Towards Preprocessing for Abstract Argumentation Frameworks

Stefan Woltran

Based on joint work with
Ringo Baumann, Wolfgang Dvořák and Thomas Linsbichler

July 3rd, 2017
Seminal Paper by Phan Minh Dung:
Seminal Paper by Phan Minh Dung:

“The purpose of this paper is to study the fundamental mechanism, humans use in argumentation, and to explore ways to implement this mechanism on computers.”
Seminal Paper by Phan Minh Dung:

“The purpose of this paper is to study the fundamental mechanism, humans use in argumentation, and to explore ways to implement this mechanism on computers.”

“The idea of argumentational reasoning is that a statement is believable if it can be argued successfully against attacking arguments.”

“...The purpose of this paper is to study the fundamental mechanism, humans use in argumentation, and to explore ways to implement this mechanism on computers.”

“The idea of argumentational reasoning is that a statement is believable if it can be argued successfully against attacking arguments.”

“[...] a formal, abstract but simple theory of argumentation is developed to capture the notion of acceptability of arguments.”
Argumentation Frameworks

...thus abstract away from everything but attacks (calculus of opposition)

Example
...thus abstract away from everything but attacks (calculus of opposition)

\[ stb(F) = \{a, d, e\}, \]
Prologue

Argumentation Frameworks

...thus abstract away from everything but attacks (calculus of opposition)

Example

\[ stb(F) = \{ \{a, d, e\}, \{b, c, e\}\} \]
Prologue

Argumentation Frameworks

... thus abstract away from everything but attacks (calculus of opposition)

Example

\[ \text{stb}(F) = \{\{a, d, e\}, \{b, c, e\}\} \]
\[ \text{pref}(F) = \{\{a, d, e\}\}, \]

\[
\begin{align*}
a & \rightarrow c \\
c & \rightarrow d \\
d & \rightarrow f \\
f & \rightarrow e \\
b & \rightarrow d \\
\end{align*}
\]
Argumentation Frameworks

...thus abstract away from everything but attacks (calculus of opposition)

Example

$$\text{stb}(F) = \left\{ \left\{ a, d, e \right\}, \left\{ b, c, e \right\} \right\}$$

$$\text{pref}(F) = \left\{ \left\{ a, d, e \right\}, \left\{ b, c, e \right\} \right\}$$
Prologue

Argumentation Frameworks

...thus abstract away from everything but attacks (calculus of opposition)

Example

\[ stb(F) = \{\{a, d, e\}, \{b, c, e\}\} \]
\[ pref(F) = \{\{a, d, e\}, \{b, c, e\}, \{a, b\}\} \]
Prologue

Nonmon. Reasoning

Logic Programming

n-Person Games

Preprocessing for Abstract Argumentation

July 3rd, 2017
Prologue

ABA

ASPIC

Nonmon. Reasoning

Logic Programming

$n$-Person Games

Preprocessing for Abstract Argumentation

July 3rd, 2017
Prologue

Preprocessing for Abstract Argumentation

Stefan Woltran (TU Wien)
Prologue

Preprocessing for Abstract Argumentation

Nonmon. Reasoning
Logic Programming
n-Person Games

ABA
ASPIC

CoQuiAAS
ArgSemSAT
ASPARTIX
ConArg
Cegartix

Outline

- Quick Background on Argumentation Frameworks
- The Role of Preprocessing
- Theoretical Foundations
- Building a Preprocessing Machine
- Conclusions and Open Questions
Definition

An argumentation framework (AF) is a pair \((A, R)\) where

1. \(A \subseteq U\) is a finite set of arguments and
2. \(R \subseteq A \times A\) is the attack relation representing conflicts.
An argumentation framework (AF) is a pair \((A, R)\) where:

- \(A \subseteq U\) is a finite set of arguments and
- \(R \subseteq A \times A\) is the attack relation representing conflicts.

**Example**

\[
F = (\{a, b, c, d, e, f\},
\{(a, c), (c, a), (c, d), (d, c), (d, b), (b, d), (c, f), (d, f), (f, f), (f, e)\})
\]
Stable Extensions

Given an AF $F = (A, R)$, a set $E \subseteq A$ is a stable extension of $F$, if

- $E$ is conflict-free in $F$ (i.e., for each $a, b \in E$, $(a, b) \notin R$),
- for each $a \in A \setminus E$, there exists some $b \in E$, such that $(b, a) \in R$. 

