end{array} \right\} \]

- \( \emptyset \) 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

\[ P = \left\{ \begin{array}{c|c}
    a & b \\
    b & a \\
\end{array} \right\} \]

- \( \emptyset \) 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

\[ P = \{ \begin{align*} a & \leftarrow b \\ b & \leftarrow a \end{align*} \} \]

- \( \emptyset \) 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

\[
P = \left\{ \begin{array}{c} a \leftarrow b \\ b \leftarrow a \end{array} \right\}
\]

- \(\emptyset\) 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
Assignments

- An assignment $A$ over $\text{dom}(A) = \text{atom}(P) \cup \text{body}(P)$ is a sequence
  $$(\sigma_1, \ldots, \sigma_n)$$

  of signed literals $\sigma_i$ of form $T_v$ or $F_v$ for $v \in \text{dom}(A)$ and $1 \leq i \leq n$

- $T_v$ expresses that $v$ is true and $F_v$ that it is false
Assignments

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

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

of signed literals $\sigma_i$ of form $T_v$ or $F_v$ for $v \in \text{dom}(A)$ and $1 \leq i \leq n$
- $T_v$ expresses that $v$ is true and $F_v$ that it is false
- The complement, $\bar{\sigma}$, of a literal $\sigma$ is defined as $\bar{T_v} = F_v$ and $\bar{F_v} = T_v$
Assignments

- An **assignment** $A$ over $\text{dom}(A) = \text{atom}(P) \cup \text{body}(P)$ is a sequence $(\sigma_1, \ldots, \sigma_n)$

  of signed literals $\sigma_i$ of form $T_v$ or $F_v$ for $v \in \text{dom}(A)$ and $1 \leq i \leq n$

- $T_v$ expresses that $v$ is true and $F_v$ that it is false

- The complement, $\overline{\sigma}$, of a literal $\sigma$ is defined as $\overline{T_v} = F_v$ and $\overline{F_v} = T_v$

- $A \circ \sigma$ stands for the result of appending $\sigma$ to $A$
Assignments

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

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

of signed literals $\sigma_i$ of form $T_v$ or $F_v$ for $v \in \text{dom}(A)$ and $1 \leq i \leq n$

- $T_v$ expresses that $v$ is true and $F_v$ that it is false

- The complement, $\overline{\sigma}$, of a literal $\sigma$ is defined as $\overline{T_v} = F_v$ and $\overline{F_v} = T_v$

- $A \circ \sigma$ stands for the result of appending $\sigma$ 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 $\text{dom}(A) = \text{atom}(P) \cup \text{body}(P)$ is a sequence $(\sigma_1, \ldots, \sigma_n)$ of signed literals $\sigma_i$ of form $T_v$ or $F_v$ for $v \in \text{dom}(A)$ and $1 \leq i \leq n$.

- $T_v$ expresses that $v$ is true and $F_v$ that it is false.

- The complement, $\bar{\sigma}$, of a literal $\sigma$ is defined as $\overline{T_v} = F_v$ and $\overline{F_v} = T_v$.

- $A \circ \sigma$ stands for the result of appending $\sigma$ 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 $\text{dom}(A) = \text{atom}(P) \cup \text{body}(P)$ is a sequence

\[(\sigma_1, \ldots, \sigma_n)\]

- $T_v$ expresses that $v$ is true and $F_v$ that it is false
- The complement, $\bar{\sigma}$, of a literal $\sigma$ is defined as $\overline{T_v} = F_v$ and $\overline{F_v} = T_v$
- $A \circ \sigma$ stands for the result of appending $\sigma$ 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

\[A^T = \{v \in \text{dom}(A) \mid T_v \in A\}\] and \[A^F = \{v \in \text{dom}(A) \mid F_v \in A\}\]
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 \( A^T \cup A^F = \text{dom}(A) \) and \( A^T \cap A^F = \emptyset \) is a solution for a set \( \Delta \) of nogoods, if \( \delta \nsubseteq A \) 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 \( A^T \cup A^F = dom(A) \) and \( A^T \cap A^F = \emptyset \) is a solution for a set \( \Delta \) of nogoods, if \( \delta \not\subseteq A \) for all \( \delta \in \Delta \)

• For a nogood \( \delta \), a literal \( \sigma \in \delta \), and an assignment \( A \), we say that \( \sigma \) is unit-resulting for \( \delta \) wrt \( A \), if
  1. \( \delta \setminus A = \{\sigma\} \) and
  2. \( \overline{\sigma} \not\in A \)
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 \( \sigma_1, \ldots, \sigma_n \).

- An assignment \( A \) such that \( A^T \cup A^F = \text{dom}(A) \) and \( A^T \cap A^F = \emptyset \) is a solution for a set \( \Delta \) of nogoods, if \( \delta \not\subseteq A \) for all \( \delta \in \Delta \).

- For a nogood \( \delta \), a literal \( \sigma \in \delta \), and an assignment \( A \), we say that \( \sigma \) is unit-resulting for \( \delta \) wrt \( A \), if
  1. \( \delta \setminus A = \{\sigma\} \) and
  2. \( \overline{\sigma} \not\in A \).

- For a set \( \Delta \) 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 \( \Delta \).
The completion of a logic program $P$ can be defined as follows:

$$\{ v_B \leftrightarrow a_1 \land \cdots \land a_m \land \neg a_{m+1} \land \cdots \land \neg a_n \mid B \in body(P), B = \{a_1, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n\} \}$$

$$\cup \{ a \leftrightarrow v_{B_1} \lor \cdots \lor v_{B_k} \mid a \in atom(P), body(a) = \{B_1, \ldots, B_k\} \},$$

where $body(a) = \{body(r) \mid r \in P, head(r) = a\}$.
Nogoods from logic programs via program completion

- The (body-oriented) equivalence

\[ v_B \leftrightarrow a_1 \land \cdots \land a_m \land \neg a_{m+1} \land \cdots \land \neg a_n \]

can be decomposed into two implications:
Nogoods from logic programs via program completion

- The (body-oriented) equivalence
  \[ v_B \iff a_1 \land \cdots \land a_m \land \neg a_{m+1} \land \cdots \land \neg a_n \]
  can be decomposed into two implications:
  \[ v_B \rightarrow a_1 \land \cdots \land a_m \land \neg a_{m+1} \land \cdots \land \neg a_n \]
  is equivalent to the conjunction of
  \[ \neg v_B \lor a_1, \ldots, \neg v_B \lor a_m, \neg v_B \lor \neg a_{m+1}, \ldots, \neg v_B \lor \neg a_n \]
  and induces the set of nogoods
  \[ \Delta(B) = \{ \{TB, F a_1\}, \ldots, \{TB, F a_m\}, \{TB, T a_{m+1}\}, \ldots, \{TB, T a_n\} \} \]
Nogoods from logic programs via program completion

- The (body-oriented) equivalence

\[ v_B \leftrightarrow a_1 \land \cdots \land a_m \land \neg a_{m+1} \land \cdots \land \neg a_n \]

can be decomposed into two implications:

\[ a_1 \land \cdots \land a_m \land \neg a_{m+1} \land \cdots \land \neg a_n \rightarrow v_B \]

gives rise to the nogood

\[ \delta(B) = \{ FB, Ta_1, \ldots, Ta_m, Fa_{m+1}, \ldots, Fa_n \} \]
Analogously, the (atom-oriented) equivalence

\[ a \leftrightarrow \lor_{i=1}^{k} v_{B_i} \]

yields the nogoods

1. \( \Delta(a) = \{ \{Fa, TB_1\}, \ldots, \{Fa, TB_k\} \} \) and

2. \( \delta(a) = \{Ta, FB_1, \ldots, FB_k\} \)
Nogoods from logic programs
atom-oriented nogoods

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

\[
\{T_a, FB_1, \ldots, FB_k\} \quad \text{and} \quad \{\{F_a, TB_1\}, \ldots, \{F_a, TB_k\}\}
\]
Nogoods from logic programs
atom-oriented nogoods

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

\[
\{ Ta, FB_1, \ldots, FB_k \} \quad \text{and} \quad \{ \{ Fa, TB_1 \}, \ldots, \{ Fa, TB_k \} \}
\]

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

\[
\begin{align*}
x & \leftarrow y & \{ Tx, F\{y\}, F\{\text{not} \ z\} \} \\
x & \leftarrow \text{not} \ z & \{ \{ Fx, T\{y\} \}, \{ Fx, T\{\text{not} \ z\} \} \}
\end{align*}
\]
Nogoods from logic programs
atom-oriented nogoods

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

  \[
  \{Ta, FB_1, \ldots, FB_k\} \quad \text{and} \quad \{\{Fa, TB_1\}, \ldots, \{Fa, TB_k\}\}
  \]

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

  \[
  \begin{array}{c}
  x \leftarrow y \\
  x \leftarrow \text{not } z
  \end{array}
  \]

  \[
  \{Tx, F\{y\}, F\{\text{not } z\}\} \quad \{\{Fx, T\{y\}\}, \{Fx, T\{\text{not } z\}\}\}
  \]

For nogood \( \{Tx, F\{y\}, F\{\text{not } z\}\} \), the signed literal
Nogoods from logic programs
atom-oriented nogoods

