TECHNISCHE UNIVERSITÄT DRESDEN

Artificial Intelligence, Computational Logic

DEDUCTION SYSTEMS

Answer-Set Programming II

*slides adapted from Torsten Schaub [Gebser et al.(2012)]

Sarah Gaggl

Dresden, 22nd May 2013

DRESDEN concept Exzellenz aus Wissenschaft und Kultur

Agenda

Solving procedure for ASP programs

- Partial Interpretations
- Unfounded sets
- Assignments
- Nogoods
- Unit-propagation
- Conflict-driven nogood learning algorithm (CDNL)

Motivation of Conflict-driven ASP Solving

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

Partial interpretations

or: 3-valued interpretations

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

TECHNISCHE UNIVERSITÄT DRESDEN

Partial interpretations

or: 3-valued interpretations

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

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

Partial interpretations

or: 3-valued interpretations

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

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

Partial interpretations

or: 3-valued interpretations

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

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

Unfounded sets

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

Unfounded sets

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

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

Unfounded sets

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

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

Unfounded sets

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

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

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

Unfounded sets

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

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

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

Unfounded sets

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

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

Unfounded sets

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

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

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

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

Unfounded sets

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

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

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

- Rules satisfying Condition 1 are not usable for further derivations

Unfounded sets

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

- A set $U \subseteq atom(P)$ is an unfounded set of P wrt $\langle T, F \rangle$
- if we have for each rule $r \in P$ such that $head(r) \in U$

  either

  1. $body(r)^+ \cap F \neq \emptyset$ or $body(r)^- \cap T \neq \emptyset$ or
  2. $body(r)^+ \cap U \neq \emptyset$
- Rules satisfying Condition 1 are not usable for further derivations
- Condition 2 is the unfounded set condition treating cyclic derivations: All rules still being usable to derive an atom in U require an(other) atom in U to be true

Example

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

TECHNISCHE UNIVERSITÄT DRESDEN

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

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

Assignments

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

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

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

Assignments

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

$$(\sigma_1, \dots, \sigma_n)$$

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

Assignments

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

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

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

Assignments

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

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

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

Assignments

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

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

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

TECHNISCHE UNIVERSITÄT DRESDEN

Assignments

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

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

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

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

TECHNISCHE UNIVERSITÄT DRESDEN

Nogoods, solutions, and unit propagation

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

Nogoods, solutions, and unit propagation

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

Nogoods, solutions, and unit propagation

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

TECHNISCHE UNIVERSITÄT DRESDEN

Nogoods, solutions, and unit propagation

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

Nogoods from logic programs
via program completion

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

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

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

Nogoods from logic programs
via program completion

- The (body-oriented) equivalence

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

can be decomposed into two implications:

Nogoods from logic programs
via program completion

- The (body-oriented) equivalence

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

can be decomposed into two implications:

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

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

   and induces the set of nogoods

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

Nogoods from logic programs
via program completion

- The (body-oriented) equivalence

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

can be decomposed into two implications:

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

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

Nogoods from logic programs
via program completion

- Analogously, the (atom-oriented) equivalence

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

yields the nogoods

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

Nogoods from logic programs
atom-oriented nogoods

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

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

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

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

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

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

TECHNISCHE UNIVERSITÄT DRESDEN

Nogoods from logic programs
atom-oriented nogoods

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

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

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

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

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

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

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

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

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

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

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

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal
- $\boldsymbol{F}x$ is unit-resulting wrt assignment $(\boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\})$ and

Nogoods from logic programs
atom-oriented nogoods

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

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

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

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

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

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal
- $\boldsymbol{F}x$ is unit-resulting wrt assignment $(\boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\})$ and

Nogoods from logic programs
atom-oriented nogoods

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

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

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

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

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

For nogood $\{\boldsymbol{T}x, \boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\}\}$, the signed literal
  - $\boldsymbol{F}x$ is unit-resulting wrt assignment $(\boldsymbol{F}\{y\}, \boldsymbol{F}\{not\ z\})$ and

