Answer Set Programming: Solving

Sebastian Rudolph

Computational Logic Group
Technische Universität Dresden

Slides based on a lecture by Martin Gebser and Torsten Schaub.

Outline

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Nogoods, solutions, and unit propagation

- A nogood is a set $\{\sigma_1, \dots, \sigma_n\}$ of signed literals, expressing a constraint violated by any assignment containing $\sigma_1, \dots, \sigma_n$
- An assignment A such that $A^{\mathbf{T}} \cup A^{\mathbf{F}} = dom(A)$ and $A^{\mathbf{T}} \cap A^{\mathbf{F}} = \emptyset$ 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, 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 $A^{\mathbf{T}} \cup A^{\mathbf{F}} = dom(A)$ and $A^{\mathbf{T}} \cap A^{\mathbf{F}} = \emptyset$ 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, solutions, and unit propagation

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

Outline

Outline

1. Motivation
2. Boolean constraints
3. Nogoods from logic programs
   - Nogoods from program completion
   - Nogoods from loop formulas
4. Conflict-driven nogood learning
   - CDNL-ASP Algorithm
   - Nogood Propagation
   - Conflict Analysis

Nogoods from logic programs
via program completion

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

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

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

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

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

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

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

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

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

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

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

$$\boxed{\begin{array}{lll} x & \leftarrow & y \\ x & \leftarrow & \sim z \end{array}} \qquad \begin{array}{l} \{\mathbf{T}x, \mathbf{F}\{y\}, \mathbf{F}\{\sim z\}\} \\ \{\,\{\mathbf{F}x, \mathbf{T}\{y\}\}, \{\mathbf{F}x, \mathbf{T}\{\sim z\}\}\,\} \end{array}$$

For nogood $\{\mathbf{T}x, \mathbf{F}\{y\}, \mathbf{F}\{\sim z\}\}$, the signed literal
- $\mathbf{F}x$ is unit-resulting wrt assignment $(\mathbf{F}\{y\}, \mathbf{F}\{\sim z\})$ and
- $\mathbf{T}\{\sim z\}$ is unit-resulting wrt assignment $(\mathbf{T}x, \mathbf{F}\{y\})$

Nogoods from logic programs
atom-oriented nogoods

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

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

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

$$\boxed{\begin{array}{lcl} x & \leftarrow & y \\ x & \leftarrow & \sim z \end{array}} \qquad \begin{array}{l} \{\mathbf{T}x, \mathbf{F}\{y\}, \mathbf{F}\{\sim z\}\} \\ \{\,\{\mathbf{F}x, \mathbf{T}\{y\}\}, \{\mathbf{F}x, \mathbf{T}\{\sim z\}\}\,\} \end{array}$$

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

$$\boxed{\begin{array}{lll} x & \leftarrow & y \\ x & \leftarrow & \sim z \end{array}} \qquad \begin{array}{l} \{\mathbf{T}x, \mathbf{F}\{y\}, \mathbf{F}\{\sim z\}\} \\ \{\,\{\mathbf{F}x, \mathbf{T}\{y\}\}, \{\mathbf{F}x, \mathbf{T}\{\sim z\}\}\,\} \end{array}$$

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

$$\boxed{\begin{array}{lll} x & \leftarrow & y \\ x & \leftarrow & \sim z \end{array}} \qquad \begin{array}{l} \{\mathbf{T}x, \mathbf{F}\{y\}, \mathbf{F}\{\sim z\}\} \\ \{\,\{\mathbf{F}x, \mathbf{T}\{y\}\}, \{\mathbf{F}x, \mathbf{T}\{\sim z\}\}\,\} \end{array}$$

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

$$\boxed{\begin{array}{lcl} x & \leftarrow & y \\ x & \leftarrow & \sim z \end{array}} \qquad \begin{array}{l} \{\mathbf{T}x, \mathbf{F}\{y\}, \mathbf{F}\{\sim z\}\} \\ \{\,\{\mathbf{F}x, \mathbf{T}\{y\}\}, \{\mathbf{F}x, \mathbf{T}\{\sim z\}\}\,\} \end{array}$$

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

$$\boxed{\begin{array}{lcl} x & \leftarrow & y \\ x & \leftarrow & \sim z \end{array}} \qquad \begin{array}{l} \{\mathbf{T}x, \mathbf{F}\{y\}, \mathbf{F}\{\sim z\}\} \\ \{\, \{\mathbf{F}x, \mathbf{T}\{y\}\}, \{\mathbf{F}x, \mathbf{T}\{\sim z\}\} \,\} \end{array}$$

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

$$\boxed{\begin{array}{lll} x & \leftarrow & y \\ x & \leftarrow & \sim z \end{array}}$$

