Structural Tractability of Enumerating CSP Solutions

Gianluigi Greco and Francesco Scarcello
University of Calabria, Italy
CSPs as Homomorphism Problems

- Set of variables \( \{X_1, \ldots, X_{26}\} \)
- Set of constraint scopes
- Set of constraint relations
CSPs as Homomorphism Problems

\[ r_{1h}(X_1, X_2, X_3, X_4, X_5) \]

\[ r_{1v}(X_1, X_7, X_{11}, X_{16}, X_{20}) \]

\( r \)-structure \( \mathbb{B} \)

\( l \)-structure \( \mathbb{A} \)
CSPs as Homomorphism Problems

\[ r_{1h}(X_1, X_2, X_3, X_4, X_5) \]

\[ r_{1v}(X_1, X_7, X_{11}, X_{16}, X_{20}) \]

\[ \text{l-structure } A \]

\[ \text{homomorphism} \]

\[ \text{r-structure } B \]
Questions
Questions

**INPUT:** CSP instance \( (A, B) \)

- Decide the *existence* of a homomorphism
- *Enumerate* all the homomorphisms \( A^B \)
- For a set of variables \( X \), enumerate the *projection* \( A^B [X] \)
Questions

**INPUT:** CSP instance \((A, B)\)

- Decide the *existence* of a homomorphism
- *Enumerate* all the homomorphisms \(A^B\)
- For a set of variables \(X\), enumerate the *projection* \(A^B[X]\)

- Tractable decision and closure properties imply tractable search
  [R. Dechter and A. Itai, 1992]
- Non-uniform case
  [D. Cohen, 2004]
CSPs and Hypergraphs

- Variables map to nodes
- Scopes map to hyperedges

\(H_A\)
Structurally Restricted CSPs

$H_A$
Structurally Restricted CSPs
Structurally Restricted CSPs

The hypergraph is acyclic

- Acyclicity is efficiently recognizable
- Acyclic CSPs can be efficiently solved
- Generalized arc consistency $\rightarrow$ Global consistency
Structurally Restricted CSPs

- Acyclicity is efficiently recognizable
- Acyclic CSPs can be efficiently solved
- Generalized arc consistency $\rightarrow$ Global consistency
Decomposition Methods

\[ \mathcal{H}_A \]
Transform the hypergraph into an acyclic one:

- Organize its edges (or nodes) in clusters
- Arrange the clusters as a tree, by satisfying the connectedness condition

\[ H_A \]
Transform the hypergraph into an acyclic one:

- Organize its edges (or nodes) in clusters
- Arrange the clusters as a tree, by satisfying the connectedness condition
Generalized Hypertree Decompositions

Transform the hypergraph into an acyclic one:
- Organize its edges (or nodes) in clusters
- Arrange the clusters as a tree, by satisfying the connectedness condition

Each cluster can be seen as a subproblem
Outline

Decomposition Methods and Tree Projections

Enumeration without Certificates

Enumeration with Certificates
Revisiting Decomposition Methods

CSP instance \((A, B)\)

\[ A_{\gamma} = \ell \text{-DM}(A) \quad B_{\gamma} = r \text{-DM}(A, B) \]

Scopes

Solutions

Work on subproblems
CSP instance \((A, B)\)

\[ A_Y = \ell\text{-DM}(A), \quad B_Y = r\text{-DM}(A, B) \]
**(Noticeable) Examples**

**CSP instance** \( (\mathcal{A}, \mathcal{B}) \)

\[
\mathcal{A}_V = \ell \text{-DM(} \mathcal{A} \text{)} \quad \mathcal{B}_V = r \text{-DM}(\mathcal{A}, \mathcal{B})
\]

- **Treewidth**: take all views that can be computed with at most \( k \) variables
- **Generalized hypertree width**: take all views that can be computed by joining at most \( k \) atoms (\( k \) query views)
- **Fractional hypertree width**: take all views that can be computed through subproblems having fractional cover at most \( k \) (or use Marx’s \( O(k^3) \) approximation to have polynomially many views)
Acyclicity in Decomposition Methods