Example image with nodes and edges is not provided in the text.
**Background**

**Stable Extensions**

Given an AF $F = (A, R)$, a set $E \subseteq A$ is a stable extension of $F$, if

1. $E$ is conflict-free in $F$ (i.e., for each $a, b \in E$, $(a, b) \notin R$),
2. for each $a \in A \setminus E$, there exists some $b \in E$, such that $(b, a) \in R$.

**Example**

$stb(F) = \{\{a, d, e\}\}$,
Stable Extensions

Given an AF $F = (A, R)$, a set $E \subseteq A$ is a stable extension of $F$, if

- $E$ is conflict-free in $F$ (i.e., for each $a, b \in E$, $(a, b) \notin R$),
- for each $a \in A \setminus E$, there exists some $b \in E$, such that $(b, a) \in R$.

Example

$$\text{stb}(F) = \{\{a, d, e\}, \{b, c, e\}\}$$
Stable Extensions

Given an AF $F = (A, R)$, a set $E \subseteq A$ is a stable extension of $F$, if

- $E$ is conflict-free in $F$ (i.e., for each $a, b \in E$, $(a, b) \notin R$),
- for each $a \in A \setminus E$, there exists some $b \in E$, such that $(b, a) \in R$.

Example

$$stb(F) = \{ \{a, d, e\}, \{b, c, e\}, \{a, b, e\} \}$$
Background

Admissible Sets

Given an AF $F = (A, R)$, a set $E \subseteq A$ is admissible in $F$, if

- $E$ is conflict-free in $F$ and
- each $a \in E$ is defended by $E$ in $F$, i.e. for each $b \in A$ with $(b, a) \in R$, there exists some $c \in E$, such that $(c, b) \in R$. 
Given an $AF F = (A, R)$, a set $E \subseteq A$ is admissible in $F$, if

- $E$ is conflict-free in $F$ and
- each $a \in E$ is defended by $E$ in $F$, i.e. for each $b \in A$ with $(b, a) \in R$, there exists some $c \in E$, such that $(c, b) \in R$.

**Example**

$$\text{adm}(F) = \{\{a, b, e\}\},$$
Admissible Sets

Given an AF $F = (A, R)$, a set $E \subseteq A$ is admissible in $F$, if

- $E$ is conflict-free in $F$ and
- each $a \in E$ is defended by $E$ in $F$, i.e. for each $b \in A$ with $(b, a) \in R$, there exists some $c \in E$, such that $(c, b) \in R$.

Example

$adm(F) = \{\{a, b, e\}, \{a, d, e\}\}$, 
Background

Admissible Sets

Given an AF $F = (A, R)$, a set $E \subseteq A$ is admissible in $F$, if

- $E$ is conflict-free in $F$ and
- each $a \in E$ is defended by $E$ in $F$, i.e. for each $b \in A$ with $(b, a) \in R$, there exists some $c \in E$, such that $(c, b) \in R$.

Example

$\text{adm}(F) = \{\{a, b, e\}, \{a, d, e\}, \{b, c, e\}\}$,
Admissible Sets

Given an AF $F = (A, R)$, a set $E \subseteq A$ is admissible in $F$, if

- $E$ is conflict-free in $F$ and
- each $a \in E$ is defended by $E$ in $F$, i.e. for each $b \in A$ with $(b, a) \in R$, there exists some $c \in E$, such that $(c, b) \in R$.

Example

$$adm(F) = \{\{a, b, e\}, \{a, d, e\}, \{b, c, e\}, \{a, b\}, \{a, d\}, \{a, e\}, \{b, c\}, \{b, e\}, \{d, e\}, \{c, e\}\}.$$
Admissible Sets

Given an $AF F = (A, R)$, a set $E \subseteq A$ is admissible in $F$, if

- $E$ is conflict-free in $F$ and
- each $a \in E$ is defended by $E$ in $F$, i.e. for each $b \in A$ with $(b, a) \in R$, there exists some $c \in E$, such that $(c, b) \in R$.

Example

$adm(F) = \{\{a, b, e\}, \{a, d, e\}, \{b, c, e\}, \{a, b\}, \{a, d\}, \{a, e\}, \{b, c\}, \{b, e\}, \{d, e\}, \{c, e\}, \{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \}$
Admissible Sets

Given an \( \mathcal{AF} F = (A, R) \), a set \( E \subseteq A \) is admissible in \( F \), if

- \( E \) is conflict-free in \( F \) and
- each \( a \in E \) is defended by \( E \) in \( F \), i.e. for each \( b \in A \) with \( (b, a) \in R \), there exists some \( c \in E \), such that \( (c, b) \in R \).