Nogoods from logic programs
atom-oriented nogoods

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

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

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

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

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

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

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

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

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

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

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

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

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

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

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

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

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

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

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

Nogoods from logic programs
body-oriented nogoods

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

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

TECHNISCHE UNIVERSITÄT DRESDEN

Nogoods from logic programs
body-oriented nogoods

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

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

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

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

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

Nogoods from logic programs
body-oriented nogoods

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

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

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

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

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

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

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

Characterization of stable models

for tight logic programs, ie. free of positive recursion

Let P be a logic program and

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

Characterization of stable models

for tight logic programs, ie. free of positive recursion

Let P be a logic program and

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

Theorem

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

Nogoods from logic programs
via loop formulas

Let P be a normal logic program and recall that:

- For $L \subseteq atom(P)$, the external supports of L for P are

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

Nogoods from logic programs
via loop formulas

Let P be a normal logic program and recall that:

- For $L \subseteq atom(P)$, the external supports of L for P are

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

- The (disjunctive) loop formula of L for P is

$$\begin{aligned} LF_P(L) \quad &= \quad (\textstyle\bigvee_{A \in L} A) \rightarrow (\textstyle\bigvee_{r \in ES_P(L)} body(r)) \\ &\equiv \quad (\textstyle\bigwedge_{r \in ES_P(L)} \neg body(r)) \rightarrow (\textstyle\bigwedge_{A \in L} \neg A) \end{aligned}$$

  – 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 $L \subseteq atom(P)$, the external supports of L for P are

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

- The (disjunctive) loop formula of L for P is

$$\begin{aligned} LF_P(L) \quad &= \quad (\bigvee_{A \in L} A) \rightarrow (\bigvee_{r \in ES_P(L)} body(r)) \\ &\equiv \quad (\bigwedge_{r \in ES_P(L)} \neg body(r)) \rightarrow (\bigwedge_{A \in L} \neg A) \end{aligned}$$

  - 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

$$EB_P(L) \quad = \quad \{body(r) \mid r \in ES_P(L)\}$$

Nogoods from logic programs
loop nogoods

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

$$\lambda(a, U) \quad = \quad \{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\}$$

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

Nogoods from logic programs
loop nogoods

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

$$\lambda(a, U) \quad = \quad \{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\}$$

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

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

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

Nogoods from logic programs
loop nogoods

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

$$\lambda(a, U) \quad = \quad \{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\}$$

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

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

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

- The set Λ_P of loop nogoods denies cyclic support among true atoms

Example

- Consider the program

$$\left\{ \begin{array}{ll} x \leftarrow not\ y & u \leftarrow x \\ & u \leftarrow v \\ y \leftarrow not\ x & v \leftarrow u, y \end{array} \right\}$$

Example

- Consider the program

$$\left\{ \begin{array}{ll} x \leftarrow \mathit{not}\ y & u \leftarrow x \\ y \leftarrow \mathit{not}\ x & u \leftarrow v \\ & v \leftarrow u, y \end{array} \right\}$$

- For u in the set $\{u, v\}$, we obtain the loop nogood:

$$\lambda(u, \{u, v\}) \quad = \quad \{\boldsymbol{T}u, \boldsymbol{F}\{x\}\}$$

Example

- Consider the program

$$\left\{ \begin{array}{ll} x \leftarrow not\ y & u \leftarrow x \\ & u \leftarrow v \\ y \leftarrow not\ x & v \leftarrow u, y \end{array} \right\}$$

- For u in the set $\{u, v\}$, we obtain the loop nogood:

$$\lambda(u, \{u, v\}) \quad = \quad \{\boldsymbol{T}u, \boldsymbol{F}\{x\}\}$$

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

$$\lambda(v, \{u, v\}) \quad = \quad \{\boldsymbol{T}v, \boldsymbol{F}\{x\}\}$$

Characterization of stable models

Theorem

Let P be a logic program. Then,

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

Characterization of stable models

Theorem

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

Some remarks

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

Towards conflict-driven search

Boolean constraint solving algorithms pioneered for SAT led to:

- Traditional DPLL-style approach
  (DPLL stands for 'Davis-Putnam-Logemann-Loveland')
  - (Unit) propagation
  - (Chronological) backtracking
  - in ASP, eg smodels
- Modern CDCL-style approach
  (CDCL stands for 'Conflict-Driven Constraint Learning')
  - (Unit) propagation
  - Conflict analysis (via resolution)
  - Learning + Backjumping + Assertion
  - in ASP, eg clasp