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

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

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

$$x \leftarrow y$$
$$x \leftarrow \text{not } z$$

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

For nogood $\{T x, F \{y\}, F \{\text{not } z\}\}$, the signed literal
Nogoods from logic programs
atom-oriented nogoods

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

$$\{ Ta, FB_1, \ldots, FB_k \} \quad \text{and} \quad \{ \{ Fa, TB_1 \}, \ldots, \{ Fa, TB_k \} \}$$

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

$$\begin{array}{c}
x \leftarrow y \\
x \leftarrow \text{not } z
\end{array}$$

$$\{{\{Tx, F\{y\}, F\{\text{not } z\}\}}\}$$

For nogood $\{ Tx, F\{y\}, F\{\text{not } z\} \}$, the signed literal

$-Fx$ is unit-resulting wrt assignment $(F\{y\}, F\{\text{not } z\})$ and
Nogoods from logic programs
atom-oriented nogoods

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

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

  $\begin{array}{c}
  \text{x} \leftarrow y \\
  \text{x} \leftarrow \text{not } z
  \end{array}$

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

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

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

- $F x$ is unit-resulting wrt assignment $(F\{y\}, F\{\text{not } z\})$ and
Nogoods from logic programs
atom-oriented nogoods

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

\[
\{Ta, FB_1, \ldots, FB_k\} \quad \text{and} \quad \{\{Fa, TB_1\}, \ldots, \{Fa, TB_k\}\}
\]

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

\[
\begin{array}{ccc}
x & \leftarrow & y \\
x & \leftarrow & \text{not } z
\end{array}
\]

\[
\{Tx, F\{y\}, F\{\text{not } z\}\}
\]

\[
\{\{Fx, T\{y\}\}, \{Fx, T\{\text{not } z\}\}\}
\]

For nogood \( \{Tx, F\{y\}, F\{\text{not } z\}\} \), the signed literal

- \( Fx \) is unit-resulting wrt assignment \( (F\{y\}, F\{\text{not } z\}) \) and
For an atom $a$ where $\text{body}(a) = \{B_1, \ldots, B_k\}$, we get

\[
\{Ta, FB_1, \ldots, FB_k\} \quad \text{and} \quad \{\{Fa, TB_1\}, \ldots, \{Fa, TB_k\}\}
\]

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

\[
\begin{align*}
x & \leftarrow y & \{Tx, F\{y\}, F\{\text{not } z\}\} \\
x & \leftarrow \text{not } z & \{\{Fx, T\{y\}\}, \{Fx, T\{\text{not } z\}\}\}
\end{align*}
\]

For nogood $\{Tx, F\{y\}, F\{\text{not } z\}\}$, the signed literal

$Fx$ is unit-resulting wrt assignment $(F\{y\}, F\{\text{not } z\})$ and
Nogoods from logic programs
atom-oriented nogoods

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

\[ \{Ta, FB_1, \ldots, FB_k\} \quad \text{and} \quad \{\{Fa, TB_1\}, \ldots, \{Fa, TB_k\}\} \]

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

$\begin{align*}
x & \leftarrow y \\
x & \leftarrow \text{not } z
\end{align*}$

$\{Tx, F\{y\}, F\{\text{not } z\}\} \quad \{\{Fx, T\{y\}\}, \{Fx, T\{\text{not } z\}\}\} \quad$

For nogood $\{Tx, F\{y\}, F\{\text{not } z\}\}$, the signed literal

$Fx$ is unit-resulting wrt assignment $(F\{y\}, F\{\text{not } z\})$ and
Nogoods from logic programs
atom-oriented nogoods

• For an atom $a$ where $\text{body}(a) = \{B_1, \ldots, B_k\}$, we get
  
  \[
  \{Ta, FB_1, \ldots, FB_k\} \quad \text{and} \quad \{\{Fa, TB_1\}, \ldots, \{Fa, TB_k\}\}
  \]

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

  $x \leftarrow y$
  $x \leftarrow \text{not } z$

  \[
  \{Tx, F\{y\}, F\{\text{not } z\}\} \quad \{\{Fx, T\{y\}\}, \{Fx, T\{\text{not } z\}\}\}
  \]

  For nogood $\{Tx, F\{y\}, F\{\text{not } z\}\}$, the signed literal

  $- T\{\text{not } z\}$ is unit-resulting wrt assignment $(Tx, F\{y\})$
Nogoods from logic programs
atom-oriented nogoods

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

\[
\{T a, F B_1, \ldots, F B_k\} \quad \text{and} \quad \{\{F a, TB_1\}, \ldots, \{F a, TB_k\}\}
\]

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

\[
\begin{array}{c}
x \leftarrow y \\
x \leftarrow \text{not } z
\end{array}
\]

\[
\{T x, F \{y\}, F \{\text{not } z\}\}
\]

\[
\{\{F x, T \{y\}\}, \{F x, T \{\text{not } z\}\}\}
\]

For nogood \( \{T x, F \{y\}, F \{\text{not } z\}\} \), the signed literal

- \( T\{\text{not } z\} \) is unit-resulting wrt assignment \( (T x, F \{y\}) \)
Nogoods from logic programs
atom-oriented nogoods

• For an atom \( a \) where \( \text{body}(a) = \{B_1, \ldots, B_k\} \), we get
  \[
  \{Ta, FB_1, \ldots, FB_k\} \quad \text{and} \quad \{\{Fa, TB_1\}, \ldots, \{Fa, TB_k\}\}
  \]

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

  \[
  \begin{array}{c}
  x \leftarrow y \\
  x \leftarrow \text{not } z
  \end{array}
  \quad \{Tx, F\{y\}, F\{\text{not } z\}\} \\
  \{\{Fx, T\{y\}\}, \{Fx, T\{\text{not } z\}\}\}
  \]

For nogood \( \{Tx, F\{y\}, F\{\text{not } z\}\} \), the signed literal

\[ T\{\text{not } z\} \] is unit-resulting wrt assignment \( (Tx, F\{y\}) \)
Nogoods from logic programs
atom-oriented nogoods

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

\[
\{Ta, FB_1, \ldots, FB_k\} \quad \text{and} \quad \{\{Fa, TB_1\}, \ldots, \{Fa, TB_k\}\}
\]

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

\[
\begin{array}{c}
x \leftarrow y \\
x \leftarrow \text{not} \ z
\end{array}
\]

\[
\{Tx, F\{y\}, F\{\text{not} \ z\}\} \quad \{\{Fx, T\{y\}\}, \{Fx, T\{\text{not} \ z\}\}\}
\]

For nogood \( \{Tx, F\{y\}, F\{\text{not} \ z\}\} \), the signed literal

- \( T\{\text{not} \ z\} \) is unit-resulting wrt assignment \( (Tx, F\{y\}) \)
Nogoods from logic programs
atom-oriented nogoods

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

\[
\{Ta, FB_1, \ldots, FB_k\} \quad \text{and} \quad \{\{Fa, TB_1\}, \ldots, \{Fa, TB_k\}\}
\]

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

\[
\begin{array}{c}
x \leftarrow y \\
x \leftarrow not \ z
\end{array}
\quad \{Tx, F\{y\}, F\{not \ z\}\}
\quad \{\{Fx, T\{y\}\}, \{Fx, T\{not \ z\}\}\}
\]

For nogood \( \{Tx, F\{y\}, F\{not \ z\}\} \), the signed literal

\[ -T\{not \ z\} \] is unit-resulting wrt assignment \((Tx, F\{y\})\)
Nogoods from logic programs

body-oriented nogoods

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

$$\{FB, Ta_1, \ldots, Ta_m, Fa_{m+1}, \ldots, Fa_n\}$$

$$\{\{TB, Fa_1\}, \ldots, \{TB, Fa_m\}, \{TB, Ta_{m+1}\}, \ldots, \{TB, Ta_n\}\}$$
Nogoods from logic programs

body-oriented nogoods

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

$$\{FB, Ta_1, \ldots, Ta_m, Fa_{m+1}, \ldots, Fa_n\}$$

$$\{\{TB, Fa_1\}, \ldots, \{TB, Fa_m\}, \{TB, Ta_{m+1}\}, \ldots, \{TB, Ta_n\}\}$$

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

\[
\begin{array}{l}
\ldots \leftarrow x, \text{not } y \\
\vdots \\
\ldots \leftarrow x, \text{not } y \\
\end{array}
\]

$$\{F\{x, \text{not } y\}, Tx, Ty\}$$

$$\{\{T\{x, \text{not } y\}, Fx\}, \{T\{x, \text{not } y\}, Ty\}\}$$
Nogoods from logic programs
body-oriented nogoods

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

  $$\{FB, Ta_1, \ldots, Ta_m, Fa_{m+1}, \ldots, Fa_n\}$$

  $$\{TB, Fa_1\}, \ldots, \{TB, Fa_m\}, \{TB, Ta_{m+1}\}, \ldots, \{TB, Ta_n\}$$

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

  $\ldots \leftarrow x, \text{not } y$

  $\ldots \leftarrow x, \text{not } y$

  $\{F\{x, \text{not } y\}, Tx, Fy\}$

  $\{\{T\{x, \text{not } y\}, Fx\}, \{T\{x, \text{not } y\}, Ty\}\}$

  For nogood $\delta(\{x, \text{not } y\}) = \{F\{x, \text{not } y\}, Tx, Fy\}$, the signed literal

