Argumentation with Bounded Tree-Width[◇]

Seminar aus Theoretischer Informatik

Wolfgang Dvořák, Reinhard Pichler, Stefan Woltran

Database and Artificial Intelligence Group
Institut für Informationssysteme
Technische Universität Wien

January 26, 2010

[◇]This work was supported by the Vienna Science and Technology Fund (WWTF) under grant ICT08-028

Outline

Decision Problems

Credulous Acceptance

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

x is in at least one preferred extension $\Leftrightarrow$ x is in at least one admissible extension.

Decision Problems

Credulous Acceptance

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

x is in at least one preferred extension $\Leftrightarrow$ x is in at least one admissible extension.

Skeptical Acceptance

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

Decision Problems

Credulous Acceptance

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

x is in at least one preferred extension $\Leftrightarrow$ x is in at least one admissible extension.

Skeptical Acceptance

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

- The credulous acceptance problem is NP-complete ([Dimopoulos and Torres(1996)]).
- The skeptical acceptance problem is Σ_2^p-complete ([Dunne and Bench-Capon(2002)]).

Theoretic Tractability

We can express the properties of admissible and preferred extensions in MSOL ([Dunne(2007)]):

$$
\begin{aligned}
\mathrm{cf}_R(U) &= \forall x, y\big(\langle x, y\rangle \in R \rightarrow (\neg x \in U \vee \neg y \in U)\big) \\
\mathrm{adm}_R(U) &= \mathrm{cf}_R(U) \wedge \forall x, y\Big((\langle x, y\rangle \in R \wedge y \in U) \rightarrow \\
&\qquad\qquad \exists z(z \in U \wedge \langle z, x\rangle \in R)\Big) \\
\mathrm{pref}_{(A,R)}(U) &= \mathrm{adm}_R(U) \wedge \neg\exists V \subseteq A : \mathrm{adm}_R(V) \wedge U \subset V
\end{aligned}
$$

The required checks for the considered decision problems can easily be added to these formulas.

Theoretic Tractability

We can express the properties of admissible and preferred extensions in MSOL ([Dunne(2007)]):

$$
\begin{aligned}
\mathrm{cf}_R(U) &= \forall x, y\big(\langle x, y\rangle \in R \rightarrow (\neg x \in U \vee \neg y \in U)\big)\\
\mathrm{adm}_R(U) &= \mathrm{cf}_R(U) \wedge \forall x, y\Big((\langle x, y\rangle \in R \wedge y \in U) \rightarrow\\
&\qquad\qquad\qquad \exists z(z \in U \wedge \langle z, x\rangle \in R)\Big)\\
\mathrm{pref}_{(A,R)}(U) &= \mathrm{adm}_R(U) \wedge \neg\exists V \subseteq A : \mathrm{adm}_R(V) \wedge U \subset V
\end{aligned}
$$

The required checks for the considered decision problems can easily be added to these formulas.

Thus by Courcelles theorem we can decide our problems in linear time on argumentation frameworks of bounded tree-width.

Tree-Decomposition

Definition

Let $G = (V, E)$ be an undirected graph.
A *tree decomposition* of G is a pair $(\mathcal{T}, \mathcal{X})$ where $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ is a tree and $\mathcal{X} = (X_t)_{t \in V_\mathcal{T}}$ is a set of so-called bags, which has to satisfy the following conditions:

1. $\bigcup_{t \in V_\mathcal{T}} X_t = V$, i.e. $\mathcal{X}$ is a cover of V;
2. For each $v \in V$, $\mathcal{T}|_{\{t | v \in X_t\}}$ is connected;
3. For each $\{v_i, v_j\} \in E$, $\{v_i, v_j\} \subseteq X_t$ for some $t \in V_\mathcal{T}$.

Tree-Decomposition

Definition

Let $G = (V, E)$ be an undirected graph.
A *tree decomposition* of G is a pair $(\mathcal{T}, \mathcal{X})$ where $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ is a tree and $\mathcal{X} = (X_t)_{t \in V_\mathcal{T}}$ is a set of so-called bags, which has to satisfy the following conditions:

1. $\bigcup_{t \in V_\mathcal{T}} X_t = V$, i.e. $\mathcal{X}$ is a cover of V;
2. For each $v \in V$, $\mathcal{T}|_{\{t \mid v \in X_t\}}$ is connected;
3. For each $\{v_i, v_j\} \in E$, $\{v_i, v_j\} \subseteq X_t$ for some $t \in V_\mathcal{T}$.