DPLL-style solving

```
loop
    propagate                                    // deterministically assign literals
    if no conflict then
        if all variables assigned then return solution
        else decide                  // non-deterministically assign some literal
    else
        if top-level conflict then return unsatisfiable
        else
            backtrack            // unassign literals made after last decision
            flip                 // assign complement of last decision literal
```

CDCL-style solving

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

Outline of CDNL-ASP algorithm

- Keep track of deterministic consequences by unit propagation on:
  - Program completion [Δ_P]
  - Loop nogoods, determined and recorded on demand [Λ_P]
  - Dynamic nogoods, derived from conflicts and unfounded sets [∇]

Outline of CDNL-ASP algorithm

- Keep track of deterministic consequences by unit propagation on:
  - Program completion [Δ_P]
  - Loop nogoods, determined and recorded on demand [Λ_P]
  - Dynamic nogoods, derived from conflicts and unfounded sets [∇]
- 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 δ
  - Backjump to the earliest (heuristic) choice such that the complement of the UIP is unit-resulting for δ
  - 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 [Δ_P]
  - Loop nogoods, determined and recorded on demand [Λ_P]
  - Dynamic nogoods, derived from conflicts and unfounded sets [∇]
- 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 δ
  - Backjump to the earliest (heuristic) choice such that the complement of the UIP is unit-resulting for δ
  - 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 $atom(P) \cup body(P)$
$\nabla := \emptyset$ // set of recorded nogoods
$dl := 0$ // decision level

loop
- $(A, \nabla) :=$ NogoodPropagation(P, ∇, A)
- **if** $\varepsilon \subseteq A$ **for some** $\varepsilon \in \Delta_P \cup \nabla$ **then** // conflict
  - **if** $\max(\{dlevel(\sigma) \mid \sigma \in \varepsilon\} \cup \{0\}) = 0$ **then return** no stable model
  - $(\delta, dl) :=$ ConflictAnalysis($\varepsilon, P, \nabla, A$)
  - $\nabla := \nabla \cup \{\delta\}$ // (temporarily) record conflict nogood
  - $A := A \setminus \{\sigma \in A \mid dl < dlevel(\sigma)\}$ // backjumping
- **else if** $A^{\boldsymbol{T}} \cup A^{\boldsymbol{F}} = atom(P) \cup body(P)$ **then** // stable model
  - **return** $A^{\boldsymbol{T}} \cap atom(P)$
- **else**
  - $\sigma_d :=$ Select(P, ∇, A) // decision
  - $dl := dl + 1$
  - $dlevel(\sigma_d) := dl$
  - $A := A \circ \sigma_d$

TECHNISCHE UNIVERSITÄT DRESDEN

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 $\sigma_d = \boldsymbol{T}a$ or $\sigma_d = \boldsymbol{F}a$, respectively, we require $a \in (atom(P) \cup body(P)) \setminus (A^{\boldsymbol{T}} \cup A^{\boldsymbol{F}})$
- For any literal $\sigma \in A$, $dl(\sigma)$ denotes the decision level of σ, viz. the value dl had when σ was assigned

Observations

- Decision level dl, initially set to 0, is used to count the number of heuristically chosen literals in assignment A
- For a heuristically chosen literal $\sigma_d = \boldsymbol{T}a$ or $\sigma_d = \boldsymbol{F}a$, respectively, we require $a \in (atom(P) \cup body(P)) \setminus (A^{\boldsymbol{T}} \cup A^{\boldsymbol{F}})$
- For any literal $\sigma \in A$, $dl(\sigma)$ denotes the decision level of σ, viz. the value dl had when σ 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 δ derived by conflict analysis is asserting, that is, some literal is unit-resulting for δ at a decision level $k < dl$