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

For nogood $\{\mathbf{T}x, \mathbf{F}\{y\}, \mathbf{F}\{\sim z\}\}$, the signed literal
- $\mathbf{F}x$ is unit-resulting wrt assignment $(\mathbf{F}\{y\}, \mathbf{F}\{\sim z\})$ and
- $\mathbf{T}\{\sim z\}$ is unit-resulting wrt assignment $(\mathbf{T}x, \mathbf{F}\{y\})$

Nogoods from logic programs
atom-oriented nogoods

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

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

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

$$\boxed{\begin{array}{lll} x & \leftarrow & y \\ x & \leftarrow & \sim z \end{array}} \qquad \begin{array}{l} \{\mathbf{T}x, \mathbf{F}\{y\}, \mathbf{F}\{\sim z\}\} \\ \{\,\{\mathbf{F}x, \mathbf{T}\{y\}\}, \{\mathbf{F}x, \mathbf{T}\{\sim z\}\}\,\} \end{array}$$

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

$$\boxed{\begin{array}{lll} x & \leftarrow & y \\ x & \leftarrow & \sim z \end{array}} \qquad \begin{array}{l} \{\mathbf{T}x, \mathbf{F}\{y\}, \mathbf{F}\{\sim z\}\} \\ \{\{\mathbf{F}x, \mathbf{T}\{y\}\}, \{\mathbf{F}x, \mathbf{T}\{\sim z\}\}\} \end{array}$$

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

$$\boxed{\begin{array}{lll} x & \leftarrow & y \\ x & \leftarrow & \sim z \end{array}}$$

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

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

$$\boxed{\begin{array}{lcl} x & \leftarrow & y \\ x & \leftarrow & \sim z \end{array}} \qquad \begin{array}{l} \{\mathbf{T}x, \mathbf{F}\{y\}, \mathbf{F}\{\sim z\}\} \\ \{\,\{\mathbf{F}x, \mathbf{T}\{y\}\}, \{\mathbf{F}x, \mathbf{T}\{\sim z\}\}\,\} \end{array}$$

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

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

Nogoods from logic programs
atom-oriented nogoods

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

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

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

$$\boxed{\begin{array}{lll} x & \leftarrow & y \\ x & \leftarrow & \sim z \end{array}} \qquad \begin{array}{l} \{\mathbf{T}x, \mathbf{F}\{y\}, \mathbf{F}\{\sim z\}\} \\ \{\,\{\mathbf{F}x, \mathbf{T}\{y\}\}, \{\mathbf{F}x, \mathbf{T}\{\sim z\}\}\,\} \end{array}$$

  For nogood $\{\mathbf{T}x, \mathbf{F}\{y\}, \mathbf{F}\{\sim z\}\}$, the signed literal
  - $\mathbf{F}x$ is unit-resulting wrt assignment $(\mathbf{F}\{y\}, \mathbf{F}\{\sim z\})$ and
  - $\mathbf{T}\{\sim z\}$ is unit-resulting wrt assignment $(\mathbf{T}x, \mathbf{F}\{y\})$

Nogoods from logic programs
atom-oriented nogoods

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

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

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

$$\boxed{\begin{array}{lll} x & \leftarrow & y \\ x & \leftarrow & \sim z \end{array}}$$

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

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

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

Nogoods from logic programs
body-oriented nogoods

- For a body $B = \{a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n\}$, we get

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

- Example Given Body $\{x, \sim y\}$, we obtain

$$\boxed{\begin{array}{r} \dots \leftarrow x, \sim y \\ \vdots \\ \dots \leftarrow x, \sim y \end{array}} \qquad \begin{array}{l} \{\mathbf{F}\{x, \sim y\}, \mathbf{T}x, \mathbf{F}y\} \\ \{\,\{\mathbf{T}\{x, \sim y\}, \mathbf{F}x\}, \{\mathbf{T}\{x, \sim y\}, \mathbf{T}y\}\,\} \end{array}$$

For nogood $\delta(\{x, \sim y\}) = \{\mathbf{F}\{x, \sim y\}, \mathbf{T}x, \mathbf{F}y\}$, the signed literal
  - $\mathbf{T}\{x, \sim y\}$ is unit-resulting wrt assignment $(\mathbf{T}x, \mathbf{F}y)$ and
  - $\mathbf{T}y$ is unit-resulting wrt assignment $(\mathbf{F}\{x, \sim y\}, \mathbf{T}x)$

