# Reasoning in Argumentation Frameworks of Bounded Clique-Width[◇]

COMMA'2010 (Desenzano del Garda, Italy)

*Wolfgang Dvořák*, Stefan Szeider, Stefan Woltran

Database and Artificial Intelligence Group
Institut für Informationssysteme
Technische Universität Wien

September 08, 2010


[◇] Dvořák's and Woltran's work was supported by the WWTF under grant ICT08-028 and Szeider's work was supported by the European Research Council, grant reference 239962.

# Motivation

## Fixed-Parameter Tractability

- Many argumentation reasoning tasks are computationally intractable.
- Often computational costs primarily depend on some problem parameters rather than on the mere size of the instances.
- Many hard problems become tractable if some problem parameter is fixed or bounded by a fixed constant.
- FPT results (in terms of treewidth) for argumentation already exist. [Dunne, 2007; Dvořák, Pichler and Woltran, 2010]
- In the arena of graphs an important parameter is clique-width, which generalizes the parameter of tree-width.

# Main Contribution

We present data-structures and algorithms for efficient reasoning in abstract Argumentation Frameworks (AFs) of bounded clique-width.

# Main Contribution

We present data-structures and algorithms for efficient reasoning in abstract Argumentation Frameworks (AFs) of bounded clique-width.

**Features of clique-width:**

- The class of AFs with bounded clique-width subsumes and significantly extends the fragment of AFs with bounded tree-width.
- There are both, sparse (e.g. tree-like AFs) and dense AFs (e.g. clique-like AFs) that possess small clique-width.
- Clique-width incorporates the orientation of attacks.
- Clique-width offers an efficient handling for modular structures.

# Clique-Width

## Definition

A $k$-AF is an AF whose arguments are labeled by integers from $\{1 \dots k\}$.

## Operations

We allow the following operations on labeled AFs:

- Vertex introduction: vertex $v$ labeled by $i$ (denoted by $i(v)$)
- Disjoint union (denoted by $\oplus$);
- Relabeling: changing all labels $i$ to $j$ (denoted by $\rho_{i \to j}$);
- Edge insertion: connecting all arguments labeled by $i$ with all arguments labeled by $j$ (denoted by $\eta_{i,j}$).

These operations treat equally labeled arguments in the same way.

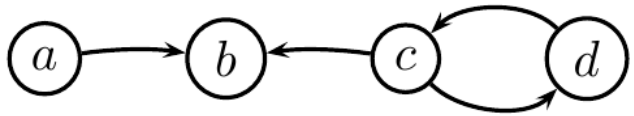
# Clique-Width

An AF can be represented by an algebraic term composed of $i(v)$, $\oplus$, $\rho_{i\to j}$, and $\eta_{i,j}$. If this construction only uses labels in $\{1,\ldots,k\}$ we call this term a $k$-expression.

# Clique-Width

An AF can be represented by an algebraic term composed of $i(v)$, $\oplus$, $\rho_{i\to j}$, and $\eta_{i,j}$. If this construction only uses labels in $\{1,\ldots,k\}$ we call this term a $k$-expression.

The AF:



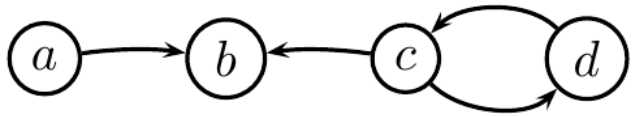

can be constructed by the 3-expression:

$$\eta_{1,2}(\rho_{1\to 3}(\eta_{1,2}(1(a)\oplus 2(b)))\oplus \eta_{2,1}(1(c)\oplus 2(d)))$$

# Clique-Width

An AF can be represented by an algebraic term composed of $i(v)$, $\oplus$, $\rho_{i\to j}$, and $\eta_{i,j}$. If this construction only uses labels in $\{1,\ldots,k\}$ we call this term a $k$-expression.

The AF:




can be constructed by the 3-expression:

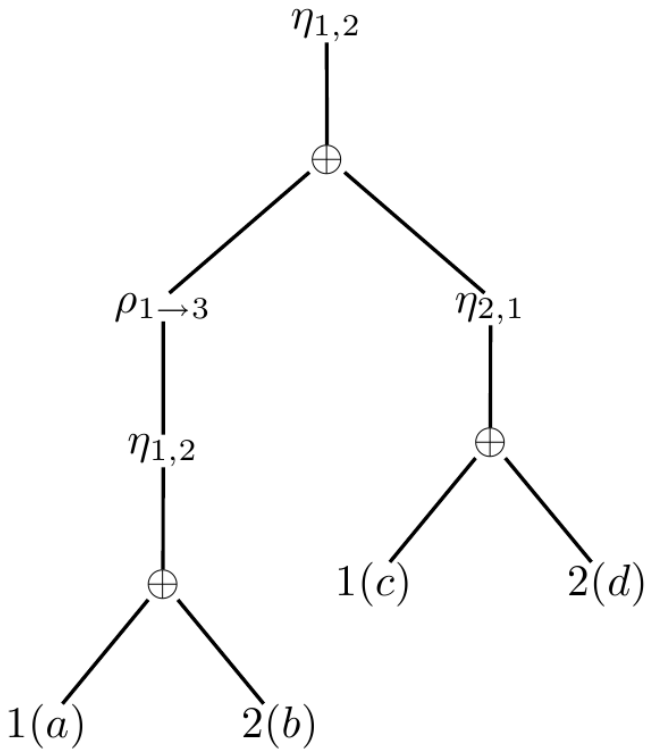
$$\eta_{1,2}(\rho_{1\to 3}(\eta_{1,2}(1(a)\oplus 2(b)))\oplus \eta_{2,1}(1(c)\oplus 2(d)))$$

## Definition

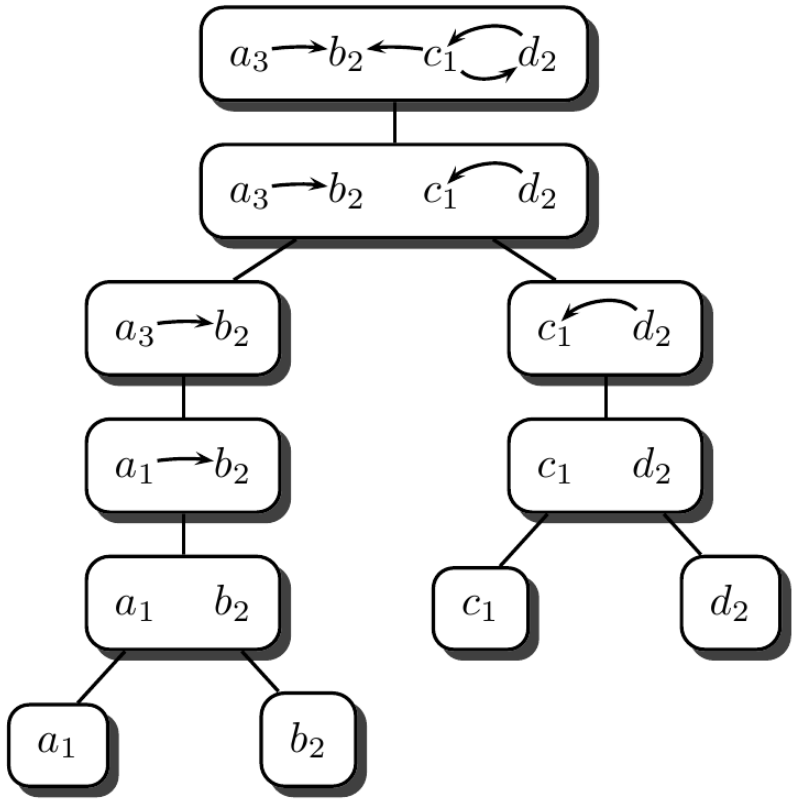
The clique-width of an AF $\mathcal{F}$, $\mathrm{cwd}(\mathcal{F})$, is the smallest integer $k$ such that $\mathcal{F}$ can be defined by a $k$-expression.