Example

\[
\text{adm}(F) = \{\{a, b, e\}, \{a, d, e\}, \{b, c, e\}, \\
\{a, b\}, \{a, d\}, \{a, e\}, \{b, c\}, \{b, e\}, \{d, e\}, \{c, e\}, \\
\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \emptyset\}
\]
Preferred Extensions

Given an \( \mathcal{A} \mathcal{F} \ A = (A, R) \), a set \( E \subseteq A \) is a preferred extension in \( \mathcal{F} \), if

- \( E \) is admissible in \( \mathcal{F} \) and
- there is no admissible set \( T \subseteq A \) of \( \mathcal{F} \) with \( T \supseteq E \).

\[ \Rightarrow \] Maximal admissible sets (w.r.t. set-inclusion).
Preferred Extensions

Given an AF $F = (A, R)$, a set $E \subseteq A$ is a preferred extension in $F$, if

- $E$ is admissible in $F$ and
- there is no admissible set $T \subseteq A$ of $F$ with $T \supset E$.

$\Rightarrow$ Maximal admissible sets (w.r.t. set-inclusion).

Example

pref$(F) = \{\{a, d, e\}, \{b, c, e\}, \{a, b\}, \{a, d\}, \{b, c\}, \{d, e\}, \{c, e\}, \{a\}, \{b\}, \{c\}, \{d\}, \emptyset\}$
For \((A, R)\) and \(E \subseteq A\), \(E_R^{\oplus} = E \cup \{b \mid (a, b) \in R\}\) denotes the range.

- **complete**: admissible sets that contain all defended arguments
- **semi-stable**: admissible sets with subset-maximal range
- **stage**: conflict-free sets with subset-maximal range

**Unique-status semantics:**
- **grounded**: subset-minimal complete set
- **ideal**: subset-maximal \(adm\) set contained in each \(pref\) extension
<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>$Cred_\sigma$</th>
<th>$Skept_\sigma$</th>
<th>$Ver_\sigma$</th>
<th>$NE_\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$cf$</td>
<td>in L</td>
<td>trivial</td>
<td>in L</td>
<td>in L</td>
</tr>
<tr>
<td>$grd$</td>
<td>P-c</td>
<td>P-c</td>
<td>P-c</td>
<td>in L</td>
</tr>
<tr>
<td>$stb$</td>
<td>NP-c</td>
<td>coNP-c</td>
<td>in L</td>
<td>NP-c</td>
</tr>
<tr>
<td>$adm$</td>
<td>NP-c</td>
<td>trivial</td>
<td>trivial</td>
<td>NP-c</td>
</tr>
<tr>
<td>$comp$</td>
<td>NP-c</td>
<td>P-c</td>
<td>in L</td>
<td>NP-c</td>
</tr>
<tr>
<td>$ideal$</td>
<td>in $\Theta_2^P$</td>
<td>in $\Theta_2^P$</td>
<td>in $\Theta_2^P$</td>
<td>in $\Theta_2^P$</td>
</tr>
<tr>
<td>$pref$</td>
<td>NP-c</td>
<td>$\Pi_2^P$-c</td>
<td>coNP-c</td>
<td>NP-c</td>
</tr>
<tr>
<td>$sem$</td>
<td>$\Sigma_2^P$-c</td>
<td>$\Pi_2^P$-c</td>
<td>coNP-c</td>
<td>NP-c</td>
</tr>
<tr>
<td>$stage$</td>
<td>$\Sigma_2^P$-c</td>
<td>$\Pi_2^P$-c</td>
<td>coNP-c</td>
<td>in L</td>
</tr>
</tbody>
</table>
The tasks supported by the solvers are summarized in the following table:

<table>
<thead>
<tr>
<th></th>
<th>D3</th>
<th>GC</th>
<th>DC</th>
<th>DG</th>
<th>CE</th>
<th>EE</th>
<th>FR</th>
<th>DC</th>
<th>DG</th>
<th>CE</th>
<th>EE</th>
<th>ST</th>
<th>DC</th>
<th>DG</th>
<th>CE</th>
<th>EE</th>
<th>QTG</th>
<th>DC</th>
<th>DG</th>
<th>CE</th>
<th>EE</th>
<th>GTG</th>
<th>DC</th>
<th>DG</th>
<th>CE</th>
<th>EE</th>
<th>OR</th>
<th>DC</th>
<th>CE</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>argmat olpb</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>argmat dvisat</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>argmat-mpg</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>argmat-sat</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ArgSemSAT</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ArgTools</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASPMin</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cegartix</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chimærarg</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CunAyg</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoQuAA3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EqArgSolver</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gg-sts</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>goDIAMOND</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>heureka</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pyglaf</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Preprocessing refers to a family of simplifications which are computationally easy to perform and are equivalence preserving:

- SAT: tautology elimination, clause subsumption, ...