  - $T\{x, \text{not } y\}$ is unit-resulting wrt assignment $(Tx, Fy)$ and
  - $Ty$ is unit-resulting wrt assignment $(F\{x, \text{not } y\}, Tx)$
Characterization of stable models
for tight logic programs, ie. free of positive recursion

Let $P$ be a logic program and

$$\Delta_P = \{\delta(a) \mid a \in \text{atom}(P)\} \cup \{\delta \in \Delta(a) \mid a \in \text{atom}(P)\} \cup \{\delta \in \Delta(B) \mid B \in \text{body}(P)\} \cup \{\delta \in \Delta(B) \mid B \in \text{body}(P)\}$$
Characterization of stable models
for tight logic programs, ie. free of positive recursion

Let $P$ be a logic program and

$$\Delta_P = \{\delta(a) \mid a \in \text{atom}(P)\} \cup \{\delta \in \Delta(a) \mid a \in \text{atom}(P)\}$$
$$\cup \{\delta(B) \mid B \in \text{body}(P)\} \cup \{\delta \in \Delta(B) \mid B \in \text{body}(P)\}$$

**Theorem**

Let $P$ be a tight logic program. Then,

- $X \subseteq \text{atom}(P)$ is a stable model of $P$ iff
- $X = A^T \cap \text{atom}(P)$ for a (unique) solution $A$ for $\Delta_P$
Nogoods from logic programs via loop formulas

Let $P$ be a normal logic program and recall that:

- For $L \subseteq \text{atom}(P)$, the external supports of $L$ for $P$ are
  
  $$
  ES_P(L) = \{ r \in P \mid \text{head}(r) \in L, \text{body}(r)^+ \cap L = \emptyset \}
  $$

- The (disjunctive) loop formula of $L$ for $P$ is
  $$
  LF_P(L) = (\bigvee A \in L) \rightarrow (\bigvee r \in ES_P(L) \text{body}(r)) \equiv (\bigwedge r \in ES_P(L) \neg \text{body}(r)) \rightarrow (\bigwedge A \in L \neg A)
  $$

  - Note: The loop formula of $L$ enforces all atoms in $L$ to be false whenever $L$ is not externally supported
Nogoods from logic programs via loop formulas

Let $P$ be a normal logic program and recall that:

- For $L \subseteq \text{atom}(P)$, the external supports of $L$ for $P$ are
  \[
  ES_P(L) = \{ r \in P \mid \text{head}(r) \in L, \text{body}(r)^+ \cap L = \emptyset \}
  \]

- The (disjunctive) loop formula of $L$ for $P$ is
  \[
  LF_P(L) = (\bigvee_{A \in L} A) \rightarrow (\bigvee_{r \in ES_P(L)} \text{body}(r)) \\
  \equiv (\bigwedge_{r \in ES_P(L)} \neg \text{body}(r)) \rightarrow (\bigwedge_{A \in L} \neg A)
  \]

  – Note: The loop formula of $L$ enforces all atoms in $L$ to be false whenever $L$ is not externally supported
Nogoods from logic programs via loop formulas

Let $P$ be a normal logic program and recall that:

- For $L \subseteq \text{atom}(P)$, the external supports of $L$ for $P$ are
  \[
  ES_P(L) = \{ r \in P \mid \text{head}(r) \in L, \text{body}(r)^+ \cap L = \emptyset \}
  \]

- The (disjunctive) loop formula of $L$ for $P$ is
  \[
  LF_P(L) = (\bigvee_{A \in L} A) \rightarrow (\bigvee_{r \in ES_P(L)} \text{body}(r)) \equiv (\bigwedge_{r \in ES_P(L)} \neg \text{body}(r)) \rightarrow (\bigwedge_{A \in L} \neg A)
  \]
  – Note: The loop formula of $L$ enforces all atoms in $L$ to be false whenever $L$ is not externally supported

- The external bodies of $L$ for $P$ are
  \[
  EB_P(L) = \{ \text{body}(r) \mid r \in ES_P(L) \} 
  \]
Nogoods from logic programs
loop nogoods

- For a logic program $P$ and some $\emptyset \subset U \subseteq \text{atom}(P)$, define the loop nogood of an atom $a \in U$ as

$$\lambda(a, U) = \{Ta, FB_1, \ldots, FB_k\}$$

where $EB_P(U) = \{B_1, \ldots, B_k\}$.
Nogoods from logic programs

loop nogoods

- For a logic program $P$ and some $\emptyset \subset U \subseteq atom(P)$, define the loop nogood of an atom $a \in U$ as
  \[
  \lambda(a, U) = \{ T_a, F_{B_1}, \ldots, F_{B_k} \}
  \]
  where $EB_P(U) = \{ B_1, \ldots, B_k \}$

- We get the following set of loop nogoods for $P$:
  \[
  \Lambda_P = \bigcup_{\emptyset \subset U \subseteq atom(P)} \{ \lambda(a, U) \mid a \in U \} 
  \]
Nogoods from logic programs

loop nogoods

- For a logic program $P$ and some $\emptyset \subset U \subseteq \text{atom}(P)$, define the loop nogood of an atom $a \in U$ as

$$
\lambda(a, U) = \{Ta, FB_1, \ldots, FB_k\}
$$

where $EB_P(U) = \{B_1, \ldots, B_k\}$

- We get the following set of loop nogoods for $P$:

$$
\Lambda_P = \bigcup_{\emptyset \subset U \subseteq \text{atom}(P)} \{\lambda(a, U) \mid a \in U\}
$$

- The set $\Lambda_P$ of loop nogoods denies cyclic support among true atoms
Example

- Consider the program

\[
\begin{align*}
  & x \leftarrow \neg y \\
  & y \leftarrow \neg x \\
  & u \leftarrow x \\
  & u \leftarrow v \\
  & v \leftarrow u, y
\end{align*}
\]
Example

• Consider the program

\[
\begin{aligned}
  x & \leftarrow \text{not } y \\
  y & \leftarrow \text{not } x \\
  u & \leftarrow x \\
  u & \leftarrow v \\
  v & \leftarrow u, y
\end{aligned}
\]

• For \( u \) in the set \( \{u, v\} \), we obtain the loop nogood:

\[
\lambda(u, \{u, v\}) = \{Tu, F\{x\}\}\]
Example

• Consider the program

\[
\begin{cases}
  x \leftarrow \text{not } y \\
  y \leftarrow \text{not } x
\end{cases}
\quad
\begin{cases}
  u \leftarrow x \\
  u \leftarrow v \\
  v \leftarrow u, y
\end{cases}
\]

• For \( u \) in the set \( \{u, v\} \), we obtain the loop nogood:

\[
\lambda(u, \{u, v\}) = \{Tu, F\{x\}\}
\]

Similarly for \( v \) in \( \{u, v\} \), we get:

\[
\lambda(v, \{u, v\}) = \{Tv, F\{x\}\}
\]
Characterization of stable models

**Theorem**

Let $P$ be a logic program. Then,

1. $X \subseteq \text{atom}(P)$ is a stable model of $P$ iff
2. $X = A^T \cap \text{atom}(P)$ for a (unique) solution $A$ for $\Delta_P \cup \Lambda_P$
Characterization of stable models

Theorem

Let $P$ be a logic program. Then,

- $X \subseteq \text{atom}(P)$ is a stable model of $P$ iff
- $X = A^T \cap \text{atom}(P)$ for a (unique) solution $A$ for $\Delta_P \cup \Lambda_P$

Some remarks

- Nogoods in $\Lambda_P$ augment $\Delta_P$ with conditions checking for unfounded sets, in particular, those being loops
- While $|\Delta_P|$ is linear in the size of $P$, $\Lambda_P$ may contain exponentially many (non-redundant) loop nogoods
Towards conflict-driven search

Boolean constraint solving algorithms pioneered for SAT led to:

- **Traditional DPLL-style approach**
  (DPLL stands for ‘Davis-Putnam-Logemann-Loveland’)
  - (Unit) propagation
  - (Chronological) backtracking
  - in ASP, eg smodels

- **Modern CDCL-style approach**
  (CDCL stands for ‘Conflict-Driven Constraint Learning’)
  - (Unit) propagation
  - Conflict analysis (via resolution)
  - Learning + Backjumping + Assertion
  - in ASP, eg clasp
DPLL-style solving

loop

propagate // deterministically assign literals

if no conflict then

if all variables assigned then return solution
else decide // non-deterministically assign some literal

else

if top-level conflict then return unsatisfiable
else

backtrack // unassign literals made after last decision
flip // assign complement of last decision literal
CDCL-style solving

loop

propagate  // deterministically assign literals
if no conflict then
  if all variables assigned then return solution
else decide  // non-deterministically assign some literal
else
  if top-level conflict then return unsatisfiable
else
  analyze  // analyze conflict and add conflict constraint
  backjump  // unassign literals until conflict constraint is unit
Outline of CDNL-ASP algorithm

