Towards Mass Spectrum Analysis with ASP

Nils Küchenmeister[0009−0004−0376−0328], Alex Ivliev[0000−0002−1604−6308], and Markus Krötzsch[0000−0002−9172−2601]

Knowledge-Based System Group, TU Dresden, Germany
{nils.kuechenmeister,alex.ivliev,markus.kroetzsch}@tu-dresden.de

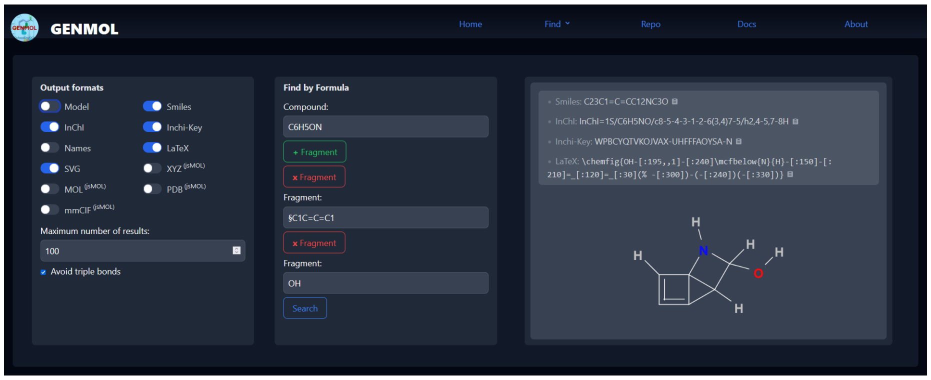
Abstract. We present a new use of Answer Set Programming (ASP) to discover the molecular structure of chemical samples based on the relative abundance of elements and structural fragments, as measured in mass spectrometry. To constrain the exponential search space for this combinatorial problem, we develop canonical representations of molecular structures and an ASP implementation that uses these definitions. We evaluate the correctness of our implementation over a large set of known molecular structures, and we compare its quality and performance to other ASP symmetry-breaking methods and to a commercial tool from analytical chemistry.

Keywords: ASP · symmetry breaking · molecular structure · chemistry.

1 Introduction

Mass spectrometry is a powerful technique to determine the chemical composition of a substance [2]. However, the mass spectrum of a substance does not reveal its exact molecular structure, but merely the possible ratios of elements in the compound and its fragments. To identify a sample, researcher may use commercial databases (for common compounds), or software tools that can discover molecular structures from the partial information available. The latter leads to a combinatorial search problem that is a natural fit for answer set programming (ASP). Molecules can be modeled as undirected graphs, representing the different elements and atomic bonds as node and edge labels, respectively. ASP is well-suited to encode chemical domain knowledge (e.g., possible number of bonds for carbon) and extra information about the sample (e.g., that it has an OH group), so that each answer set encodes a valid molecular graph.

Unfortunately, this does not work: a direct ASP encoding yields exponentially many answer sets for each molecular graph due to the large number of symmetries (automorphisms) in such graphs. For example, $C_6H_{12}O$ admits 211 distinct molecule structures but leads to 111,870 answer sets. Removing redundant solutions and limiting the search to unique representations are common techniques used in the ASP community where they have motivated research on *symmetry breaking*. Related approaches work by adding additional rules to ASP [1,5,11], by rewriting the ground program before solving [6,4,3], or by introducing dedicated solvers [9]. However, our experiments with some of these approaches still produced 10–10,000 times more answer sets than molecules even in simple cases.



Output formats

- Model [off]
- Smiles [✓]
- InChI [✓]
- Inchi-Key [✓]
- Names [off]
- LaTeX [✓]
- SVG [✓]
- XYZ (jsMOL) [off]
- MOL (jsMOL) [off]
- PDB (jsMOL) [off]
- mmCIF (jsMOL) [off]

Maximum number of results: 100

[✓] Avoid triple bonds

Find by Formula

Compound: C6H5ON

Fragment: §C1C=C=C1

Fragment: OH

- Smiles: C23C1=C=CC12NC3O
- InChI: InChI=1S/C6H5NO/c8-5-4-3-1-2-6(3,4)7-5/h2,4-5,7-8H
- Inchi-Key: WPBCYQTVKOJVAX-UHFFFAOYSA-N
- LaTeX: `\chemfig{OH-[:195,,1]-[:240]\mcfbelow{N}{H}-[:150]-[:210]=_[:120]=_[:30](% -[:300])-(-[:240])(-[:330])}`