Working on subproblems is not necessarily beneficial…

CSP instance \((A, B)\)

\[ A_{\mathcal{V}} = \ell - \text{DM}(A) \quad B_{\mathcal{V}} = r - \text{DM}(A, B) \]
Tree Projections (by Example)

\[ A \Delta : \ r_1(A, B, C) \quad r_2(A, F) \quad r_3(C, D) \quad r_4(D, E, F) \]
\[ r_5(E, F, G) \quad r_6(G, H, I) \quad r_7(I, J) \quad r_8(J, K) \]

Structure of the CSP
Tree Projections (by Example)

\[ \Delta : r_1(A, B, C) \quad r_2(A, F) \quad r_3(C, D) \quad r_4(D, E, F) \]
\[ r_5(E, F, G) \quad r_6(G, H, I) \quad r_7(I, J) \quad r_8(J, K) \]

Structure of the CSP

Available Views
Tree Projections (by Example)

\[ \Delta \] : \quad r_1(A, B, C) \quad r_2(A, F) \quad r_3(C, D) \quad r_4(D, E, F) \\
\; r_5(E, F, G) \quad r_6(G, H, I) \quad r_7(I, J) \quad r_8(J, K) \\

Structure of the CSP

Tree Projection

Available Views
Tree Projections (by Example)

\[ \Delta : \ r_1(A, B, C) \quad r_2(A, F) \quad r_3(C, D) \quad r_4(D, E, F) \\
\quad r_5(E, F, G) \quad r_6(G, H, I) \quad r_7(I, J) \quad r_8(J, K) \]

\[ \mathcal{H}_\Delta, \ \mathcal{H}_\alpha, \ \mathcal{H}_{\alpha_{\forall}} \]

Structure of the CSP  Tree Projection  Available Views
Deciding whether a tree projection exists is NP-complete

The existence of a tree projection ensures polynomial-time solvability
[N. Goodman and O. Shmueli, 1984], [Y. Sagiv and O. Shmueli, 1993]

Acyclicity is efficiently recognizable
Acyclic CSPs can be efficiently solved
Generalized arc consistency $\rightarrow$ Global consistency
Outline

Decomposition Methods and Tree Projections

Enumeration without Certificates

Enumeration with Certificates
The Algorithm: Backtracking and Propagation

Input: An ECSP instance \((A, B, O)\), where \(O = \{X_1, \ldots, X_m\}\);
Output: \(A^B[O]\);
Method: update \((A, B, O)\) with any of its domain-restricted versions;
    let \(A_V := \ell\text{-DM}(A)\), \(B_V := r\text{-DM}(A, B)\);
    invoke \text{Propagate}(1, (A_V, B_V), m, \langle \rangle);

Procedure \text{Propagate}(i: integer, (A_V, B_V): pair of structures, m: integer, \langle a_1, \ldots, a_{i-1}\rangle: tuple of values in \(A^i\));
begin
1. let \(B'_V := \text{GAC}(A_V, B_V)\);
2. let activeValues := \text{dom}(X_i)^{B'_V};
3. for each element \(\langle a_i \rangle \in \text{activeValues} \) do
4.    if \(i = m\) then
5.        output \(\langle a_1, \ldots, a_{m-1}, a_m \rangle\);
6.    else
7.        update dom \((X_i)^{B'_V}\) with \{\(\langle a_i \rangle\}\}; /* \(X_i\) is fixed to value \(a_i\) */
8.        \text{Propagate}(i + 1, (A_V, B'_V), m, \langle a_1, \ldots, a_{i-1}, a_i \rangle);
end.
The Algorithm: Backtracking and Propagation

Input: An ECSP instance \((A, B, O)\), where \(O = \{X_1, \ldots, X_m\}\);
Output: \(A^B[O]\);
Method: update \((A, B, O)\) with any of its domain-restricted versions;
let \(A_V := \ell\text{-DM}(A)\), \(B_V := r\text{-DM}(A, B)\);
invoke Propagate(1, \((A_V, B_V), m, \langle\rangle\));