# Parse-Tree

3 - expression: $\eta_{1,2}(\rho_{1\to 3}(\eta_{1,2}(1(a) \oplus 2(b))) \oplus \eta_{2,1}(1(c) \oplus 2(d)))$



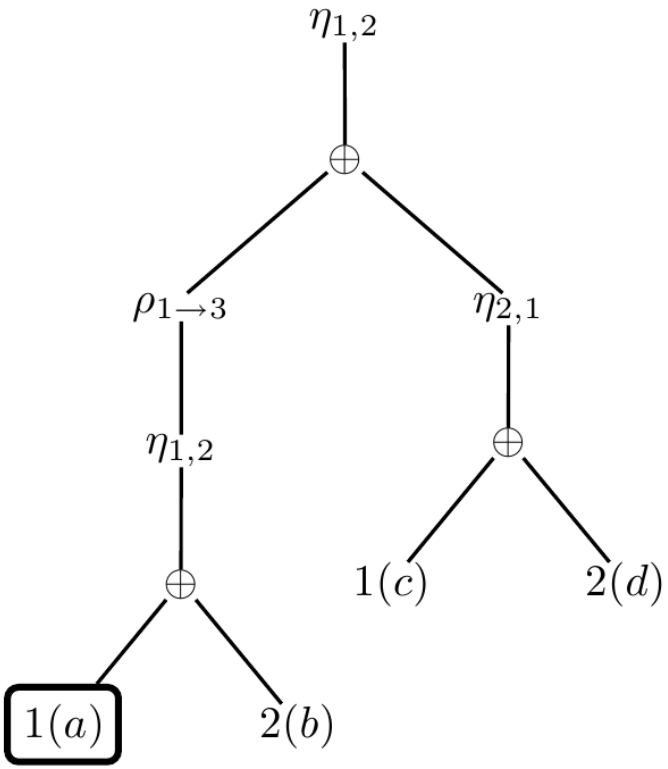

parse-tree



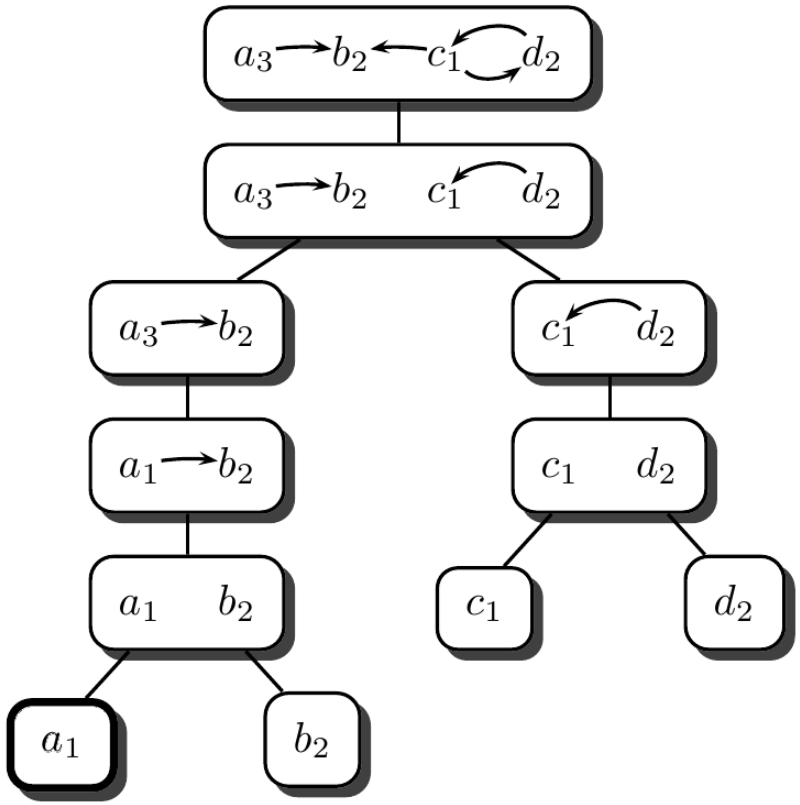

associated AFs

# Parse-Tree

3 - expression: $\eta_{1,2}(\rho_{1\to 3}(\eta_{1,2}(1(a) \oplus 2(b))) \oplus \eta_{2,1}(1(c) \oplus 2(d)))$



