Computational Aspects of $cf_2$ and $stage_2$ Argumentation Semantics

COMMA 2012 (Vienna)

Wolfgang Dvořák¹, Sarah Alice Gaggl²

¹Theory and Applications of Algorithms Group, University of Vienna
²Institute of Information Systems, Vienna University of Technology

Sept. 11th, 2012

Supported by the Vienna Science and Technology Fund (WWTF) through project ICT08-028.
Motivation

*cf2-Semantics*:

- Uniform treatment of odd- and even-length cycles.
- Fulfills most evaluation criteria proposed in [Baroni & Giacomin 07].
- But produces questionable results on several frameworks.
  → fixed by *stage2* semantics
Motivation

**cf2-Semantics:**
- Uniform treatment of odd- and even-length cycles.
- Fulfills most evaluation criteria proposed in [Baroni & Giacomin 07].
- But produces questionable results on several frameworks.
  → fixed by stage2 semantics

**stage2-Semantics:**
- Instantiates the SCC-recursive schema of cf2 semantics with stage semantics.
- Satisfies the evaluation criteria.
- Coincides with stable semantics in the absence of odd-length cycles.
1. Motivation

Motivation

Computational Issues:

- For \( cf2 \) and \( stage2 \) typical reasoning tasks are computationally hard, i.e. NP/coNP-hard.

- But this is worst case complexity – there might be classes of instances that show milder complexity ⇒ tractable fragments.

- The analysis of computational complexity and identifying tractable cases are indispensable for building efficient systems.

- So far only the general complexity of the main reasoning tasks was studied [Gaggl & Woltran 12; Dvořák & Gaggl 12].
Contributions

**Complexity analysis** of $cf2$ and $stage2$ semantics:

- We consider four graph classes for being tractable fragments and provide either polynomial time algorithms or hardness results.
- We discuss possible parameters for fixed-parameter tractability.

Discussion of **implementation methods**:

- A labeling based algorithm for $cf2$ and $stage2$ semantics.
2. Background

Argumentation Frameworks

Abstract Argumentation Framework [Dung 95]

An argumentation framework (AF) is a pair $F = (A, R)$, where $A$ is a finite set of arguments and $R \subseteq A \times A$ a attack relation.

Example

$F = (A, R), A = \{a, b, c\}, R = \{(a, b), (b, c), (c, b), (c, c)\}$.
2. Background

cf2-Semantics

*cf2-extensions* are given as follows:

**While** the AF is non-empty

1. Pick a minimal strongly connected component $C$.
2. Compute a *maximal conflict free set* $S$ of $C$ and add it to the extension.
3. Delete $C$ and all arguments attacked by $S$.

---

![Graph](attachment:image.png)
2. Background

**cf2-Semantics**

*cf2-extensions* are given as follows:

While the AF is non-empty

1. Pick a minimal strongly connected component \( C \).
2. Compute a maximal conflict free set \( S \) of \( C \) and add it to the extension.
3. Delete \( C \) and all arguments attacked by \( S \).
**cf2-Semantics**

*cf2-extensions* are given as follows:

**While** the AF is non-empty

1. Pick a minimal strongly connected component $C$.
2. Compute a *maximal conflict free set* $S$ of $C$ and add it to the extension.
3. Delete $C$ and all arguments attacked by $S$. 

![Diagram of connected components](attachment:image.png)

---

1. Specification
2. Background
3. Computational Aspects of *cf2* and *stage2* Semantics
4. Conclusion and Future Work
5. References
**cf2-Semantics**

*cf2-extensions* are given as follows:

**While** the AF is non-empty

1. Pick a minimal strongly connected component $C$.
2. Compute a **maximal conflict free set** $S$ of $C$ and add it to the extension.
3. Delete $C$ and all arguments attacked by $S$. 

---

**Diagram:**

- Vertices: $a, b, c, d, e, f, g, h, i, j$
- Edges: $a ightarrow b$, $b ightarrow c$, $c ightarrow a$, $d ightarrow e$, $e ightarrow g$, $g ightarrow h$, $f ightarrow f$ (loop), $d ightarrow d$ (loop), $j ightarrow i$, $i ightarrow j$. 

---

**Notes:**

- The diagram represents the structure of the minimal strongly connected components and the flow of arguments in the conflict-free set process.
- The loops indicate the internal connections within the component, while the arrows represent the conflict-free flow of arguments.
2. Background

**cf2-Semantics**

*cf2-extensions* are given as follows:

**While** the AF is non-empty

1. Pick a minimal strongly connected component $C$.
2. Compute a maximal conflict free set $S$ of $C$ and add it to the extension.
3. Delete $C$ and all arguments attacked by $S$. 

![Diagram](attachment)
**cf2-Semantics**

*cf2-extensions* are given as follows:

**While** the AF is non-empty

1. Pick a minimal strongly connected component $C$.
2. Compute a maximal conflict free set $S$ of $C$ and add it to the extension.
3. Delete $C$ and all arguments attacked by $S$.

