Answer Set Programming: Solving

Sebastian Rudolph

Computational Logic Group
Technische Universität Dresden

Slides based on a lecture by Martin Gebser and Torsten Schaub. Potassco Slide Packages are licensed under a Creative Commons Attribution 3.0 Unported License.
1 Motivation

2 Boolean constraints

3 Nogoods from logic programs
   - Nogoods from program completion
   - Nogoods from loop formulas

4 Conflict-driven nogood learning
   - CDNL-ASP Algorithm
   - Nogood Propagation
   - Conflict Analysis
Motivation

- **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

1 Motivation

2 Boolean constraints

3 Nogoods from logic programs
   - Nogoods from program completion
   - Nogoods from loop formulas

4 Conflict-driven nogood learning
   - CDNL-ASP Algorithm
   - Nogood Propagation
   - Conflict Analysis
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 $T\nu$ or $F\nu$ for $\nu \in \text{dom}(A)$ and $1 \leq i \leq n$
- $T\nu$ expresses that $\nu$ is true and $F\nu$ that it is false
- The complement, $\overline{\sigma}$, of a literal $\sigma$ is defined as $\overline{T\nu} = F\nu$ and $\overline{F\nu} = T\nu$
- $A \circ \sigma$ 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
  
  $A^T = \{\nu \in \text{dom}(A) \mid T\nu \in A\}$ and $A^F = \{\nu \in \text{dom}(A) \mid F\nu \in A\}$
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 $A[\sigma_k] = (\sigma_1, \ldots, \sigma_{k-1})$.
- We sometimes identify an assignment with the set of its literals.

Given this, we access true and false propositions in $A$ via

$$A^T = \{v \in \text{dom}(A) \mid T_v \in A\} \text{ and } A^F = \{v \in \text{dom}(A) \mid F_v \in A\}.$$
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 $A[\sigma_k] = (\sigma_1, \ldots, \sigma_{k-1})$.
- 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 $A^F = \{v \in \text{dom}(A) \mid F_v \in A\}$. 

Sebastian Rudolph (TUD)
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 $A[\sigma_k] = (\sigma_1, \ldots, \sigma_{k-1})$.

- We sometimes identify an assignment with the set of its literals.

- Given this, we access true and false propositions in $A$ via

$$A^T = \{ v \in \text{dom}(A) \mid T_v \in A \} \text{ and } A^F = \{ v \in \text{dom}(A) \mid F_v \in A \}$$
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 $Tv$ or $Fv$ for $v \in \text{dom}(A)$ and $1 \leq i \leq n$.

- $Tv$ expresses that $v$ is true and $Fv$ that it is false.

- The complement, $\overline{\sigma}$, of a literal $\sigma$ is defined as $\overline{Tv} = Fv$ and $\overline{Fv} = Tv$.

- $A \circ \sigma$ 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

$$A^T = \{v \in \text{dom}(A) \mid Tv \in A\} \text{ and } A^F = \{v \in \text{dom}(A) \mid Fv \in A\}$$
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 $A[\sigma_k] = (\sigma_1, \ldots, \sigma_{k-1})$.

- We sometimes identify an assignment with the set of its literals.

- Given this, we access true and false propositions in $A$ via

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

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 $T\nu$ or $F\nu$ for $\nu \in \text{dom}(A)$ and $1 \leq i \leq n$
- $T\nu$ expresses that $\nu$ is true and $F\nu$ that it is false
- The complement, $\bar{\sigma}$, of a literal $\sigma$ is defined as $\overline{T\nu} = F\nu$ and $\overline{F\nu} = T\nu$
- $A \circ \sigma$ 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
  $$A^T = \{ \nu \in \text{dom}(A) \mid T\nu \in A \} \text{ and } A^F = \{ \nu \in \text{dom}(A) \mid F\nu \in A \}$$
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 \(\Delta\).
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 \( \Delta \).
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 \( \overline{\sigma} \) 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 \( \Delta \).
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 \(\overline{\sigma}\) 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 \(\Delta\).
Outline

1 Motivation

2 Boolean constraints

3 Nogoods from logic programs
   - Nogoods from program completion
   - Nogoods from loop formulas

4 Conflict-driven nogood learning
   - CDNL-ASP Algorithm
   - Nogood Propagation
   - Conflict Analysis
Outline

1 Motivation

2 Boolean constraints

3 Nogoods from logic programs
   - Nogoods from program completion
     - Nogoods from loop formulas

4 Conflict-driven nogood learning
   - CDNL-ASP Algorithm
   - Nogood Propagation
   - Conflict Analysis
Nogoods from logic programs
via program completion

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

$$\{v_B \leftrightarrow a_1 \land \cdots \land a_m \land \neg a_{m+1} \land \cdots \land \neg a_n \mid B \in body(P) \text{ and } B = \{a_1, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n\}\}$$

$$\bigcup \{a \leftrightarrow v_{B_1} \lor \cdots \lor v_{B_k} \mid a \in atom(P) \text{ and } body_P(a) = \{B_1, \ldots, B_k\}\},$$

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

The (body-oriented) equivalence

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

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

The (body-oriented) equivalence

\[ \nu_B \iff a_1 \land \cdots \land a_m \land \neg a_{m+1} \land \cdots \land \neg a_n \]

can be decomposed into two implications:

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

is equivalent to the conjunction of

\[ \neg \nu_B \lor a_1, \ldots, \neg \nu_B \lor a_m, \neg \nu_B \lor \neg a_{m+1}, \ldots, \neg \nu_B \lor \neg a_n \]

and induces the set of nogoods

\[ \Delta(B) = \{ \{TB, Fa_1\}, \ldots, \{TB, Fa_m\}, \{TB, Ta_{m+1}\}, \ldots, \{TB, Ta_n\} \} \]
The (body-oriented) equivalence

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

can be decomposed into two implications:

\[ 2 \quad a_1 \land \cdots \land a_m \land \neg a_{m+1} \land \cdots \land \neg a_n \rightarrow \nu_B \]
gives rise to the nogood

\[ \delta(B) = \{ F_B, T_{a_1}, \ldots, T_{a_m}, F_{a_{m+1}}, \ldots, F_{a_n} \} \]
Analogously, the (atom-oriented) equivalence

\[ a \leftrightarrow v_{B_1} \lor \cdots \lor v_{B_k} \]