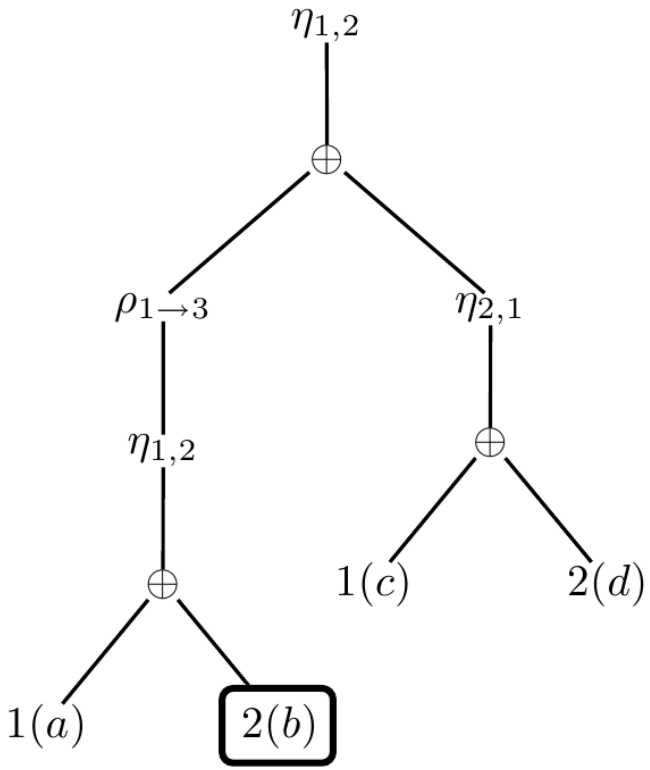

parse-tree



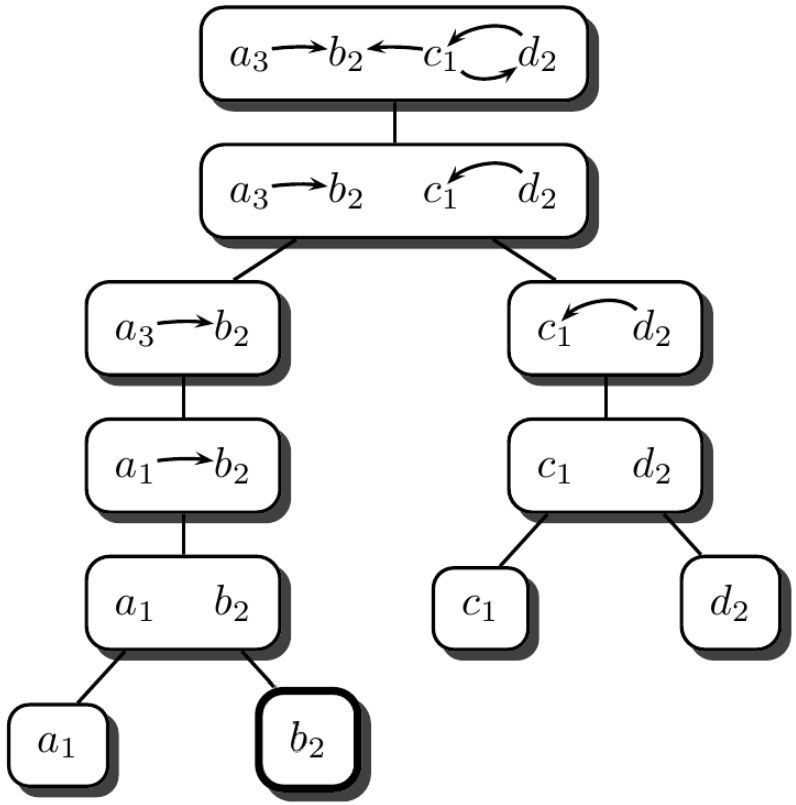

associated AFs

# Parse-Tree

3 - expression: $\eta_{1,2}(\rho_{1\to 3}(\eta_{1,2}(1(a) \oplus 2(b))) \oplus \eta_{2,1}(1(c) \oplus 2(d)))$



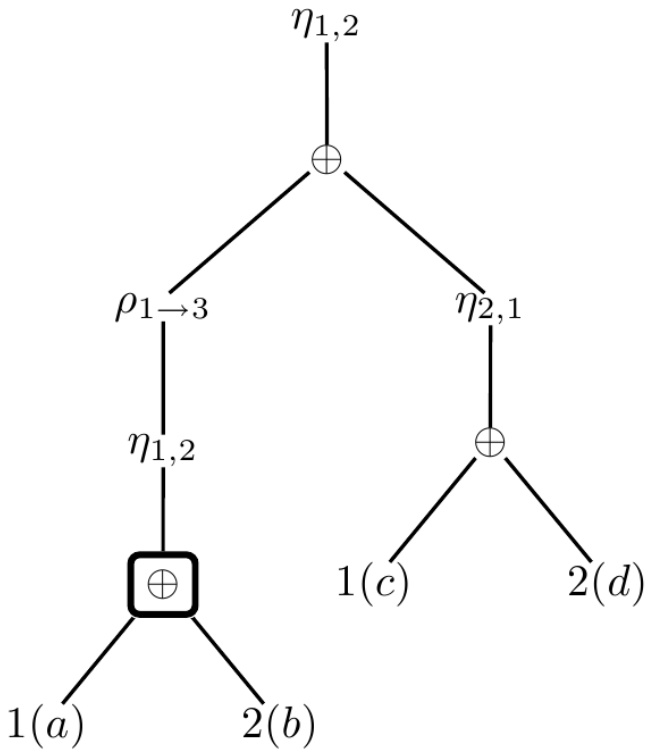

parse-tree



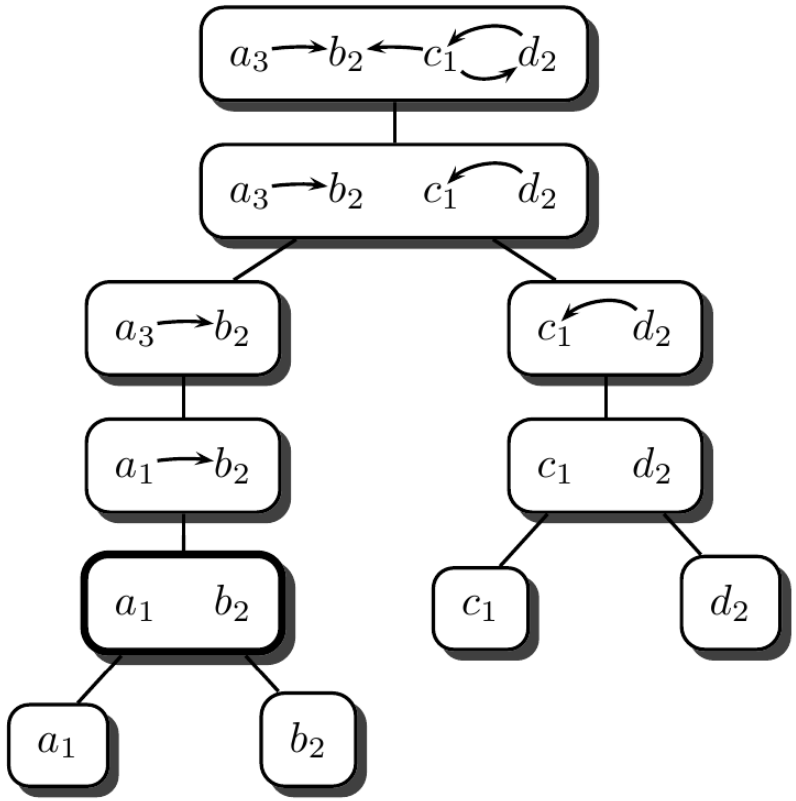

associated AFs

# Parse-Tree

3 - expression: $\eta_{1,2}(\rho_{1\to 3}(\eta_{1,2}(1(a) \oplus 2(b))) \oplus \eta_{2,1}(1(c) \oplus 2(d)))$



