TECHNISCHE UNIVERSITÄT DRESDEN

DEDUCTION SYSTEMS

Answer Set Programming: Solving

Markus Krötzsch

Chair for Knowledge-Based Systems

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

TU Dresden, 2 July 2018

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

ASP Solving: Overview

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

Outline

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

TECHNISCHE UNIVERSITÄT DRESDEN

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 $\boldsymbol{T}v$ 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

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

  of signed literals σ_i of form $\boldsymbol{T}v$ or $\boldsymbol{F}v$ for $v \in dom(A)$ and $1 \leq i \leq n$

- 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 $1 \leq i \leq n$

- $A \circ \sigma$ 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 $1 \leq i \leq n$

- 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 $1 \leq i \leq n$

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

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 $1 \leq i \leq n$

- We sometimes identify an assignment with the set of its literals
- Given this, we access true and false propositions in A via

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

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

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

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) $\overline{\sigma} \notin A$

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

Outline

Outline

Nogoods from logic programs
via program completion

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

$$
\begin{aligned}
& \{v_B \leftrightarrow a_1 \wedge \cdots \wedge a_m \wedge \neg a_{m+1} \wedge \cdots \wedge \neg a_n \mid \\
& \qquad B \in \mathit{body}(P) \text{ and } B = \{a_1, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n\}\} \\
\cup \quad & \{a \leftrightarrow v_{B_1} \vee \cdots \vee v_{B_k} \mid \\
& \qquad a \in \mathit{atom}(P) \text{ and } \mathit{body}_P(a) = \{B_1, \ldots, B_k\}\}\ ,
\end{aligned}
$$

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

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

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) $\delta(a) = \{\boldsymbol{T}a, \boldsymbol{F}B_1, \ldots, \boldsymbol{F}B_k\}$

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

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

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

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

- 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

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

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

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

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

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

Example

- Consider the program

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

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

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

Example

- Consider the program

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

- 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
$X = A^{\boldsymbol{T}} \cap atom(P)$ 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

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
```

TECHNISCHE UNIVERSITÄT DRESDEN

Outline

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 := ∅                                              // assignment over atom(P) ∪ body(P)
∇ := ∅                                              // set of recorded nogoods
dl := 0                                             // decision level

loop
    (A, ∇) := NogoodPropagation(P, ∇, A)
    if ε ⊆ A for some ε ∈ Δ_P ∪ ∇ then              // conflict
        if max({dlevel(σ) | σ ∈ ε} ∪ {0}) = 0 then return no stable model

        (δ, dl) := ConflictAnalysis(ε, P, ∇, A)
        ∇ := ∇ ∪ {δ}                                // (temporarily) record conflict nogood
        A := A \ {σ ∈ A | dl < dlevel(σ)}           // backjumping
    else if A^T ∪ A^F = atom(P) ∪ body(P) then      // stable model
        return A^T ∩ atom(P)
    else
        σ_d := Select(P, ∇, A)                      // decision
        dl := dl + 1
        dlevel(σ_d) := dl
        A := A ∘ σ_d
```

Explanations

- Decision level dl, initially set to 0, is used to count the number of heuristically chosen literals in assignment A
- For a heuristically chosen literal $\sigma_d = \boldsymbol{T}a$ 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 $\sigma \in A$, $dlevel(\sigma)$ denotes the decision level of σ, i.e. the value dl had when σ was assigned

Explanations

- Decision level dl, initially set to 0, is used to count the number of heuristically chosen literals in assignment A
- For a heuristically chosen literal $\sigma_d = \boldsymbol{T}a$ 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 $\sigma \in A$, $dlevel(\sigma)$ denotes the decision level of σ, i.e. 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 $k < dl$

Explanations

- Decision level dl, initially set to 0, is used to count the number of heuristically chosen literals in assignment A
- For a heuristically chosen literal $\sigma_d = \boldsymbol{T}a$ 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 $\sigma \in A$, $dlevel(\sigma)$ denotes the decision level of σ, i.e. 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

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	$\boldsymbol{T}u$		

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	$\boldsymbol{T}u$	
2	$\boldsymbol{F}\{{\sim}x, {\sim}y\}$	

Example: CDNL-ASP

Consider

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

dl	σ_d	$\overline{\sigma}$	δ
1	$\boldsymbol{T}u$		
2	$\boldsymbol{F}\{{\sim}x, {\sim}y\}$		
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{{\sim}x, {\sim}y\}\} = \delta(w)$

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	$\boldsymbol{T}u$		
2	$\boldsymbol{F}\{{\sim}x, {\sim}y\}$		
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{{\sim}x, {\sim}y\}\} = \delta(w)$
3	$\boldsymbol{F}\{{\sim}y\}$		

