DEDUCTION SYSTEMS

Answer Set Programming: Solving

Markus Krötzsch
Chair for Knowledge-Based Systems

Slides by Sebastian Rudolph, and based on a lecture by Martin Gebser and Torsten Schaub (CC-By 3.0)

TU Dresden, 2 July 2018
ASP Solving: Overview

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
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 $T_v$ 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

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

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

• The complement, $\overline{\sigma}$, of a literal $\sigma$ is defined as $\overline{Tv} = Fv$ and $\overline{Fv} = Tv$
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 $1 \leq i \leq n$

- $A \circ \sigma$ 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

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

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

• 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 $1 \leq i \leq n$

• 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

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

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

• 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 $A^F = \{v \in \text{dom}(A) \mid F_v \in A\}$
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 $\bar{T_v} = F_v$ and $\bar{F_v} = T_v$
- $A \circ \sigma$ 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

  $$A^T = \{ v \in \text{dom}(A) \mid T_v \in A \} \text{ and } A^F = \{ v \in \text{dom}(A) \mid F_v \in A \}$$
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 \(\delta \not\subseteq A\) for all \(\delta \in \Delta\).
Nogoods, solutions, and unit propagation

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

- An assignment \( A \) such that \( A^T \cup A^F = \text{dom}(A) \) 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. \( \overline{\sigma} \not\in A \).
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
  
  \( 2 \) \( \overline{\sigma} \not\in A \)

- 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

When introducing auxiliary atoms $v_B$ for rule bodies $B$, the completion of a logic program $P$ can be defined as follows:

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

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

where $\text{body}_P(a) = \{\text{body}(r) \mid r \in P \text{ and } \text{head}(r) = a\}$.
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:
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:

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

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

\[ \Delta(B) = \{ \{TB, Fa_1\}, \ldots, \{TB, Fa_m\}, \{TB, Ta_{m+1}\}, \ldots, \{TB, Ta_n\} \} \]
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:

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

gives rise to the nogood

\[ \delta(B) = \{FB, Ta_1, \ldots, Ta_m, Fa_{m+1}, \ldots, Fa_n\} \]
Nogoods from logic programs via program completion

• Analogously, the (atom-oriented) equivalence

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

yields the nogoods

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

2. \( \delta(a) = \{T a, F B_1, \ldots, F B_k\} \)
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 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

$$ES_P(L) = \{ r \in P \mid \text{head}(r) \in L \text{ and } \text{body}(r)^+ \cap L = \emptyset \}$$
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

$$ES_P(L) = \{ r \in P \mid \text{head}(r) \in L \text{ and } \text{body}(r)^+ \cap L = \emptyset \}$$

- 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
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
  \[ ES_P(L) = \{ r \in P \mid \text{head}(r) \in L \text{ and } \text{body}(r)^+ \cap L = \emptyset \} \]

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

  \textbf{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 $EB_P(U) = \{B_1, \ldots, B_k\}$
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 $P$:

$$\Lambda_P = \bigcup_{\emptyset \subset U \subseteq \text{atom}(P)} \{\lambda(a, U) \mid a \in U\}$$
For a logic program $P$ and some $\emptyset \subseteq U \subseteq \text{atom}(P)$, 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 $P$:

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

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

- Consider the program

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

- Consider the program
  \[
  \begin{cases}
  x &\leftarrow \neg y \\
  y &\leftarrow \neg x \\
  u &\leftarrow x \\
  u &\leftarrow v \\
  v &\leftarrow u, y
  \end{cases}
  \]

- For \( u \) in the set \( \{u, v\} \), we obtain the loop nogood:
  \[
  \lambda(u, \{u, v\}) = \{Tu, F\{x\}\} 
  \]
Example

- Consider the program

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

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

- 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:

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

Theorem

Let \( P \) be a logic program. Then,
\[
X \subseteq \text{atom}(P) \text{ is a stable model of } P \text{ iff } \\
X = A^T \cap \text{atom}(P) \text{ for a (unique) solution } A \text{ for } \Delta_P \cup \Lambda_P
\]
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
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
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
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 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
  - 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