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 $\text{body}_P(a) = \{B_1, \ldots, B_k\}$, we get

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

Example Given Atom $x$ with $\text{body}(x) = \{\{y\}, \{\sim z\}\}$, we obtain

\[
\begin{align*}
{x} & \leftarrow \ y \\
{x} & \leftarrow \sim z
\end{align*}
\]

$\{T x, F\{y\}, F\{\sim z\}\}$

$\{\{F x, T\{y\}\}, \{F x, T\{\sim z\}\}\}$

For nogood $\{T x, F\{y\}, F\{\sim z\}\}$, the signed literal

$F x$ is unit-resulting wrt assignment $\{F\{y\}, F\{\sim z\}\}$ and $T\{\sim z\}$ is unit-resulting wrt assignment $\{T x, F\{y\}\}$
Nogoods from logic programs
atom-oriented nogoods

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

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

**Example** Given Atom $x$ with $\text{body}(x) = \{\{y\}, \{\sim z\}\}$, we obtain

\[
\begin{align*}
x & \leftarrow y \\
x & \leftarrow \sim z
\end{align*}
\]

\[
\{Tx, F\{y\}, F\{\sim z\}\} \\
\{\{Fx, T\{y\}\}, \{Fx, T\{\sim z\}\}\}
\]

For nogood $\{Tx, F\{y\}, F\{\sim z\}\}$, the signed literal $Fx$ is unit-resulting wrt assignment $(Fx, F\{y\})$ and $T\{\sim z\}$ is unit-resulting wrt assignment $(Tx, F\{y\})$. 
Nogoods from logic programs
atom-oriented nogoods

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

$$\{\top a, \bot B_1, \ldots, \bot B_k\} \quad \text{and} \quad \{\{\bot a, \top B_1\}, \ldots, \{\bot a, \top B_k\}\}$$

Example Given Atom $x$ with $\text{body}(x) = \{\{y\}, \{\sim z\}\}$, we obtain

$$\begin{align*}
x &\leftarrow y \\
x &\leftarrow \sim z
\end{align*}$$

For nogood $\{\top x, \bot \{y\}, \bot \{\sim z\}\}$, the signed literal

- $\bot x$ is unit-resulting wrt assignment $(\bot \{y\}, \bot \{\sim z\})$ and
- $\top \{\sim z\}$ is unit-resulting wrt assignment $(\top x, \bot \{y\})$
Nogoods from logic programs

atom-oriented nogoods

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

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

Example Given Atom $x$ with $\text{body}(x) = \{\{y\}, \{\sim z\}\}$, we obtain

$$x \leftarrow y$$
$$x \leftarrow \sim z$$

$$\{T x, F \{y\}, F \{\sim z\}\}$$
$$\{\{F x, T \{y\}\}, \{F x, T \{\sim z\}\}\}$$

For nogood $\{T x, F \{y\}, F \{\sim z\}\}$, the signed literal

- $F x$ is unit-resulting wrt assignment $(F \{y\}, F \{\sim z\})$ and
- $T \{\sim z\}$ is unit-resulting wrt assignment $(T x, F \{y\})$
For an atom $a$ where $\text{body}_P(a) = \{B_1, \ldots, B_k\}$, we get

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

**Example** Given Atom $x$ with $\text{body}(x) = \{\{y\}, \{\sim z\}\}$, we obtain

$$x \leftarrow y$$
$$x \leftarrow \sim z$$

\begin{align*}
\{Tx, F\{y\}, F\{\sim z\}\} \\
\{\{Fx, T\{y\}\}, \{Fx, T\{\sim z\}\}\}
\end{align*}

For nogood $\{Tx, F\{y\}, F\{\sim z\}\}$, the signed literal

- $Fx$ is unit-resulting wrt assignment $(F\{y\}, F\{\sim z\})$ and
- $T\{\sim z\}$ is unit-resulting wrt assignment $(Tx, F\{y\})$
Nogoods from logic programs
atom-oriented nogoods

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

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

Example Given Atom $x$ with $body(x) = \{\{y\}, \{\sim z\}\}$, we obtain

$$x \leftarrow y$$
$$x \leftarrow \sim z$$

For nogood $\{Tx, F\{y\}, F\{\sim z\}\}$, the signed literal

- $Fx$ is unit-resulting wrt assignment $(F\{y\}, F\{\sim z\})$ and
- $T\{\sim z\}$ is unit-resulting wrt assignment $(Tx, F\{y\})$
Nogoods from logic programs
atom-oriented nogoods

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

**Example** Given Atom \( x \) with \( \text{body}(x) = \{\{y\}, \{\sim z\}\} \), we obtain
\[
\begin{align*}
x & \leftarrow y \\
x & \leftarrow \sim z
\end{align*}
\[
\{Tx, F\{y\}, F\{\sim z\}\} \\
\{\{Fx, T\{y\}\}, \{Fx, T\{\sim z\}\}\}
\]

For nogood \( \{Tx, F\{y\}, F\{\sim z\}\} \), the signed literal
\[
\begin{align*}
Fx \quad \text{is unit-resulting wrt assignment} \ (F\{y\}, F\{\sim z\}) \quad \text{and} \\
T\{\sim z\} \quad \text{is unit-resulting wrt assignment} \ (Tx, F\{y\})
\end{align*}
\]
For an atom $a$ where $\text{body}_P(a) = \{B_1, \ldots, B_k\}$, we get