The width of such a tree decomposition is given by $\max\{|X_t| \mid t \in V_\mathcal{T}\} - 1$. The *tree-width* of a graph G is the minimum width over all tree decompositions of G.

Nice Tree-Decomposition

Definition ([Kloks(1994)])

A tree decomposition $(\mathcal{T}, \mathcal{X})$ is called *nice* if $\mathcal{T}$ is a rooted tree and if each node $t \in \mathcal{T}$ is of one of the following types:

1. LEAF: t is a leaf of $\mathcal{T}$
2. FORGET: t has only one child t' and $X_t = X_{t'} \dot{\cup} \{v\}$
3. INSERT: t has only one child t' and $X_t \dot{\cup} \{v\} = X_{t'}$
4. JOIN: t has two children t', t'' and $X_t = X_{t'} = X_{t''}$

Additional we will assume that $X_R = \emptyset$ for the root node R.

Nice Tree-Decomposition

Definition ([Kloks(1994)])

A tree decomposition $(\mathcal{T}, \mathcal{X})$ is called *nice* if $\mathcal{T}$ is a rooted tree and if each node $t \in \mathcal{T}$ is of one of the following types:

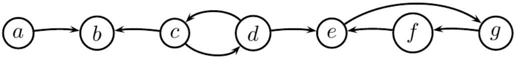
1. LEAF: t is a leaf of $\mathcal{T}$
2. FORGET: t has only one child t' and $X_t = X_{t'} \dot{\cup} \{v\}$
3. INSERT: t has only one child t' and $X_t \dot{\cup} \{v\} = X_{t'}$
4. JOIN: t has two children t', t'' and $X_t = X_{t'} = X_{t''}$

Additional we will assume that $X_R = \emptyset$ for the root node R.

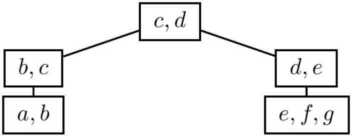
Lemma ([Kloks(1994)])

A tree decomposition $((\mathcal{T}), (\mathcal{X}))$ of a graph $\mathcal{G}$ with n nodes can be transformed in time $O(n)$ into a nice tree decomposition $((\mathcal{T})', (\mathcal{X})')$ of $\mathcal{G}$ which has the same width as $((\mathcal{T}), (\mathcal{X}))$ and where $(\mathcal{T})'$ has $O(n)$ nodes.

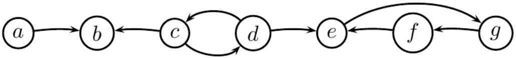
Example:



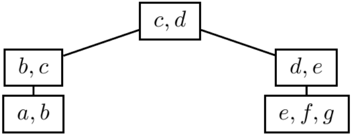
Tree-decomposition:



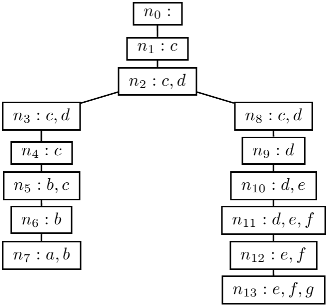
Example:



Tree-decomposition:



nice tree-decomposition:



More Definitions

Definition

Let $F = (A, R)$ be an AF and $B \subseteq A$.
A set $E \subseteq A$ is a *B-restricted admissible* set, iff E is conflict-free in F and E defends itself against all $a \in B$.

More Definitions

Definition

Let $F = (A, R)$ be an AF and $B \subseteq A$.
A set $E \subseteq A$ is a *B-restricted admissible* set, iff E is conflict-free in F and E defends itself against all $a \in B$.

- E admissible $\Leftrightarrow$ E is A-restricted admissible.
- E conflict-free $\Leftrightarrow$ E is $\emptyset$-restricted admissible.

More Definitions

Definition

Let $F = (A, R)$ be an AF and $B \subseteq A$.
A set $E \subseteq A$ is a *B-restricted admissible* set, iff E is conflict-free in F and E defends itself against all $a \in B$.