Nogoods from logic programs
body-oriented nogoods

- For a body $B = \{a_1, \dots, a_m, {\sim}a_{m+1}, \dots, {\sim}a_n\}$, we get

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

- Example Given Body $\{x, {\sim}y\}$, we obtain

$$\boxed{\begin{array}{c} \dots \leftarrow x, {\sim}y \\ \vdots \\ \dots \leftarrow x, {\sim}y \end{array}}$$

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

For nogood $\delta(\{x, {\sim}y\}) = \{\mathbf{F}\{x, {\sim}y\}, \mathbf{T}x, \mathbf{F}y\}$, the signed literal
- $\mathbf{T}\{x, {\sim}y\}$ is unit-resulting wrt assignment $(\mathbf{T}x, \mathbf{F}y)$ and
- $\mathbf{T}y$ is unit-resulting wrt assignment $(\mathbf{F}\{x, {\sim}y\}, \mathbf{T}x)$

Nogoods from logic programs
body-oriented nogoods

- For a body $B = \{a_1, \dots, a_m, {\sim}a_{m+1}, \dots, {\sim}a_n\}$, we get

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

- Example Given Body $\{x, {\sim}y\}$, we obtain

$$\boxed{\begin{array}{l} \dots \leftarrow x, {\sim}y \\ \qquad\quad \vdots \\ \dots \leftarrow x, {\sim}y \end{array}}$$

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

For nogood $\delta(\{x, {\sim}y\}) = \{\mathbf{F}\{x, {\sim}y\}, \mathbf{T}x, \mathbf{F}y\}$, the signed literal

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

Characterization of stable models

for tight logic programs

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*

Characterization of stable models

for tight logic programs

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

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

Outline

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

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

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

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

  - 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

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

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

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

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

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

  - 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

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

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

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

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

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

  - 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
$$\lambda(a, U) \quad = \quad \{\mathbf{T}a, \mathbf{F}B_1, \dots, \mathbf{F}B_k\}$$
where $EB_P(U) = \{B_1, \dots, B_k\}$

- 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

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
$$\lambda(a, U) = \{\mathbf{T}a, \mathbf{F}B_1, \ldots, \mathbf{F}B_k\}$$
where $EB_P(U) = \{B_1, \ldots, B_k\}$
- We get the following set of loop nogoods for P:
$$\Lambda_P = \bigcup_{\emptyset \subset U \subseteq atom(P)} \{\lambda(a, U) \mid a \in U\}$$
- The set Λ_P of loop nogoods denies cyclic support among *true* atoms

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

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

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

- 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

$$\left\{ \begin{array}{ll} x \leftarrow {\sim}y & u \leftarrow x \\ & u \leftarrow v \\ y \leftarrow {\sim}x & v \leftarrow u, y \end{array} \right\}$$

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

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

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

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

Example

- Consider the program

$$\left\{\begin{array}{ll} x \leftarrow \sim y & u \leftarrow x \\ & u \leftarrow v \\ y \leftarrow \sim x & v \leftarrow u, y \end{array}\right\}$$

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

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

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

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

Example

- Consider the program

$$\left\{ \begin{array}{ll} x \leftarrow \sim y & u \leftarrow x \\ & u \leftarrow v \\ y \leftarrow \sim x & v \leftarrow u, y \end{array} \right\}$$

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

$$\lambda(u, \{u, v\}) \;=\; \{\mathbf{T}u, \mathbf{F}\{x\}\}$$

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

$$\lambda(v, \{u, v\}) \;=\; \{\mathbf{T}v, \mathbf{F}\{x\}\}$$

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* $\Delta_P \cup \Lambda_P$

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

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* $\Delta_P \cup \Lambda_P$

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

Outline

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

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 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

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

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
    $(A, \nabla) := \textsc{NogoodPropagation}(P, \nabla, A)$
    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
        $(\delta, dl) := \textsc{ConflictAnalysis}(\varepsilon, P, \nabla, A)$
        $\nabla := \nabla \cup \{\delta\}$ // *(temporarily) record conflict nogood*
        $A := A \setminus \{\sigma \in A \mid dl < dlevel(\sigma)\}$ // *backjumping*
    else if $A^{\mathbf{T}} \cup A^{\mathbf{F}} = atom(P) \cup body(P)$ **then** // *stable model*
        return $A^{\mathbf{T}} \cap atom(P)$
    else
        $\sigma_d := \textsc{Select}(P, \nabla, A)$ // *decision*
        $dl := dl + 1$
        $dlevel(\sigma_d) := dl$
        $A := A \circ \sigma_d$

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 $\sigma_d = \mathbf{T}a$ or $\sigma_d = \mathbf{F}a$, respectively, we require $a \in (atom(P) \cup body(P)) \setminus (A^{\mathbf{T}} \cup A^{\mathbf{F}})$
- 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 !