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

**Example** Given Atom $x$ with $\text{body}(x) = \{\{y\}, \{\sim z\}\}$, we obtain

$$x \leftarrow y$$

$$x \leftarrow \sim z$$

$$\{Tx, F\{y\}, F\{\sim z\}\}$$

$$\{\{Fx, T\{y\}\}, \{Fx, T\{\sim z\}\}\}$$

For nogood $\{Tx, F\{y\}, F\{\sim z\}\}$, the signed literal

- $Fx$ is unit-resulting wrt assignment $(F\{y\}, F\{\sim z\})$ and
- $T\{\sim z\}$ is unit-resulting wrt assignment $(Tx, F\{y\})$
Nogoods from logic programs

atom-oriented nogoods

For an atom \( a \) where \( \text{body}_P(a) = \{ B_1, \ldots, B_k \} \), we get
\[
\{ T_\alpha, F B_1, \ldots, F B_k \} \quad \text{and} \quad \{ \{ F_\alpha, T B_1 \}, \ldots, \{ F_\alpha, T B_k \} \}
\]

Example: Given Atom \( x \) with \( \text{body}(x) = \{ \{ y \}, \{ \sim z \} \} \), we obtain
\[
\begin{align*}
&x \leftarrow y \\
x \leftarrow \sim z
\end{align*}
\]
\[
\{ T_\alpha, F \{ y \}, F \{ \sim z \} \} \\
\{ \{ F_\alpha, T \{ y \} \}, \{ F_\alpha, T \{ \sim z \} \} \}
\]

For nogood \( \{ T_\alpha, F \{ y \}, F \{ \sim z \} \} \), the signed literal
\[
\begin{align*}
&\text{F}_x \text{ is unit-resulting wrt assignment } (F \{ y \}, F \{ \sim z \}) \text{ and} \\
&T \{ \sim z \} \text{ is unit-resulting wrt assignment } (T_\alpha, F \{ y \})
\end{align*}
\]
Nogoods from logic programs

atom-oriented nogoods

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

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

Example

Given Atom $x$ with $\text{body}(x) = \{\{y\}, \{\sim z\}\}$, we obtain

$$x \leftarrow y \quad \{Tx, F\{y\}, F\{\sim z\}\}$$

$$x \leftarrow \sim z \quad \{\{Fx, T\{y\}\}, \{Fx, T\{\sim z\}\}\}$$

For nogood $\{Tx, F\{y\}, F\{\sim z\}\}$, the signed literal

$F_x$ is unit-resulting wrt assignment $(F\{y\}, F\{\sim z\})$ and

$T\{\sim z\}$ is unit-resulting wrt assignment $(Tx, F\{y\})$
Nogoods from logic programs

atom-oriented nogoods

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

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

**Example** Given Atom $x$ with $\text{body}(x) = \{\{y\}, \{\sim z\}\}$, we obtain

$$x \leftarrow y \quad \{T x, F\{y\}, F\{\sim z\}\}$$

$$x \leftarrow \sim z \quad \{\{F x, T\{y\}\}, \{F x, T\{\sim z\}\}\}$$

For nogood $\{T x, F\{y\}, F\{\sim z\}\}$, the signed literal

$F x$ is unit-resulting wrt assignment $(F\{y\}, F\{\sim z\})$ and

$T\{\sim z\}$ is unit-resulting wrt assignment $(T x, F\{y\})$
For an atom $a$ where $\text{body}_P(a) = \{B_1, \ldots, B_k\}$, we get
\[
\{Ta, FB_1, \ldots, FB_k\} \quad \text{and} \quad \{\{Fa, TB_1\}, \ldots, \{Fa, TB_k\}\}
\]

**Example** Given Atom $x$ with $\text{body}(x) = \{\{y\}, \{\sim z\}\}$, we obtain
\[
\begin{align*}
x & \leftarrow y \\
x & \leftarrow \sim z
\end{align*}
\]
\[
\begin{align*}
\{Tx, F\{y\}, F\{\sim z\}\} \\
\{\{Fx, T\{y\}\}, \{Fx, T\{\sim z\}\}\}
\end{align*}
\]

For nogood $\{Tx, F\{y\}, F\{\sim z\}\}$, the signed literal $Fx$ is unit-resulting wrt assignment $(F\{y\}, F\{\sim z\})$ and $T\{\sim z\}$ is unit-resulting wrt assignment $(Tx, F\{y\})$.
Nogoods from logic programs
atom-oriented nogoods

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

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

Example: Given Atom $x$ with $\text{body}(x) = \{\{y\}, \{\sim z\}\}$, we obtain

$$x \leftarrow y \quad \{Tx, F\{y\}, F\{\sim z\}\} \quad \{\{Fx, T\{y\}\}, \{Fx, T\{\sim z\}\}\}$$

For nogood $\{Tx, F\{y\}, F\{\sim z\}\}$, the signed literal $Fx$ is unit-resulting wrt assignment $(F\{y\}, F\{\sim z\})$ and $T\{\sim z\}$ is unit-resulting wrt assignment $(Tx, F\{y\})$.
Nogoods from logic programs

atom-oriented nogoods

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

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

Example

Given Atom \( x \) with \( \text{body}(x) = \{\{y\}, \{\sim z\}\} \), we obtain

\[
\begin{align*}
x & \leftarrow y \\
x & \leftarrow \sim z
\end{align*}
\]

{\(T x, F\{y\}, F\{\sim z\}\)}

{\(\{F x, T\{y\}\}, \{F x, T\{\sim z\}\}\) }

For nogood \( \{T x, F\{y\}, F\{\sim z\}\} \), the signed literal

\(F x\) is unit-resulting wrt assignment \((F\{y\}, F\{\sim z\})\) and

\(T\{\sim z\}\) is unit-resulting wrt assignment \((T x, F\{y\})\)
Nogoods from logic programs
body-oriented nogoods

For a body \( B = \{a_1, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n\} \), 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 \( \{x, \neg y\} \), we obtain

\[
\begin{align*}
\ldots & \leftarrow x, \neg y \\
\ldots & \leftarrow x, \neg y
\end{align*}
\]

\[
\{F\{x, \neg y\}, Tx, Fy\} \\
\{\{T\{x, \neg y\}, Fx\}, \{T\{x, \neg y\}, Ty\}\}
\]