- Keep track of deterministic consequences by unit propagation on:
  - Program completion \([\Delta_P]\)
  - Loop nogoods, determined and recorded on demand \([\Lambda_P]\)
  - Dynamic nogoods, derived from conflicts and unfounded sets \([\nabla]\)

  • When a nogood in \([\Delta_P] \cup [\nabla]\) becomes violated:
    - Analyze the conflict by resolution (until reaching a Unique Implication Point, short: UIP)
    - Learn the derived conflict nogood \([\delta]\)
    - Backjump to the earliest (heuristic) choice such that the complement of the UIP is unit-resulting for \([\delta]\)
    - Assert the complement of the UIP and proceed (by unit propagation)

• Terminate when either:
  - Finding a stable model (a solution for \([\Delta_P] \cup [\Lambda_P]\))
  - Deriving a conflict independently of (heuristic) choices
Outline of CDNL-ASP algorithm

- Keep track of deterministic consequences by unit propagation on:
  - Program completion $[\Delta_P]$
  - Loop nogoods, determined and recorded on demand $[\Lambda_P]$
  - Dynamic nogoods, derived from conflicts and unfounded sets $[\nabla]$

- When a nogood in $\Delta_P \cup \nabla$ becomes violated:
  - Analyze the conflict by resolution
    (until reaching a Unique Implication Point, short: UIP)
  - Learn the derived conflict nogood $\delta$
  - Backjump to the earliest (heuristic) choice such that the complement of the UIP is unit-resulting for $\delta$
  - Assert the complement of the UIP and proceed
    (by unit propagation)
Outline of CDNL-ASP algorithm

- Keep track of deterministic consequences by unit propagation on:
  - Program completion \([\Delta_P]\)
  - Loop nogoods, determined and recorded on demand \([\Lambda_P]\)
  - Dynamic nogoods, derived from conflicts and unfounded sets \([\nabla]\)

- When a nogood in \(\Delta_P \cup \nabla\) becomes violated:
  - Analyze the conflict by resolution (until reaching a Unique Implication Point, short: UIP)
  - Learn the derived conflict nogood \(\delta\)
  - Backjump to the earliest (heuristic) choice such that the complement of the UIP is unit-resulting for \(\delta\)
  - Assert the complement of the UIP and proceed (by unit propagation)

- Terminate when either:
  - Finding a stable model (a solution for \(\Delta_P \cup \Lambda_P\))
  - Deriving a conflict independently of (heuristic) choices
Algorithm 1: CDNL-ASP

Input: A normal program \( P \)
Output: A stable model of \( P \) or “no stable model”

\[
A := \emptyset \quad \text{// assignment over } \text{atom}(P) \cup \text{body}(P) \\
\nabla := \emptyset \quad \text{// set of recorded nogoods} \\
dl := 0 \quad \text{// decision level}
\]

\[
\text{loop} \\
\quad (A, \nabla) := \text{NogoodPropagation}(P, \nabla, A) \\
\quad \text{if } \varepsilon \subseteq A \text{ for some } \varepsilon \in \Delta P \cup \nabla \text{ then} \\
\qquad \text{if } \max(\{dlevel(\sigma) | \sigma \in \varepsilon \cup \{0\}\}) = 0 \text{ then return no stable model} \\
\qquad (\delta, dl) := \text{ConflictAnalysis}(\varepsilon, P, \nabla, A) \\
\qquad \nabla := \nabla \cup \{\delta\} \quad \text{// (temporarily) record conflict nogood} \\
\qquad A := A \setminus \{\sigma \in A | dl < dlevel(\sigma)\} \quad \text{// backjumping} \\
\quad \text{else if } A^T \cup A^F = \text{atom}(P) \cup \text{body}(P) \text{ then} \\
\qquad \text{return } A^T \cap \text{atom}(P) \quad \text{// stable model} \\
\quad \text{else} \\
\qquad \sigma_d := \text{Select}(P, \nabla, A) \\
\qquad dl := dl + 1 \\
\qquad dlevel(\sigma_d) := dl \\
\qquad A := A \circ \sigma_d \quad \text{// decision}
\]
Observations

- Decision level $dl$, initially set to 0, is used to count the number of heuristically chosen literals in assignment $A$
- For a heuristically chosen literal $\sigma_d = Ta$ or $\sigma_d = Fa$, respectively, we require $a \in \left(\text{atom}(P) \cup \text{body}(P)\right) \setminus (A^T \cup A^F)$
- For any literal $\sigma \in A$, $dl(\sigma)$ denotes the decision level of $\sigma$, viz. the value $dl$ had when $\sigma$ was assigned
Observations

- Decision level $dl$, initially set to 0, is used to count the number of heuristically chosen literals in assignment $A$.
- For a heuristically chosen literal $\sigma_d = Ta$ or $\sigma_d = Fa$, respectively, we require $a \in (\text{atom}(P) \cup \text{body}(P)) \setminus (A^T \cup A^F)$.
- For any literal $\sigma \in A$, $dl(\sigma)$ denotes the decision level of $\sigma$, viz. the value $dl$ had when $\sigma$ was assigned.
- A conflict is detected from violation of a nogood $\varepsilon \subseteq \Delta_P \cup \nabla$.
- A conflict at decision level 0 (where $A$ contains no heuristically chosen literals) indicates non-existence of stable models.
- A nogood $\delta$ derived by conflict analysis is asserting, that is, some literal is unit-resulting for $\delta$ at a decision level $k < dl$. 

TU Dresden, 22nd May 2013  Deduction Systems  slide 78 of 119
Observations

- Decision level $dl$, initially set to 0, is used to count the number of heuristically chosen literals in assignment $A$.
- For a heuristically chosen literal $\sigma_d = Ta$ or $\sigma_d = Fa$, respectively, we require $a \in (\text{atom}(P) \cup \text{body}(P)) \setminus (A^T \cup A^F)$.
- For any literal $\sigma \in A$, $dl(\sigma)$ denotes the decision level of $\sigma$, viz. the value $dl$ had when $\sigma$ was assigned.
- A conflict is detected from violation of a nogood $\varepsilon \subseteq \Delta_P \cup \nabla$.
- A conflict at decision level 0 (where $A$ contains no heuristically chosen literals) indicates non-existence of stable models.
- A nogood $\delta$ derived by conflict analysis is asserting, that is, some literal is unit-resulting for $\delta$ at a decision level $k < dl$.
  - After learning $\delta$ and backjumping to decision level $k$, at least one literal is newly derivable by unit propagation.
  - No explicit flipping of heuristically chosen literals!
Example: CDNL-ASP

Consider

\[ P = \{ \begin{array}{llll} x \leftarrow \text{not } y & u \leftarrow x, y & v \leftarrow x & w \leftarrow \text{not } x, \text{not } y \\ y \leftarrow \text{not } x & u \leftarrow v & v \leftarrow u, y \end{array} \} \]

<table>
<thead>
<tr>
<th>( dl )</th>
<th>( \sigma_d )</th>
<th>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example: CDNL-ASP

Consider

\[ P = \{ x \leftarrow \text{not } y, u \leftarrow x, y, v \leftarrow x, w \leftarrow \text{not } x, \text{not } y \} \]

<table>
<thead>
<tr>
<th>dl</th>
<th>( \sigma_d )</th>
<th>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: CDNL-ASP

Consider

\[ P = \{ \begin{array}{llll}
    x & \leftarrow & \text{not } y & \\
    u & \leftarrow & x, y & \\
    v & \leftarrow & x & \\
    w & \leftarrow & \text{not } x, \text{not } y & \\
    y & \leftarrow & \text{not } x & \\
    u & \leftarrow & v & \\
    v & \leftarrow & u, y & \\
  \end{array} \} \]

<table>
<thead>
<tr>
<th>dl</th>
<th>( \sigma_d )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
</tr>
<tr>
<td>2</td>
<td>( F { \text{not } x, \text{not } y } )</td>
</tr>
</tbody>
</table>

\[ dl \quad \sigma_d \quad \overline{\sigma} \quad \delta \]

\[ dl \quad \sigma_d \quad \overline{\sigma} \quad \delta \]

\[ dl \quad \sigma_d \quad \overline{\sigma} \quad \delta \]

\[ dl \quad \sigma_d \quad \overline{\sigma} \quad \delta \]
Example: CDNL-ASP

Consider

\[ P = \{ x \leftarrow \text{not } y, \quad u \leftarrow x, y, \quad v \leftarrow x, \quad w \leftarrow \text{not } x, \text{not } y \\
\quad y \leftarrow \text{not } x, \quad u \leftarrow v, \quad v \leftarrow u, y \} \]

<table>
<thead>
<tr>
<th>(dl)</th>
<th>(\sigma_d)</th>
<th>(\bar{\sigma})</th>
<th>(\delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(Tu)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(F{\text{not } x, \text{not } y})</td>
<td>(Fw)</td>
<td>({Tw, F{\text{not } x, \text{not } y}} = \delta(w))</td>
</tr>
</tbody>
</table>