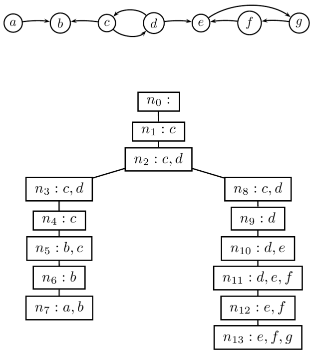
- E admissible $\Leftrightarrow$ E is A-restricted admissible.
- E conflict-free $\Leftrightarrow$ E is $\emptyset$-restricted admissible.

Some Notation

Let $(\mathcal{T}, \mathcal{X})$ be a tree-decomposition and $t \in \mathcal{T}$ then :

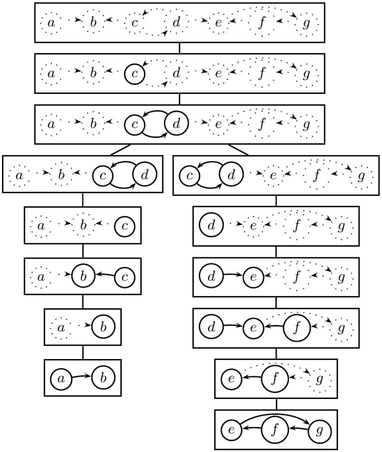
- $X_{\geq t} = \bigcup_{t' \geq t} X_{t'}$ (union over the subtree rooted in t),
- $X_{>t} = X_{\geq t} \setminus X_t$

Dynamic Programming



Argumentation Framework
& nice tree-decomposition

$\Rightarrow$



Nice tree-decomposition with
induced sub-frameworks

Dynamic Programming

Basic Ideas:

- Compute the $X_{>t}$-restricted admissible sets for each bag with a bottom-up algorithm on the tree-decomposition
  - For a bag t we only store information about nodes in X_t
  - The information about the nodes in $X_{>t}$ is implicitly encoded
- The results for the entire problem can be read of the root.

Dynamic Programming

Basic Ideas:

- Compute the $X_{>t}$-restricted admissible sets for each bag with a bottom-up algorithm on the tree-decomposition
  - For a bag t we only store information about nodes in X_t
  - The information about the nodes in $X_{>t}$ is implicitly encoded
- The results for the entire problem can be read of the root.

Bag - Colorings

A coloring for a bag is a function $C_t : X_t \rightarrow \{in, out, att, def\}$. A coloring corresponds to an $X_{>t}$-restricted admissible set S in the following way:

$$x \in X_t : C(x) = \begin{cases} in & iff \;\; x \in S \\ out & iff \;\; x \notin S \wedge x \not\rightarrowtail S \wedge S \not\rightarrowtail x \\ att & iff \;\; x \notin S \wedge x \rightarrowtail S \wedge S \not\rightarrowtail x \\ def & iff \;\; x \notin S \wedge S \rightarrowtail x \end{cases}$$

DP - Leaf-Node

Leaf Nodes

We compute all conflict-free sets over X_t.
As $X_{>t} = \emptyset$ the conflict-free sets coincide with the $X_{>t}$-restricted admissible sets.

DP - Leaf-Node

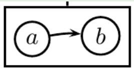
Leaf Nodes

We compute all conflict-free sets over X_t.

As $X_{>t} = \emptyset$ the conflict-free sets coincide with the $X_{>t}$-restricted admissible sets.

Tree Decomposition

$n_7 : \{a, b\}$



Colorings for n_7

There are three conflict-free sets $\emptyset, \{a\}, \{b\}$

a	*b*	#
in	*def*	1
att	*in*	1
out	*out*	1

DP - Forget-Node

Forget-Node for argument x

Eliminate all colorings C with $C(x) = att$. Remove the variable x from the remaining colorings.

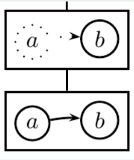
DP - Forget-Node

Forget-Node for argument x

Eliminate all colorings C with $C(x) = att$. Remove the variable x from the remaining colorings.