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 $\sigma_d = \mathbf{T}a$ or $\sigma_d = \mathbf{F}a$, respectively, we require $a \in (atom(P) \cup body(P)) \setminus (A^{\mathbf{T}} \cup A^{\mathbf{F}})$
- 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 !

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 $\sigma_d = \mathbf{T}a$ or $\sigma_d = \mathbf{F}a$, respectively, we require $a \in (atom(P) \cup body(P)) \setminus (A^{\mathbf{T}} \cup A^{\mathbf{F}})$
- 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

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

dl	σ_d	$\overline{\sigma}$	δ
1	$\mathbf{T}u$		
2	$\mathbf{F}\{\sim x, \sim y\}$		
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$
3	$\mathbf{F}\{\sim y\}$		
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{\sim y\}\} = \delta(x)$
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$
		$\vdots$	$\vdots$
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$ ✗

Example: CDNL-ASP

Consider

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

dl	σ_d	$\overline{\sigma}$	δ
1	$\mathbf{T}u$		
2	$\mathbf{F}\{\sim x, \sim y\}$		
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$
3	$\mathbf{F}\{\sim y\}$		
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{\sim y\}\} = \delta(x)$
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$
		$\vdots$	$\vdots$
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$ ✘

Example: CDNL-ASP

Consider

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

dl	σ_d	$\overline{\sigma}$	δ
1	$\mathbf{T}u$		
2	$\mathbf{F}\{\sim x, \sim y\}$		
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$
3	$\mathbf{F}\{\sim y\}$		
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{\sim y\}\} = \delta(x)$
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$
		$\vdots$	$\vdots$
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$ ✘

Example: CDNL-ASP

Consider

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

dl	σ_d	$\overline{\sigma}$	δ
1	$\mathbf{T}u$		
2	$\mathbf{F}\{\sim x, \sim y\}$		
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$
3	$\mathbf{F}\{\sim y\}$		
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{\sim y\}\} = \delta(x)$
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$
		$\vdots$	$\vdots$
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$ ✗

Example: CDNL-ASP

Consider

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

dl	σ_d	$\overline{\sigma}$	δ
1	$\mathbf{T}u$		
2	$\mathbf{F}\{{\sim}x, {\sim}y\}$		
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{{\sim}x, {\sim}y\}\} = \delta(w)$
3	$\mathbf{F}\{{\sim}y\}$		
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{{\sim}y\}\} = \delta(x)$
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$
		$\vdots$	$\vdots$
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$ ✘

Example: CDNL-ASP

Consider

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

dl	σ_d	$\overline{\sigma}$	δ
1	$\mathbf{T}u$		
2	$\mathbf{F}\{{\sim}x, {\sim}y\}$		
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{{\sim}x, {\sim}y\}\} = \delta(w)$
3	$\mathbf{F}\{{\sim}y\}$		
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{{\sim}y\}\} = \delta(x)$
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$
		$\vdots$	$\vdots$
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$ ✘

Example: CDNL-ASP

Consider

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

dl	σ_d	$\overline{\sigma}$	δ
1	$\mathbf{T}u$		
2	$\mathbf{F}\{\sim x, \sim y\}$		
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$
3	$\mathbf{F}\{\sim y\}$		
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{\sim y\}\} = \delta(x)$
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$
		$\vdots$	$\vdots$
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$ ✘

Example: CDNL-ASP

Consider

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

dl	σ_d	$\overline{\sigma}$	δ
1	$\mathbf{T}u$		
		$\mathbf{T}x$	$\{\mathbf{T}u, \mathbf{F}x\} \in \nabla$
		$\vdots$	$\vdots$
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{x\}\} \in \Delta(v)$
		$\mathbf{F}y$	$\{\mathbf{T}y, \mathbf{F}\{\sim x\}\} = \delta(y)$
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$

Example: CDNL-ASP

Consider

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

dl	σ_d	$\overline{\sigma}$	δ
1	$\mathbf{T}u$		
		$\mathbf{T}x$	$\{\mathbf{T}u, \mathbf{F}x\} \in \nabla$
		$\vdots$	$\vdots$
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{x\}\} \in \Delta(v)$
		$\mathbf{F}y$	$\{\mathbf{T}y, \mathbf{F}\{\sim x\}\} = \delta(y)$
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$

Example: CDNL-ASP

Consider

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

dl	σ_d	$\overline{\sigma}$	δ
1	$\mathbf{T}u$		
		$\mathbf{T}x$	$\{\mathbf{T}u, \mathbf{F}x\} \in \nabla$
		$\vdots$	$\vdots$
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{x\}\} \in \Delta(v)$
		$\mathbf{F}y$	$\{\mathbf{T}y, \mathbf{F}\{{\sim}x\}\} = \delta(y)$
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{{\sim}x, {\sim}y\}\} = \delta(w)$