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 $\sigma_d = \boldsymbol{T}a$ or $\sigma_d = \boldsymbol{F}a$, respectively, we require $a \in (atom(P) \cup body(P)) \setminus (A^{\boldsymbol{T}} \cup A^{\boldsymbol{F}})$
- For any literal $\sigma \in A$, $dl(\sigma)$ denotes the decision level of σ, viz. the value dl had when σ 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 δ derived by conflict analysis is asserting, that is, some literal is unit-resulting for δ at a decision level $k < dl$
  - After learning δ 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

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d $\overline{\sigma}$	δ

Example: CDNL-ASP

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		

TECHNISCHE UNIVERSITÄT DRESDEN

Example: CDNL-ASP

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
2	$\boldsymbol{F}\{not\ x, not\ y\}$		

Example: CDNL-ASP

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
2	$\boldsymbol{F}\{not\ x, not\ y\}$		
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$

Example: CDNL-ASP

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
2	$\boldsymbol{F}\{not\ x, not\ y\}$		
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$
3	$\boldsymbol{F}\{not\ y\}$		

Example: CDNL-ASP

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
2	$\boldsymbol{F}\{not\ x, not\ y\}$		
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$
3	$\boldsymbol{F}\{not\ y\}$		
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{not\ y\}\} = \delta(x)$
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$

Example: CDNL-ASP

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
2	$\boldsymbol{F}\{not\ x, not\ y\}$		
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$
3	$\boldsymbol{F}\{not\ y\}$		
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{not\ y\}\} = \delta(x)$
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$
		$\vdots$	$\vdots$
			$\{\boldsymbol{T}u, \boldsymbol{F}\{x\}, \boldsymbol{F}\{x, y\}\} = \lambda(u, \{u, v\})$ ✘

Example: CDNL-ASP

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		

Example: CDNL-ASP

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
		$\boldsymbol{T}x$	$\{\boldsymbol{T}u, \boldsymbol{F}x\} \in \nabla$

Example: CDNL-ASP

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
		$\boldsymbol{T}x$	$\{\boldsymbol{T}u, \boldsymbol{F}x\} \in \nabla$
		$\vdots$	$\vdots$
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{x\}\} \in \Delta(v)$
		$\boldsymbol{F}y$	$\{\boldsymbol{T}y, \boldsymbol{F}\{not\ x\}\} = \delta(y)$
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$

Example: CDNL-ASP

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
		$\boldsymbol{T}x$	$\{\boldsymbol{T}u, \boldsymbol{F}x\} \in \nabla$
		$\vdots$	$\vdots$
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{x\}\} \in \Delta(v)$
		$\boldsymbol{F}y$	$\{\boldsymbol{T}y, \boldsymbol{F}\{not\ x\}\} = \delta(y)$
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$

Outline of NogoodPropagation

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

Outline of NogoodPropagation

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

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

- Wrt a fixpoint of unit propagation,

Outline of NogoodPropagation

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

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

- 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 Δ_P and ∇;
  - Unfounded sets $U \subseteq atom(P)$
- Note that U is unfounded if $EB_P(U) \subseteq A^{\boldsymbol{F}}$
  - Note: For any $a \in U$, we have $(\lambda(a, U) \setminus \{\boldsymbol{T}a\}) \subseteq A$
- An "interesting" unfounded set U satisfies:

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

- 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 ∇ 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 ∇ of nogoods, and an assignment A.
Output : An extended assignment and set of nogoods.

```
U := ∅                                                        // unfounded set
loop
    repeat
        if δ ⊆ A for some δ ∈ Δ_P ∪ ∇ then return (A, ∇)                  // conflict
        Σ := {δ ∈ Δ_P ∪ ∇ | δ \ A = {σ̄}, σ ∉ A}              // unit-resulting nogoods
        if Σ ≠ ∅ then let σ̄ ∈ δ \ A for some δ ∈ Σ in
            dlevel(σ) := max({dlevel(ρ) | ρ ∈ δ \ {σ̄}} ∪ {0})
            A := A ∘ σ
    until Σ = ∅
    if loop(P) = ∅ then return (A, ∇)
    U := U \ A^F
    if U = ∅ then U := UnfoundedSet(P, A)
    if U = ∅ then return (A, ∇)           // no unfounded set ∅ ⊂ U ⊆ atom(P) \ A^F
    let a ∈ U in
        ∇ := ∇ ∪ {{Ta} ∪ {FB | B ∈ EB_P(U)}}                    // record loop nogood
```