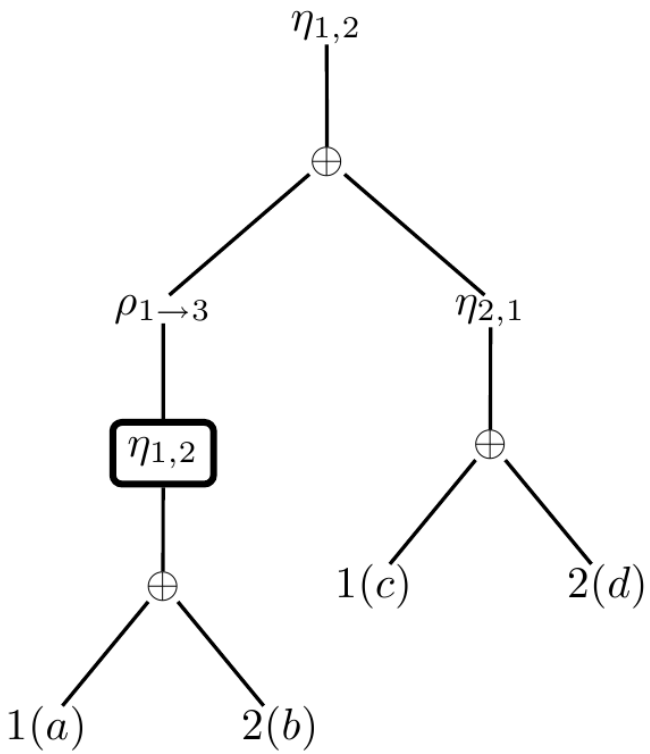

parse-tree



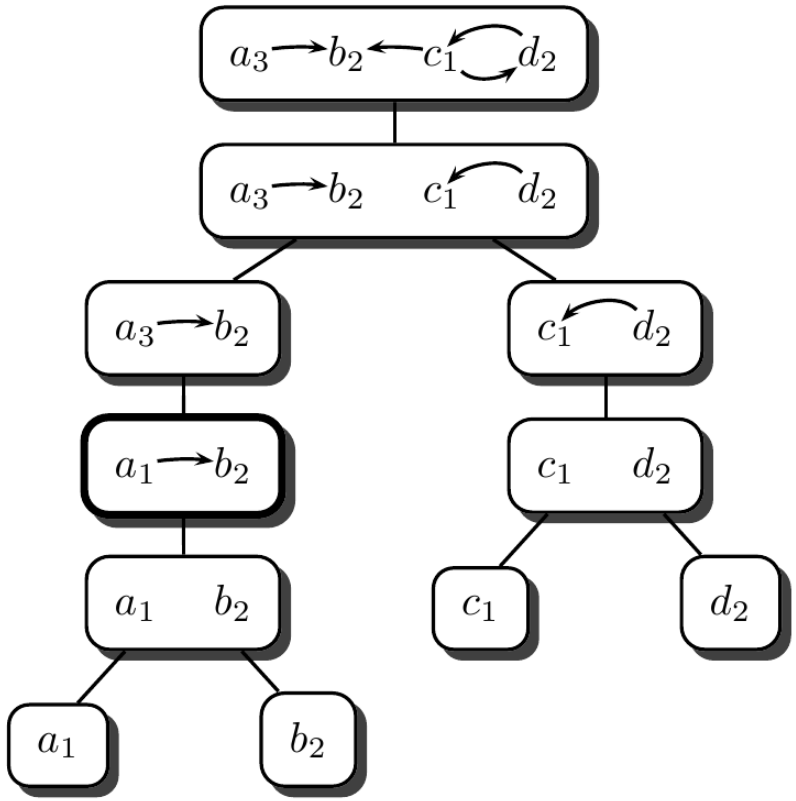

associated AFs

# Parse-Tree

3 - expression: $\eta_{1,2}(\rho_{1\to 3}(\eta_{1,2}(1(a) \oplus 2(b))) \oplus \eta_{2,1}(1(c) \oplus 2(d)))$



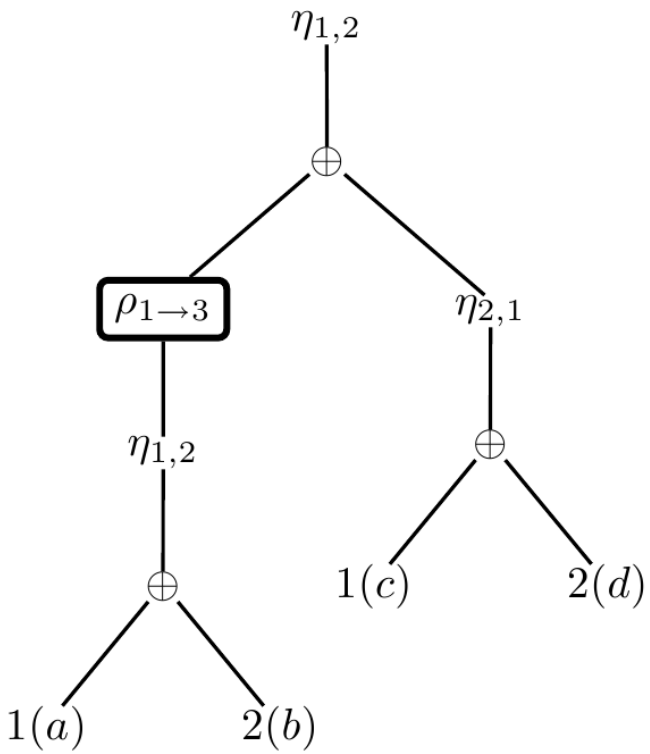

parse-tree



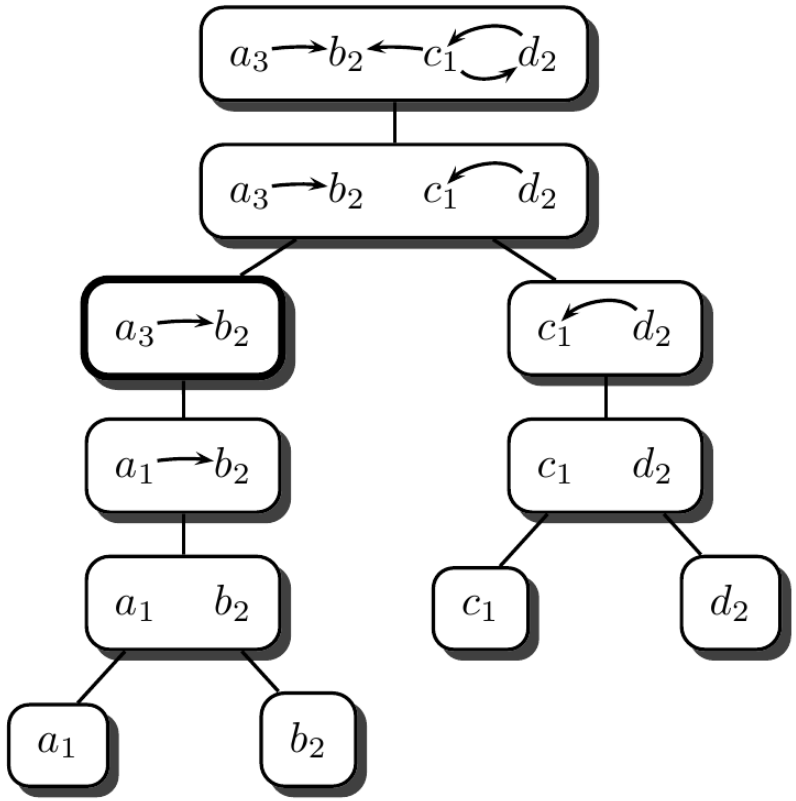

associated AFs

# Parse-Tree

3 - expression: $\eta_{1,2}(\rho_{1\to 3}(\eta_{1,2}(1(a) \oplus 2(b))) \oplus \eta_{2,1}(1(c) \oplus 2(d)))$