TU Dresden, 22nd May 2013

Deduction Systems

slide 83 of 119
Example: CDNL-ASP

Consider

\[ P = \left\{ \begin{array}{c}
x \leftarrow \text{not } y \\
y \leftarrow \text{not } x \\
u \leftarrow x, y \\
v \leftarrow x \\
w \leftarrow \text{not } x, \text{not } y \\
\end{array} \right\} \]

\[
\begin{array}{c|c|c|c}
   dl & \sigma_d & \bar{\sigma} & \delta \\
1 & Tu & & \\
2 & F\{\text{not } x, \text{not } y\} & Fw & \{Tw, F\{\text{not } x, \text{not } y\}\} = \delta(w) \\
3 & F\{\text{not } y\} & & \\
\end{array}
\]
Example: CDNL-ASP

Consider

\[
\begin{align*}
P &= \left\{ 
\begin{array}{llll}
x & \leftarrow & \text{not } y & \quad u \leftarrow x, y \\
y & \leftarrow & \text{not } x & \quad v \leftarrow x \\
\text{not } y & \leftarrow & u & \quad w \leftarrow \text{not } x, \text{not } y
\end{array}
\right\}
\end{align*}
\]

\[
\begin{array}{|c|c|c|}
\hline
dl & \sigma_d & \bar{\sigma} & \delta \\
\hline
1 & Tu & & \\
2 & F\{\text{not } x, \text{not } y\} & Fw & \{Tw, F\{\text{not } x, \text{not } y\}\} = \delta(w) \\
3 & F\{\text{not } y\} & Fx & \{Tx, F\{\text{not } y\}\} = \delta(x) \\
& & F\{x\} & \{T\{x\}, Fx\} \in \Delta(\{x\}) \\
& & F\{x, y\} & \{T\{x, y\}, Fx\} \in \Delta(\{x, y\}) \\
\hline
\end{array}
\]
Example: CDNL-ASP

Consider

\[ P = \{ \begin{array}{l} x \leftarrow \text{not } y \quad u \leftarrow x, y \quad v \leftarrow x \quad w \leftarrow \text{not } x, \text{not } y \\ y \leftarrow \text{not } x \\ u \leftarrow v \\ v \leftarrow u, y \end{array} \} \]

<table>
<thead>
<tr>
<th>( dl )</th>
<th>( \sigma_d )</th>
<th>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\text{not } x, \text{not } y} )</td>
<td>( Fw )</td>
<td>{( Tw, F{\text{not } x, \text{not } y} }} = \delta(( w ))</td>
</tr>
<tr>
<td>3</td>
<td>( F{\text{not } y} )</td>
<td>( Fx )</td>
<td>{( Tx, F{\text{not } y} }} = \delta(( x ))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x} )</td>
<td>{( T{x}, Fx }} \in \Delta({x})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x, y} )</td>
<td>{( T{x, y}, Fx }} \in \Delta({x, y})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
</tbody>
</table>
|       |       | \( \{Tu, F\{x\}, F\{x, y\} \}\} = \lambda(\( u, \{u, v\} \) \times
Example: CDNL-ASP

Consider

\[ P = \{ x \leftarrow \text{not } y \quad u \leftarrow x, y \quad v \leftarrow x \quad w \leftarrow \text{not } x, \text{not } y \\
y \leftarrow \text{not } x \quad u \leftarrow v \quad v \leftarrow u, y \} \]

<table>
<thead>
<tr>
<th>( dl )</th>
<th>( \sigma_d )</th>
<th>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td>( )</td>
<td>( )</td>
</tr>
</tbody>
</table>
Example: CDNL-ASP

Consider

\[ P = \{ x \leftarrow \text{not } y \quad u \leftarrow x, y \quad v \leftarrow x \quad w \leftarrow \text{not } x, \text{not } y \\
   y \leftarrow \text{not } x \quad u \leftarrow v \quad v \leftarrow u, y \} \]

\[
\begin{array}{|c|c|c|c|}
\hline
   dl & \sigma_d & \bar{\sigma} & \delta \\
\hline
   1  & Tu & Tx & \{Tu, Fx\} \in \nabla \\
\hline
\end{array}
\]
Example: CDNL-ASP

Consider

\[ P = \begin{cases} 
  x \leftarrow \text{not } y & u \leftarrow x, y \\
  y \leftarrow \text{not } x & v \leftarrow x \\
  u \leftarrow v & v \leftarrow u, y \\
  w \leftarrow \text{not } x, \text{not } y 
\end{cases} \]

<table>
<thead>
<tr>
<th>(dl)</th>
<th>(\sigma_d)</th>
<th>(\overline{\sigma})</th>
<th>(\delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(Tu)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Tx)</td>
<td>({Tu, Fx} \in \nabla)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\vdots)</td>
<td>(\vdots)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Tv)</td>
<td>({Fv, T{x}} \in \Delta(v))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Fy)</td>
<td>({Ty, F{\text{not } x}} = \delta(y))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Fw)</td>
<td>({Tw, F{\text{not } x, \text{not } y}} = \delta(w))</td>
<td></td>
</tr>
</tbody>
</table>
Example: CDNL-ASP

Consider

\[ P = \{ x \leftarrow \text{not } y \quad u \leftarrow x, y \quad v \leftarrow x \quad w \leftarrow \text{not } x, \text{not } y \\
y \leftarrow \text{not } x \quad u \leftarrow v \quad v \leftarrow u, y \} \]

<table>
<thead>
<tr>
<th>( dl )</th>
<th>( \sigma_d )</th>
<th>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tu</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T_x )</td>
<td>( { Tu, F_x } \in \nabla )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F_y )</td>
<td>( { T_y, F{ \text{not } x } } = \delta(y) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F_w )</td>
<td>( { T_w, F{ \text{not } x, \text{not } y } } = \delta(w) )</td>
<td></td>
</tr>
</tbody>
</table>
Outline of NogoodPropagation

- Derive deterministic consequences via:
  - Unit propagation on $\Delta_P$ and $\nabla$;
  - Unfounded sets $U \subseteq \text{atom}(P)$
- Note that $U$ is unfounded if $EB_P(U) \subseteq A^F$
  - Note: For any $a \in U$, we have $(\lambda(a, U) \setminus \{Ta\}) \subseteq A$

TU Dresden, 22nd May 2013

Deduction Systems
Outline of NogoodPropagation

- Derive deterministic consequences via:
  - Unit propagation on $\Delta_P$ and $\nabla$;
  - Unfounded sets $U \subseteq \text{atom}(P)$
- Note that $U$ is unfounded if $EB_P(U) \subseteq A^F$
  - Note: For any $a \in U$, we have $(\lambda(a, U) \setminus \{T_a\}) \subseteq A$
- An “interesting” unfounded set $U$ satisfies:

$$\emptyset \subset U \subseteq (\text{atom}(P) \setminus A^F)$$

- Wrts a fixpoint of unit propagation,
Outline of NogoodPropagation

- Derive deterministic consequences via:
  - Unit propagation on $\Delta_P$ and $\nabla$;
  - Unfounded sets $U \subseteq \text{atom}(P)$
- Note that $U$ is unfounded if $EB_P(U) \subseteq A^F$
  - Note: For any $a \in U$, we have $(\lambda(a, U) \setminus \{Ta\}) \subseteq A$
- An “interesting” unfounded set $U$ satisfies:
  $$\emptyset \subset U \subseteq (\text{atom}(P) \setminus A^F)$$
- Wrt a fixpoint of unit propagation, such an unfounded set contains some loop of $P$
  - Note: Tight programs do not yield “interesting” unfounded sets!
Outline of NogoodPropagation

- Derive deterministic consequences via:
  - Unit propagation on $\Delta_P$ and $\nabla$;
  - Unfounded sets $U \subseteq \text{atom}(P)$
- Note that $U$ is unfounded if $EB_P(U) \subseteq A^F$
  - Note: For any $a \in U$, we have $(\lambda(a, U) \setminus \{Ta\}) \subseteq A$
- An “interesting” unfounded set $U$ satisfies:
  \[
  \emptyset \subset U \subseteq (\text{atom}(P) \setminus A^F)
  \]
- Wrt a fixpoint of unit propagation, such an unfounded set contains some loop of $P$
  - Note: Tight programs do not yield “interesting” unfounded sets!
- Given an unfounded set $U$ and some $a \in U$, adding $\lambda(a, U)$ to $\nabla$ triggers
  a conflict or further derivations by unit propagation
  - Note: Add loop nogoods atom by atom to eventually falsify all $a \in U$
Algorithm 2: NogoodPropagation

Input: A normal program $P$, a set $\nabla$ of nogoods, and an assignment $A$.
Output: An extended assignment and set of nogoods.