Fig. 1: User interface of Genmol

We therefore develop a new approach that prevents symmetries in graph representations already during grounding, and use it as the core of an ASP-based prototype implementation for enumerating molecular structures based on partial chemical information. In Section 2, we explain the problem and our prototype tool from a user perspective. We then define the problem formally in Section 3, using an abstract notion of *tree representations* of molecular graphs that takes inspiration from the chemical notation SMILES [12]. We then derive a new canonical representation for molecular graphs (Section 4) to guide our ASP implementation (Section 5). In Section 6, we evaluate the correctness, symmetry-breaking capabilities, and performance of our tool in comparison to other ASP-based approaches and a leading commercial software for analytical chemistry [8]. We achieve perfect symmetry-breaking for acyclic graph structures, and up to three orders of magnitude reduction in answer sets for cyclic cases in comparison to other ASP approaches. Overall, ASP therefore appears to be a promising basis for this use case, and possibly for other use cases concerned with undirected graph structures.

Our ASP source code, evaluation helpers, and data sets are available online at `https://github.com/knowsys/eval-2024-asp-molecules`. The sources of our prototype application are at `https://gitlab.com/nkuechen/genmol/`.

2 Analysis of Mass Spectra with Genmol

Many mass spectrometers break up samples into smaller fragments and measure their relative abundance. The resulting mass spectrum forms a characteristic pattern, enabling inferences about the underlying sample. High-resolution spectra may contain information such as "the molecule has six carbon atoms" or "there is an *OH* group", but cannot reveal the samples's full molecular structure. In chemical analysis, we are looking for molecular structures that are consistent with the measured mass spectrum.

To address this task, we have developed Genmol, a prototype application for enumerating molecular structures for a given composition of fragments. It is available as a command-line tool and as a progressive web application (PWA), shown in Fig. 1. Genmol is implemented in Rust, with the web front-end using the Yew framework on top of a JSON-API, whereas the search for molecular structures is implemented in Answer

Set Programming (ASP) and solved using *clingo* [7]. An online demo of GENMOL is available for review at `https://tools.iccl.inf.tu-dresden.de/genmol/`.

The screenshot shows the use of GENMOL with a *sum formula* C_6H_5ON and two fragments as input. Specifying detected fragments and restricting bond types helps to reduce the search space. Alternatively, users can provide a molecular mass or a complete mass spectrum, which will then be associated with possible chemical formulas using, e.g., information about the abundance of isotopes.

The core task of GENMOL then is to find molecules that match the given input constraints. Molecules in this context are viewed as undirected graphs of atoms, linked by covalent bonds that result from sharing electrons.[1] Many chemical elements admit a fixed number of bonds, the so-called *valence*, according to the number of electrons available for binding (e.g., carbon has a valence of 4). Bonds may involve several electrons, leading to single, double, triple bonds, etc. The graph structure of molecules, the assignment of elements, and the possible types of bonds can lead to a large number of possible molecules for a single chemical formula, and this combinatorial search task is a natural match for ASP.

3 Problem Definition: Enumeration of Molecules

We begin by formalizing the problem of molecule enumeration, and by introducing a chemistry-inspired representation of molecules. We consider a set $\mathbb{E}$ of *elements*, with each $e \in \mathbb{E}$ associated with a *valence* $\mathbb{V}(e) \in \mathbb{N}_{>0}$.[2] We assume that $\mathbb{E}$ contains a distinguished element $H \in \mathbb{E}$ (hydrogen) with $\mathbb{V}(H) = 1$. We model molecules as undirected graphs with edges labelled by natural numbers to indicate the type of bond.

Definition 1. *A* molecular graph *G is a tuple $G = \langle V, E, \ell, b\rangle$ with vertices $V = \{1, \ldots, k\}$ for some $k \geq 1$, undirected edges $E \subseteq \binom{V}{2}$, where $\binom{V}{2}$ is the set of all 2-element subsets of V, and labelling functions $\ell\colon V \to \mathbb{E}$ and $b\colon E \to \mathbb{N}_{>0}$.*

The degree $\mathsf{deg}(v)$ *of a vertex $v \in V$ is defined as $\mathsf{deg}(v) = \sum\{b(e) \mid e \in E, v \in e\}$. A list of n distinct vertices $v_1, \ldots, v_n$ is a* simple path *in G if $\{v_i, v_{i+1}\} \in E$ for every $1 \leq i < n$. G is connected if there is a simple path from v to w for every pair $v, w \in V$.*