For nogood \( \delta(\{x, \neg y\}) = \{F\{x, \neg y\}, Tx, Fy\} \), the signed literal

\[
\begin{align*}
\text{T}\{x, \neg y\} & \text{ is unit-resulting wrt assignment } (Tx, Fy) \text{ and} \\
\text{T}y & \text{ is unit-resulting wrt assignment } (F\{x, \neg y\}, Tx)
\end{align*}
\]
Nogoods from logic programs

body-oriented nogoods

For a body $B = \{a_1, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n\}$, we get

\[
\{FB, Ta_1, \ldots, Tam, Fa_{m+1}, \ldots, Fa_n\} \\
\{\{TB, Fa_1\}, \ldots, \{TB, Fa_m\}, \{TB, Ta_{m+1}\}, \ldots, \{TB, Ta_n\}\}
\]

Example

Given Body $\{x, \neg y\}$, we obtain

\[
\cdots \leftarrow x, \neg y \\
\quad \vdots \\
\cdots \leftarrow x, \neg y
\]

\[
\{F\{x, \neg y\}, Tx, Fy\} \\
\{\{T\{x, \neg y\}, Fx\}, \{T\{x, \neg y\}, Ty\}\}
\]

For nogood $\delta(\{x, \neg y\}) = \{F\{x, \neg y\}, Tx, Fy\}$, the signed literal

- $T\{x, \neg y\}$ is unit-resulting wrt assignment $(Tx, Fy)$ and
- $Ty$ is unit-resulting wrt assignment $(F\{x, \neg y\}, Tx)$
Nogoods from logic programs

body-oriented nogoods

For a body \( B = \{a_1, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n\} \), we get

\[
\{F_B, T_{a_1}, \ldots, T_{a_m}, F_{a_{m+1}}, \ldots, F_{a_n}\}
\]
\[
\{\{T_B, F_{a_1}\}, \ldots, \{T_B, F_{a_m}\}, \{T_B, T_{a_{m+1}}\}, \ldots, \{T_B, T_{a_n}\}\}
\]

Example

Given Body \( \{x, \neg y\} \), we obtain

\[
\ldots \leftarrow x, \neg y
\]
\[
\ldots \leftarrow x, \neg y
\]
\[
\{F\{x, \neg y\}, T_x, F_y\}
\]
\[
\{\{T\{x, \neg y\}, F_x\}, \{T\{x, \neg y\}, T_y\}\}
\]

For nogood \( \delta(\{x, \neg y\}) = \{F\{x, \neg y\}, T_x, F_y\} \), the signed literal

- \( T\{x, \neg y\} \) is unit-resulting wrt assignment (\( T_x, F_y \)) and
- \( T_y \) is unit-resulting wrt assignment (\( F\{x, \neg y\}, T_x \))
Characterization of stable models for tight logic programs

Let $P$ be a logic program and

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

Theorem

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

$X \subseteq atom(P)$ is a stable model of $P$ iff

$X = A^T \cap atom(P)$ for a (unique) solution $A$ for $\Delta_P$
Characterization of stable models
for tight logic programs

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 $\Delta_P$
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 $\Delta_P$
Outline

1 Motivation

2 Boolean constraints

3 Nogoods from logic programs
   - Nogoods from program completion
   - Nogoods from loop formulas

4 Conflict-driven nogood learning
   - CDNL-ASP Algorithm
   - Nogood Propagation
   - Conflict Analysis
Let $P$ be a normal logic program and recall that:

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

- 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
  \[ EB_P(L) = \{ \text{body}(r) \mid r \in ES_P(L) \} \]
Let \( P \) be a normal logic program and recall that:

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

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

  **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
  \[
  \text{EB}_P(L) = \{ \text{body}(r) \mid r \in \text{ES}_P(L) \} 
  \]
Let $P$ be a normal logic program and recall that:

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

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

- 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
  $$EB_P(L) = \{ \text{body}(r) \mid r \in ES_P(L) \}$$
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.
For a logic program $P$ and some $\emptyset \subset U \subseteq \text{atom}(P)$, define the \textit{loop nogood} of an atom $a \in U$ as

$$\lambda(a, U) = \{T a, \text{FB}_1, \ldots, \text{FB}_k\}$$

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 \textit{true} atoms.
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

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

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:

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

Consider the program

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

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

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

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

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

Consider the program

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

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

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

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

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

Theorem

Let $P$ be a logic program. Then,

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

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.
Characterization of stable models

Theorem

Let $P$ be a logic program. Then,

\[ X \subseteq \text{atom}(P) \text{ is a stable model of } P \text{ iff } \]

\[ X = A^T \cap \text{atom}(P) \text{ for a (unique) solution } A \text{ for } \Delta_P \cup \Lambda_P \]

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.
Outline

1 Motivation

2 Boolean constraints

3 Nogoods from logic programs
   - Nogoods from program completion
   - Nogoods from loop formulas

4 Conflict-driven nogood learning
   - CDNL-ASP Algorithm
   - Nogood Propagation
   - Conflict Analysis
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*
Conflict-driven nogood learning

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 propagated 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
1 Motivation

2 Boolean constraints

3 Nogoods from logic programs
   - Nogoods from program completion
   - Nogoods from loop formulas

4 Conflict-driven nogood learning
   - CDNL-ASP Algorithm
   - Nogood Propagation
   - Conflict Analysis
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)

- 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)

- 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”

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

loop

$(A, \nabla) := \text{NogoodPropagation}(P, \nabla, A)$

if $\varepsilon \subseteq A$ for some $\varepsilon \in \Delta_P \cup \nabla$ then

if $\max(\{\text{dlevel}(\sigma) \mid \sigma \in \varepsilon\} \cup \{0\}) = 0$ then return no stable model

$(\delta, dl) := \text{ConflictAnalysis}(\varepsilon, P, \nabla, A)$
$\nabla := \nabla \cup \{\delta\}$  // (temporarily) record conflict nogood
$A := A \setminus \{\sigma \in A \mid dl < \text{dlevel}(\sigma)\}$  // backjumping

else if $A^T \cup A^F = \text{atom}(P) \cup \text{body}(P)$ then

return $A^T \cap \text{atom}(P)$  // stable model

else

$\sigma_d := \text{Select}(P, \nabla, A)$  // decision
$dl := dl + 1$
$d\text{level}(\sigma_d) := dl$
$A := A \circ \sigma_d$

end
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 $\sigma_d = T_a$ or $\sigma_d = F_a$, 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!
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!