$U := \emptyset$ // unfounded set

loop
repeat
  if $\delta \subseteq A$ for some $\delta \in \Delta_P \cup \nabla$ then return $(A, \nabla)$ // conflict
  $\Sigma := \{\delta \in \Delta_P \cup \nabla \mid \delta \setminus A = \{\sigma\}, \sigma \notin A\}$ // unit-resulting nogoods
  if $\Sigma \neq \emptyset$ then let $\sigma \in \delta \setminus A$ for some $\delta \in \Sigma$ in
    $dlevel(\sigma) := \max(\{dlevel(\rho) \mid \rho \in \delta \setminus \{\sigma\}\} \cup \{0\})$
    $A := A \cup \sigma$
until $\Sigma = \emptyset$

if $\text{loop}(P) = \emptyset$ then return $(A, \nabla)$

$U := U \setminus A^F$
if $U = \emptyset$ then $U := \text{UnfoundedSet}(P, A)$
if $U = \emptyset$ then return $(A, \nabla)$ // no unfounded set $\emptyset \subset U \subseteq \text{atom}(P) \setminus A^F$

let $a \in U$ in
  $\nabla := \nabla \cup \{Ta\} \cup \{FB \mid B \in \text{EB}_P(U)\}$ // record loop nogood
Requirements for UnfoundedSet

- Implementations of UnfoundedSet must guarantee the following for a result $U$
  1. $U \subseteq (\text{atom}(P) \setminus A^F)$
  2. $EB_P(U) \subseteq A^F$
  3. $U = \emptyset$ iff there is no nonempty unfounded subset of $(\text{atom}(P) \setminus A^F)$
Requirements for UnfoundedSet

- Implementations of UnfoundedSet must guarantee the following for a result $U$
  1. $U \subseteq (\text{atom}(P) \setminus A^F)$
  2. $\text{EB}_P(U) \subseteq A^F$
  3. $U = \emptyset$ iff there is no nonempty unfounded subset of $(\text{atom}(P) \setminus A^F)$

- Beyond that, there are various alternatives, such as:
  - Calculating the greatest unfounded set
  - Calculating unfounded sets within strongly connected components of the positive atom dependency graph of $P$
  - Usually, the latter option is implemented in ASP solvers
Example: NogoodPropagation

Consider

\[ P = \{ \begin{align*} x &\leftarrow \text{not } y \\ u &\leftarrow x, y \\ v &\leftarrow x \\ w &\leftarrow \text{not } x, \text{not } y \\ y &\leftarrow \text{not } x \\ u &\leftarrow v \\ v &\leftarrow u, y \end{align*} \} \]

<table>
<thead>
<tr>
<th>( dl )</th>
<th>( \sigma_d )</th>
<th>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\text{not } x, \text{not } y} )</td>
<td>( Fw )</td>
<td>( {Tw, F{\text{not } x, \text{not } y}} = \delta(w) )</td>
</tr>
</tbody>
</table>
| 3 | \( F\{\text{not } y\} \) | \( Fx, F\{x\}, F\{x, y\}, T\{\text{not } x\}, T\{y\}, T\{v\}, T\{u, y\}, T\{v\} \) | \( \{Tx, F\{\text{not } y\}\} = \delta(x) \)  
\( \{T\{x\},Fx\} \in \Delta(\{x\}) \)  
\( \{T\{x, y\},Fx\} \in \Delta(\{x, y\}) \)  
\( \{F\{\text{not } x\},Fx\} = \delta(\{\text{not } x\}) \)  
\( \{F\{\text{not } y\}, Fy\} = \delta(\{\text{not } y\}) \)  
\( \{Tu, F\{x, y\}, F\{v\}\} = \delta(u) \)  
\( \{F\{u, y\}, Tu, Ty\} = \delta(\{u, y\}) \)  
\( \{Fv, T\{u, y\}\} \in \Delta(v) \)  
\( \{Tu, F\{x\}, F\{x, y\}\} = \lambda(u, \{u, v\}) \) |
Conflict analysis is triggered whenever some nogood $\delta \in \Delta_P \cup \nabla$ becomes violated, viz. $\delta \subseteq A$, at a decision level $dl > 0$

- Note that all but the first literal assigned at $dl$ have been unit-resulting for nogoods $\varepsilon \in \Delta_P \cup \nabla$
- If $\sigma \in \delta$ has been unit-resulting for $\varepsilon$, we obtain a new violated nogood by resolving $\delta$ and $\varepsilon$ as follows:

\[(\delta \setminus \{\sigma\}) \cup (\varepsilon \setminus \{\overline{\sigma}\})\]
Outline of Conflict Analysis

- Conflict analysis is triggered whenever some nogood $\delta \in \Delta_P \cup \nabla$ becomes violated, viz. $\delta \subseteq A$, at a decision level $dl > 0$
  - Note that all but the first literal assigned at $dl$ have been unit-resulting for nogoods $\varepsilon \in \Delta_P \cup \nabla$
  - If $\sigma \in \delta$ has been unit-resulting for $\varepsilon$, we obtain a new violated nogood by resolving $\delta$ and $\varepsilon$ as follows:

$$ (\delta \setminus \{\sigma\}) \cup (\varepsilon \setminus \{\overline{\sigma}\}) $$

- Resolution is directed by resolving first over the literal $\sigma \in \delta$ derived last, viz. $(\delta \setminus A[\sigma]) = \{\sigma\}$
  - Iterated resolution progresses in inverse order of assignment
Outline of Conflict Analysis

- Conflict analysis is triggered whenever some nogood $\delta \in \Delta_p \cup \nabla$ becomes violated, viz. $\delta \subseteq A$, at a decision level $dl > 0$
  - Note that all but the first literal assigned at $dl$ have been unit-resulting for nogoods $\varepsilon \in \Delta_p \cup \nabla$
  - If $\sigma \in \delta$ has been unit-resulting for $\varepsilon$, we obtain a new violated nogood by resolving $\delta$ and $\varepsilon$ as follows:
    $$(\delta \setminus \{\sigma\}) \cup (\varepsilon \setminus \{\overline{\sigma}\})$$

- Resolution is directed by resolving first over the literal $\sigma \in \delta$ derived last, viz. $(\delta \setminus A[\sigma]) = \{\sigma\}$
  - Iterated resolution progresses in inverse order of assignment
- Iterated resolution stops as soon as it generates a nogood $\delta$ containing exactly one literal $\sigma$ assigned at decision level $dl$
  - This literal $\sigma$ is called First Unique Implication Point (First-UIP)
  - All literals in $(\delta \setminus \{\sigma\})$ are assigned at decision levels smaller than $dl$
Algorithm 3: ConflictAnalysis

Input: A non-empty violated nogood $\delta$, a normal program $P$, a set $\nabla$ of nogoods, and an assignment $A$.

Output: A derived nogood and a decision level.

Loop

let $\sigma \in \delta$ such that $\delta \setminus A[\sigma] = \{\sigma\}$ in

$k := \max(\{dlevel(\rho) \mid \rho \in \delta \setminus \{\sigma\}\} \cup \{0\})$

if $k = dlevel(\sigma)$ then

let $\varepsilon \in \Delta_P \cup \nabla$ such that $\varepsilon \setminus A[\sigma] = \{\sigma\}$ in

$\delta := (\delta \setminus \{\sigma\}) \cup (\varepsilon \setminus \{\sigma\})$ // resolution

else return $(\delta, k)$
Example: ConflictAnalysis

Consider

\[ P = \left\{ \begin{array}{l}
  x \leftarrow \text{not } y \\
  u \leftarrow x, y \\
  v \leftarrow x \\
  w \leftarrow \text{not } x, \text{not } y \\
  y \leftarrow \text{not } x \\
  u \leftarrow v \\
  v \leftarrow u, y
\end{array} \right\} \]

<table>
<thead>
<tr>
<th>dl</th>
<th>σ_d</th>
<th>\bar{σ}</th>
<th>δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\text{not } x, \text{not } y} )</td>
<td>( Fw )</td>
<td>( {Tw, F{\text{not } x, \text{not } y}} = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\text{not } y} )</td>
<td>( Fx )</td>
<td>( {Tx, F{\text{not } y}} = \delta(x) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x} )</td>
<td>( {T{x}, Fx} \in \Delta({x}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x, y} )</td>
<td>( {T{x, y}, Fx} \in \Delta({x, y}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T{\text{not } x} )</td>
<td>( {F{\text{not } x}, Fx} = \delta({\text{not } x}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( Ty )</td>
<td>( {F{\text{not } y}, Fy} = \delta({\text{not } y}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T{v} )</td>
<td>( {Tu, F{x, y}, F{v}} = \delta(u) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T{u, y} )</td>
<td>( {F{u, y}, Tu, Ty} = \delta({u, y}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( Tv )</td>
<td>( {Fv, T{u, y}} \in \Delta(v) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( {Tu, F{x}, F{x, y}} = \lambda(u, {u, v}) )</td>
<td>×</td>
</tr>
</tbody>
</table>
Example: ConflictAnalysis

Consider

\[ P = \{ x \leftarrow \neg y, \quad u \leftarrow x, y, \quad v \leftarrow x, \quad w \leftarrow \neg x, \neg y, \quad y \leftarrow \neg x, \quad u \leftarrow v, \quad v \leftarrow u, y \} \]

<table>
<thead>
<tr>
<th>dl</th>
<th>( \sigma_d )</th>
<th>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\neg x, \neg y} )</td>
<td>( Fw )</td>
<td>{( Tw, F{\neg x, \neg y}}} = \delta(w)</td>
</tr>
<tr>
<td>3</td>
<td>( F{\neg y} )</td>
<td>( Fx )</td>
<td>{( Tx, F{\neg y}}} = \delta(x)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x} )</td>
<td>{( T{x}, Fx}} \in \Delta({x})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x, y} )</td>
<td>{( T{x, y}, Fx}} \in \Delta({x, y})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T{\neg x} )</td>
<td>{( F{\neg x}, Fx}} = \delta({\neg x})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( Ty )</td>
<td>{( F{\neg y}, Fy}} = \delta({\neg y})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T{v} )</td>
<td>{( Tu, F{x, y}, F{v}}} = \delta(u)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T{u, y} )</td>
<td>{( F{u, y}, Tu, Ty}} = \delta({u, y})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( Tv )</td>
<td>{( Fv, T{u, y}}} \in \Delta(v)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{( Tu, F{x}, F{x, y}}} = \lambda(u, {u, v})</td>
<td></td>
</tr>
</tbody>
</table>
Example: ConflictAnalysis