Proved very successful in SAT and QSAT solving.

Preprocessing in the context of argumentation poses some additional challenges (nonmonotonicity!)
Clause Elimination for SAT and QSAT

Marijn Heule  
Department of Computer Science,  
The University of Texas at Austin, USA

Matti Järvisalo  
HIIT, Department of Computer Science,  
University of Helsinki, Finland

Florian Lonsing  
Institute of Information Systems,  
Vienna University of Technology, Austria

Martina Seidl  
Armin Biere  
Institute for Formal Models and Verification,  
Johannes Kepler University Linz, Austria

MARIJN@CS.UTEXAS.EDU  
MATTI.JARVISALO@CS.HELSEINKI.FI  
FLORIAN.LONSING@TUWIEI.AC.AT  
MARTINA.SEIDL@JKU.AT  
BIERE@JKU.AT

Abstract

The famous archetypical NP-complete problem of Boolean satisfiability (SAT) and its PSPACE-complete generalization of quantified Boolean satisfiability (QSAT) have become central declarative programming paradigms through which real-world instances of various computationally hard problems can be efficiently solved. This success has been achieved through several breakthroughs in practical implementations of decision procedures for SAT and QSAT, that is, in SAT and QSAT solvers. Here, simplification techniques for conjunctive normal form (CNF) for SAT and for prenex conjunctive normal form (PCNF) for QSAT—the standard input formats of SAT and QSAT solvers—have recently proven very effective in increasing solver efficiency when applied before (i.e., in pre-processing) or during (i.e., in in-processing) satisfiability search.

In this article, we develop and analyze clause elimination procedures for pre- and in-processing. Clause elimination procedures form a family of (P)CNF formula simplification techniques which remove clauses that have specific (in practice polynomial-time) redundancy properties while maintaining the satisfiability status of the formulas. Extending known procedures such as tautology, subsumption, and blocked clause elimination, we introduce novel elimination procedures based...
Clause Elimination for SAT and QSAT

Marijn Heule
Department of Computer Science,
The University of Texas at Austin, USA

Matti Järvisalo
HIIT, Department of Computer Science,
University of Helsinki, Finland

Florian Lonsing
Institute of Information Systems,
Vienna University of Technology, Austria

Martina Scidl
Armin Biere
Institute for Formal Models and Verification,
Johannes Kepler University Linz, Austria

Abstract

The famous archetypical NP-complete problem of Boolean satisfiability has been the focus of extensive research over the years leading to the development of different techniques and programming paradigms through which real-world instances of NP-complete problems can be efficiently solved. This success has been achieved through several breakthroughs in practical implementations of decision procedures for SAT and QSAT, that is, in SAT and QSAT solvers. Here, simplification techniques for conjunctive normal form (CNF) for SAT and for prenex conjunctive normal form (PCNF) for QSAT—the standard input formats of SAT and QSAT solvers—have recently proven very effective in increasing solver efficiency when applied before (i.e., in preprocessing) or during (i.e., in inprocessing) satisfiability search.

In this article, we develop and analyze clause elimination procedures for pre- and inprocessing. Clause elimination procedures form a family of (P)CNF formula simplification techniques which remove clauses that have specific (in practice polynomial-time) redundancy properties while maintaining the satisfiability status of the formulas. Extending known procedures such as tautology, subsumption, and blocked clause elimination, we introduce several elimination procedures based

Example from the QBF world:

- Preprocessor Bloqer solved 471 of 1130 instances from QBFEVAL’16.
- DepQBF solves 556 instances without preprocessing, but 817 with preprocessing.
Effect of (non equivalence-preserving) modifications with instances from the ICCMA 2015 stable generator.
In order to define possible preprocessing steps, we require a suitable notion of equivalence... which allows to verify which subparts of AFs can be simplified... under different semantics.