Tree Decomposition

$n_7 : \{a, b\} \rightarrow n_6 : \{b\}$



Colorings for n_7

a	b	#
in	*def*	1
att	*in*	1
out	*out*	1

Colorings for n_6

b	#
def	1
out	1

DP - Insert-Node

Insert-Node for argument x

For each coloring C of the child-node there may be two colorings.
1) C extended by $C(x) \in \{out, att, def\}$
2) C extended by $C(x) = in$ (if $\{y \in X_t : C(y) = in\} \cup \{x\}$ is conflict free)

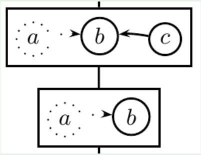
DP - Insert-Node

Insert-Node for argument x

For each coloring C of the child-node there may be two colorings.
1) C extended by $C(x) \in \{out, att, def\}$
2) C extended by $C(x) = in$ (if $\{y \in X_t : C(y) = in\} \cup \{x\}$ is conflict free)

Tree Decomposition

$n_6 : \{b\} \rightarrow n_5 : \{b, c\}$



Colorings for n_6

b	#
def	1
out	1

Colorings for n_5

b	c	#
def	*in*	2
def	*out*	1
out	*out*	1

DP - Join-Node

Join-Node

Combine the colorings of the child-nodes that map the same arguments to *in*.

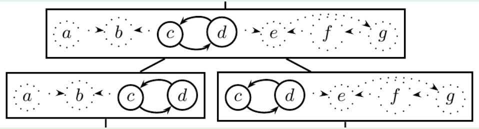
DP - Join-Node

Join-Node

Combine the colorings of the child-nodes that map the same arguments to *in*.

Tree Decomposition

$n_3 : \{a, b\}$; $n_8 : \{a, b\} \longrightarrow n_2 : \{a, b\}$



Colorings for n_3, n_8

n_3:

c	d	#
in	*def*	2
def	*in*	2
out	*out*	2

n_8:

c	d	#
in	*def*	1
def	*in*	2
out	*out*	1

Colorings for n_2

c	d	#
in	*def*	2
def	*in*	4
out	*out*	2

DP - Root-Node

Root-Node

As $X_{>t}$ equals A we have that the $X_{>t}$-restricted admissible sets of $X_{\geq t}$ are the admissible sets of F.

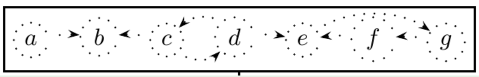
DP - Root-Node

Root-Node

As $X_{>t}$ equals A we have that the $X_{>t}$-restricted admissible sets of $X_{\geq t}$ are the admissible sets of F.

Tree Decomposition

n_0 : $\{\}$



Colorings for n_0

–	#
ϵ	8

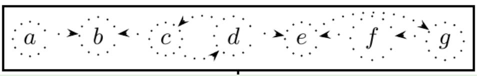
DP - Root-Node

Root-Node

As $X_{>t}$ equals A we have that the $X_{>t}$-restricted admissible sets of $X_{\geq t}$ are the admissible sets of F.

Tree Decomposition

$n_0 : \{\}$



Colorings for n_0

$-$	$\#$
ϵ	8

Credulous Acceptance

For credulous acceptance of an argument x we only consider colorings C with $C(x) = in$ (for bags X_t with $x \in X_t$).
Then the argument x is credulously accepted iff $\#_{root} > 0$.

Complexity

Complexity

Given an AF of tree-width w and an argument x, our algorithm decides if x is sceptical accepted in time $O(f(w) \cdot |AF|)$

Complexity

Complexity

Given an AF of tree-width w and an argument x, our algorithm decides if x is sceptical accepted in time $O(f(w) \cdot |AF|)$

Proof Ideas.

- The size of a coloring is bounded by $O(w)$.
- The number of colorings for every bag is bounded by 4^w.
- Thus the computation of all colorings for an bag can be done in time $f(w)$.

□

Future Work

Future and Ongoing Work:

- Implementation of these algorithms
- Systematic Comparison with existing Frameworks.
- Adapting this algorithm to other semantics for AFs.
- Identifying larger tractable fragments (e.g. directed graph measures like clique width)

  $\hookrightarrow$ Developing / Implementing fixed-parameter tractable algorithms

WE WANT YOU!

Yannis Dimopoulos and Alberto Torres.
Graph theoretical structures in logic programs and default theories.
Theor. Comput. Sci., 170(1-2):209–244, 1996.

Paul E. Dunne.
Computational properties of argument systems satisfying graph-theoretic constraints.
Artif. Intell., 171(10-15):701–729, 2007.
ISSN 0004-3702.
doi: http://dx.doi.org/10.1016/j.artint.2007.03.006.

Paul E. Dunne and Trevor J. M. Bench-Capon.
Coherence in finite argument systems.
Artif. Intell., 141(1/2):187–203, 2002.

Ton Kloks.
Treewidth, Computations and Approximations, volume 842 of *Lecture Notes in Computer Science*.
Springer, 1994.