Consider

\[
P = \left\{ \begin{array}{ll}
x & \leftrightarrow \text{not } y \\
u & \leftrightarrow x, y \\
v & \leftrightarrow x \\
w & \leftrightarrow \text{not } x, \text{not } y \\
y & \leftrightarrow \text{not } x \\
u & \leftrightarrow v \\
v & \leftrightarrow u, y \\
\end{array} \right\}
\]

<table>
<thead>
<tr>
<th>(dl)</th>
<th>(\sigma_d)</th>
<th>(\bar{\sigma})</th>
<th>(\delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(Tu)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(F{\text{not } x, \text{not } y})</td>
<td>(Fw)</td>
<td>({Tw, F{\text{not } x, \text{not } y}} = \delta(w))</td>
</tr>
<tr>
<td>3</td>
<td>(F{\text{not } y})</td>
<td>(Fx)</td>
<td>({Tx, F{\text{not } y}} = \delta(x))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(F{x})</td>
<td>({T{x},Fx} \in \Delta({x}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(F{x, y})</td>
<td>({T{x, y},Fx} \in \Delta({x, y}))</td>
</tr>
<tr>
<td></td>
<td>(T{\text{not } x})</td>
<td>(F{\text{not } x},Fx} = \delta({\text{not } x}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Ty)</td>
<td>(F{\text{not } y},Fy} = \delta({\text{not } y}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(T{v})</td>
<td>({Tu, F{x, y},F{v}} = \delta(u))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(T{u, y})</td>
<td>({F{u, y},Tu,Ty} = \delta({u, y}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Tv)</td>
<td>({Fv,T{u, y}} \in \Delta(v))</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>({Tu,F{x},F{x, y}} = \lambda(u, {u, v}))</td>
<td>(\times)</td>
</tr>
</tbody>
</table>
Example: ConflictAnalysis

Consider

\[ P = \left\{ \begin{array}{l}
x \leftrightarrow \text{not } y \\
y \leftrightarrow \text{not } x \\
u \leftrightarrow x, y \\
v \leftrightarrow x \\
w \leftrightarrow \text{not } x, \text{not } y
\end{array} \right. \]

<table>
<thead>
<tr>
<th>(d)</th>
<th>(\sigma_d)</th>
<th>(\bar{\sigma})</th>
<th>(\delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(Tu)</td>
<td>({})</td>
<td>({})</td>
</tr>
<tr>
<td>2</td>
<td>(F{\text{not } x, \text{not } y})</td>
<td>(F_w)</td>
<td>({Tw, F{\text{not } x, \text{not } y}} = \delta(w))</td>
</tr>
<tr>
<td>3</td>
<td>(F{\text{not } y})</td>
<td>(Fx)</td>
<td>({Tx, F{\text{not } y}} = \delta(x))</td>
</tr>
<tr>
<td></td>
<td>(F{x})</td>
<td>({T{x},Fx} \in \Delta({x}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(F{x, y})</td>
<td>({T{x, y},Fx} \in \Delta({x, y}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(T{\text{not } x})</td>
<td>({F{\text{not } x},Fx} = \delta({\text{not } x}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Ty)</td>
<td>({F{\text{not } y},Fy} = \delta({\text{not } y}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(T{v})</td>
<td>({Tu,F{x, y},F{v}} = \delta(u))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(T{u, y})</td>
<td>({F{u, y},Tu,Ty} = \delta({u, y}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Tv)</td>
<td>({Fv,T{u, y}} \in \Delta(v))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>({Tu,F{x},F{x, y}} = \lambda(u, {u, v}))</td>
<td><strong>X</strong></td>
<td></td>
</tr>
</tbody>
</table>
Example: ConflictAnalysis

Consider

\[ P = \{ x \leftarrow \text{not } y \quad u \leftarrow x, y \quad v \leftarrow x \quad w \leftarrow \text{not } x, \text{not } y \\
    y \leftarrow \text{not } x \quad u \leftarrow v \quad v \leftarrow u, y \} \]

<table>
<thead>
<tr>
<th>dl</th>
<th>( \sigma_d )</th>
<th>( \sigma )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\text{not } x, \text{not } y} )</td>
<td>( Fw )</td>
<td>( {Tw, F{\text{not } x, \text{not } y}} = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\text{not } y} )</td>
<td>( Fx )</td>
<td>( {Tx, F{\text{not } y}} = \delta(x) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x} )</td>
<td>( {T{x}, Fx} \in \Delta({x}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x, y} )</td>
<td>( {T{x, y}, Fx} \in \Delta({x, y}) )</td>
</tr>
<tr>
<td></td>
<td>( T{\text{not } x} )</td>
<td>( F{\text{not } x}, Fx )</td>
<td>( \delta({\text{not } x}) )</td>
</tr>
<tr>
<td></td>
<td>( Ty )</td>
<td>( F{\text{not } y}, Fy )</td>
<td>( \delta({\text{not } y}) )</td>
</tr>
<tr>
<td></td>
<td>( T{v} )</td>
<td>( F{u, y}, Tu, Ty )</td>
<td>( \delta({u, y}) )</td>
</tr>
<tr>
<td></td>
<td>( T{u, y} )</td>
<td>( F{u, y}, Tu, Ty )</td>
<td>( \delta({u, y}) )</td>
</tr>
<tr>
<td></td>
<td>( Tv )</td>
<td>( F{u, y}, Tu, Ty )</td>
<td>( \delta(u, {u, v}) )</td>
</tr>
</tbody>
</table>

\[ \{Tu, Fx, F\{x\}\} \]
Example: ConflictAnalysis

Consider

\[ P = \left\{ \begin{array}{l}
  x \leftarrow \text{not } y \\
  u \leftarrow x, y \\
  v \leftarrow x \\
  w \leftarrow \text{not } x, \text{not } y \\
  y \leftarrow \text{not } x \\
  u \leftarrow v \\
  v \leftarrow u, y
\end{array} \right\} \]

<table>
<thead>
<tr>
<th>dl</th>
<th>( \sigma_d )</th>
<th>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\text{not } x, \text{not } y} )</td>
<td>( Fw )</td>
<td>( { Tw, F{\text{not } x, \text{not } y}} = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\text{not } y} )</td>
<td></td>
<td>( { Tu, Fx, F{x} } )</td>
</tr>
<tr>
<td></td>
<td>( Fx )</td>
<td>( { Tx, F{\text{not } y}} = \delta(x) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F{x} )</td>
<td>( { T{x}, Fx} \in \Delta({x}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F{x, y} )</td>
<td>( { T{x, y}, Fx} \in \Delta({x, y}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T{\text{not } x} )</td>
<td>( { F{\text{not } x}, Fx} = \delta({\text{not } x}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( Ty )</td>
<td>( { F{\text{not } y}, Fy} = \delta({\text{not } y}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T{v} )</td>
<td>( { Tu, F{x, y}, F{v}} = \delta(u) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T{u, y} )</td>
<td>( { F{u, y}, Tu, Ty} = \delta({u, y}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( Tv )</td>
<td>( { Fv, T{u, y}} \in \Delta(v) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( { Tu, F{x}, F{x, y}} = \lambda(u, {u, v}) )</td>
<td>( \times )</td>
</tr>
</tbody>
</table>

TU Dresden, 22nd May 2013

Deduction Systems
Example: ConflictsAnalysis

Consider

\[ P = \{ x \leftarrow \text{not } y, u \leftarrow x, y, v \leftarrow x, w \leftarrow \text{not } x, \text{not } y \\
y \leftarrow \text{not } x, u \leftarrow v, v \leftarrow u, y \} \]

<table>
<thead>
<tr>
<th>dl</th>
<th>( \sigma_d )</th>
<th>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\text{not } x, \text{not } y} )</td>
<td>( Fw )</td>
<td>( {Tw, F{\text{not } x, \text{not } y}} = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\text{not } y} )</td>
<td>(Fx)</td>
<td>( {Tx, F{\text{not } y}} = \delta(x) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x} )</td>
<td>( {T{x}, Fx} \in \Delta{x} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x, y} )</td>
<td>( {T{x, y}, Fx} \in \Delta{x, y} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T{\text{not } x} )</td>
<td>( {F{\text{not } x}, Fx} = \delta{\text{not } x} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( Ty )</td>
<td>( {F{\text{not } y}, Fy} = \delta{\text{not } y} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T{v} )</td>
<td>( {Tu, F{x, y}, F{v}} = \delta(u) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T{u, y} )</td>
<td>( {F{u, y}, Tu, Ty} = \delta{u, y} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( Tv )</td>
<td>( {Fv, T{u, y}} \in \Delta(v) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( {Tu, F{x}, F{x, y}} = \lambda(u, {u, v}) )</td>
<td>×</td>
</tr>
</tbody>
</table>
Example: ConflictAnalysis