More precisely, we want to find pairs $(F, F')$ such that replacing $F$ by $F'$ in any AF $G$ does not change the extensions of $G$ (under certain assumptions).
In order to define possible preprocessing steps, we require

- a suitable notion of equivalence . . .
- which allows to verify which subparts of AFs can be simplified . . .
- under different semantics
Theoretical Foundations

In order to define possible preprocessing steps, we require

- a suitable notion of equivalence . . .
- which allows to verify which subparts of AFs can be simplified . . .
- under different semantics

More precisely, we want to find pairs \((F, F')\) such that replacing \(F\) by \(F'\) in any AF \(G\) does not change the extensions of \(G\) (under certain assumptions)
**Definition**

Given a semantics $\sigma$. Two AFs $F$ and $F'$ are **(standard) equivalent** w.r.t. $\sigma$ (in symbols $F \equiv^\sigma F'$) iff $\sigma(F) = \sigma(F')$.

**Definition**

Given a semantics $\sigma$. Two AFs $F$ and $F'$ are **strongly equivalent** w.r.t. $\sigma$ (in symbols $F \equiv^\sigma_S F'$) iff $F \cup H \equiv^\sigma F' \cup H$ holds for each AF $H$. 
Example

\[ stb(F) = \{\{a, d, e\}, \{b, c, e\}\} \quad \equiv \quad stb(F') = \{\{a, d, e\}, \{b, c, e\}\} \]
Theoretical Foundations – Notions of Equivalence

Example

\[ a \leftarrow c \rightarrow d \leftarrow b \leftarrow f \rightarrow e \]
\[ \equiv_{S}^{stb} \]

Follows from results in [Oikarinen & W, 2011].
Theoretical Foundations – Notions of Equivalence

Example

\[
\text{pref}(F) = \{\{a, d, e\}, \{b, c, e\}, \{a, b\}\}
\]

\[
\text{pref}(F') = \{\{a, d, e\}, \{b, c, e\}, \{a, b, e\}\}
\]

\[\neq^{\text{pref}}\]
Example

\[ stb(F) = \{\{a, d, e\}, \{b, c, e\}\} \]

\[ stb(F') = \{\{a, d, e\}, \{b, c, e\}\} \]

\[ stb(F) \equiv_{stb} stb(F') \]
Theoretical Foundations – Notions of Equivalence

Example

\[ a \rightarrow c \rightarrow f \rightarrow e \]

\[ b \rightarrow d \]

\[ \not\equiv_{stb}^{S} \]

\[ a \rightarrow c \]

\[ b \leftarrow d \]

\[ e \]
Theoretical Foundations – Notions of Equivalence

Example

\[ \text{stb}(F \cup H) = \{\{a, d, e\}, \{b, c, e\}\} \]

but \[ \text{stb}(F' \cup H) = \{\{a, d, e\}, \{b, c, e\},\{a, b, e\}, \{a, d, f\}, \{b, c, f\}\} \]

\[ \neq_{\text{stb}} \]
Theoretical Foundations – Main Results

Observations:
- Standard equivalence is too weak for our purpose
- Strong equivalence is too restricted
  - For self-loop free AFs $F, F'$: $F \equiv^s F' \text{ iff } F = F'$

We thus require a notion of equivalence which takes into account the neighborhood in an adequate way.

Definition
Given a semantics $\sigma$ and arguments $C \subseteq U$. Two AFs $F$ and $F'$ are $C$-relativized equivalent w.r.t. $\sigma$ (in symbols $F \equiv^C \sigma F'$) iff $F \cup H \equiv^C \sigma F' \cup H$ holds for each AF $H$ not containing arguments from $C$.

For $C = \emptyset$, $C$-relativized equivalence coincides with strong equivalence for $C = U$, $C$-relativized equivalence is just standard equivalence
Observations:

- Standard equivalence is too weak for our purpose
- Strong equivalence is too restricted
  - For self-loop free AFs $F, F'$: $F \equiv^\sigma_S F'$ iff $F = F'$!

We thus require a notion of equivalence which takes into account the neighborhood in an adequate way.

**Definition**

Given a semantics $\sigma$ and arguments $C \subseteq U$. Two AFs $F$ and $F'$ are $C$-relativized equivalent w.r.t. $\sigma$ (in symbols $F \equiv^\sigma_C F'$) iff $F \cup H \equiv^\sigma F' \cup H$ holds for each AF $H$ not containing arguments from $C$. 
Theoretical Foundations – Main Results

Observations:

- Standard equivalence is too weak for our purpose
- Strong equivalence is too restricted
  - For self-loop free AFs $F, F': F \equiv_{S} F' \iff F = F'$!