Requirements for UnfoundedSet

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

Requirements for UnfoundedSet

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

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
2	$\boldsymbol{F}\{not\ x, not\ y\}$		
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$
3	$\boldsymbol{F}\{not\ y\}$		
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{not\ y\}\} = \delta(x)$
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$
		$\boldsymbol{T}\{not\ x\}$	$\{\boldsymbol{F}\{not\ x\}, \boldsymbol{F}x\} = \delta(\{not\ x\})$
		$\boldsymbol{T}y$	$\{\boldsymbol{F}\{not\ y\}, \boldsymbol{F}y\} = \delta(\{not\ y\})$
		$\boldsymbol{T}\{v\}$	$\{\boldsymbol{T}u, \boldsymbol{F}\{x, y\}, \boldsymbol{F}\{v\}\} = \delta(u)$
		$\boldsymbol{T}\{u, y\}$	$\{\boldsymbol{F}\{u, y\}, \boldsymbol{T}u, \boldsymbol{T}y\} = \delta(\{u, y\})$
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{u, y\}\} \in \Delta(v)$
			$\{\boldsymbol{T}u, \boldsymbol{F}\{x\}, \boldsymbol{F}\{x, y\}\} = \lambda(u, \{u, v\})$ ✘

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 ε, we obtain a new violated nogood by resolving δ and ε as follows:

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

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 ε, we obtain a new violated nogood by resolving δ and ε 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 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 ε, we obtain a new violated nogood by resolving δ and ε 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 δ containing exactly one literal σ assigned at decision level dl
  - This literal σ 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 δ, a normal program P, a set ∇ of nogoods, and an assignment A.

Output : A derived nogood and a decision level.

```
loop
    let σ ∈ δ such that δ \ A[σ] = {σ} in
        k := max({dlevel(ρ) | ρ ∈ δ \ {σ}} ∪ {0})
        if k = dlevel(σ) then
            let ε ∈ Δ_P ∪ ∇ such that ε \ A[σ] = {σ̄} in
                δ := (δ \ {σ}) ∪ (ε \ {σ̄})                    // resolution
        else return (δ, k)
```

Example: ConflictAnalysis

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ	
1	$\boldsymbol{T}u$			
2	$\boldsymbol{F}\{not\ x, not\ y\}$			
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$	
3	$\boldsymbol{F}\{not\ y\}$			
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{not\ y\}\} = \delta(x)$	
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$	
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$	
		$\boldsymbol{T}\{not\ x\}$	$\{\boldsymbol{F}\{not\ x\}, \boldsymbol{F}x\} = \delta(\{not\ x\})$	
		$\boldsymbol{T}y$	$\{\boldsymbol{F}\{not\ y\}, \boldsymbol{F}y\} = \delta(\{not\ y\})$	
		$\boldsymbol{T}\{v\}$	$\{\boldsymbol{T}u, \boldsymbol{F}\{x, y\}, \boldsymbol{F}\{v\}\} = \delta(u)$	
		$\boldsymbol{T}\{u, y\}$	$\{\boldsymbol{F}\{u, y\}, \boldsymbol{T}u, \boldsymbol{T}y\} = \delta(\{u, y\})$	
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{u, y\}\} \in \Delta(v)$	
			$\{\boldsymbol{T}u, \boldsymbol{F}\{x\}, \boldsymbol{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
2	$\boldsymbol{F}\{not\ x, not\ y\}$		
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$
3	$\boldsymbol{F}\{not\ y\}$		
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{not\ y\}\} = \delta(x)$
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$
		$\boldsymbol{T}\{not\ x\}$	$\{\boldsymbol{F}\{not\ x\}, \boldsymbol{F}x\} = \delta(\{not\ x\})$
		$\boldsymbol{T}y$	$\{\boldsymbol{F}\{not\ y\}, \boldsymbol{F}y\} = \delta(\{not\ y\})$
		$\boldsymbol{T}\{v\}$	$\{\boldsymbol{T}u, \boldsymbol{F}\{x, y\}, \boldsymbol{F}\{v\}\} = \delta(u)$
		$\boldsymbol{T}\{u, y\}$	$\{\boldsymbol{F}\{u, y\}, \boldsymbol{T}u, \boldsymbol{T}y\} = \delta(\{u, y\})$
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{u, y\}\} \in \Delta(v)$
			$\{\boldsymbol{T}u, \boldsymbol{F}\{x\}, \boldsymbol{F}\{x, y\}\} = \lambda(u, \{u, v\})$ ✘