- 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”

$A := \emptyset$  
$\n := \emptyset$  
$dl := 0$

\[\text{loop}\]

\[(A, \n) := \text{NogoodPropagation}(P, \n, A)\]

\[\text{if } \varepsilon \subseteq A \text{ for some } \varepsilon \in \Delta_P \cup \n \text{ then} \]

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

\[(\delta, dl) := \text{ConflictAnalysis}(\varepsilon, P, \n, A)\]
$\n := \n \cup \{\delta\}$
$A := A \setminus \{\sigma \in A | dl < \text{dlevel}(\sigma)\}$  

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

\[\text{return } A^T \cap \text{atom}(P) \]

\[\text{else} \]

\[\sigma_d := \text{Select}(P, \n, A)\]
$dl := dl + 1$
$d\text{level}(\sigma_d) := dl$
$A := A \circ \sigma_d$  

\[\text{end loop}\]
Explanations

- 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 $\sigma \in A$, $dlevel(\sigma)$ denotes the decision level of $\sigma$, i.e. the value $dl$ had when $\sigma$ was assigned
Explanations

- 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 $\sigma \in A$, $dlevel(\sigma)$ denotes the decision level of $\sigma$, i.e. 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 $k < dl$. 
Explanations

- 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 $\sigma \in A$, $dlevel(\sigma)$ denotes the decision level of $\sigma$, i.e. 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

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

<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>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: CDNL-ASP

Consider

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

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

Consider

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

<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></td>
<td></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>(\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>( F_w )</td>
<td>( { Tw, F{\sim x, \sim y}} = \delta(w) )</td>
</tr>
</tbody>
</table>
Example: CDNL-ASP

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>( \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>( F_w )</td>
<td>( Tw, F{\neg x, \neg y} ) = ( \delta(w) )</td>
</tr>
<tr>
<td>3</td>
<td>( F{\neg y} )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: CDNL-ASP

Consider

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

<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>( 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></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>
</tbody>
</table>
Example: CDNL-ASP

Consider

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

<table>
<thead>
<tr>
<th>dl</th>
<th>(\sigma_d)</th>
<th>(\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>(F_w)</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>(T{x}, Fx)</td>
<td>(\Delta({x}))</td>
</tr>
<tr>
<td></td>
<td>(F{x, y})</td>
<td>(T{x, y}, Fx)</td>
<td>(\Delta({x, y}))</td>
</tr>
<tr>
<td></td>
<td>:::</td>
<td>:::</td>
<td>:::</td>
</tr>
<tr>
<td></td>
<td>{(Tu, F{x}, F{x, y}}} = \lambda(u, {u, v}) \times</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: CDNL-ASP

Consider

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

\[
\begin{array}{|c|c|c|}
\hline
dl & \sigma_d & \bar{\sigma} \\
\hline
1 & T u & \delta \\
\hline
\end{array}
\]
Example: CDNL-ASP

Consider

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

\[
\begin{array}{|c|c|c|}
\hline
dl & \sigma_d & \vec{\sigma} & \delta \\
\hline
1 & Tu & Tx & \{Tu, Fx\} \in \nabla \\
\hline
\end{array}
\]
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>( \bar{\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>
<tr>
<td></td>
<td></td>
<td>( \vdots )</td>
<td>( \vdots )</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{\neg x} } = \delta(y) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( Fw )</td>
<td>( { Tw, F{\neg x, \neg y} } = \delta(w) )</td>
<td></td>
</tr>
</tbody>
</table>
Example: CDNL-ASP

Consider

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

<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>( Tx )</td>
<td>( {Tu, Fx} \in \nabla )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \vdots )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( Tv )</td>
<td>( {Fv, Tx} \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>
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 $(\lambda(a, U) \setminus \{Ta\}) \subseteq A$
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,
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

\textbf{Input}: A normal program $P$, a set $\nabla$ of nogoods, and an assignment $A$.