Procedure Propagate\((i: \text{integer}, \ (A_V, B_V): \text{pair of structures}, \ m: \text{integer}, \ \langle a_1, \ldots, a_{i-1}\rangle: \text{tuple of values in } A^i)\);
begin
1. let \(B'_V := \text{GAC}(A_V, B_V)\);
2. let activeValues := \text{dom}(X_i)^{B'_V};
3. for each element \(\langle a_i \rangle \in \text{activeValues} \) do
4. \quad if \(i = m\) then
5. \quad \quad output \(\langle a_1, \ldots, a_{m-1}, a_m\rangle\);
6. \quad else
7. \quad \quad update \text{dom}(X_i)^{B'_V} \text{ with } \{\langle a_i \rangle\}; \quad \quad /* X_i is fixed to value \(a_i\) */
8. \quad Propagate\((i + 1, \ (A_V, B'_V), m, \langle a_1, \ldots, a_{i-1}, a_i\rangle)\);
end.
The Algorithm: Backtracking and Propagation

Input: An ECSP instance \((\mathbb{A}, \mathbb{B}, O)\), where \(O = \{X_1, \ldots, X_m\}\);
Output: \(\mathbb{A}^\mathbb{B}[O]\);
Method: update \((\mathbb{A}, \mathbb{B}, O)\) with any of its domain-restricted versions;
\[
\text{let } \mathbb{A}_\mathcal{V} := \ell\text{-DM}(\mathbb{A}), \quad \mathbb{B}_\mathcal{V} := r\text{-DM}(\mathbb{A}, \mathbb{B});
\]
\[
\text{invoke } \text{Propagate}(1, (\mathbb{A}_\mathcal{V}, \mathbb{B}_\mathcal{V}), m, \langle \rangle);
\]

Procedure \text{Propagate}(i: \text{integer}, (\mathbb{A}_\mathcal{V}, \mathbb{B}_\mathcal{V}): \text{pair of structures}, m: \text{integer},
\langle a_1, \ldots, a_{i-1}\rangle: \text{tuple of values in } A^i);

begin
1. let \(\mathbb{B}'_\mathcal{V} := \text{GAC}(\mathbb{A}_\mathcal{V}, \mathbb{B}_\mathcal{V})\);
2. let \(\text{activeValues} := \text{dom}(X_i)^{\mathbb{B}'_\mathcal{V}}\);
3. for each element \(\langle a_i\rangle \in \text{activeValues} \) do
   4. if \(i = m\) then
      5. output \(\langle a_1, \ldots, a_{m-1}, a_m\rangle\);
   6. else
      7. update \(\text{dom}(X_i)^{\mathbb{B}'_\mathcal{V}}\) with \(\{\langle a_i\rangle\}\);  \(\text{/* } X_i \text{ is fixed to value } a_i */\)
     8. \text{Propagate}(i + 1, (\mathbb{A}_\mathcal{V}, \mathbb{B}'_\mathcal{V}), m, \langle a_1, \ldots, a_{i-1}, a_i\rangle);
end.
The Algorithm: Backtracking and Propagation

Input: An ECSP instance \((A, B, O)\), where \(O = \{X_1, \ldots, X_m\}\);
Output: \(A^B[O]\);
Method: update \((A, B, O)\) with any of its domain-restricted versions;
   let \(A_V := \ell\text{-DM}(A), \ B_V := r\text{-DM}(A, B)\);
   invoke Propagate(1, \((A_V, B_V)\), \(m\), \(\langle \rangle\));