Since we assume that atoms in a molecule use all available bonds according to their valence, the (very frequent) hydrogen atoms do not need to be mentioned explicitly. Such *hydrogen-suppressed* molecular graphs are common in computational chemistry:

Definition 2. *A* molecular formula *is a function $f\colon \mathbb{E} \to \mathbb{N}$. We say that a molecular graph $G = \langle V, E, \ell, b\rangle$ is* valid *for f, if it satisfies the following properties:*

1. *G is connected,*
2. *for every $e \in \mathbb{E}$ with $e \neq H$, $\#\{v \in V \mid \ell(v) = e\} = f(e)$,*
3. *for every $v \in V$, $\mathsf{deg}(v) \leq \mathbb{V}(\ell(v))$,*
4. *$\sum_{v \in V} (\mathbb{V}(\ell(v)) - \mathsf{deg}(v)) = f(H)$.*

[1] This graph does not always determine the spacial configuration of molecules, which cannot be determined by mass spectrometry alone, yet it suffices for many applications.

[2] The formalization ignores multi-valence elements, even though supported in the ASP program.

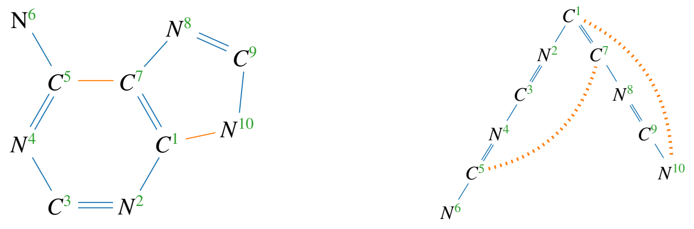


Fig. 2: Hydrogen-suppressed molecular graph of adenine ($C_5H_5N_5$) and corresponding spanning tree with cycle edges (dotted); subscripts indicate correspondence of vertices

The enumeration problem can now be stated as follows: for a given molecular formula f, enumerate, up to isomorphism, all valid molecular graphs for f. In general, the number of distinct isomorphic molecular graphs is exponential in the number of atoms. It is therefore important to reduce the enumeration of redundant isomorphic graphs.

A first step towards this is the use of a more restricted representation of molecular graphs. Here, we take inspiration from the *simplified molecular-input line-entry system* (*SMILES*), a widely used serialization format for molecular graphs. SMILES strings start from an (arbitrary) spanning tree of the molecular graph, serialized in a depth-first order, with subtrees enclosed in parentheses. Edges not covered by the spanning tree (since they would complete a cycle) are indicated by pairs of numeric *cycle markers*.

Example 1. Adenine ($C_5H_5N_5$) has the graph structure shown in Figure 2 (left), with a spanning tree on the right. The SMILES is `C1(=C2N=CN1)N=CN=C2N` where consecutive atoms are connected by bonds and double bonds are marked (=). The segment from vertex 7 to 10 is in parentheses to indicate a branch. Additionally, the two non-sequential (dotted) connections are denoted by matching pairs of numerical markers.

Definition 3. *A molecular graph $G = \langle V, E, \ell, b\rangle$ is a* molecular tree *if it is free of cycles and the natural order of vertices $V = \{1, \ldots, k\}$ corresponds to a depth-first search of the tree (in particular, vertex* 1 *is the root).*

A tree representation *of an arbitrary molecular graph $G = \langle V, E, \ell, b\rangle$ is a set $T \subseteq E$ such that $\langle V, T, \ell, b\rangle$ is a molecular tree. In this case, we denote G as $\langle V, T \cup C, \ell, b\rangle$ where T are the* tree edges *and $C = E \setminus T$ are the* cycle edges*.*

Note that the tree edges T by definition visit every vertex of V, and the cycle edges C merely make additional connections between vertices of the tree. In SMILES, the edges in C and their labels (bond types) are encoded with special markers, while the order of vertices is given by the order of appearance in the SMILES string.

A tree representation for a given molecular graph is uniquely determined by the following choices: (a) a spanning tree (corresponding to a choice of tree edges and cycle edges), (b) a root vertex, and (c) for every vertex, an order of visiting its child vertices in depth first search. For a given graph structure, the number of tree representations can be significantly lower than the number of isomorphic molecular graphs. For example, a graph that is a chain has only linearly many tree representations, but still admits exponentially many graphs. Nevertheless, the number of tree representations can still be exponential, and we will investigate below how the choices in (a)–(c) can be further constrained to reduce redundancy.