![Diagram](attachment:diagram.png)
2. Background

**cf2-Semantics**

*cf2-extensions* are given as follows:

**While** the AF is non-empty

1. Pick a minimal strongly connected component $C$.
2. Compute a maximal conflict free set $S$ of $C$ and add it to the extension.
3. Delete $C$ and all arguments attacked by $S$.
**cf2-Semantics**

*cf2-extensions* are given as follows:

**While** the AF is non-empty

1. Pick a minimal strongly connected component $C$.
2. Compute a *maximal conflict free set* $S$ of $C$ and add it to the extension.
3. Delete $C$ and all arguments attacked by $S$. 
2. Background

**cf2-Semantics**

**cf2-extensions** are given as follows:

**While** the AF is non-empty

1. Pick a minimal strongly connected component $C$.
2. Compute a **maximal conflict free set** $S$ of $C$ and add it to the extension.
3. Delete $C$ and all arguments attacked by $S$.
**cf2-Semantics**

*cf2-extensions* are given as follows:

**While** the AF is non-empty

1. Pick a minimal strongly connected component $C$.
2. Compute a *maximal conflict free set* $S$ of $C$ and add it to the extension.
3. Delete $C$ and all arguments attacked by $S$. 

![Diagram of a network with labeled nodes and edges indicating the flow and connections between nodes. The diagram illustrates the process of picking a minimal strongly connected component, computing a maximal conflict free set, and removing the component and its attacked arguments.]
**cf2-Semantics**

*cf2-extensions* are given as follows:

**While** the AF is non-empty

1. Pick a minimal strongly connected component $C$.
2. Compute a *maximal conflict free set* $S$ of $C$ and add it to the extension.
3. Delete $C$ and all arguments attacked by $S$.
stage2-Semantics

stage2-extensions are given as follows:

While the AF is non-empty

1. Pick a minimal strongly connected component \( C \).
2. Compute a stage extension \( S \) of \( C \) and add it to the extension.
3. Delete \( C \) and all arguments attacked by \( S \).
## Reasoning Problems

**Credulous Acceptance**

\[ Cred_\sigma: \text{Given } AF \ F = (A, R) \text{ and } a \in A; \text{ is } a \text{ contained in at least one } \sigma\text{-extension of } F? \]

**Skeptical Acceptance**

\[ Skept_\sigma: \text{Given } AF \ F = (A, R) \text{ and } a \in A; \text{ is } a \text{ contained in every } \sigma\text{-extension of } F? \]

**Verifying an extension**

\[ Ver_\sigma: \text{Given } AF \ F = (A, R) \text{ and } S \subseteq A; \text{ is } S \text{ a } \sigma\text{-extension of } F? \]
2. Background

Reasoning Problems

Credulous Acceptance

\[ Cred_\sigma: \text{Given } AF \ F = (A, R) \text{ and } a \in A; \text{ is } a \text{ contained in } \textit{at least one} \ \sigma\text{-extension of } F? \]

Skeptical Acceptance

\[ Skept_\sigma: \text{Given } AF \ F = (A, R) \text{ and } a \in A; \text{ is } a \text{ contained in } \textit{every} \ \sigma\text{-extension of } F? \]

Verifying an extension

\[ Ver_\sigma: \text{Given } AF \ F = (A, R) \text{ and } S \subseteq A; \text{ is } S \text{ a } \sigma\text{-extension of } F? \]

<table>
<thead>
<tr>
<th></th>
<th>naive</th>
<th>stable</th>
<th>stage</th>
<th>cf2</th>
<th>stage2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Cred_\sigma )</td>
<td>in P</td>
<td>NP-c</td>
<td>( \Sigma^P_2 )-c</td>
<td>NP-c</td>
<td>( \Sigma^P_2 )-c</td>
</tr>
<tr>
<td>( Skept_\sigma )</td>
<td>in P</td>
<td>coNP-c</td>
<td>( \Pi^P_2 )-c</td>
<td>coNP-c</td>
<td>( \Pi^P_2 )-c</td>
</tr>
<tr>
<td>( Ver_\sigma )</td>
<td>in P</td>
<td>in P</td>
<td>coNP-c</td>
<td>in P</td>
<td>coNP-c</td>
</tr>
</tbody>
</table>
Acyclic AFs

On Acyclic AFs both semantics coincide with grounded semantics. Complexity classification follows immediately from grounded semantics.

Theorem

For acyclic AFs and $\sigma \in \{\text{cf2}, \text{stage2}\}$ the problems Cred$_{\sigma}$ and Skept$_{\sigma}$ are in $P$. 
Even Cycle Free Argumentation Frameworks

By [Dunne & Bench-Capon 01], reasoning with admissible-based semantics in AFs without even-length cycles is tractable.