Procedure Propagate(i: integer, \((A_V, B_V)\): pair of structures, \(m\): integer,
   \(\langle a_1, \ldots, a_{i-1}\rangle\): tuple of values in \(A^i\));
begin
1. let \(B'_V := \text{GAC}(A_V, B_V)\);
2. let activeValues := \(\text{dom}(X_i)_{B'_V}\);
3. for each element \(\langle a_i\rangle \in \text{activeValues} \) do
4. \(\text{if } i = \text{m} \text{ then}
5. \| \text{output } \langle a_1, \ldots, a_{m-1}, a_m\rangle;
6. \| \text{else}
7. \| \text{update } \text{dom}(X_i)_{B'_V} \text{ with } \{\langle a_i\rangle\}; \quad /\!\!/ X_i \text{ is fixed to value } a_i \quad /\!\!/
8. \| \text{Propagate}(i + 1, (A_V, B'_V), \text{m, } \langle a_1, \ldots, a_{i-1}, a_i\rangle)\);
end.
The Algorithm: Backtracking and Propagation

Input: An ECSP instance \((A, B, O)\), where \(O = \{X_1, \ldots, X_m\}\);
Output: \(A^B[O]\);
Method: update \((A, B, O)\) with any of its domain-restricted versions;
\[
\text{let } A\_V := \ell\text{-DM}(A), \quad B\_V := r\text{-DM}(A, B);
\]
invoke Propagate\((1, (A\_V, B\_V), m, \langle \rangle)\);

Procedure Propagate\((i: \text{integer}, (A\_V, B\_V): \text{pair of structures}, m: \text{integer}, \langle a_1, \ldots, a_{i-1}\rangle: \text{tuple of values in } A^i)\);

begin
1. let \(B\_V' := \text{GAC}(A\_V, B\_V)\);
2. let \(\text{activeValues} := \text{dom}(X_i)^{B\_V'}\);
3. for each element \(\langle a_i \rangle \in \text{activeValues}\) do
   4. if \(i = m\) then
   5. \quad output \(\langle a_1, \ldots, a_{m-1}, a_m \rangle\);
   6. else
   7. \quad \text{update } \text{dom}(X_i)^{B\_V'} \text{ with } \{\langle a_i \rangle\}; \quad /* X_i is fixed to value } a_i */
   8. \quad Propagate\((i + 1, (A\_V, B\_V'), m, \langle a_1, \ldots, a_{i-1}, a_i \rangle)\);
end.

The solution is not certified!
The Algorithm: Backtracking and Propagation

When is the algorithm correct?
Tp-covering
{B, C} is individually tp-covered

{D} is not individually tp-covered
Thm.

Let $\mathbb{A}$ be an $\ell$-structure, and let $O \subseteq A$ be a set of variables. The following are equivalent:

1. $O$ is individually $tp$-covered
2. For every $r$-structure $\mathbb{B}$, $\text{ComputeAllSolutions}_{DM}$ computes $A^B[O]$. 
The existence of a tree projection implies "GAC $\rightarrow$ Global"
[Y. Sagiv and O. Shmueli, 1983]
The existence of a tree projection implies "GAC $\rightarrow$ Global"

[Y. Sagiv and O. Shmueli, 1983]
The existence of a tree projection implies ”GAC → Global”
[Y. Sagiv and O. Shmueli, 1983]

For *generalized hypertree decompositions*, the existence of a tree projection for the core of the left-structure implies ”GAC → Global”
[H. Chen and V. Dalmau, 2005]
The existence of a tree projection implies "GAC → Global"
[Y. Sagiv and O. Shmueli, 1983]

For generalized hypertree decompositions, the existence of a tree projection for the core of the left-structure implies "GAC → Global"
[H. Chen and V. Dalmau, 2005]

Over the views, i.e., k-consistency
The existence of a tree projection implies "GAC → Global"
[Y. Sagiv and O. Shmueli, 1983]

For generalized hypertree decompositions, the existence of a tree projection for the core of the left-structure implies "GAC → Global"
[H. Chen and V. Dalmau, 2005]

For tree decompositions, and on classes of CSPs having bounded arity, the existence of a tree projection for the core of the left-structure is a sufficient and necessary condition to imply "GAC → Global"
[A. Atserias, A. Bulatov, and V. Dalmau, 2007]
The existence of a tree projection implies "GAC $\rightarrow$ Global"
[Y. Sagiv and O. Shmueli, 1983]

For generalized hypertree decompositions, the existence of a tree projection for the core of the left-structure implies "GAC $\rightarrow$ Global"
[H. Chen and V. Dalmau, 2005]