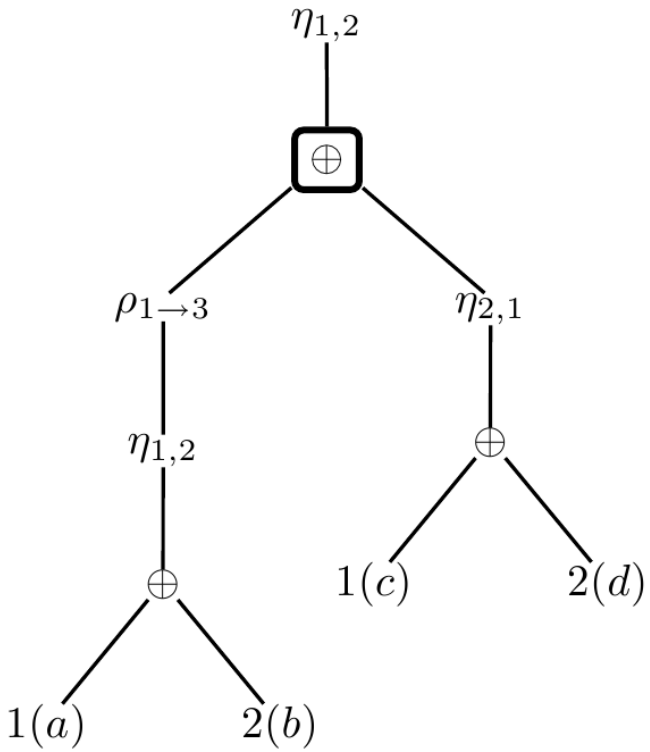

parse-tree



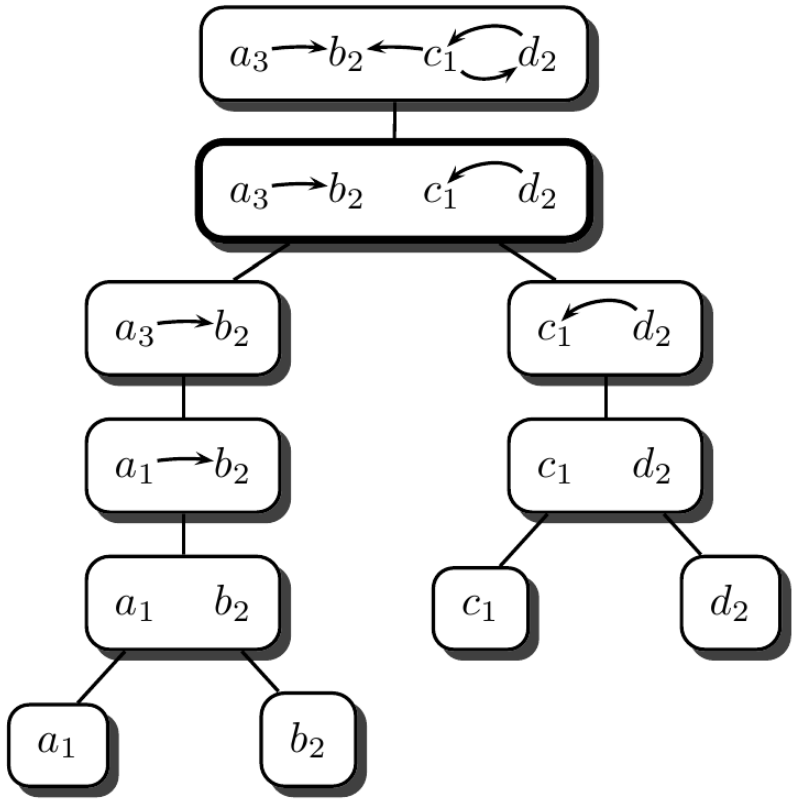

associated AFs

# Parse-Tree

3 - expression: $\eta_{1,2}(\rho_{1\to 3}(\eta_{1,2}(1(a) \oplus 2(b))) \oplus \eta_{2,1}(1(c) \oplus 2(d)))$



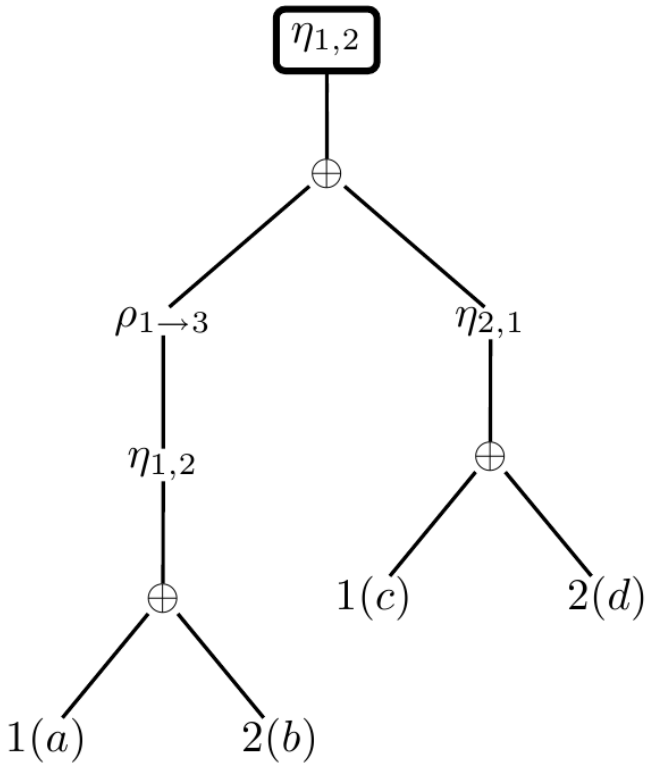

parse-tree



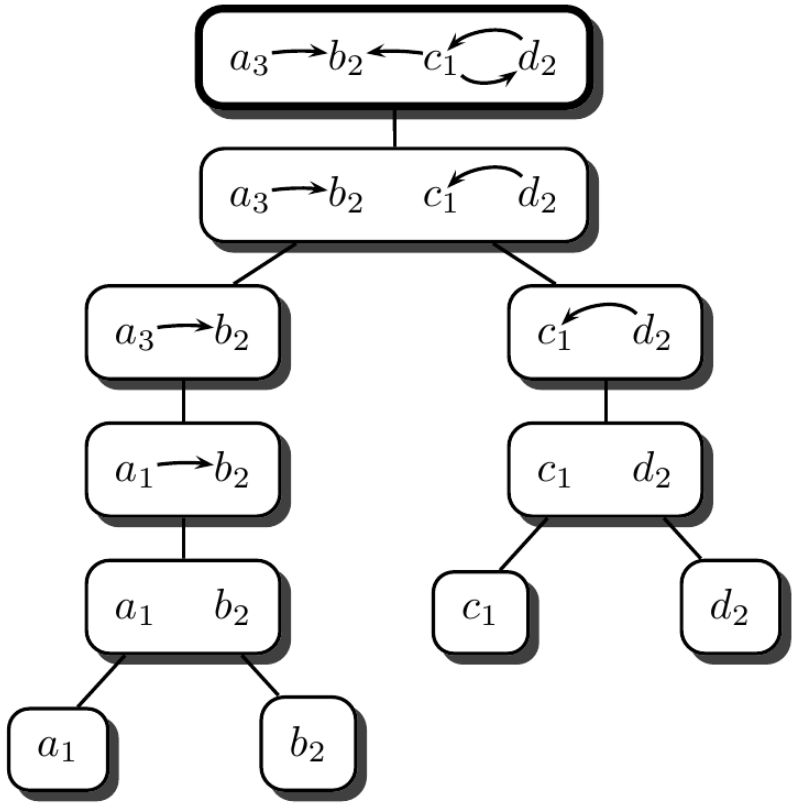

associated AFs

# Parse-Tree

3 - expression: $\eta_{1,2}(\rho_{1\to 3}(\eta_{1,2}(1(a) \oplus 2(b))) \oplus \eta_{2,1}(1(c) \oplus 2(d)))$