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	$\boldsymbol{T}u$		
2	$\boldsymbol{F}\{\sim x, \sim y\}$		
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{\sim x, \sim y\}\} = \delta(w)$
3	$\boldsymbol{F}\{\sim y\}$		
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{\sim y\}\} = \delta(x)$
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$

Example: CDNL-ASP

Consider

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

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

TECHNISCHE UNIVERSITÄT DRESDEN

Example: CDNL-ASP

Consider

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

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

TECHNISCHE UNIVERSITÄT DRESDEN

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	$\boldsymbol{T}u$		
		$\boldsymbol{T}x$	$\{\boldsymbol{T}u, \boldsymbol{F}x\} \in \nabla$

Example: CDNL-ASP

Consider

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

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

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	$\boldsymbol{T}u$		
		$\boldsymbol{T}x$	$\{\boldsymbol{T}u, \boldsymbol{F}x\} \in \nabla$
		$\vdots$	$\vdots$
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{x\}\} \in \Delta(v)$
		$\boldsymbol{F}y$	$\{\boldsymbol{T}y, \boldsymbol{F}\{\sim x\}\} = \delta(y)$
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{\sim x, \sim y\}\} = \delta(w)$

Outline

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.

$U := \emptyset$ // 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, ∇)

  $U := U \setminus A^{\boldsymbol{F}}$
  if $U = \emptyset$ **then** $U :=$ UnfoundedSet(P, A)

  if $U = \emptyset$ **then return** (A, ∇) // no unfounded set $\emptyset \subset U \subseteq atom(P) \setminus A^{\boldsymbol{F}}$

  let $a \in U$ **in**
    $\nabla := \nabla \cup \{\{\boldsymbol{T}a\} \cup \{\boldsymbol{F}B \mid B \in EB_P(U)\}\}$ // record loop nogood

Requirements for UnfoundedSet

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

Requirements for UnfoundedSet

- Implementations of UnfoundedSet must guarantee the following for a result U
  (1) $U \subseteq (atom(P) \setminus A^{\boldsymbol{F}})$
  (2) $EB_P(U) \subseteq A^{\boldsymbol{F}}$
  (3) $U = \emptyset$ iff there is no nonempty unfounded subset of $(atom(P) \setminus A^{\boldsymbol{F}})$
- 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	$\boldsymbol{T}u$		
2	$\boldsymbol{F}\{\sim x, \sim y\}$		
		$\boldsymbol{F}w$	$\{\boldsymbol{T}w, \boldsymbol{F}\{\sim x, \sim y\}\} = \delta(w)$
3	$\boldsymbol{F}\{\sim y\}$		
		$\boldsymbol{F}x$	$\{\boldsymbol{T}x, \boldsymbol{F}\{\sim y\}\} = \delta(x)$
		$\boldsymbol{F}\{x\}$	$\{\boldsymbol{T}\{x\}, \boldsymbol{F}x\} \in \Delta(\{x\})$
		$\boldsymbol{F}\{x, y\}$	$\{\boldsymbol{T}\{x, y\}, \boldsymbol{F}x\} \in \Delta(\{x, y\})$
		$\boldsymbol{T}\{\sim x\}$	$\{\boldsymbol{F}\{\sim x\}, \boldsymbol{F}x\} = \delta(\{\sim x\})$
		$\boldsymbol{T}y$	$\{\boldsymbol{F}\{\sim y\}, \boldsymbol{F}y\} = \delta(\{\sim y\})$
		$\boldsymbol{T}\{v\}$	$\{\boldsymbol{T}u, \boldsymbol{F}\{x, y\}, \boldsymbol{F}\{v\}\} = \delta(u)$
		$\boldsymbol{T}\{u, y\}$	$\{\boldsymbol{F}\{u, y\}, \boldsymbol{T}u, \boldsymbol{T}y\} = \delta(\{u, y\})$
		$\boldsymbol{T}v$	$\{\boldsymbol{F}v, \boldsymbol{T}\{u, y\}\} \in \Delta(v)$
			$\{\boldsymbol{T}u, \boldsymbol{F}\{x\}, \boldsymbol{F}\{x, y\}\} = \lambda(u, \{u, v\})$ ✘

Outline

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

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

    if $k = dlevel(\sigma)$ **then**

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

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

    else return (δ, k)

Example: ConflictAnalysis

Consider

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

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

Example: ConflictAnalysis

Consider

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

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

Example: ConflictAnalysis

Consider

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

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

Example: ConflictAnalysis

Consider

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

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

Example: ConflictAnalysis

Consider

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

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

Example: ConflictAnalysis

Consider

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

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

Example: ConflictAnalysis

Consider

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

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

Example: ConflictAnalysis

Consider

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

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

Example: ConflictAnalysis

Consider

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

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

Example: ConflictAnalysis

Consider

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

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

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 !