Sebastian Rudolph (TUD) Answer Set Programming: Solving 27 / 39
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

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

<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>( T_u )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\neg x, \neg y} )</td>
<td>( F_w )</td>
<td>( { T_w, F{\neg x, \neg y}} = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\neg y} )</td>
<td>( F_x )</td>
<td>( { T_x, F{\neg y}} = \delta(x) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x} )</td>
<td>( { T{x}, F_x} \in \Delta({x}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x, y} )</td>
<td>( { T{x, y}, F_x} \in \Delta({x, y}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( { T_u, F{x}, F{x, y}} = \lambda(u, {u, v}) )</td>
<td></td>
</tr>
</tbody>
</table>
Consider

\[
P = \begin{cases} 
  x \leftarrow \neg y & u \leftarrow x, y & v \leftarrow x & w \leftarrow \neg x, \neg y \\
  y \leftarrow \neg x & u \leftarrow v & v \leftarrow u, y 
\end{cases}
\]

<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>(T_u)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(F{\neg x, \neg y})</td>
<td>(F_w)</td>
<td>({T_w, F{\neg x, \neg y}} = \delta(w))</td>
</tr>
<tr>
<td>3</td>
<td>(F{\neg y})</td>
<td>(F_x)</td>
<td>({T_x, F{\neg y}} = \delta(x))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(F{x})</td>
<td>({T{x}, F_x} \in \Delta({x}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(F{x, y})</td>
<td>({T{x, y}, F_x} \in \Delta({x, y}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(\vdots)</td>
<td>(\vdots)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>({T_u, F{x}, F{x, y}} = \lambda(u, {u, v}))</td>
<td></td>
</tr>
</tbody>
</table>
Example: CDNL-ASP

Consider

\[ P = \left\{ \begin{array}{l}
x \leftarrow \sim y \\
u \leftarrow x, y \\
v \leftarrow x \\
w \leftarrow \sim x, \sim y \\
y \leftarrow \sim x \\
u \leftarrow v \\
v \leftarrow u, y \\
\end{array} \right\} \]

<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>( T_u )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\sim x, \sim y} )</td>
<td>( F_w )</td>
<td>( {T_w, F{\sim x, \sim y}} = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\sim y} )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \{T_x, F\{\sim y\}\} = \delta(x) \]
\[ \{T\{x\}, Fx\} \in \Delta(\{x\}) \]
\[ \{T\{x, y\}, Fx\} \in \Delta(\{x, y\}) \]

\[ \{T_u, F\{x\}, F\{x, y\}\} = \lambda(u, \{u, v\}) \]
Example: CDNL-ASP

Consider

\[ P = \left\{ \begin{array}{l}
x \leftarrow \neg y \\
y \leftarrow \neg x \\
u \leftarrow x, y \\
v \leftarrow x \\
w \leftarrow \neg x, \neg y \\
\end{array} \right\} \]

<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>( T_u )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\neg x, \neg y} )</td>
<td>( F_w )</td>
<td>( { T_w, F{\neg x, \neg y} } = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\neg y} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F_x )</td>
<td>( { T_x, F{\neg y} } = \delta(x) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F{x} )</td>
<td>( { T{x}, F_x } \in \Delta({x}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F{x, y} )</td>
<td>( { T{x, y}, F_x } \in \Delta({x, y}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \vdots )</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( { T_u, F{x}, F{x, y} } = \lambda(u, {u, v}) )</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>
Example: CDNL-ASP

Consider

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

<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>2</td>
<td>( F{\sim x, \sim y} )</td>
<td>( F_w )</td>
<td>( { Tw, F{\sim x, \sim y}} = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\sim y} )</td>
<td>( F_x )</td>
<td>( { Tx, F{\sim y}} = \delta(x) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x} )</td>
<td>( { Tx, F{x}} \in \Delta({x}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F{x, y} )</td>
<td>( { Tx, y}, F_x } \in \Delta({x, y}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \vdots )</td>
<td>( \vdots )</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: CDNL-ASP

Consider

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

<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>( T_u )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\neg x, \neg y} )</td>
<td>( F_w )</td>
<td>( { T_w, F{\neg x, \neg y} } = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\neg y} )</td>
<td>( F_x )</td>
<td>( { T_x, F{\neg 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>( \vdots )</td>
<td>( \vdots )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( { T_u, F{x}, F{x, y} } = \lambda(u, {u, v}) )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: CDNL-ASP

Consider

\[ P = \left\{ \begin{array}{c}
  x \leftarrow \neg y \\
  u \leftarrow x, y \\
  v \leftarrow x \\
  w \leftarrow \neg x, \neg y \\
  y \leftarrow \neg x \\
  u \leftarrow v \\
  v \leftarrow u, y
\end{array} \right\} \]

<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>( T_u )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\neg x, \neg y} )</td>
<td>( F_w )</td>
<td>( { T_w, F{\neg x, \neg y} } = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\neg y} )</td>
<td>( F_x )</td>
<td>( { T_x, 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>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( { T_u, F{x}, F{x, y} } = \lambda(u, {u, v}) )</td>
<td>( \times )</td>
</tr>
</tbody>
</table>
Example: CDNL-ASP

Consider

\[ P = \left\{ \begin{array}{l}
  x \leftarrow \sim y \\
  y \leftarrow \sim x \\
  u \leftarrow x, y \\
  v \leftarrow x \\
  w \leftarrow \sim x, \sim y \\
  u \leftarrow v \\
  v \leftarrow u, y
\end{array} \right\} \]

<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 \n</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{\sim x}} = \delta(y)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( Fw )</td>
<td>{Tw,F{\sim x,\sim y}} = \delta(w)</td>
<td></td>
</tr>
</tbody>
</table>
Consider

\[ P = \left\{ \begin{array}{llll}
  x & \leftarrow & \neg y \\
  y & \leftarrow & \neg x \\
  u & \leftarrow & x, y \\
  v & \leftarrow & x \\
  w & \leftarrow & \neg x, \neg y
\end{array} \right\} \]