Example: ConflictAnalysis

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow \textit{not } y & u \leftarrow x, y & v \leftarrow x & w \leftarrow \textit{not } x, \textit{not } y \\ y \leftarrow \textit{not } x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
2	$\boldsymbol{F}\{\textit{not } x, \textit{not } y\}$		
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{\textit{not } x, \textit{not } y\}\} = \delta(w)$
3	$\boldsymbol{F}\{\textit{not } y\}$		
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{\textit{not } y\}\} = \delta(x)$
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$
		$\boldsymbol{T}\{\textit{not } x\}$	$\{\boldsymbol{F}\{\textit{not } x\}, \boldsymbol{F}x\} = \delta(\{\textit{not } x\})$
		$\boldsymbol{T}y$	$\{\boldsymbol{F}\{\textit{not } y\}, \boldsymbol{F}y\} = \delta(\{\textit{not } y\})$
		$\boldsymbol{T}\{v\}$	$\{\boldsymbol{T}u, \boldsymbol{F}\{x, y\}, \boldsymbol{F}\{v\}\} = \delta(u)$
		$\boldsymbol{T}\{u, y\}$	$\{\boldsymbol{F}\{u, y\}, \boldsymbol{T}u, \boldsymbol{T}y\} = \delta(\{u, y\})$
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{u, y\}\} \in \Delta(v)$
			$\{\boldsymbol{T}u, \boldsymbol{F}\{x\}, \boldsymbol{F}\{x, y\}\} = \lambda(u, \{u, v\})$ ✘

Example: ConflictAnalysis

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
2	$\boldsymbol{F}\{not\ x, not\ y\}$		
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$
3	$\boldsymbol{F}\{not\ y\}$		
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{not\ y\}\} = \delta(x)$
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$
		$\boldsymbol{T}\{not\ x\}$	$\{\boldsymbol{F}\{not\ x\}, \boldsymbol{F}x\} = \delta(\{not\ x\})$
		$\boldsymbol{T}y$	$\{\boldsymbol{F}\{not\ y\}, \boldsymbol{F}y\} = \delta(\{not\ y\})$
		$\boldsymbol{T}\{v\}$	$\{\boldsymbol{T}u, \boldsymbol{F}\{x, y\}, \boldsymbol{F}\{v\}\} = \delta(u)$
		$\boldsymbol{T}\{u, y\}$	$\{\boldsymbol{F}\{u, y\}, \boldsymbol{T}u, \boldsymbol{T}y\} = \delta(\{u, y\})$
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{u, y\}\} \in \Delta(v)$
			$\{\boldsymbol{T}u, \boldsymbol{F}\{x\}, \boldsymbol{F}\{x, y\}\} = \lambda(u, \{u, v\})$ ✘