parse-tree



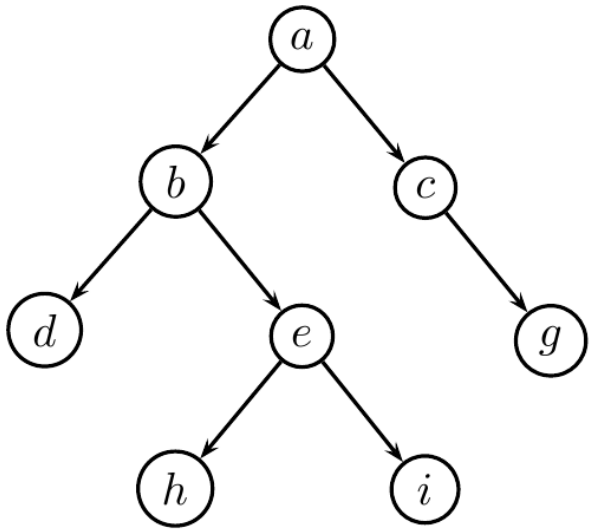

associated AFs

# AFs with Bounded Clique-Width

- trees/forests (clique-width 3)
- graphs of bounded tree-width: including trees, forests, series parallel networks, outer-planar graphs, Halin graphs, ...
- co-graphs (clique-width 2) including complete graphs, complete bipartite graphs, Threshold graphs, Turán graphs, ...
- transitive tournaments (clique-width 2)

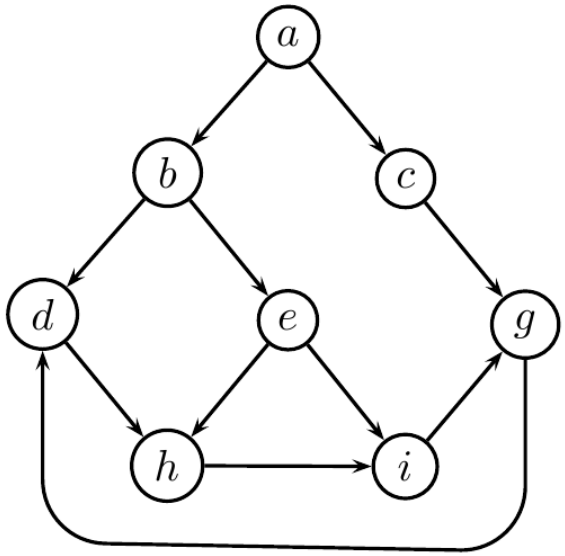
# AFs with Bounded Clique-Width

- **trees**/forests (clique-width 3)
- graphs of bounded tree-width: including trees, forests, series parallel networks, outer-planar graphs, Halin graphs, ...
- co-graphs (clique-width 2) including complete graphs, complete bipartite graphs, Threshold graphs, Turán graphs, ...
- transitive tournaments (clique-width 2)



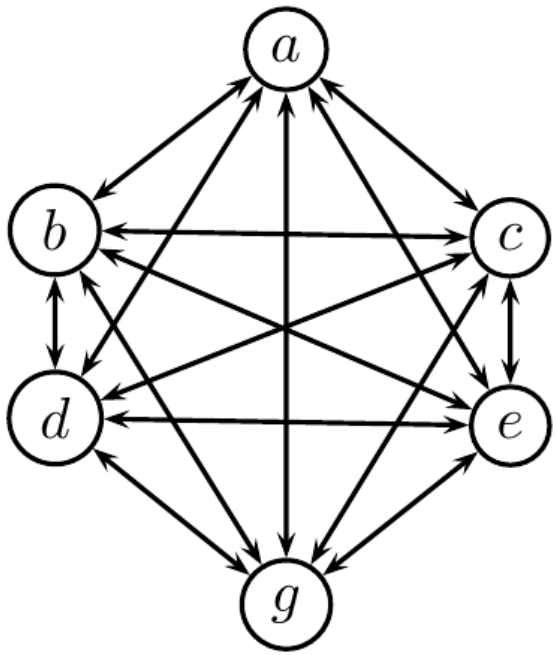
# AFs with Bounded Clique-Width

- trees/forests (clique-width 3)
- graphs of bounded **tree-width**: including trees, forests, series parallel networks, outer-planar graphs, Halin graphs, ...
- co-graphs (clique-width 2) including complete graphs, complete bipartite graphs, Threshold graphs, Turán graphs, ...
- transitive tournaments (clique-width 2)



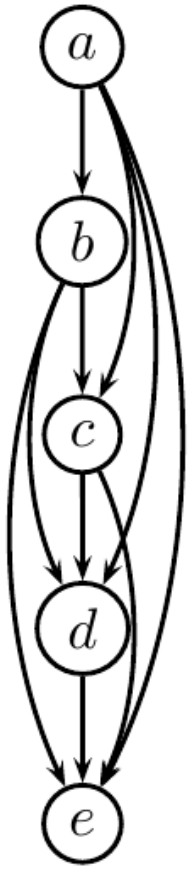
# AFs with Bounded Clique-Width

- trees/forests (clique-width 3)
- graphs of bounded tree-width: including trees, forests, series parallel networks, outer-planar graphs, Halin graphs, ...
- **co-graphs** (clique-width 2) including complete graphs, complete bipartite graphs, Threshold graphs, Turán graphs, ...
- transitive tournaments (clique-width 2)



# AFs with Bounded Clique-Width

- trees/forests (clique-width 3)
- graphs of bounded tree-width: including trees, forests, series parallel networks, outer-planar graphs, Halin graphs, ...
- co-graphs (clique-width 2) including complete graphs, complete bipartite graphs, Threshold graphs, Turán graphs, ...
- **transitive tournaments** (clique-width 2)



# Reasoning

## Credulous Acceptance

Given an AF $F = (A, R)$ and an argument $x \in A$.
Is $x$ in at least one preferred extension ?

For credulous acceptance it suffices to consider admissible extensions.

## Skeptical Acceptance

Given an AF $F = (A, R)$ and an argument $x \in A$.
Is $x$ in every preferred extension ?

# Reasoning

## Credulous Acceptance

Given an AF $F = (A, R)$ and an argument $x \in A$.
Is $x$ in at least one preferred extension ?

For credulous acceptance it suffices to consider admissible extensions.

## Skeptical Acceptance

Given an AF $F = (A, R)$ and an argument $x \in A$.
Is $x$ in every preferred extension ?

**Complexity**:

- The credulous acceptance problem is NP-complete (Dimopoulos and Torres, 1996).
- The skeptical acceptance problem is $\Pi_2^p$-complete (Dunne and Bench-Capon, 2002).

# Fixed-Parameter Tractability

## Theorem (Courcelle, Makowsky, Rotics 2000)

*Any graph problem that can be expressed in Monadic Second Order Logic with second-order quantification on vertex sets ($MSO_1$) can be solved in linear time for graphs of clique-width bounded by some constant $k$.*

# Fixed-Parameter Tractability

## Theorem (Courcelle, Makowsky, Rotics 2000)

*Any graph problem that can be expressed in Monadic Second Order Logic with second-order quantification on vertex sets ($MSO_1$) can be solved in linear time for graphs of clique-width bounded by some constant $k$.*

[Dunne 2007] already provided $MSO_1$ characterizations to show FPT for tree-width.