4 Canonical tree representations of molecular graphs

To eliminate redundant isomorphic solutions, we first define a canonical tree representation of any molecular graph. The defining conditions of this unique representation will then be used to constrain the search for possible graphs in our implementation. We first consider the simpler case of molecular trees.

4.1 Canonical Molecular Trees

We define a total order on molecular trees, which will allow us to define a largest tree among a set of candidates. To define this order inductively, we need to consider subtrees that may not have 1 as their root. For a molecular tree $G = \langle V, E, \ell, b \rangle$ with vertex $v \in V$, let $\mathsf{subtree}(G, v)$ be the tuple $\langle V', E', \ell, b, \mathsf{in}(G, v) \rangle$ where V' and E' are the restriction of V and E, respectively, to vertices that are part of the subtree with root v in G, and either $\mathsf{in}(G, v) = 0$ if $v = 1$ is the root of G, or $\mathsf{in}(G, v) = b(e)$ is the edge label $b(e) \geq 1$ of the edge e between v and its parent in G. Moreover, if $\langle c_1, \ldots, c_k \rangle$ are the ordered children of v, then $\mathsf{childtrees}(G, v) = \langle \mathsf{subtree}(G, c_1), \ldots, \mathsf{subtree}(G, c_k) \rangle$.

Finally, let $R(G, v) = \langle d, s, c, \ell, b \rangle$ be the tuple with $d \geq 1$ the depth of $\mathsf{subtree}(G, v)$; $s \geq 1$ the size (number of vertices) of $\mathsf{subtree}(G, v)$; $c \geq 0$ the number of children of v; $\ell \in \mathbb{E}$ the element $\ell(v)$ of v; and $b = \mathsf{in}(G, v) \geq 0$.

For the following definition, recall that the *lexicographic extension* of a strict order $\prec$ to tuples of the same size k is defined by setting $\boldsymbol{t} \prec \boldsymbol{u}$ if there is $i \in \{1, \ldots, k\}$ such that $\boldsymbol{t}[i] \prec \boldsymbol{u}[i]$ and $\boldsymbol{t}[j] = \boldsymbol{u}[j]$ for all $j < i$.

Definition 4. *Let $\sqsubset$ be an arbitrary but fixed strict total order on $\mathbb{E}$, and let $\sqsubset$ denote the usual order $<$ on natural numbers. We extend $\sqsubset$ to 5-tuples of the form $R(G, v)$ lexicographically.*

We define a strict order $\prec$ on subtrees as the smallest relation where, for each pair of subtrees $S_i = \mathsf{subtree}(G_i, v_i)$ of molecular trees G_i ($i = 1, 2$), $S_1 \prec S_2$ holds if

1. *$R(G_1, v_1) \sqsubset R(G_2, v_2)$, or*
2. *$R(G_1, v_1) = R(G_2, v_2)$, i.e. S_1 and S_2 are locally indistinguishable, with (necessarily equal) number of children k, such that $\mathsf{childtrees}(G_1, v_1) \prec \mathsf{childtrees}(G_2, v_2)$ where $\prec$ is the lexicographic extension of $\prec$ to k-tuples of subtrees.*

For molecular trees G_1 and G_2, we define $G_1 \prec G_2$ if $\mathsf{subtree}(G_1, 1) \prec \mathsf{subtree}(G_2, 1)$.

Proposition 1. *The relation $\prec$ of Definition 4 is a strict total order on molecular trees.*

Proof. The claim follows by showing that $\prec$ is a strict order on subtrees. The order $\sqsubset$ on tuples is strict since it is the lexicographic extension of strict orders. Hence, all subtrees $S_i = \mathsf{subtree}(G_i, v_i)$ ($i = 1, 2$) with $R(G_1, v_1) \neq R(G_2, v_2)$ are $\prec$-comparable by 1. Totality for case $R(G_1, v_1) = R(G_2, v_2)$ is shown by induction on the (equal) depth of the subtrees S_1 and S_2. For depth 1, S_1 and S_2 have no children, and $R(G_1, v_1) = R(G_2, v_1)$ implies $S_1 = S_2$. For depth i with $i > 1$, we can assume all subtrees of depth $\leq i - 1$ to be $\prec$-comparable unless equal. If $\mathsf{childtrees}(G_1, v_1)$ and $\mathsf{childtrees}(G_2, v_2)$ are comparable under the lexicographic extension of $\prec$, then S_1 and S_2 are $\prec$-comparable by 2. Otherwise, $\mathsf{childtrees}(G_1, v_1) = \mathsf{childtrees}(G_2, v_2)$, and therefore $S_1 = S_2$.