Consider

\[ P = \{ \begin{align*}
  x & \leftarrow \text{not } y \\
  u & \leftarrow x, y \\
  v & \leftarrow x \\
  y & \leftarrow \text{not } x \\
  u & \leftarrow v \\
  v & \leftarrow u, y \\
\end{align*} \} \]

<table>
<thead>
<tr>
<th>dl</th>
<th>( \sigma_d )</th>
<th>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\text{not } x, \text{not } y} )</td>
<td>( Fw )</td>
<td>( {Tw, F{\text{not } x, \text{not } y}} = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\text{not } y} )</td>
<td>( Fx )</td>
<td>( {Tx, F{\text{not } y}} = \delta(x) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x} )</td>
<td>( {T{x}, Fx} \in \Delta({x}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x, y} )</td>
<td>( {T{x, y}, Fx} \in \Delta({x, y}) )</td>
</tr>
<tr>
<td></td>
<td>( T{\text{not } x} )</td>
<td>( F{\text{not } x}, Fx)</td>
<td>( \delta({\text{not } x}) )</td>
</tr>
<tr>
<td></td>
<td>( Ty )</td>
<td>( F{\text{not } y}, Fy)</td>
<td>( \delta({\text{not } y}) )</td>
</tr>
<tr>
<td></td>
<td>( T{v} )</td>
<td>( Fv, T{u, y}, F{v})</td>
<td>( \delta(u) )</td>
</tr>
<tr>
<td></td>
<td>( T{u, y} )</td>
<td>( F{u, y}, Tu, Ty)</td>
<td>( \delta({u, y}) )</td>
</tr>
<tr>
<td></td>
<td>( Tv )</td>
<td>( Fv, T{u, y})</td>
<td>( \Delta(v) )</td>
</tr>
<tr>
<td></td>
<td>( F{x}, F{x, y})</td>
<td>( \lambda(u, {u, v}) )</td>
<td>( \times )</td>
</tr>
</tbody>
</table>
Example: ConflictAnalysis

Consider

\[ P = \begin{cases} 
  x &\leftrightarrow not y \\
  u &\leftarrow x, y \\
  v &\leftarrow x \\
  w &\leftarrow not x, not y \\
  y &\leftrightarrow not x \\
  u &\leftarrow v \\
  v &\leftarrow u, y 
\end{cases} \]

<table>
<thead>
<tr>
<th>dl</th>
<th>( \sigma_d )</th>
<th>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{not x, not y} )</td>
<td>( Tw, F{not x, not y} )</td>
<td>( \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{not y} )</td>
<td>( Tx, F{not y} )</td>
<td>( \delta(x) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T{x},Fx )</td>
<td>( {Tu,Fx} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T{x,y},Fx )</td>
<td>( {Tu,Fx,F{x}} )</td>
</tr>
<tr>
<td></td>
<td>( T{not x} )</td>
<td>( F{not x},Fx )</td>
<td>( \delta({not x}) )</td>
</tr>
<tr>
<td></td>
<td>( Ty )</td>
<td>( F{not y},Fy )</td>
<td>( \delta({not y}) )</td>
</tr>
<tr>
<td></td>
<td>( T{v} )</td>
<td>( Tu,F{x,y},F{v} )</td>
<td>( \delta(u) )</td>
</tr>
<tr>
<td></td>
<td>( T{u,y} )</td>
<td>( F{u,y},Tu,Ty )</td>
<td>( \delta({u,y}) )</td>
</tr>
<tr>
<td></td>
<td>( Tv )</td>
<td>( Fv,T{u,y} )</td>
<td>( \Delta(v) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( Tu,F{x},F{x,y} )</td>
<td>( \lambda(u,{u,v}) )</td>
</tr>
</tbody>
</table>

TU Dresden, 22nd May 2013

Deduction Systems
Example: ConflictAnalysis

Consider

\[
P = \{ \begin{array}{l}
x \leftrightarrow \text{not } y \\
u \leftrightarrow x, y \\
v \leftrightarrow x \\
w \leftrightarrow \text{not } x, \text{not } y \\
y \leftrightarrow \text{not } x \\
u \leftrightarrow v \\
v \leftrightarrow u, y \\
\end{array} \}
\]

<table>
<thead>
<tr>
<th>dl</th>
<th>(\sigma_d)</th>
<th>(\bar{\sigma})</th>
<th>(\delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(Tu)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(F{\text{not } x, \text{not } y})</td>
<td>(Fw)</td>
<td>({Tw, F{\text{not } x, \text{not } y}} = \delta(w))</td>
</tr>
<tr>
<td>3</td>
<td>(F{\text{not } y})</td>
<td>(Fx)</td>
<td>({Tx, F{\text{not } y}} = \delta(x))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(F{x})</td>
<td>({T{x}, Fx} \in \Delta({x}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(F{x, y})</td>
<td>({T{x, y}, Fx} \in \Delta({x, y}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(T{\text{not } x})</td>
<td>({F{\text{not } x}, Fx} = \delta({\text{not } x}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Ty)</td>
<td>({F{\text{not } y}, Fy} = \delta({\text{not } y}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(T{v})</td>
<td>({Tu, F{x, y}, F{v}} = \delta({u}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(T{u, y})</td>
<td>({F{u, y}, Tu, Ty} = \delta({u, y}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Tv)</td>
<td>({Fv, T{u, y}} \in \Delta(v))</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>({Tu, F{x}, F{x, y}} = \lambda(u, {u, v}))</td>
</tr>
</tbody>
</table>

TU Dresden, 22nd May 2013

Deduction Systems
Example: ConflictAnalysis ctd.

Consider

\[
P = \{ \begin{array}{llll}
x & \leftarrow & \text{not } y & u \leftarrow x, y \\
y & \leftarrow & \text{not } x & v \leftarrow x \\
\end{array} \begin{array}{llll}
w & \leftarrow & \text{not } x, \text{not } y & v \leftarrow u, y \\
\end{array} \}
\]

<table>
<thead>
<tr>
<th>( dl )</th>
<th>( \sigma_d )</th>
<th>( \overline{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td>( Tx )</td>
<td>( { Tu, Fx } \in \nabla )</td>
</tr>
</tbody>
</table>
Example: ConflictAnalysis ctd.

Consider

\[
P = \left\{ \begin{array}{l}
x \leftarrow \text{not } y \\
u \leftarrow x, y \\
v \leftarrow x \\
w \leftarrow \text{not } x, \text{not } y \\
y \leftarrow \text{not } x \\
u \leftarrow v \\
v \leftarrow u, y \\
\end{array} \right. \]

<table>
<thead>
<tr>
<th>dl</th>
<th>( \sigma_d )</th>
<th>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td>( T_x )</td>
<td>( { Tu, F_x } \in \nabla )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td></td>
<td>( Tv )</td>
<td>( { F_v, T{x} } \in \Delta(v) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F_y )</td>
<td>( { T_y, F{\text{not } x} } = \delta(y) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F_w )</td>
<td>( { T_w, F{\text{not } x, \text{not } y} } = \delta(w) )</td>
<td></td>
</tr>
</tbody>
</table>
Remarks

- There always is a First-UIP at which conflict analysis terminates
  - In the worst, resolution stops at the heuristically chosen literal assigned at decision level $dl$
Remarks

- There always is a First-UIP at which conflict analysis terminates
  - In the worst, resolution stops at the heuristically chosen literal assigned at decision level $dl$
- The nogood $\delta$ containing First-UIP $\sigma$ is violated by $A$, viz. $\delta \subseteq A$
- We have $k = \max(\{dl(\rho) \mid \rho \in \delta \setminus \{\sigma\}\} \cup \{0\}) < dl$
Remarks

- There always is a First-UIP at which conflict analysis terminates
  - In the worst, resolution stops at the heuristically chosen literal assigned at decision level $dl$
- The nogood $\delta$ containing First-UIP $\sigma$ is violated by $A$, viz. $\delta \subseteq A$
- We have $k = \max(\{dl(\rho) \mid \rho \in \delta \setminus \{\sigma\}\} \cup \{0\}) < dl$
  - After recording $\delta$ in $\nabla$ and backjumping to decision level $k$, $\overline{\sigma}$ is unit-resulting for $\delta$!
  - Such a nogood $\delta$ is called **asserting**
Remarks

- There always is a First-UIP at which conflict analysis terminates
  - In the worst, resolution stops at the heuristically chosen literal assigned at decision level $dl$

- The nogood $\delta$ containing First-UIP $\sigma$ is violated by $A$, viz. $\delta \subseteq A$

- We have $k = \max\{\text{dl}(\rho) \mid \rho \in \delta \setminus \{\sigma\} \cup \{0\}\} < \text{dl}$
  - After recording $\delta$ in $\nabla$ and backjumping to decision level $k$, $\overline{\sigma}$ is unit-resulting for $\delta$!
  - Such a nogood $\delta$ is called asserting

- Asserting nogoods direct conflict-driven search into a different region of the search space than traversed before, without explicitly flipping any heuristically chosen literal!
References

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

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