Example: CDNL-ASP

Consider

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

dl	σ_d	$\overline{\sigma}$	δ
1	$\mathbf{T}u$		
		$\mathbf{T}x$	$\{\mathbf{T}u, \mathbf{F}x\} \in \nabla$
		$\vdots$	$\vdots$
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{x\}\} \in \Delta(v)$
		$\mathbf{F}y$	$\{\mathbf{T}y, \mathbf{F}\{\sim x\}\} = \delta(y)$
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$

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

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

- 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 $a \in U$

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 $EB_P(U) \subseteq A^{\mathbf{F}}$
  - Note For any $a \in U$, we have $(\lambda(a, U) \setminus \{\mathbf{T}a\}) \subseteq A$
- An "interesting" unfounded set U satisfies:

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

- 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 $a \in U$

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 $EB_P(U) \subseteq A^{\mathbf{F}}$
  - Note For any $a \in U$, we have $(\lambda(a, U) \setminus \{\mathbf{T}a\}) \subseteq A$
- An "interesting" unfounded set U satisfies:

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

- 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 $a \in U$

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 $EB_P(U) \subseteq A^{\mathbf{F}}$
  - Note For any $a \in U$, we have $(\lambda(a, U) \setminus \{\mathbf{T}a\}) \subseteq A$
- An "interesting" unfounded set U satisfies:

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

- 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.

$U := \emptyset$ // *unfounded set*

loop
  repeat
    if $\delta \subseteq A$ **for some** $\delta \in \Delta_P \cup \nabla$ **then return** (A, ∇) // *conflict*
    $\Sigma := \{\delta \in \Delta_P \cup \nabla \mid \delta \setminus A = \{\overline{\sigma}\}, \sigma \notin A\}$ // *unit-resulting nogoods*
    if $\Sigma \neq \emptyset$ **then let** $\overline{\sigma} \in \delta \setminus A$ **for some** $\delta \in \Sigma$ **in**
      $dlevel(\sigma) := \max(\{dlevel(\rho) \mid \rho \in \delta \setminus \{\overline{\sigma}\}\} \cup \{0\})$
      $A := A \circ \sigma$
  until $\Sigma = \emptyset$
  if $loop(P) = \emptyset$ **then return** (A, ∇)
  $U := U \setminus A^{\mathbf{F}}$
  if $U = \emptyset$ **then** $U := \textsc{UnfoundedSet}(P, A)$
  if $U = \emptyset$ **then return** (A, ∇) // *no unfounded set* $\emptyset \subset U \subseteq atom(P) \setminus A^{\mathbf{F}}$
  let $a \in U$ **in**
    $\nabla := \nabla \cup \{\{\mathbf{T}a\} \cup \{\mathbf{F}B \mid B \in EB_P(U)\}\}$ // *record loop nogood*

Requirements for UNFOUNDEDSET

- Implementations of UNFOUNDEDSET must guarantee the following for a result U
  1. $U \subseteq (atom(P) \setminus A^{\mathbf{F}})$
  2. $EB_P(U) \subseteq A^{\mathbf{F}}$
  3. $U = \emptyset$ iff there is no nonempty unfounded subset of $(atom(P) \setminus A^{\mathbf{F}})$
- Beyond that, there are various alternatives, such as:
  - Calculating the greatest unfounded set
  - Calculating unfounded sets within strongly connected components of the positive atom dependency graph of P
  - Usually, the latter option is implemented in ASP solvers

Requirements for UnfoundedSet

- Implementations of UnfoundedSet must guarantee the following for a result U
  1. $U \subseteq (atom(P) \setminus A^{\mathbf{F}})$
  2. $EB_P(U) \subseteq A^{\mathbf{F}}$
  3. $U = \emptyset$ iff there is no nonempty unfounded subset of $(atom(P) \setminus A^{\mathbf{F}})$
- Beyond that, there are various alternatives, such as:
  - Calculating the greatest unfounded set
  - Calculating unfounded sets within strongly connected components of the positive atom dependency graph of P