By Proposition 1, we could define the canonical molecular tree to be the $\prec$-largest tree among a set of isomorphic trees. However, this would force us to select a root that is the start (or end) of a longest path in the graph to maximize the depth of the tree. It is more efficient to compare a smaller set of potential roots that are closer together:

Definition 5. *Let $G = \langle V, E, \ell, b\rangle$ be a molecular tree. A vertex v_i is* central *in a simple path $v_1, \ldots, v_n$ in G if $i \in \{\lceil (n+1)/2\rceil, \lfloor (n+1)/2\rfloor\}$ (a singleton set if n is odd). A vertex is central in G if it is central in any longest simple path in G.*

The canonical molecular tree *C of G is the $\prec$-largest molecular tree that is obtained by permutation of vertices in G such that the root of C is central in G.*

In every tree, the central vertices of all longest simple paths are the same, and hence there are at most two. Indeed, two distinct longest paths always share at least one vertex in a tree. So if two such paths $\boldsymbol{v}$ and $\boldsymbol{w}$ would have different central vertices $v_a \neq w_b$, and a shared vertex $v_i = w_j$ with (w.l.o.g.) $a > i$ and $b > j$, then the path $v_1, \ldots, v_a, \ldots, v_i = w_j, \ldots, w_b, \ldots, w_1$ would be longer than $\boldsymbol{v}$ and $\boldsymbol{w}$, contradicting their assumed maximal length. Using this insight, our implementation can find the canonical molecular tree by considering at most two possible roots.

4.2 Canonical Molecular Graphs

Next, we define canonical tree representations of arbitrary molecular graphs. For a tree representation $G = \langle V, T \cup C, \ell, b\rangle$ with $V = \{1, \ldots, k\}$, we define a molecular tree $\mathsf{tr}(G)$ by replacing each edge in C by two edges to fresh vertices. Hence, let $V_C = \bigcup C$ be the set of all vertices in edges of C, and let $\mu : \{k+1, \ldots, k+|V_C|\} \to V_C$ be an arbitrary bijective mapping. For $v \in V$ (i.e., $1 \leq v \leq k$), let $\mu(v) = v$. Then $\mathsf{tr}(G) = \langle V', T', \ell', b'\rangle$ with $V' = \{1, \ldots, k+|V_C|\}$, $T' = T \cup \{\{v, w\} \mid \{v, \mu(w)\} \in C\}$, $\ell'(v) = \ell(\mu(v))$, and $b'(\{v, w\}) = b(\mu(v), \mu(w))$. Note that T' contains two edges for each edge in C since edges are undirected.

Given two tree representations G_1 and G_2, we define $G_1 \prec G_2$ if $\mathsf{tr}(G_1) \prec \mathsf{tr}(G_2)$. This does not define a total order, since tr is not injective. However, on any set of tree representations with the same number of vertices (and especially on any set of isomorphic tree representations), tr is injective and $\prec$ is total.

Though $\prec$ defines a largest tree representation of any molecular graph, it is impractical to consider every possible such representation in search of this optimum. We therefore restrict to tree representations where the tree edges are identified by iterative addition of longest simple paths that do not create cycles.

Definition 6. *A* pre-tree representation *is a molecular graph $G = \langle V, E, \ell, b\rangle$ where E is a disjoint union $E = T \cup C$ such that the edges of T define a tree (possibly not a spanning tree for G).*

An extension *of G is a simple path $v_1, \ldots, v_n$ such that $v_1 \in T$ and $v_2, \ldots, v_n \in C$. A* longest extension *is one of maximal length among all extensions of G. A* refinement *of G is a pre-tree representation $G' = \langle V, T' \cup C', \ell, b\rangle$, where $T' = T \cup P$ and $C' = C \setminus P$ for a set of edges P of some longest extension of G.*

We can view any molecular graph as a *pre-tree representation* with $T = \emptyset$ and refine it iteratively. Refinements exist whenever there is a vertex that is not reached in T. Hence, a pre-tree representation admits no further refinement exactly if it is a tree representation.