Example: ConflictAnalysis

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ	
1	$\boldsymbol{T}u$			
2	$\boldsymbol{F}\{not\ x, not\ y\}$			
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$	
3	$\boldsymbol{F}\{not\ y\}$			
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{not\ y\}\} = \delta(x)$	
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$	
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$	$\{\boldsymbol{T}u, \boldsymbol{F}x, \boldsymbol{F}\{x\}\}$
		$\boldsymbol{T}\{not\ x\}$	$\{\boldsymbol{F}\{not\ x\}, \boldsymbol{F}x\} = \delta(\{not\ x\})$	
		$\boldsymbol{T}y$	$\{\boldsymbol{F}\{not\ y\}, \boldsymbol{F}y\} = \delta(\{not\ y\})$	
		$\boldsymbol{T}\{v\}$	$\{\boldsymbol{T}u, \boldsymbol{F}\{x, y\}, \boldsymbol{F}\{v\}\} = \delta(u)$	
		$\boldsymbol{T}\{u, y\}$	$\{\boldsymbol{F}\{u, y\}, \boldsymbol{T}u, \boldsymbol{T}y\} = \delta(\{u, y\})$	
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{u, y\}\} \in \Delta(v)$	
			$\{\boldsymbol{T}u, \boldsymbol{F}\{x\}, \boldsymbol{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ	
1	$\boldsymbol{T}u$			
2	$\boldsymbol{F}\{not\ x, not\ y\}$			
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$	
3	$\boldsymbol{F}\{not\ y\}$			
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{not\ y\}\} = \delta(x)$	
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$	
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$	$\{\boldsymbol{T}u, \boldsymbol{F}x, \boldsymbol{F}\{x\}\}$
		$\boldsymbol{T}\{not\ x\}$	$\{\boldsymbol{F}\{not\ x\}, \boldsymbol{F}x\} = \delta(\{not\ x\})$	
		$\boldsymbol{T}y$	$\{\boldsymbol{F}\{not\ y\}, \boldsymbol{F}y\} = \delta(\{not\ y\})$	
		$\boldsymbol{T}\{v\}$	$\{\boldsymbol{T}u, \boldsymbol{F}\{x, y\}, \boldsymbol{F}\{v\}\} = \delta(u)$	
		$\boldsymbol{T}\{u, y\}$	$\{\boldsymbol{F}\{u, y\}, \boldsymbol{T}u, \boldsymbol{T}y\} = \delta(\{u, y\})$	
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{u, y\}\} \in \Delta(v)$	
			$\{\boldsymbol{T}u, \boldsymbol{F}\{x\}, \boldsymbol{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ	
1	$\boldsymbol{T}u$			
2	$\boldsymbol{F}\{not\ x, not\ y\}$			
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$	
3	$\boldsymbol{F}\{not\ y\}$			
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{not\ y\}\} = \delta(x)$	
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$	
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$	$\{\boldsymbol{T}u, \boldsymbol{F}x, \boldsymbol{F}\{x\}\}$
		$\boldsymbol{T}\{not\ x\}$	$\{\boldsymbol{F}\{not\ x\}, \boldsymbol{F}x\} = \delta(\{not\ x\})$	
		$\boldsymbol{T}y$	$\{\boldsymbol{F}\{not\ y\}, \boldsymbol{F}y\} = \delta(\{not\ y\})$	
		$\boldsymbol{T}\{v\}$	$\{\boldsymbol{T}u, \boldsymbol{F}\{x, y\}, \boldsymbol{F}\{v\}\} = \delta(u)$	
		$\boldsymbol{T}\{u, y\}$	$\{\boldsymbol{F}\{u, y\}, \boldsymbol{T}u, \boldsymbol{T}y\} = \delta(\{u, y\})$	
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{u, y\}\} \in \Delta(v)$	
			$\{\boldsymbol{T}u, \boldsymbol{F}\{x\}, \boldsymbol{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ	
1	$\boldsymbol{T}u$			
2	$\boldsymbol{F}\{not\ x, not\ y\}$			
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$	
3	$\boldsymbol{F}\{not\ y\}$			
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{not\ y\}\} = \delta(x)$	
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$	$\{\boldsymbol{T}u, \boldsymbol{F}x\}$
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$	$\{\boldsymbol{T}u, \boldsymbol{F}x, \boldsymbol{F}\{x\}\}$
		$\boldsymbol{T}\{not\ x\}$	$\{\boldsymbol{F}\{not\ x\}, \boldsymbol{F}x\} = \delta(\{not\ x\})$	
		$\boldsymbol{T}y$	$\{\boldsymbol{F}\{not\ y\}, \boldsymbol{F}y\} = \delta(\{not\ y\})$	
		$\boldsymbol{T}\{v\}$	$\{\boldsymbol{T}u, \boldsymbol{F}\{x, y\}, \boldsymbol{F}\{v\}\} = \delta(u)$	
		$\boldsymbol{T}\{u, y\}$	$\{\boldsymbol{F}\{u, y\}, \boldsymbol{T}u, \boldsymbol{T}y\} = \delta(\{u, y\})$	
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{u, y\}\} \in \Delta(v)$	
			$\{\boldsymbol{T}u, \boldsymbol{F}\{x\}, \boldsymbol{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ	
1	$\boldsymbol{T}u$			
2	$\boldsymbol{F}\{not\ x, not\ y\}$			
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$	
3	$\boldsymbol{F}\{not\ y\}$			
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{not\ y\}\} = \delta(x)$	
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$	$\{\boldsymbol{T}u, \boldsymbol{F}x\}$
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$	$\{\boldsymbol{T}u, \boldsymbol{F}x, \boldsymbol{F}\{x\}\}$
		$\boldsymbol{T}\{not\ x\}$	$\{\boldsymbol{F}\{not\ x\}, \boldsymbol{F}x\} = \delta(\{not\ x\})$	
		$\boldsymbol{T}y$	$\{\boldsymbol{F}\{not\ y\}, \boldsymbol{F}y\} = \delta(\{not\ y\})$	
		$\boldsymbol{T}\{v\}$	$\{\boldsymbol{T}u, \boldsymbol{F}\{x, y\}, \boldsymbol{F}\{v\}\} = \delta(u)$	
		$\boldsymbol{T}\{u, y\}$	$\{\boldsymbol{F}\{u, y\}, \boldsymbol{T}u, \boldsymbol{T}y\} = \delta(\{u, y\})$	
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{u, y\}\} \in \Delta(v)$	
			$\{\boldsymbol{T}u, \boldsymbol{F}\{x\}, \boldsymbol{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ	
1	$\boldsymbol{T}u$			
2	$\boldsymbol{F}\{not\ x, not\ y\}$			
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$	
3	$\boldsymbol{F}\{not\ y\}$			
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{not\ y\}\} = \delta(x)$	
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$	$\{\boldsymbol{T}u, \boldsymbol{F}x\}$
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$	$\{\boldsymbol{T}u, \boldsymbol{F}x, \boldsymbol{F}\{x\}\}$
		$\boldsymbol{T}\{not\ x\}$	$\{\boldsymbol{F}\{not\ x\}, \boldsymbol{F}x\} = \delta(\{not\ x\})$	
		$\boldsymbol{T}y$	$\{\boldsymbol{F}\{not\ y\}, \boldsymbol{F}y\} = \delta(\{not\ y\})$	
		$\boldsymbol{T}\{v\}$	$\{\boldsymbol{T}u, \boldsymbol{F}\{x, y\}, \boldsymbol{F}\{v\}\} = \delta(u)$	
		$\boldsymbol{T}\{u, y\}$	$\{\boldsymbol{F}\{u, y\}, \boldsymbol{T}u, \boldsymbol{T}y\} = \delta(\{u, y\})$	
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{u, y\}\} \in \Delta(v)$	
			$\{\boldsymbol{T}u, \boldsymbol{F}\{x\}, \boldsymbol{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis ctd.