  - Usually, the latter option is implemented in ASP solvers

Example: NogoodPropagation

Consider

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

dl	σ_d	$\overline{\sigma}$	δ
1	$\mathbf{T}u$		
2	$\mathbf{F}\{\sim x, \sim y\}$		
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$
3	$\mathbf{F}\{\sim y\}$		
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{\sim y\}\} = \delta(x)$
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$
		$\mathbf{T}\{\sim x\}$	$\{\mathbf{F}\{\sim x\}, \mathbf{F}x\} = \delta(\{\sim x\})$
		$\mathbf{T}y$	$\{\mathbf{F}\{\sim y\}, \mathbf{F}y\} = \delta(\{\sim y\})$
		$\mathbf{T}\{v\}$	$\{\mathbf{T}u, \mathbf{F}\{x, y\}, \mathbf{F}\{v\}\} = \delta(u)$
		$\mathbf{T}\{u, y\}$	$\{\mathbf{F}\{u, y\}, \mathbf{T}u, \mathbf{T}y\} = \delta(\{u, y\})$
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{u, y\}\} \in \Delta(v)$
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$ ✘

Outline

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

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

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

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

dl	σ_d	$\overline{\sigma}$	δ	
1	$\mathbf{T}u$			
2	$\mathbf{F}\{{\sim}x, {\sim}y\}$			
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{{\sim}x, {\sim}y\}\} = \delta(w)$	
3	$\mathbf{F}\{{\sim}y\}$			
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{{\sim}y\}\} = \delta(x)$	
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$	$\{\mathbf{T}u, \mathbf{F}x\}$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$	$\{\mathbf{T}u, \mathbf{F}x, \mathbf{F}\{x\}\}$
		$\mathbf{T}\{{\sim}x\}$	$\{\mathbf{F}\{{\sim}x\}, \mathbf{F}x\} = \delta(\{{\sim}x\})$	
		$\mathbf{T}y$	$\{\mathbf{F}\{{\sim}y\}, \mathbf{F}y\} = \delta(\{{\sim}y\})$	
		$\mathbf{T}\{v\}$	$\{\mathbf{T}u, \mathbf{F}\{x, y\}, \mathbf{F}\{v\}\} = \delta(u)$	
		$\mathbf{T}\{u, y\}$	$\{\mathbf{F}\{u, y\}, \mathbf{T}u, \mathbf{T}y\} = \delta(\{u, y\})$	
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{u, y\}\} \in \Delta(v)$	
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

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

dl	σ_d	$\overline{\sigma}$	δ	
1	$\mathbf{T}u$			
2	$\mathbf{F}\{\sim x, \sim y\}$			
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$	
3	$\mathbf{F}\{\sim y\}$			
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{\sim y\}\} = \delta(x)$	
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$	$\{\mathbf{T}u, \mathbf{F}x\}$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$	$\{\mathbf{T}u, \mathbf{F}x, \mathbf{F}\{x\}\}$
		$\mathbf{T}\{\sim x\}$	$\{\mathbf{F}\{\sim x\}, \mathbf{F}x\} = \delta(\{\sim x\})$	
		$\mathbf{T}y$	$\{\mathbf{F}\{\sim y\}, \mathbf{F}y\} = \delta(\{\sim y\})$	
		$\mathbf{T}\{v\}$	$\{\mathbf{T}u, \mathbf{F}\{x, y\}, \mathbf{F}\{v\}\} = \delta(u)$	
		$\mathbf{T}\{u, y\}$	$\{\mathbf{F}\{u, y\}, \mathbf{T}u, \mathbf{T}y\} = \delta(\{u, y\})$	
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{u, y\}\} \in \Delta(v)$	
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

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

dl	σ_d	$\overline{\sigma}$	δ	
1	$\mathbf{T}u$			
2	$\mathbf{F}\{\sim x, \sim y\}$			
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$	
3	$\mathbf{F}\{\sim y\}$			
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{\sim y\}\} = \delta(x)$	
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$	$\{\mathbf{T}u, \mathbf{F}x\}$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$	$\{\mathbf{T}u, \mathbf{F}x, \mathbf{F}\{x\}\}$
		$\mathbf{T}\{\sim x\}$	$\{\mathbf{F}\{\sim x\}, \mathbf{F}x\} = \delta(\{\sim x\})$	
		$\mathbf{T}y$	$\{\mathbf{F}\{\sim y\}, \mathbf{F}y\} = \delta(\{\sim y\})$	
		$\mathbf{T}\{v\}$	$\{\mathbf{T}u, \mathbf{F}\{x, y\}, \mathbf{F}\{v\}\} = \delta(u)$	
		$\mathbf{T}\{u, y\}$	$\{\mathbf{F}\{u, y\}, \mathbf{T}u, \mathbf{T}y\} = \delta(\{u, y\})$	
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{u, y\}\} \in \Delta(v)$	
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