We thus require a notion of equivalence which takes into account the neighborhood in an adequate way.

Definition

Given a semantics $\sigma$ and arguments $C \subseteq U$. Two AFs $F$ and $F'$ are **$C$-relativized equivalent** w.r.t. $\sigma$ (in symbols $F \equiv_{C}^{\sigma} F'$) iff $F \cup H \equiv_{\sigma}^{C} F' \cup H$ holds for each AF $H$ not containing arguments from $C$.

- for $C = \emptyset$, $C$-relativized equivalence coincides with strong equivalence
- for $C = U$, $C$-relativized equivalence is just standard equivalence
Definition

Given a semantics $\sigma$ and arguments $C \subseteq U$. Two AFs $F$ and $F'$ are $C$-relativized equivalent w.r.t. $\sigma$ (in symbols $F \equiv^C_\sigma F'$) iff $F \cup H \equiv^\sigma F' \cup H$ holds for each AF $H$ not containing arguments from $C$. 
Definition

Given a semantics $\sigma$ and arguments $C \subseteq U$. Two AFs $F$ and $F'$ are $C$-relativized equivalent w.r.t. $\sigma$ (in symbols $F \equiv^C_\sigma F'$) iff $F \cup H \equiv^\sigma C F' \cup H$ holds for each AF $H$ not containing arguments from $C$.

Example with $C = \{c, d, e, f\}$
Theoretical Foundations – Main Results

We first define a parameterized notion of the semantics.

**Definition**

Let $F = (A, R)$, $C \subseteq U$. The **$C$-restricted stable extensions** of $F$ are

$$stb_C(F) = \{ E \in cf(F) | A \cap C \subseteq E^\oplus_F \}$$
We first define a parameterized notion of the semantics.

**Definition**

Let \( F = (A, R) \), \( C \subseteq U \). The \( C \)-restricted stable extensions of \( F \) are

\[
stb_C(F) = \{ E \in cf(F) \mid A \cap C \subseteq E_F^+ \}\]

**Example with** \( C = \{c, d, e, f\} \)

\[
stb_C(F) = \{\{a, d, e\}, \{b, c, e\}, \{d, e\}, \{c, e\}\}
\]

\[
stb_C(F') = \{\{a, d, e\}, \{b, c, e\}, \{d, e\}, \{c, e\}\}
\]
We first define a parameterized notion of the semantics.

**Definition**

Let $F = (A, R)$, $C \subseteq U$. The $C$-restricted stable extensions of $F$ are

$$stb_C(F) = \{E \in cf(F) \mid A \cap C \subseteq E^+_F\}$$

**Example with $C = \{e, f\}$**

$$stb_C(F) = \{\{a, d, e\}, \{b, c, e\}, \{d, e\}, \{c, e\}\}$$

$$stb_C(F') = \{\{a, d, e\}, \{b, c, e\}, \{d, e\}, \{c, e\}, \{a, e\}, \{b, e\}, \{e\}\}$$
Theoretical Foundations – Main Results

For other semantics, such variants can be defined accordingly.

**Definition**

Let $F$ be an AF, $C \subseteq U$. We define

\[
\begin{align*}
\text{adm}_C(F) &= \{ E \in \text{cf}(F) \mid E^- \cap C \subseteq E^+ \} \\
\text{pref}_C(F) &= \{ E \in \text{adm}_C(F) \mid \text{for all } D \in \text{adm}_C(F) \text{ with } E \setminus C = D \setminus C, E^+_F \setminus C \subseteq D^+_F \setminus C, E^-_F \setminus E^+_F \supseteq D^-_F \setminus D^+_F : E \cap C \not\subset D \cap C \} 
\end{align*}
\]

For complete and grounded semantics, similar definitions can be given by using a parameterized version of the characteristic function.

**Theorem**

Let $F$ be an AF and $C \subseteq U$. Then, the following relations hold:

\[
\begin{align*}
\text{stb}_C(F) &\subseteq \text{pref}_C(F) \subseteq \text{comp}_C(F) \subseteq \text{adm}_C(F) \\
\text{grd}_C(F) &\subseteq \text{comp}_C(F)
\end{align*}
\]
Theoretical Foundations – Main Results

For other semantics, such variants can be defined accordingly.

