**Motivation**
- Argumentation is a general issue in AI
- Many argumentation problems are in general computationally intractable
- We are interested in tractable fragments
- By Courcelle’s theorem we know that there exists tractable algorithms for bounded tree-width but it doesn’t give us practical algorithms.
- Dynamic Programming algorithms are approved for bounded tree-width problems.

**Tree-Width**
Let \( G(V,E) \) be an undirected graph.
A tree decomposition of \( G \) is a pair \((T,X)\) where \( T = (V,E) \) is a tree and \( X = \{X_v : v \in V \} \) such that:
1. \( \bigcup_{v \in V} X_v = V \), \( X \) is a cover of \( V \)
2. For each vertex \( v \in V \) the subgraph of \( T \) induced by \( \{t : v \in X_t\} \) is connected
3. For each edge \( \{v,v'\} \in E \) there exists an \( X_v \) with \( \{v,v'\} \subseteq X_v \)
We say that the width of such a decomposition is
\[ \max \{|X_t| : t \in V_T\} - 1 \]
The tree-width of a graph is the minimum width over all tree decompositions.
A nice tree decomposition is a tree decomposition where each bag \( t \) is of the following types:
- Leaf: \( t \) is a leaf of \( T \)
- Forget: \( t \) has exactly one child \( t' \) and \( \{v\} \subseteq X_{t'} \)
- Insert: \( t \) has exactly one child \( t' \) and \( \{v\} \subseteq X_{t'} \)
- Join: \( t \) has exactly two children \( t',t'' \) and \( \{v\} \subseteq X_{t'} \)
Given a tree decomposition we can easily compute a nice tree decomposition of the same width.
Example: A nice tree decomposition of our example argumentation framework.

**Counting Stable Extensions**
Following these ideas we get the following algorithm:
- **Leaf Nodes:**
  - compute all possible bag extensions
  - label arguments and test if the extension is cf
- **Forget Nodes:**
  - delete extensions where the forgotten argument is undefeated
  - delete the column of the forgotten argument
  - union rows representing the same extension (sum the values #a)
- **Insert Nodes:**
  - add column for the new argument
  - duplicate each extension - one version including the new argument and one that not
  - update labels and test if the extension is cf
- **Join Nodes:**
  - copy extension which are in both successor tables
  - multiply the values #a from the successors
  - update labels, an argument is undefeated only if it is undefeated in both successor extensions
- **Root Node:**
  - sum over all bag extensions without undefeated arguments to get the number of stable extensions

**Theorem**
Given an argumentation framework \( F \) together with a tree decomposition of width \( t \), our DP algorithm computes the number of stable extensions in time \( O(t^t) / (t!) \) (assuming constant cost for arithmetic).

**Fixed Parameter Tractability**
As argumentation problems are computationally intractable we are interested in tractable fragments.
- Often the computational complexity primarily depends on some problem parameters rather than on the size of the instance.
- Many hard problems become tractable if some problem parameters are fixed or bounded.
- In the area of graphs tree-width is such a parameter, e.g. there are many hard problems which are tractable for graphs of bounded tree-width.

**Complexity**
Reasoning Problems in AFs for e(a,c,d,e) and R=(a,b),(b,c),(d,c),(d,e),(e,e).

**Semantics**
Let \( F=(A,R) \) be an AF. A set \( SC(A) \) is
- conflict-free (cf), if there are no a, b, c, s, such that \((a,b) \in R \)
- a stable extension of \( F \), if \( S \) is conflict-free and each \( a,c \) is defended by \( SC(A) \) (in \( F \) for, if for each \( b,c \), it holds that, if \((b,a) \in R \), then \( S \) defeats \( b \) (in \( F \)).

**Example**
Let \( F=(A,R) \) an AF with \( A=\{a,b,c,d,e\} \) and \( R=(a,b),(b,c),(d,c),(d,e),(e,e) \).

**Counting Stable Extensions**
We have presented a DP-algorithm that counts stable extensions for our example AF get the following tables:

**Summary**
- We have presented a DP-algorithm that counts stable extensions in linear time for bounded tree-width.
- With similar techniques we get DP-algorithms for admissible and preferred extensions.
- (and further for stage, semi-stable, ideal)
- Future work:
  - Implementation of these DP-algorithms
  - Evaluate the DP-algorithm’s efficiency by comparing it with other systems

---

Poster Session ACAI'09 Belfast
Argumentation with Bounded Tree-Width

Wolfgang Dvořák