For tree decompositions, and on classes of CSPs having bounded arity, the existence of a tree projection for the core of the left-structure is a sufficient and necessary condition to imply "GAC $\rightarrow$ Global"
[A. Atserias, A. Bulatov, and V. Dalmau, 2007]

For generalized hypertree decompositions, the existence of a tree projection for the core is also necessary to imply "GAC $\rightarrow$ Global"
[G. Greco and F. Scarcello, 2010]
Proof: Discussion on the Base (Decision) Case

**Thm** [G. Greco and F. Scarcello, 2010].

Let $A$ be an $\ell$-structure, and let $A_\gamma$ be a $v$-structure. The following are equivalent:

1. There is a core $A'$ of $A$ such that $(H_{A'}, H_{A_\gamma})$ has a tree projection
2. For every $r$-structure $B$, for every $r$-structure $B_\gamma$ that is legal, enforcing generalized arc consistency on $B_\gamma$ is a correct decision procedure

For *generalized hypertree decompositions*, the existence of a tree projection for the core is also **necessary** to imply "GAC $\rightarrow$ Global" [G. Greco and F. Scarcello, 2010]
Complexity Issues of ComputeAllSolutions_DMD
If $O$ is tp-covered, the algorithm runs \textbf{With} Polynomial \textbf{Delay}…
Complexity Issues of ComputeAllSolutions

If $O$ is tp-covered, the algorithm runs \textbf{With Polynomial Delay}…

…and the result is essentially tight

\textbf{Thm.}

Assume $\text{FPT} \neq \text{W}[1]$. Let $\mathbf{A}$ be any class of $\ell$-structures of bounded arity. Then, the following are equivalent:

1. $\mathbf{A}$ has bounded treewidth modulo homomorphic equivalence;
2. For every $\mathbf{A} \in \mathbf{A}$, for every $r$-structure $\mathbf{B}$, and for every set of variables $O \subseteq \text{drv}(\mathbf{A})$, the ECSP instance $(\mathbf{A}, \mathbf{B}, O)$ is solvable WPD.
If $O$ is tp-covered, the algorithm runs \textbf{With Polynomial Delay}…

…and the result is essentially tight

\textbf{Thm.}

Assume $\text{FPT} \neq \text{W}[1]$. Let $A$ be any class of $\ell$-structures of bounded arity. Then, the following are equivalent:

1. $A$ has bounded treewidth modulo homomorphic equivalence;
2. For every $A \in A$, for every $r$-structure $B$, and for every set of variables $O \subseteq \text{drv}(A)$, the ECSP instance $(A, B, O)$ is solvable WPD.
If \( O \) is tp-covered, the algorithm runs with polynomial delay…

…and the result is essentially tight.

\[
\text{Thm.}
\]

Assume \( \text{FPT} \neq \text{W}[1] \). Let \( \mathbf{A} \) be any class of \( \ell \)-structures of bounded arity. Then, the following are equivalent:

1. \( \mathbf{A} \) has bounded treewidth modulo homomorphic equivalence;
2. For every \( \mathbf{A} \in \mathbf{A} \), for every \( r \)-structure \( \mathbf{B} \), and for every set of variables \( O \subseteq \text{drv}(\mathbf{A}) \), the ECSP instance \( (\mathbf{A}, \mathbf{B}, O) \) is solvable WPD.

There is no efficient algorithm for the no-promise problem.
Outline

- Decomposition Methods and Tree Projections
- Enumeration without Certificates
- Enumeration with Certificates
A Simple Modification

**Input:** An ECSP instance \((A, B, O)\), where \(O = \{X_1, \ldots, X_m\}\);

**Output:** for each solution \(h \in A^B[O]\), a certified solution \((h, h')\);

**Method:** let \(A = \{X_1, \ldots, X_m, X_{m+1}, \ldots, X_n\}\) be the variables of \(A\);
update \((A, B, A)\) with any of its domain restricted versions;
let \(A'_V := \ell\text{-DM}(A), B'_V := r\text{-DM}(A, B)\);
invoke CPropagate\(1, (A'_V, B'_V), m, \langle \rangle \);