## Theorem

*For AFs of clique-width bounded by a constant, credulous and skeptical acceptance are decidable in linear time.*

# Fixed-Parameter Tractability

## Theorem (Courcelle, Makowsky, Rotics 2000)

*Any graph problem that can be expressed in Monadic Second Order Logic with second-order quantification on vertex sets ($MSO_1$) can be solved in linear time for graphs of clique-width bounded by some constant $k$.*

[Dunne 2007] already provided $MSO_1$ characterizations to show FPT for tree-width.

## Theorem

*For AFs of clique-width bounded by a constant, credulous and skeptical acceptance are decidable in linear time.*

But the theorem doesn't lead us to efficient algorithms.

# Dynamic Programming

**Basic Ideas:**

Given: $AF_\sigma$ and the corresponding $k$-expression $\sigma$

- Traverse the parse-tree of $\sigma$ with a bottom up algorithm
- Compute the admissible sets for each node
  - Instead of computing the admissible sets, we compute a succinct representation of all admissible sets, via tables
  - We use the tables of the successors to compute the table for the current node.
  - To decide credulous acceptance we mark rows representing extensions containing the specified argument.
- The results for the entire problem can be read of the root

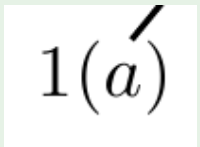
# DP-Example

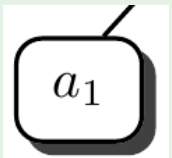
**Problem:** We want to decide the credulous acceptance of argument $b$ in our Example AF.

## Leaf-Node i(v)

We have two conflict-free sets, i.e. $\{v\}$ and $\emptyset$.

## Example




## Table 1($a$)

| *in* | *att* | *out* | *def* | |
|---|---|---|---|---|
| 1 | - | - | - | $\{a\}$ |
| - | - | 1 | - | $\emptyset$ |

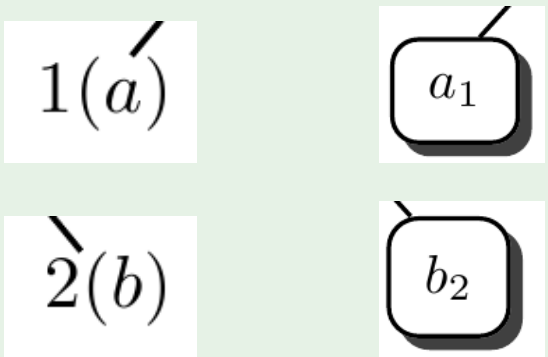
# DP-Example

**Problem:** We want to decide the credulous acceptance of argument $b$ in our Example AF.

## Leaf-Node i(v)

We have two conflict-free sets, i.e. $\{v\}$ and $\emptyset$.

## Example




## Table 1($a$)

| *in* | *att* | *out* | *def* | |
|---|---|---|---|---|
| 1 | - | - | - | $\{a\}$ |
| - | - | 1 | - | $\emptyset$ |

## Table 2($b$)

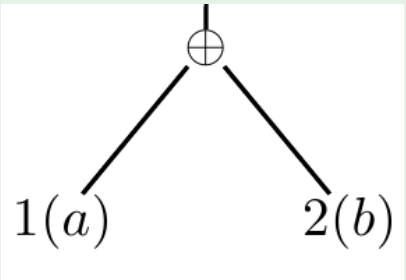
| *in* | *att* | *out* | *def* | |
|---|---|---|---|---|
| 2 | - | - | - | $\{b\}$ |
| - | - | 2 | - | $\emptyset$ |

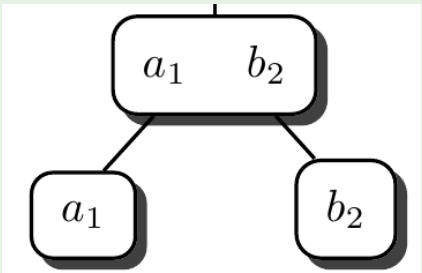
# DP-Example

## Union-Node $\oplus$

We combine the tables of the successors to get the table of the $\oplus$-node.

## Example




## Table 1($a$) $\oplus$ 2($b$)

| *in* | *att* | *out* | *def* | |
|---|---|---|---|---|
| 1,2 | - | - | - | $\{a, b\}$ |
| 1 | - | 2 | - | $\{a\}$ |
| 2 | - | 1 | - | $\{b\}$ |
| - | - | 1,2 | - | $\emptyset$ |

## Table 1($a$)

| *in* | *att* | *out* | *def* | |
|---|---|---|---|---|
| 1 | - | - | - | $\{a\}$ |
| - | - | 1 | - | $\emptyset$ |

## Table 2($b$)

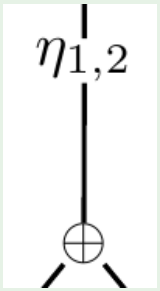
| *in* | *att* | *out* | *def* | |
|---|---|---|---|---|
| 2 | - | - | - | $\{b\}$ |
| - | - | 2 | - | $\emptyset$ |

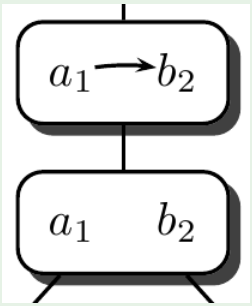
# DP-Example

## Edge Insertion-Node $\eta_{i,j}$

Eliminate rows with $\{i, j\} \subseteq in$.
Update sets $att, out, def$

## Example




## $\eta_{1,2}(\ 1(a) \oplus 2(b)\ )$

| $in$ | $att$ | $out$ | $def$ | |
|---|---|---|---|---|
| 1 | - | - | 2 | $\{a\}$ |
| 2 | 1 | - | - | $\{b\}$ |
| - | - | 1,2 | - | $\emptyset$ |

## $1(a) \oplus 2(b)$

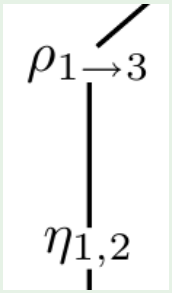
| $in$ | $att$ | $out$ | $def$ | |
|---|---|---|---|---|
| 1,2 | - | - | - | $\{a, b\}$ |
| 1 | - | 2 | - | $\{a\}$ |
| 2 | - | 1 | - | $\{b\}$ |
| - | - | 1,2 | - | $\emptyset$ |

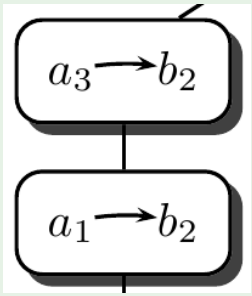
# DP-Example

## Relabeling-Node $\rho_{i\to j}$

We update the sets *in*, *att*, *out*, *def* of according to the renaming.