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

dl	σ_d	$\overline{\sigma}$	δ	
1	$\mathbf{T}u$			
2	$\mathbf{F}\{\sim x, \sim y\}$			
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$	
3	$\mathbf{F}\{\sim y\}$			
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{\sim y\}\} = \delta(x)$	
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$	$\{\mathbf{T}u, \mathbf{F}x\}$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$	$\{\mathbf{T}u, \mathbf{F}x, \mathbf{F}\{x\}\}$
		$\mathbf{T}\{\sim x\}$	$\{\mathbf{F}\{\sim x\}, \mathbf{F}x\} = \delta(\{\sim x\})$	
		$\mathbf{T}y$	$\{\mathbf{F}\{\sim y\}, \mathbf{F}y\} = \delta(\{\sim y\})$	
		$\mathbf{T}\{v\}$	$\{\mathbf{T}u, \mathbf{F}\{x, y\}, \mathbf{F}\{v\}\} = \delta(u)$	
		$\mathbf{T}\{u, y\}$	$\{\mathbf{F}\{u, y\}, \mathbf{T}u, \mathbf{T}y\} = \delta(\{u, y\})$	
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{u, y\}\} \in \Delta(v)$	
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

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

dl	σ_d	$\overline{\sigma}$	δ	
1	$\mathbf{T}u$			
2	$\mathbf{F}\{\sim x, \sim y\}$			
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$	
3	$\mathbf{F}\{\sim y\}$			
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{\sim y\}\} = \delta(x)$	
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$	$\{\mathbf{T}u, \mathbf{F}x\}$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$	$\{\mathbf{T}u, \mathbf{F}x, \mathbf{F}\{x\}\}$
		$\mathbf{T}\{\sim x\}$	$\{\mathbf{F}\{\sim x\}, \mathbf{F}x\} = \delta(\{\sim x\})$	
		$\mathbf{T}y$	$\{\mathbf{F}\{\sim y\}, \mathbf{F}y\} = \delta(\{\sim y\})$	
		$\mathbf{T}\{v\}$	$\{\mathbf{T}u, \mathbf{F}\{x, y\}, \mathbf{F}\{v\}\} = \delta(u)$	
		$\mathbf{T}\{u, y\}$	$\{\mathbf{F}\{u, y\}, \mathbf{T}u, \mathbf{T}y\} = \delta(\{u, y\})$	
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{u, y\}\} \in \Delta(v)$	
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

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

dl	σ_d	$\overline{\sigma}$	δ	
1	$\mathbf{T}u$			
2	$\mathbf{F}\{\sim x, \sim y\}$			
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$	
3	$\mathbf{F}\{\sim y\}$			
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{\sim y\}\} = \delta(x)$	
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$	$\{\mathbf{T}u, \mathbf{F}x\}$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$	$\{\mathbf{T}u, \mathbf{F}x, \mathbf{F}\{x\}\}$
		$\mathbf{T}\{\sim x\}$	$\{\mathbf{F}\{\sim x\}, \mathbf{F}x\} = \delta(\{\sim x\})$	
		$\mathbf{T}y$	$\{\mathbf{F}\{\sim y\}, \mathbf{F}y\} = \delta(\{\sim y\})$	
		$\mathbf{T}\{v\}$	$\{\mathbf{T}u, \mathbf{F}\{x, y\}, \mathbf{F}\{v\}\} = \delta(u)$	
		$\mathbf{T}\{u, y\}$	$\{\mathbf{F}\{u, y\}, \mathbf{T}u, \mathbf{T}y\} = \delta(\{u, y\})$	
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{u, y\}\} \in \Delta(v)$	
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

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

dl	σ_d	$\overline{\sigma}$	δ	
1	$\mathbf{T}u$			
2	$\mathbf{F}\{{\sim}x, {\sim}y\}$			
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{{\sim}x, {\sim}y\}\} = \delta(w)$	
3	$\mathbf{F}\{{\sim}y\}$			
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{{\sim}y\}\} = \delta(x)$	
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$	$\{\mathbf{T}u, \mathbf{F}x\}$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$	$\{\mathbf{T}u, \mathbf{F}x, \mathbf{F}\{x\}\}$
		$\mathbf{T}\{{\sim}x\}$	$\{\mathbf{F}\{{\sim}x\}, \mathbf{F}x\} = \delta(\{{\sim}x\})$	
		$\mathbf{T}y$	$\{\mathbf{F}\{{\sim}y\}, \mathbf{F}y\} = \delta(\{{\sim}y\})$	
		$\mathbf{T}\{v\}$	$\{\mathbf{T}u, \mathbf{F}\{x, y\}, \mathbf{F}\{v\}\} = \delta(u)$	
		$\mathbf{T}\{u, y\}$	$\{\mathbf{F}\{u, y\}, \mathbf{T}u, \mathbf{T}y\} = \delta(\{u, y\})$	
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{u, y\}\} \in \Delta(v)$	
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