<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>( T_u )</td>
<td>( T_x )</td>
<td>( { T_u, F_x } \in \nabla )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \vdots )</td>
<td>( { F_v, T{x} } \in \Delta(v) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T_v )</td>
<td>( { T_y, F{\neg x} } = \delta(y) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F_y )</td>
<td>( { T_w, F{\neg x, \neg y} } = \delta(w) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( F_w )</td>
<td></td>
</tr>
</tbody>
</table>
Example: CDNL-ASP

Consider

\[ P = \begin{cases} x \leftarrow \sim y & u \leftarrow x, y & v \leftarrow x & w \leftarrow \sim x, \sim y \\ y \leftarrow \sim x & u \leftarrow v & v \leftarrow u, y \end{cases} \]

<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>\textbf{T}u</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>\textbf{T}x</td>
<td>{\textbf{T}u, \textbf{F}x} \in \nabla</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\vdots</td>
<td>\vdots</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\textbf{T}v</td>
<td>{\textbf{F}v, \textbf{T}{x}} \in \Delta(v)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\textbf{F}y</td>
<td>{\textbf{T}y, \textbf{F}{\sim x}} = \delta(y)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\textbf{F}w</td>
<td>{\textbf{T}w, \textbf{F}{\sim x, \sim y}} = \delta(w)</td>
<td></td>
</tr>
</tbody>
</table>
Example: CDNL-ASP

Consider

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

<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>( T_u )</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T_x )</td>
<td>{ ( T_u ), ( F_x ) } \in \nabla</td>
<td></td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T_v )</td>
<td>{ ( F_v ), ( T{x} ) } \in \Delta(v)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F_y )</td>
<td>{ ( T_y ), ( F{\neg x} ) } = \delta(y)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F_w )</td>
<td>{ ( T_w ), ( F{\neg x, \neg y} ) } = \delta(w)</td>
<td></td>
</tr>
</tbody>
</table>
Outline

1. Motivation

2. Boolean constraints

3. Nogoods from logic programs
   - Nogoods from program completion
   - Nogoods from loop formulas

4. Conflict-driven nogood learning
   - CDNL-ASP Algorithm
   - Nogood Propagation
   - Conflict Analysis
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 $a \in U$
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 $a \in U$
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 $a \in U$
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 $\text{EB}_P(U) \subseteq A^F$
  - 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

\[ \text{repeat} \]

\[ \text{if } \delta \subseteq A \text{ for some } \delta \in \Delta_P \cup \nabla \text{ then return } (A, \nabla) \]  // conflict

\[ \Sigma := \{ \delta \in \Delta_P \cup \nabla \mid \delta \setminus A = \{ \sigma \}, \sigma \notin A \} \]  // unit-resulting nogoods

\[ \text{if } \Sigma \neq \emptyset \text{ then let } \overline{\sigma} \in \delta \setminus A \text{ for some } \delta \in \Sigma \text{ in} \]

\[ \text{dlevel}(\sigma) := \max(\{\text{dlevel}(\rho) \mid \rho \in \delta \setminus \{\overline{\sigma}\}\}) \cup \{0\} \]

\[ A := A \circ \sigma \]

until \( \Sigma = \emptyset \)

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

\[ U := U \setminus A^F \]

\[ \text{if } U = \emptyset \text{ then } U := \text{UNFOUNDEDSET}(P, A) \]

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

\[ \text{let } a \in U \text{ in} \]

\[ \nabla := \nabla \cup \{Ta\} \cup \{FB \mid B \in EB_P(U)\} \]

// 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 \( (\text{atom}(P) \setminus A^F) \)

- 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
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 $(\text{atom}(P) \setminus A^F)$

- 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

\[
P = \left\{ \begin{array}{llll}
x & \leftarrow & \sim y \\
u & \leftarrow & x, y \\
v & \leftarrow & x \\
w & \leftarrow & \sim x, \sim y \\
y & \leftarrow & \sim x \\
u & \leftarrow & v \\
v & \leftarrow & u, y \\
\end{array} \right\}
\]

<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>2</td>
<td>(F{\sim x, \sim y})</td>
<td>(Fw)</td>
<td>({Tw, F{\sim x, \sim y}} = \delta(w))</td>
</tr>
<tr>
<td>3</td>
<td>(F{\sim y})</td>
<td>(Fx)</td>
<td>({Tx, F{\sim 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{\sim x})</td>
<td>({F{\sim x}, Fx} = \delta({\sim x}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Ty)</td>
<td>({F{\sim y}, Fy} = \delta({\sim 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>(\times)</td>
</tr>
</tbody>
</table>
Outline

1. Motivation

2. Boolean constraints

3. Nogoods from logic programs
   - Nogoods from program completion
   - Nogoods from loop formulas

4. Conflict-driven nogood learning
   - CDNL-ASP Algorithm
   - Nogood Propagation
   - Conflict Analysis
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:

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

- 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 $dl$
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 $dl$
Conflict-driven nogood learning

Outline of ConflictAnalysis

- 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: \textsc{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.

\begin{verbatim}
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 $\varepsilon \setminus A[\sigma] = \{\overline{\sigma}\}$ in
        $\delta := (\delta \setminus \{\sigma\}) \cup (\varepsilon \setminus \{\overline{\sigma}\})$ // resolution
    else return $(\delta, k)$
\end{verbatim}
**Example: ConflictAnalysis**

Consider

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

<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{\sim x, \sim y} )</td>
<td>( Tw, F{\sim x, \sim y} )</td>
<td>( \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\sim y} )</td>
<td>( Tx, F{\sim y} )</td>
<td>( \delta(x) )</td>
</tr>
<tr>
<td></td>
<td>( F{x} )</td>
<td>( {Tx, F{x}} \in \Delta({x}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F{x, y} )</td>
<td>( {Tx, y, Fx} \in \Delta({x, y}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( Tw )</td>
<td>( F{\sim x}, Fx )</td>
<td>( \delta({\sim x}) )</td>
</tr>
<tr>
<td></td>
<td>( Ty )</td>
<td>( F{\sim y}, Fy )</td>
<td>( \delta({\sim 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>( T{v} )</td>
<td>( {Fv, Tu, y} )</td>
<td>( \delta({v}) )</td>
</tr>
<tr>
<td></td>
<td>( T{u, y} )</td>
<td>( {Tu, F{x}, F{y}} )</td>
<td>( \lambda(u, {u, v}) )</td>
</tr>
</tbody>
</table>