## Example




## $\rho_{1\to 3}(\ \eta_{1,2}(1(a) \oplus 2(b))\ )$

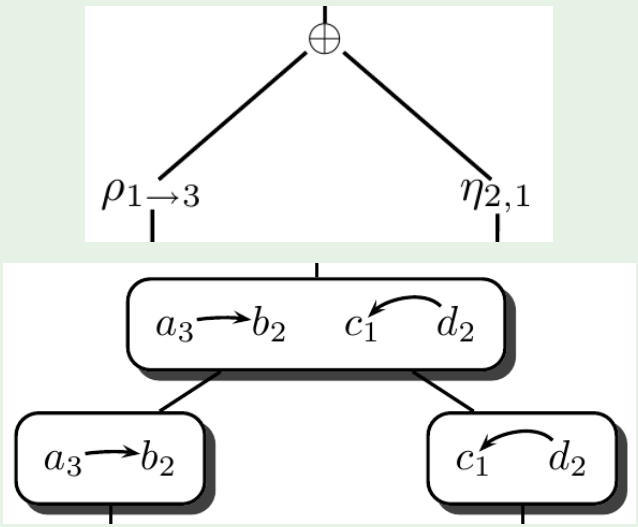
| *in* | *att* | *out* | *def* | |
|---|---|---|---|---|
| 3 | - | - | 2 | $\{a\}$ |
| 2 | 3 | - | - | $\{b\}$ |
| - | - | 3,2 | - | $\emptyset$ |

## $\eta_{1,2}(1(a) \oplus 2(b))$

| *in* | *att* | *out* | *def* | |
|---|---|---|---|---|
| 1 | - | - | 2 | $\{a\}$ |
| 2 | 1 | - | - | $\{b\}$ |
| - | - | 1,2 | - | $\emptyset$ |

# DP-Example

## Example




## $\rho_{1\to 3}(\dots) \oplus \eta_{1,2}(\dots)$

| *in* | *att* | *out* | *def* | |
|---|---|---|---|---|
| 1,3 | 2 | - | - | $\{a, c\}$ |
| 2,3 | - | - | 1,2 | $\{a, d\}$ |
| 3 | - | 1,2 | - | $\{a\}$ |
| 1,2 | 2,3 | - | - | $\{b, c\}$ |
| 2 | 3 | - | 1 | $\{b, d\}$ |
| 2 | 3 | 1,2 | - | $\{b\}$ |
| ⋮ | ⋮ | ⋮ | ⋮ | |

## $\rho_{1\to 3}(\ \eta_{1,2}(1(a) \oplus 2(b))\ )$

| *in* | *att* | *out* | *def* | |
|---|---|---|---|---|
| 3 | - | - | 2 | $\{a\}$ |
| 2 | 3 | - | - | $\{b\}$ |
| - | - | 3,2 | - | $\emptyset$ |

## $\eta_{1,2}(1(c) \oplus 2(d))$

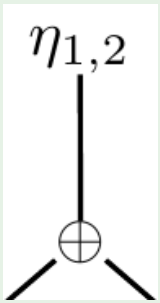
| *in* | *att* | *out* | *def* | |
|---|---|---|---|---|
| 1 | 2 | - | - | $\{c\}$ |
| 2 | - | - | 1 | $\{d\}$ |
| - | - | 1,2 | - | $\emptyset$ |

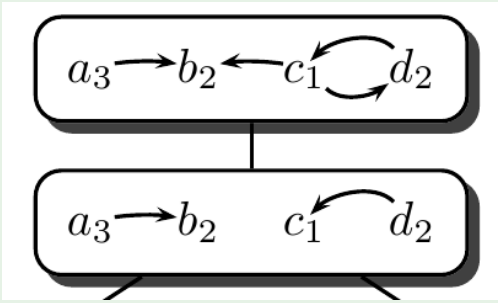
# DP-Example

## Root-Node

$att = \emptyset \Leftrightarrow$ admissible sets.
marked row with $att = \emptyset$ if and only if $a$ is credulous accepted

## Example




$\eta_{1,2}(\dots)$

| *in* | *att* | *out* | *def* | |
|---|---|---|---|---|
| 1,3 | - | - | 2 | $\{a, c\}$ |
| 2,3 | - | - | 1,2 | $\{a, d\}$ |
| 3 | - | 1,2 | - | $\{a\}$ |
| 2 | 3 | - | 1 | $\{b, d\}$ |
| 2 | 1,3 | 2 | - | $\{b\}$ |
| ⋮ | ⋮ | ⋮ | ⋮ | |

| *in* | *att* | *out* | *def* | |
|---|---|---|---|---|
| 1,3 | 2 | - | - | $\{a, c\}$ |
| 2,3 | - | - | 1,2 | $\{a, d\}$ |
| 3 | - | 1,2 | - | $\{a\}$ |
| 1,2 | 2,3 | - | - | $\{b, c\}$ |
| 2 | 3 | - | 1 | $\{b, d\}$ |
| 2 | 3 | 1,2 | - | $\{b\}$ |
| ⋮ | ⋮ | ⋮ | ⋮ | |

# Complexity

## Complexity

Given an AF $\mathcal{F}_\sigma$ with a $k$-expression $\sigma$ and an argument $x$, our algorithm decides if $x$ is credulously accepted in time $O(f(k) \cdot |AF|)$.

# Complexity

## Complexity

Given an AF $\mathcal{F}_\sigma$ with a $k$-expression $\sigma$ and an argument $x$, our algorithm decides if $x$ is credulously accepted in time $O(f(k) \cdot |AF|)$.

## Skeptical Acceptance

By extending our data-structure to characterize preferred extensions we get a similar algorithm for skeptical reasoning.

# Conclusion

**Main Contributions** of the paper:

- We identified AFs of bounded clique-width as new tractable fragment for abstract argumentation.
- Fixed-parameter tractable algorithms for reasoning in AFs of bounded clique-width.
  - Credulous / Skeptical Reasoning w.r.t. preferred semantics.
  - These algorithms can be extended for Computing extensions (with linear delay) and Counting extensions.
- The techniques presented for preferred semantics are prototypical, i.e. can be easily applied to several other semantics.

# Conclusion

**Main Contributions** of the paper:

- We identified AFs of bounded clique-width as new tractable fragment for abstract argumentation.
- Fixed-parameter tractable algorithms for reasoning in AFs of bounded clique-width.
  - Credulous / Skeptical Reasoning w.r.t. preferred semantics.
  - These algorithms can be extended for Computing extensions (with linear delay) and Counting extensions.
- The techniques presented for preferred semantics are prototypical, i.e. can be easily applied to several other semantics.

**Future and Ongoing Work:**

- Implementation of these algorithms.
- Identifying further tractable fragments.

# Modularity

## Definition

Consider an AF $\mathcal{F} = (A, R)$. A subset $M \subseteq A$ is a module of $\mathcal{F}$ if any two arguments in $M$ are "indistinguishable from the outside" i.e., for any $x, x' \in M$ and $y \in A \setminus M$ we have that $(x, y) \in R$ iff $(x', y) \in R$, and $(y, x) \in R$ iff $(y, x') \in R$.

**Example**: A Group of agents that share the same beliefs and opinions regarding the world outside the group (but possibly attack each other for some "internal" reason).

Efficient handle of modules via clique-width:

- Find a $k$-expression for the subframework induced by $M$.
- Give all arguments in $M$ the same label and treat them for subsequent considerations as one single argument.