\textbf{Output}: An extended assignment and set of nogoods.

$U := \emptyset$  // unfounded set

\textbf{loop}

\textbf{repeat}

\hspace{1em} \textbf{if} $\delta \subseteq A$ \textbf{for some} $\delta \in \Delta_P \cup \nabla$ \textbf{then return} $(A, \nabla)$  // conflict

\hspace{1em} $\Sigma := \{\delta \in \Delta_P \cup \nabla \mid \delta \setminus A = \{\sigma\}, \sigma /\in A\}$  // unit-resulting nogoods

\hspace{2em} \textbf{if} $\Sigma \neq \emptyset$ \textbf{then let} $\sigma \in \delta \setminus A$ \textbf{for some} $\delta \in \Sigma$ \textbf{in}

\hspace{3em} $dlevel(\sigma) := \max\{dlevel(\rho) \mid \rho \in \delta \setminus \{\sigma\}\} \cup \{0\}$

\hspace{3em} $A := A \circ \sigma$

\hspace{1em} \textbf{until} $\Sigma = \emptyset$

\hspace{1em} \textbf{if} $\text{loop}(P) = \emptyset$ \textbf{then return} $(A, \nabla)$

\hspace{1em} $U := U \setminus A^F$

\hspace{2em} \textbf{if} $U = \emptyset$ \textbf{then} $U := \text{UnfoundedSet}(P, A)$

\hspace{1em} \textbf{if} $U = \emptyset$ \textbf{then return} $(A, \nabla)$  // no unfounded set $\emptyset \subset U \subseteq \text{atom}(P) \setminus A^F$

\hspace{2em} \textbf{let} $a \in U$ \textbf{in}

\hspace{3em} $\nabla := \nabla \cup \{Ta\} \cup \{FB \mid B \in 
\text{EB}_P(U)\}$  // 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. \( E_{B_P}(U) \subseteq A^F \)
  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

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

<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{\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></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></td>
<td>( Tw, F{x, y} = \lambda(u, {u, v}) )</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:

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

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

Consider

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

<table>
<thead>
<tr>
<th>dl</th>
<th>( \sigma_d )</th>
<th>( \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} ) = ( \delta(w) )</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>( F{\sim y} )</td>
<td>( Tx, F{\sim 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{\sim x} )</td>
<td>( F{\sim x}, Fx = \delta({\sim x}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( T{y} )</td>
<td>( F{\sim y}, Fy \in \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>
<tr>
<td></td>
<td>( T{u, F{x} } )</td>
<td>( F{x, y} ) = ( \lambda(u, {u, v}) )</td>
<td></td>
</tr>
</tbody>
</table>

TU Dresden, 2 July 2018
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>\text{T}_u</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>\text{F}{\sim x, \sim y}</td>
<td>\text{F}_w</td>
<td>{\text{T}_w, \text{F}{\sim x, \sim y}} = \delta(w)</td>
</tr>
<tr>
<td>3</td>
<td>\text{F}{\sim y}</td>
<td>\text{F}_x</td>
<td>{\text{T}_x, \text{F}{\sim y}} = \delta(x)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\text{F}{x}</td>
<td>{\text{T}{x}, \text{F}_x} \in \Delta({x})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\text{F}{x, y}</td>
<td>{\text{T}{x, y}, \text{F}_x} \in \Delta({x, y})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\text{T}{\sim x}</td>
<td>{\text{F}{\sim x}, \text{F}_x} = \delta({\sim x})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\text{T}_y</td>
<td>{\text{F}{\sim y}, \text{F}_y} = \delta({\sim y})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\text{T}{v}</td>
<td>{\text{T}_u, \text{F}{x, y}, \text{F}{v}} = \delta(u)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\text{T}{u, y}</td>
<td>{\text{F}_v, \text{T}{u, y}} \in \Delta(v)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\text{T}_v</td>
<td>{\text{T}_u, \text{F}{x}, \text{F}{x, y}} = \lambda(u, {u, v})</td>
</tr>
</tbody>
</table>