Sebastian Rudolph (TUD)  
Answer Set Programming: Solving
Example: ConflictAnalysis

Consider

\[ P = \left\{ \begin{array}{llll}
x & \leftarrow & \sim y \\
y & \leftarrow & \sim x \\
 u & \leftarrow & x, y \\
v & \leftarrow & x \\
w & \leftarrow & \sim x, \sim y \\
u & \leftarrow & v \\
v & \leftarrow & u, y \\
\end{array} \right\} \]

<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>2</td>
<td>( F{\sim x, \sim y} )</td>
<td>( Fw )</td>
<td>( { Tw, F{\sim x, \sim y}} = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\sim y} )</td>
<td>( Fx )</td>
<td>( { Tx, F{\sim 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{\sim x} )</td>
<td>( { F{\sim x}, Fx } = \delta({\sim x}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( Ty )</td>
<td>( { F{\sim y}, Fy } = \delta({\sim 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>( x )</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>( \overline{\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>( Tw, F{\neg x, \neg y} ) = ( \delta(w) )</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>( F{\neg y} )</td>
<td>( Tx, F{\neg y} ) = ( \delta(x) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F{x} )</td>
<td>( Tw, F{x} \in \Delta({x}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F{x, y} )</td>
<td>( Tw, F{x, y} \in \Delta({x, y}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T{\neg x} )</td>
<td>( F{\neg x}, Fx = \delta({\neg x}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( Ty )</td>
<td>( F{\neg y}, Fy = \delta({\neg 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>( Fu, Tu, Ty ) = ( \delta({u, y}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( Tv )</td>
<td>( Fv, Tu, Tu ) \in ( \Delta(v) )</td>
<td></td>
</tr>
</tbody>
</table>

\( \{Tu, Fx\} \cap \{Tu, F\{x\}, F\{x, y\}\} = \lambda(u, \{u, v\}) \)
Example: ConflictAnalysis

Consider

\[ P = \left\{ \begin{array}{l}
  x \leftarrow \sim y \\
  u \leftarrow x, y \\
  v \leftarrow x \\
  w \leftarrow \sim x, \sim y \\
  y \leftarrow \sim x \\
  u \leftarrow v \\
  v \leftarrow u, y
\end{array} \right\} \]

<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>2</td>
<td>(F{\sim x, \sim y})</td>
<td>(Fw)</td>
<td>({Tw, F{\sim x, \sim y}} = \delta(w))</td>
</tr>
<tr>
<td>3</td>
<td>(F{\sim y})</td>
<td>(Fx)</td>
<td>({Tx, F{\sim y}} = \delta(x))</td>
</tr>
<tr>
<td></td>
<td>(F{x})</td>
<td>({Tx, F{x, y}, F{v}} = \delta(v))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(F{x, y})</td>
<td>({Tx, F{x, y}} \in \Delta({x, y}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(T{\sim x})</td>
<td>({F{\sim x}, Fx} = \delta({\sim x}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Ty)</td>
<td>({F{\sim y}, Fy} = \delta({\sim 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, Tu, w} \in \Delta(v))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>({Tu, F{x}, F{x, y}} = \lambda(u, {u, v}))</td>
<td>(\times)</td>
<td></td>
</tr>
</tbody>
</table>
Example: ConflictAnalysis

Consider

\[ P = \left\{ \begin{array}{llll}
  x \leftarrow \sim y & u \leftarrow x, y & v \leftarrow x & w \leftarrow \sim x, \sim y \\
  y \leftarrow \sim x & u \leftarrow v & v \leftarrow u, y
\end{array} \right\} \]

<table>
<thead>
<tr>
<th>dl</th>
<th>( \sigma )</th>
<th>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td>[ F{\sim x, \sim y} ]</td>
<td>( {Tw, F{\sim x, \sim y}} = \delta(w) )</td>
</tr>
<tr>
<td>2</td>
<td>( F{\sim x, \sim y} )</td>
<td>( Fw )</td>
<td>( {T{x}, Fw} \in \Delta({x}) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\sim y} )</td>
<td>( Fx )</td>
<td>( {Tx, F{\sim 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{\sim x} )</td>
<td>( {F{\sim x}, Fx} = \delta({\sim x}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( Ty )</td>
<td>( {F{\sim y}, Fy} = \delta({\sim 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>
</tbody>
</table>

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

Consider

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

<table>
<thead>
<tr>
<th>dl</th>
<th>( \sigma )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\sim x, \sim y} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F_w )</td>
<td>( { Tw, F{\sim x, \sim y}} = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\sim y} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F_x )</td>
<td>( { Tx, F{\sim y}} = \delta(x) )</td>
</tr>
<tr>
<td></td>
<td>( F{x} )</td>
<td>( { Tx, F{x}} \in \Delta({x}) )</td>
</tr>
<tr>
<td></td>
<td>( F{x, y} )</td>
<td>( { Tx, y}, F{x}} \in \Delta({x, y}) )</td>
</tr>
<tr>
<td></td>
<td>( T{\sim x} )</td>
<td>( { F{\sim x}, F{x}} = \delta({\sim x}) )</td>
</tr>
<tr>
<td></td>
<td>( T_y )</td>
<td>( { F{\sim y}, F{y}} = \delta({\sim y}) )</td>
</tr>
<tr>
<td></td>
<td>( T{v} )</td>
<td>( { Tu, F{x, y}, F{v}} = \delta(u) )</td>
</tr>
<tr>
<td></td>
<td>( T{u, y} )</td>
<td>( { F{u, y}, Tu, Ty} = \delta({u, y}) )</td>
</tr>
<tr>
<td></td>
<td>( T{v} )</td>
<td>( { F{v}, Tu, T{u, y}}\in \Delta({v}) )</td>
</tr>
<tr>
<td></td>
<td>( { Tu, F{x}, F{x, y}} = \lambda(u, {u, v}) )</td>
<td>( \times )</td>
</tr>
</tbody>
</table>
Example: ConflictAnalysis