**Definition**

Let $F$ be an AF, $C \subseteq U$. We define

$$adm_C(F) = \{ E \in cf(F) \mid E^-_F \cap C \subseteq E^+_F \}$$

$$pref_C(F) = \{ E \in adm_C(F) \mid \text{for all } D \in adm_C(F) \text{ with } E \setminus C = D \setminus C, E^+_F \setminus C \subseteq D^+_F \setminus C, E^-_F \setminus E^+_F \supseteq D^-_F \setminus D^+_F : E \cap C \not\subset D \cap C \}$$

For complete and grounded semantics, similar definitions can be given by using a parameterized version of the characteristic function.
Theoretical Foundations – Main Results

For other semantics, such variants can be defined accordingly.

**Definition**

Let $F$ be an AF, $C \subseteq U$. We define

\[
\begin{align*}
\text{adm}_C(F) &= \{ E \in \text{cf}(F) \mid E_F^- \cap C \subseteq E_F^+ \} \\
\text{pref}_C(F) &= \{ E \in \text{adm}_C(F) \mid \text{for all } D \in \text{adm}_C(F) \text{ with } E \setminus C = D \setminus C, E_F^+ \setminus C \subseteq D_F^+ \setminus C, E_F^- \setminus E_F^+ \supseteq D_F^- \setminus D_F^+ : E \cap C \nsubseteq D \cap C \} 
\end{align*}
\]

For complete and grounded semantics, similar definitions can be given by using a parameterized version of the characteristic function.

**Theorem**

Let $F$ be an AF and $C \subseteq U$. Then, the following relations hold:

\[
\text{stb}_C(F) \subseteq \text{pref}_C(F) \subseteq \text{comp}_C(F) \subseteq \text{adm}_C(F); \quad \text{grd}_C(F) \subseteq \text{comp}_C(F).
\]
Theoretical Foundations – Main Results

Theorem

Let $F, F'$ be AFs and $C \subseteq U$. Then, $F \equiv_C^{stb} F'$ iff jointly

1. $stb_C(F) = stb_C(F')$;
2. if $stb_C(F) \neq \emptyset$, $A(F) \setminus C = A(F') \setminus C$;
3. for all $E \in stb_C(F)$, $E_F^+ \setminus C = E_{F'}^+ \setminus C$. 

Stefan Woltrant (TU Wien)
Preprocessing for Abstract Argumentation July 3rd, 2017 28 / 37
Theoretical Foundations – Main Results

Theorem

Let $F, F'$ be AFs and $C \subseteq U$. Then, $F \equiv_{C}^{stb} F'$ iff jointly

1. $stb_{C}(F) = stb_{C}(F')$;
2. if $stb_{C}(F) \neq \emptyset$, $A(F) \setminus C = A(F') \setminus C$;
3. for all $E \in stb_{C}(F)$, $E_{F}^{\uparrow} \setminus C = E_{F'}^{\uparrow} \setminus C$.

Example with $C = \{c, d, e, f\}$

Recall (1) $stb_{C}(F) = stb_{C}(F') = \{\{a, d, e\}, \{b, c, e\}, \{d, e\}, \{c, e\}\}$; (2) and (3) also hold.
Theoretical Foundations – Main Results

**Theorem**

Let $F, F'$ be AFs and $C \subseteq U$. Then, $F \equiv_{C}^{stb} F'$ iff jointly

1. $stb_{C}(F) = stb_{C}(F')$;
2. if $stb_{C}(F) \neq \emptyset$, $A(F) \setminus C = A(F') \setminus C$;
3. for all $E \in stb_{C}(F)$, $E_{F}^{+} \setminus C = E_{F'}^{+} \setminus C$.

- For $C = U$, (1)–(3) reduce to $stb(F) = stb(F')$;
- For $C = \emptyset$, we have
  1. $cf(F) = cf(F')$,
  2. $A(F) = A(F')$,
  3. for all $E \in cf(F)$, $E_{F}^{+} = E_{F'}^{+}$, i.e. $F, F'$ coincide except for attacks from self-attacking arguments

(equals known results for strong equivalence).
Theoretical Foundations – Main Results

**Theorem**

Let $F, F'$ be AFs and $C \subseteq U$. Then, $F \equiv^{stb}_C F'$ iff jointly

1. $stb_C(F) = stb_C(F')$;
2. if $stb_C(F) \neq \emptyset$, $A(F) \setminus C = A(F') \setminus C$;
3. for all $E \in stb_C(F)$, $E_F^+ \setminus C = E_{F'}^+ \setminus C$.