TU Dresden, 2 July 2018
Example: ConflictAnalysis

Consider

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

<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>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></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>( T_y )</td>
<td>( {F{\sim y}, Fy} = \delta({\sim y}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T{v} )</td>
<td>( {T_u, F{x, y}, F{v}} = \delta(u) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T{u, y} )</td>
<td>( {F{u, y}, T_u, T_y} = \delta({u, y}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( T_v )</td>
<td>( {F_v, T{u, y}} \in \Delta(v) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( {T_u, F{x}, F{x, y}} = \lambda(u, {u, v}) )</td>
</tr>
</tbody>
</table>

TU Dresden, 2 July 2018  Deduction Systems  slide 75 of 86
Example: ConflictAnalysis

Consider

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

<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>( 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>( {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>( T_y )</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>( T_v )</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}) \times )</td>
</tr>
</tbody>
</table>
Example: ConflictAnalysis

Consider

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

<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>$F{\sim x, \sim y}$</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>${Tu, F{x}, F{x, y}} = \lambda(u, {u, v}) \times$</td>
<td></td>
</tr>
</tbody>
</table>
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>( \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>( 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>( { 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{\sim x} )</td>
<td>( F{\sim x}, Fx )</td>
<td>( \delta({\sim x}) )</td>
</tr>
<tr>
<td></td>
<td>( T{y} )</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} } = \delta(u) )</td>
<td></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>( F_v, T{u, y} )</td>
<td>( \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>
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>(d_{\ell})</th>
<th>(\sigma_d)</th>
<th>(\bar{\sigma})</th>
<th>(\delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Tu)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 (F{\sim x, \sim y})</td>
<td>(F_w)</td>
<td>({Tw, F{\sim x, \sim y}} = \delta(w))</td>
<td></td>
</tr>
<tr>
<td>3 (F{\sim y})</td>
<td>(Fx)</td>
<td>({Tx, F{\sim 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{\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>({Tv, 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>(\times)</td>
<td></td>
</tr>
</tbody>
</table>
Example: ConflictAnalysis

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>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( Tu )</td>
<td></td>
<td>( \delta )</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></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>( T_y )</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>( T_v )</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>
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>(\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>(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>(Tv)</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>
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>( \bar{\sigma} )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( T_u )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 2  | \( F\{\sim x, \sim y\} \) | \( F_w \) | \( \{T_w, F\{\sim x, \sim y\}\} = \delta(w) \)
| 3  | \( F\{\sim y\} \) | \( F_{x} \) | \( \{T_{x}, F\{\sim y\}\} = \delta(x) \)
|    | \( F\{x\} \) | \( \{T\{x\}, F_{x}\} \in \Delta(\{x\}) \)
|    | \( F\{x, y\} \) | \( \{T\{x, y\}, F_{x}\} \in \Delta(\{x, y\}) \)
|    | \( T\{\sim x\} \) | \( \{F\{\sim x\}, F_{x}\} = \delta(\{\sim x\}) \)
|    | \( T\{y\} \) | \( \{F\{\sim y\}, F_{y}\} = \delta(\{\sim y\}) \)
|    | \( T\{v\} \) | \( \{T_{u}, F\{x, y\}, F\{v\}\} = \delta(u) \)
|    | \( T\{u, y\} \) | \( \{F\{u, y\}, T_{u}, T_{y}\} = \delta(\{u, y\}) \)
|    | \( T_{v} \) | \( \{F_{v}, T\{u, y\}\} \in \Delta(\nu) \)
|    |                 | \( \{T_{u}, F\{x\}, F\{x, y\}\} = \lambda(u, \{u, v\}) \) | \( \times \)
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 $k = \max(\{dl(\rho) \mid \rho \in \delta \setminus \{\sigma\}\} \cup \{0\}) < dl$
  - 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!