Consider

\[ P = \left\{ \begin{align*}
x & \leftarrow \neg y \\
y & \leftarrow \neg x \\
u & \leftarrow x, y \\
v & \leftarrow x \\
w & \leftarrow \neg x, \neg y \\
\end{align*} \right\} \]

<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>( T_u )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\neg x, \neg y} )</td>
<td>( F_w )</td>
<td>( {T_w, F{\neg x, \neg y}} = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\neg y} )</td>
<td>( F_x )</td>
<td>( {T_x, F{\neg y}} = \delta(x) )</td>
</tr>
<tr>
<td></td>
<td>( F{x} )</td>
<td>( {T{x}, F_x} \in \Delta({x}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F{x, y} )</td>
<td>( {T{x, y}, F_x} \in \Delta({x, y}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T{\neg x} )</td>
<td>( F{\neg x}, F_x = \delta({\neg x}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T_y )</td>
<td>( F{\neg y}, F_y = \delta({\neg y}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T{v} )</td>
<td>( {T_u, F{x, y}, F{v}} = \delta(u) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T{u, y} )</td>
<td>( F{u, y}, T_u, T_y = \delta({u, y}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T_v )</td>
<td>( {F_v, T{u, y}} \in \Delta(v) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( {T_u, F{x}, F{x, y}} = \lambda(u, {u, v}) )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sebastian Rudolph (TUD)  Answer Set Programming: Solving  38 / 39
Example: ConflictAnalysis

Consider

\[ P = \{ \begin{align*}
    x & \leftarrow \sim y \\
    u & \leftarrow x, y \\
    v & \leftarrow x \\
    w & \leftarrow \sim x, \sim y \\
    y & \leftarrow \sim x \\
    u & \leftarrow v \\
    v & \leftarrow u, y
  \end{align*} \} \]

<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>( T_u )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( F{\sim x, \sim y} )</td>
<td>( F_w )</td>
<td>( { T_w, F{\sim x, \sim y} } = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\sim y} )</td>
<td>( F_x )</td>
<td>( { T_x, F{\sim y} } = \delta(x) )</td>
</tr>
<tr>
<td></td>
<td>( F{x} )</td>
<td>( { T{x}, F_x } \in \Delta({x}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( F{x, y} )</td>
<td>( { T{x, y}, F_x } \in \Delta({x, y}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T{\sim x} )</td>
<td>( { F{\sim x}, F_x } = \delta({\sim x}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T_y )</td>
<td>( { F{\sim y}, F_y } = \delta({\sim y}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T{v} )</td>
<td>( { T_u, F{x, y}, F{v} } = \delta(u) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T{u, y} )</td>
<td>( { F{u, y}, T_u, T_y } = \delta({u, y}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T_v )</td>
<td>( { F_v, T{u, y} } \in \Delta(v) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( { T_u, F{x}, F{x, y} } = \lambda(u, {u, v}) )</td>
<td>( \times )</td>
<td></td>
</tr>
</tbody>
</table>
Example: ConflictAnalysis

Consider

\[
P = \begin{cases} x \leftarrow \sim y & u \leftarrow x, y & v \leftarrow x & w \leftarrow \sim x, \sim y \\ y \leftarrow \sim x & u \leftarrow v & v \leftarrow u, y \end{cases}
\]

<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>2</td>
<td>(F{\sim x, \sim y})</td>
<td>(Fw)</td>
<td>({Tw, F{\sim x, \sim y}} = \delta(w))</td>
</tr>
<tr>
<td>3</td>
<td>(F{\sim y})</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Fx)</td>
<td>({Tx, F{\sim y}} = \delta(x))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(F{x})</td>
<td>({Tx, Fx} \in \Delta({x}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(F{x, y})</td>
<td>({Tx, y}, Fx} \in \Delta({x, y}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(T{\sim x})</td>
<td>(F{\sim x}, Fx} = \delta({\sim x}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Ty)</td>
<td>(F{\sim y}, Fy} = \delta({\sim y}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(T{\sim y})</td>
<td>({Tu, F{x, y}, F{v}} = \delta(u))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(T{u, y})</td>
<td>({Fu, y}, Tu, Ty} = \delta({u, y}))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Tv)</td>
<td>({Fv, Tu, {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></td>
<td></td>
</tr>
</tbody>
</table>

Sebastian Rudolph (TUD)  Answer Set Programming: Solving
Consider

\[ P = \left\{ \begin{array}{l}
x \leftarrow \neg y \\
y \leftarrow \neg x \\
u \leftarrow x, y \\
v \leftarrow x \\
w \leftarrow \neg x, \neg y \\
u \leftarrow v \\
v \leftarrow u, y \\
\end{array} \right\} \]

<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>( \delta(w) )</td>
</tr>
<tr>
<td>2</td>
<td>( F{\neg x, \neg y} )</td>
<td>( F_w )</td>
<td>( {Tw, F{\neg x, \neg y}} = \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\neg y} )</td>
<td>( F_x )</td>
<td>( {Tx, F{\neg 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{\neg x} )</td>
<td>( {F{\neg x}, Fx} = \delta({\neg x}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( Ty )</td>
<td>( {F{\neg y}, Fy} = \delta({\neg 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></td>
</tr>
</tbody>
</table>

Example: ConflictAnalysis
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

- Asserting nogoods direct conflict-driven search into a different region of the search space than traversed before, without explicitly flipping any heuristically chosen literal!
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 $k$, $\bar{\sigma}$ 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!
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 $k$, $\bar{\sigma}$ 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!
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
- Asserting nogoods direct conflict-driven search into a different region of the search space than traversed before, without explicitly flipping any heuristically chosen literal!