Similar characterization results can be shown for the other main semantics.
Replacement Theorem

For AFs $F, F', G$ and $C \subseteq U$ such that $A(F) \cup A(F') \subseteq C$, $(A(G) \setminus A(F)) \cap C = \emptyset$, and $F$ is a sub-AF of $G$, let $B = (A(F))_G^\oplus \cup (A(F))_G^-$ and $F^G = (B, R(G) \cap (B \times B))$. Then, $F^G \equiv_C F^G[F/F']$ implies $G \equiv\sigma G[F/F']$.
Replacement Theorem

For AFs $F, F', G$ and $C \subseteq U$ such that $A(F) \cup A(F') \subseteq C$, $(A(G) \setminus A(F)) \cap C = \emptyset$, and $F$ is a sub-AF of $G$, let $B = (A(F))^\oplus_G \cup (A(F))^\neg_G$ and $F^G = (B, R(G) \cap (B \times B))$. Then, $F^G \equiv_C^\sigma F^G[F/F']$ implies $G \equiv^\sigma G[F/F']$.

Example

Odd-length cycles $(a_1, \ldots, a_n, a_1)$ can be simplified under the stable semantics for any AF where the cycle has exactly one incoming attack $(b, a_1)$ as follows:

- make $a_1$ self-attacking
- remove all $a_i$ with odd $i \neq 1$ plus adjacent attacks
Some complexity results:

<table>
<thead>
<tr>
<th>( \sigma )</th>
<th>( \text{grd} )</th>
<th>( \text{stb} )</th>
<th>( \text{adm} )</th>
<th>( \text{comp} )</th>
<th>( \text{pref} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F \equiv^\sigma_S G )</td>
<td>in L</td>
<td>in L</td>
<td>in L</td>
<td>in L</td>
<td>in L</td>
</tr>
<tr>
<td>( F \equiv^\sigma G )</td>
<td>P-c</td>
<td>coNP-c.</td>
<td>coNP-c.</td>
<td>coNP-c.</td>
<td>( \Pi^P_2 )-c.</td>
</tr>
<tr>
<td>( F \equiv^\sigma_C G )</td>
<td>coNP-c.</td>
<td>coNP-c.</td>
<td>coNP-c.</td>
<td>coNP-c.</td>
<td>( \Pi^P_2 )-c.</td>
</tr>
</tbody>
</table>
Building a Preprocessing Machine - Our Vision
1. Collect patterns \((F^G, F, F')\) which apply for the replacement theorem

- This can be done in an offline-phase
- employ the equivalence characterizations
- different patterns for different semantics
2. Build a tool that scans a given AF for possible application of the replacement patterns \( (F^G, F, F') \)

- Requires efficient implementation of subgraph-isomorphism problem
- sort out which size of subgraphs allow for efficient scanning for patterns
- integrate other known simplifications (computation of grounded extension) and interleave this with the applied replacements
3. Experimental Evaluation and Fine-Tuning

- which replacements actually help solvers?
  - Preprocessing on the argumentation level should go beyond preprocessing on encodings
- identification of “promising regions” (e.g. potential separation into SCCs)
- integration of ML techniques
Abstract argumentation a central formalism in AI
ICCMA has stipulated development of solvers
In other domains, preprocessing recognized as a crucial step to improve efficiency
Nonmonotonic nature of argumentation semantics makes life complicated

In this talk:
- Introduced a suitable notion of equivalence to seek for simplification patterns
- Discussion of next steps towards practical realization of a preprocessing tool
  ▶ Recall: this is beneficial for all solvers!
Future Work and Open Questions

- Understand $C$-relativized equivalence for further semantics
- What can be done for acceptance problems?
- Claim: preprocessing could be more powerful if we allow to shift from AFs to a more general formalism (for instance, SETAFs)
  - however, this requires solvers for this general formalism

Thanks for your attention and enjoy LPNMR!

Stefan Woltran (TU Wien)
Future Work and Open Questions

- Understand $C$-relativized equivalence for further semantics
- What can be done for acceptance problems?
- Claim: preprocessing could be more powerful if we allow to shift from AFs to a more general formalism (for instance, SETAFs)
  - however, this requires solvers for this general formalism

Thanks for your attention and enjoy LPNMR!
**References**