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

dl	σ_d	$\overline{\sigma}$	δ	
1	$\mathbf{T}u$			
2	$\mathbf{F}\{\sim x, \sim y\}$			
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{\sim x, \sim y\}\} = \delta(w)$	
3	$\mathbf{F}\{\sim y\}$			
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{\sim y\}\} = \delta(x)$	
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$	$\{\mathbf{T}u, \mathbf{F}x\}$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$	$\{\mathbf{T}u, \mathbf{F}x, \mathbf{F}\{x\}\}$
		$\mathbf{T}\{\sim x\}$	$\{\mathbf{F}\{\sim x\}, \mathbf{F}x\} = \delta(\{\sim x\})$	
		$\mathbf{T}y$	$\{\mathbf{F}\{\sim y\}, \mathbf{F}y\} = \delta(\{\sim y\})$	
		$\mathbf{T}\{v\}$	$\{\mathbf{T}u, \mathbf{F}\{x, y\}, \mathbf{F}\{v\}\} = \delta(u)$	
		$\mathbf{T}\{u, y\}$	$\{\mathbf{F}\{u, y\}, \mathbf{T}u, \mathbf{T}y\} = \delta(\{u, y\})$	
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{u, y\}\} \in \Delta(v)$	
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

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

dl	σ_d	$\overline{\sigma}$	δ	
1	$\mathbf{T}u$			
2	$\mathbf{F}\{{\sim}x, {\sim}y\}$			
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{{\sim}x, {\sim}y\}\} = \delta(w)$	
3	$\mathbf{F}\{{\sim}y\}$			
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{{\sim}y\}\} = \delta(x)$	
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$	$\{\mathbf{T}u, \mathbf{F}x\}$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$	$\{\mathbf{T}u, \mathbf{F}x, \mathbf{F}\{x\}\}$
		$\mathbf{T}\{{\sim}x\}$	$\{\mathbf{F}\{{\sim}x\}, \mathbf{F}x\} = \delta(\{{\sim}x\})$	
		$\mathbf{T}y$	$\{\mathbf{F}\{{\sim}y\}, \mathbf{F}y\} = \delta(\{{\sim}y\})$	
		$\mathbf{T}\{v\}$	$\{\mathbf{T}u, \mathbf{F}\{x, y\}, \mathbf{F}\{v\}\} = \delta(u)$	
		$\mathbf{T}\{u, y\}$	$\{\mathbf{F}\{u, y\}, \mathbf{T}u, \mathbf{T}y\} = \delta(\{u, y\})$	
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{u, y\}\} \in \Delta(v)$	
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

Example: ConflictAnalysis

Consider

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

dl	σ_d	$\overline{\sigma}$	δ	
1	$\mathbf{T}u$			
2	$\mathbf{F}\{{\sim}x, {\sim}y\}$			
		$\mathbf{F}w$	$\{\mathbf{T}w, \mathbf{F}\{{\sim}x, {\sim}y\}\} = \delta(w)$	
3	$\mathbf{F}\{{\sim}y\}$			
		$\mathbf{F}x$	$\{\mathbf{T}x, \mathbf{F}\{{\sim}y\}\} = \delta(x)$	
		$\mathbf{F}\{x\}$	$\{\mathbf{T}\{x\}, \mathbf{F}x\} \in \Delta(\{x\})$	$\{\mathbf{T}u, \mathbf{F}x\}$
		$\mathbf{F}\{x, y\}$	$\{\mathbf{T}\{x, y\}, \mathbf{F}x\} \in \Delta(\{x, y\})$	$\{\mathbf{T}u, \mathbf{F}x, \mathbf{F}\{x\}\}$
		$\mathbf{T}\{{\sim}x\}$	$\{\mathbf{F}\{{\sim}x\}, \mathbf{F}x\} = \delta(\{{\sim}x\})$	
		$\mathbf{T}y$	$\{\mathbf{F}\{{\sim}y\}, \mathbf{F}y\} = \delta(\{{\sim}y\})$	
		$\mathbf{T}\{v\}$	$\{\mathbf{T}u, \mathbf{F}\{x, y\}, \mathbf{F}\{v\}\} = \delta(u)$	
		$\mathbf{T}\{u, y\}$	$\{\mathbf{F}\{u, y\}, \mathbf{T}u, \mathbf{T}y\} = \delta(\{u, y\})$	
		$\mathbf{T}v$	$\{\mathbf{F}v, \mathbf{T}\{u, y\}\} \in \Delta(v)$	
			$\{\mathbf{T}u, \mathbf{F}\{x\}, \mathbf{F}\{x, y\}\} = \lambda(u, \{u, v\})$	✘

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 !

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 !

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 !

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 !