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
		$\boldsymbol{T}x$	$\{\boldsymbol{T}u, \boldsymbol{F}x\} \in \nabla$

Example: ConflictAnalysis ctd.

Consider

$$P = \left\{ \begin{array}{llll} x \leftarrow not\ y & u \leftarrow x, y & v \leftarrow x & w \leftarrow not\ x, not\ y \\ y \leftarrow not\ x & u \leftarrow v & v \leftarrow u, y & \end{array} \right\}$$

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
		$\boldsymbol{T}x$	$\{\boldsymbol{T}u, \boldsymbol{F}x\} \in \nabla$
		$\vdots$	$\vdots$
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{x\}\} \in \Delta(v)$
		$\boldsymbol{F}y$	$\{\boldsymbol{T}y, \boldsymbol{F}\{not\ x\}\} = \delta(y)$
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{not\ x, not\ y\}\} = \delta(w)$

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 δ containing First-UIP σ 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 δ containing First-UIP σ 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 δ in ∇ and backjumping to decision level k, $\overline{\sigma}$ is unit-resulting for δ !
  - Such a nogood δ 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 δ containing First-UIP σ 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 δ in ∇ and backjumping to decision level k, $\overline{\sigma}$ is unit-resulting for δ !
  - Such a nogood δ 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 !

TECHNISCHE UNIVERSITÄT DRESDEN

References

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

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