**Procedure** CPropagate\((i: \text{integer}, (A'_V, B'_V): \text{pair of structures}, m: \text{integer}, \langle a_1, \ldots, a_{i-1}\rangle: \text{tuple of values in} A^i)\);

\[
\begin{align*}
\text{begin} & \\
1. & \text{let } B'_V := \text{GAC}(A'_V, B'_V); \\
2. & \text{if } i > 1 \text{ and } B'_V \text{ is empty then output "DM failure" and HALT;} \\
3. & \text{let } \text{activeValues} := \text{dom}(X_i)^{B'_V}; \\
4. & \text{for each element } \langle a_i \rangle \in \text{activeValues do} \\
5. & \quad \text{if } i = n \text{ then} \\
6. & \quad \quad \text{output} \text{ the certified solution } (\langle a_1, \ldots, a_m \rangle, \langle a_{m+1}, \ldots, a_n \rangle); \\
7. & \quad \text{else} \\
8. & \quad \quad \text{update } \text{dom}(X_i)^{B'_V} \text{ with } \{\langle a_i \rangle\}; \quad /* X_i \text{ is fixed to value } a_i */ \\
9. & \quad \quad \text{CPropagate} (i + 1, (A'_V, B'_V), m, \langle a_1, \ldots, a_{i-1}, a_i \rangle); \\
10. & \quad \quad \text{if } i > m \text{ then BREAK; }
\end{align*}
\]

end.
**Input**: An ECSP instance \((A, B, O)\), where \(O = \{X_1, \ldots, X_m\}\);

**Output**: for each solution \(h \in A^B[G]\), a certified solution \((h, h')\);

**Method**: let \(A = \{X_1, \ldots, X_m, X_{m+1}, \ldots, X_n\}\) be the variables of \(A\);
update \((A, B, A)\) with any of its domain restricted versions;
let \(A_V := \ell-DM(A), \ B_V := r-DM(A, B)\);
invoke CPropagate\((1, (A_V, B_V), m, \langle \rangle)\);

**Procedure** CPropagate\((i: integer, (A_V, B_V): pair of structures, m: integer, \langle a_1, \ldots, a_{i-1}\rangle: tuple of values in A^i)\);

begin
1. let \(B'_V := GAC(A_V, B_V)\);
2. if \(i > 1\) and \(B'_V\) is empty then output “DM failure” and HALT;
3. let \(activeValues := dom(X_i)^B'_V\);
4. for each element \(\langle a_i\rangle \in activeValues\) do
5.      if \(i = n\) then
6.         output the certified solution \((\langle a_1, \ldots, a_m\rangle, \langle a_{m+1}, \ldots, a_n\rangle)\);
7.      else
8.          update \(dom(X_i)^B'_V\) with \(\{\langle a_i\rangle\}\); \textit{/* }X_i\text{ is fixed to value } a_i \text{ */} \)
9.          CPropagate\((i + 1, (A_V, B'_V), m, \langle a_1, \ldots, a_{i-1}, a_i\rangle)\);
10.     if \(i > m\) then BREAK;
end.
Input: An ECSP instance \((A, B, O)\), where \(O = \{X_1, \ldots, X_m\}\);
Output: for each solution \(h \in A^B[G]\), a certified solution \((h, h')\);
Method: let \(A = \{X_1, \ldots, X_m, X_{m+1}, \ldots, X_n\}\) be the variables of \(A\);
update \((A, B, A)\) with any of its domain restricted versions;
let \(A_V := \ell-DM(A), B_V := r-DM(A, B)\);
invoke CPropagate(1, (A_V, B_V), m, \langle \rangle);