But as cf2 and stage2 treat cycles uniformly this result does not extend.

Theorem

For AFs without even-length cycles: $Cred_{cf2}$ is NP-complete, $Skept_{cf2}$ is coNP-complete, $Cred_{stage2}$ is NP-hard, and $Skept_{stage2}$ is coNP-hard.
Even Cycle Free Argumentation Frameworks

By [Dunne & Bench-Capon 01], reasoning with admissible-based semantics in AFs without even-length cycles is tractable.

But as cf2 and stage2 treat cycles uniformly this result does not extend.

**Theorem**

*For AFs without even-length cycles: Cred<sub>cf2</sub> is NP-complete, Skept<sub>cf2</sub> is coNP-complete, Cred<sub>stage2</sub> is NP-hard, and Skept<sub>stage2</sub> is coNP-hard.*

**Proof.**
Bipartite AFs have been shown to be tractable for admissible-based semantics in [Dunne 07]. We show that on bipartite AFs the credulously (skeptically) accepted arguments w.r.t. $cf_2$ are exactly those credulously (skeptically) accepted w.r.t. stable semantics.

**Theorem**

*For bipartite AFs the problems $\text{Cred}_{cf_2}$ and $\text{Skept}_{cf_2}$ are in $P$.***

Moreover on bipartite AFs $stage_2$ and stable semantics coincides.

**Theorem**

*For bipartite AFs the problems $\text{Cred}_{stage_2}$ and $\text{Skept}_{stage_2}$ are in $P$.***
Symmetric AFs

In symmetric AFs there are no attacks between different strongly connected components. Hence,

- $cf_2$ coincides with naive semantics and;
- $stage_2$ coincides with stage semantics.

**Theorem**

*For symmetric AFs the problems $Cred_{cf_2}$ and $Skept_{cf_2}$ are in $P$.***

**Theorem**

*For symmetric, irreflexive AFs the problems $Cred_{stage_2}$ and $Skept_{stage_2}$ are in $P$.***

But for symmetric AFs with self-attacks the problems $Cred_{stage_2}$ and $Skept_{stage_2}$ are still $\Sigma_2^K / \Pi_2^K$ complete.
Fixed-Parameter Tractability

- Fixed parameter tractability w.r.t. parameters **Tree-Width / Clique-Width** follows from MSO\textsubscript{1} characterizations in [Dvořák, Szeider, Woltran 12].
Fixed-Parameter Tractability

- Fixed parameter tractability w.r.t. parameters Tree-Width / Clique-Width follows from MSO$_1$ characterizations in [Dvořák, Szeider, Woltran 12].

- **Backdoor approach** [Ordyniak & Szeider 11]:
  - Negative Results for stage semantics extends to stage2 semantics.
  - Negative result for bipartite AFs extends also to cf2 semantics.
  - Open: backdoors for cf2 and acyclic or symmetric AFs.
A Labelling based Algorithm

Require: $AF F = (A, R)$, labeling $L = (L_{in}, L_{out}, L_{undec})$;
Ensure: Return all cf2 labelings of $F$;

$X = \{a \in L_{undec} \mid att(a) \subseteq L_{out}\}$;
$Y = \{a \in L_{undec} \mid \exists b \in L_{in}, (b, a) \in R, a \not\Rightarrow^A_{\setminus L_{out}} b\}$;

while $(X \cup Y) \neq \emptyset$ do

$L_{in} = L_{in} \cup X, L_{out} = L_{out} \cup Y, L_{undec} = L_{undec} \setminus (X \cup Y)$;
update $X$ and $Y$;

end while

$B = \{a \in L_{undec} \mid L_{in} \cup \{a\} \in cf(F)\}$;
if $B \neq \emptyset$ then

$C = \{a \in B \mid \forall b \in B : b \Rightarrow^A_{\setminus L_{out}} a, a \not\Rightarrow^A_{\setminus L_{out}} b\}$;
$E = \emptyset$;
for all $L' \in naive_L(F|C)$ do

update $L$ with $L'$;
$E = E \cup cf2_L(F, L)$;
end for
return $E$;
else

return $\{(L_{in}, L_{out}, L_{undec})\}$;
end if
Conclusion

Summary

- **Complexity analysis** of reasoning with \textit{cf2} and \textit{stage2} semantics:
  - consider four graph classes for being tractable fragments;
  - discuss possibilities for fixed-parameter tractability.

- **Discussion of implementation methods**:
  - provide a labelling based algorithm.
Conclusion

Summary

- **Complexity analysis** of reasoning with cf2 and stage2 semantics:
  - consider four graph classes for being tractable fragments;
  - discuss possibilities for fixed-parameter tractability.

- Discussion of **implementation methods**:
  - provide a labelling based algorithm.

Future Work

- Considering characterizations via the equational approach for further tractable fragments.