Procedure CPropagate\((i: \text{integer}, (A_V, B_V): \text{pair of structures}, m: \text{integer}, \langle a_1, \ldots, a_{i-1} \rangle: \text{tuple of values in } A^i)\);

begin
1. let \(B'_V := \cap C(A_V, B_V)\);
2. if \(i > 1\) and \(B'_V\) is empty then output "DM failure" and HALT;
3. let activeValues := \(\text{dom}(X_i)^{B'_V}\);
4. for each element \(\langle a_i \rangle \in \text{activeValues} \) do
5. | if \(i = n\) then
6. | | output the certified solution \((\langle a_1, \ldots, a_m \rangle, \langle a_{m+1}, \ldots, a_n \rangle)\);
7. | else
8. | | update \(\text{dom}(X_i)^{B'_V}\) with \(\{\langle a_i \rangle\}\); /* \(X_i\) is fixed to value \(a_i\) */
9. | | CPropagate\((i + 1, (A_V, B'_V), m, \langle a_1, \ldots, a_{i-1}, a_i \rangle)\);
10. | | if \(i > m\) then BREAK;
end.
A Simple Modification

When is the algorithm correct?
Let $A$ be an $\ell$-structure, and $O \subseteq A$ be a set of variables. Then, for every $r$-structure $B$, $\text{ComputeCertifiedSolutions}_{DM}^{B}$ computes WPD a subset of the solutions in $A^{B}[O]$, with a certificate for each of them. Moreover,

- If $\text{ComputeCertifiedSolutions}_{DM}^{B}$ outputs “DM failure”, then $(H_{A}, H_{\ell-DM(A)})$ does not have a tree projection;
- otherwise, $\text{ComputeCertifiedSolutions}_{DM}^{B}$ computes WPD $A^{B}[O]$. 
Assume \( \text{FPT} \neq \text{W}[1] \). Let \( \mathbf{A} \) be any bounded-arity recursively-enumerable class of \( \ell \)-structures closed under taking minors. Then, the following are equivalent:

1. \( \mathbf{A} \) has bounded treewidth;
2. For every \( \mathbf{A} \in \mathbf{A} \), for every \( r \)-structure \( \mathbf{B} \), and for every set of variables \( O \subseteq A \), the ECSP instance \( (\mathbf{A}, \mathbf{B}, O) \) is solvable WPD.
Comparing The Results

Thm.

Assume $\text{FPT} \neq W[1]$. Let $A$ be any class of $\ell$-structures of bounded arity. Then, the following are equivalent:

1. $A$ has bounded treewidth modulo homomorphic equivalence;
2. For every $A \in A$, for every $r$-structure $B$, and for every set of variables $O \subseteq \text{drv}(A)$, the ECSP instance $(A, B, O)$ is solvable WPD.

Thm.

Assume $\text{FPT} \neq W[1]$. Let $A$ be any bounded-arity recursively-enumerable class of $\ell$-structures closed under taking minors. Then, the following are equivalent:

1. $A$ has bounded treewidth;
2. For every $A \in A$, for every $r$-structure $B$, and for every set of variables $O \subseteq A$, the ECSP instance $(A, B, O)$ is solvable WPD.
Comparing The Results

**Thm.**

Assume FPT $\neq W[1]$. Let $\mathbf{A}$ be any class of $\ell$-structures of bounded arity. Then, the following are equivalent:

1. $\mathbf{A}$ has bounded treewidth modulo homomorphic equivalence;
2. For every $\mathbf{A} \in \mathbf{A}$, for every $r$-structure $\mathbf{B}$, and for every set of variables $O \subseteq \text{drv}(\mathbf{A})$, the ECSP instance $(\mathbf{A}, \mathbf{B}, O)$ is solvable WPD.

**Thm.**

Assume FPT $\neq W[1]$. Let $\mathbf{A}$ be any bounded-arity recursively-enumerable class of $\ell$-structures closed under taking minors. Then, the following are equivalent:

1. $\mathbf{A}$ has **bounded treewidth**;
2. For every $\mathbf{A} \in \mathbf{A}$, for every $r$-structure $\mathbf{B}$, and for every set of variables $O \subseteq \mathbf{A}$, the ECSP instance $(\mathbf{A}, \mathbf{B}, O)$ is solvable WPD.
Thank you!