Definition 7. *A* maximal refinement *of a molecular graph G is a tree representation that is obtained from G by a finite sequence of refinements. A* centralized maximal refinement *is a tree representation obtained from a maximal refinement by a permutation of vertices such that the root is central in its spanning tree (analogous to Definition 5). The canonical tree representation of a molecular graph G is its $\prec$-largest centralized maximal refinement.*

In particular, the canonical tree representation coincides with the canonical molecular tree if G is free of cycles.

5 ASP Implementation

Our implementation incorporates many of the conditions on canonical tree representations in the rules that infer these structures, rather than relying on constraints to filter redundant representations later. For molecular trees, our implementation achieves full symmetry breaking along the lines of Definition 5. For graphs with cycles, we merely approximate the conditions from Definition 7, since the required $\prec$-maximality in this case seems to require a computationally prohibitive search in ASP.[3]

As in Definition 3, our implementation identifies vertices with integers, with `1` being the root. Input molecular formulas are encoded in facts `molecular_formula`$(e, f(e))$ and `element`$(e, n_e, V(e))$, for elements $e \in \mathbb{E}$ with atomic number n_e (for usual ordering) and valence $\mathbb{V}(e)$. We encode tree representations $G = \langle V, T \cup C, \ell, b\rangle$ by facts `atom`(v) and `symbol`$(v, \ell(v))$ for $v \in V$; `parent`(v, i, v') for $\{v, v'\} \in T$ where v' is the ith child of v; and `cycle_start`(v, c) and `cycle_end`(v', c) for $\{v, v'\} \in C$ with $v < v'$, where c is a unique integer id for this pair of facts. Multiplicities of bonds $b(e)$ are encoded only if $b(e) > 1$: for $e \in T$, we associate bonds with the child $v = \max(e)$ using `multi_bond`$(v, b(e))$, whereas for $e \in C$, we encode $b(e)$ single-bond cycles for the same e, which showed better performance in this case.

Given the input, we guess facts for `symbol`, `multi_bond`, `cycle_start`, as well as `cycle_end`. For efficiency, we avoid aggregates and instead proceed iteratively, updating counters as we make guesses. We constrain possible guesses based on Definition 2. For example, given a molecular formula f, the number of "additional" bonds used in cycles and multi-bonds $|C| + \sum_{e \in E}(b(e) - 1)$, known as the *degree of unsaturation* in chemistry, and can be computed as $\sum_{e \in \mathbb{E}}^{f(e)>0}(f(e)(\mathbb{V}(e) - 2))/2 + 1$.

When guessing multi-bonds and cycles, we therefore ensure that this number is met. Multi-bonds `multi_bond`$(v,$`2`$)$ or `multi_bond`$(v,$`3`$)$ are guessed for non-root vertices v.[4] For cycle markers, first we guess the number of `cycle_start`s at each

[3] Section 6 shows that the reduction in symmetry is still significant. Remaining isomorphic results might be more efficiently removed by post-processing the set of all answer sets.

[4] Higher bond multiplicities are not implemented in our prototype.

vertex and thereafter generate these facts with their unique cycle ids. Second, we guess the number of cycle_ends at each vertex, making sure to not exceed the total count of cycle_starts at smaller vertices. Using additional constraints, we ensure that each cycle has a single end, start vertices are always smaller than end vertices, cycles never span a single edge e (which should be represented as a multi-bond), and two cycle edges with the same start are indexed according to their end vertex.

This completes the initial guessing phase for ℓ, b, and C. Facts that have the form preset_bonds(v, $pre(v)$) store the number of bonding places $\mathsf{pre}(v)$ that have been used up in the process for vertex v. In the next phase, the program specifies possible spanning trees to establish Definition 2 (1). Choices are limited since we aim at $\prec$-maximal tree representations, e.g., the subtree depth cannot increase from left to right.

We first guess the length of the longest path in the tree representation, whose central elements are the only possible roots by Definition 7. This length is encoded as main_chain_len($length$). It ranges from 1 to $|V|$, but performance is gained by a better lower bound estimate with $X = \max_{e \in \mathbb{E}} \mathbb{V}(e)$ and a tree s.t. $\mathsf{deg}(v) \in \{1, X\} \forall v \in V$:

$$\min\left\{2 \cdot \left\lceil \log_{X-1}\left((X-2) \cdot \frac{N-1}{X} + 1\right)\right\rceil + 1, 2 \cdot \left\lceil \log_{X-1}\left((X-2) \cdot \frac{N}{2} + 1\right)\right\rceil\right\}$$

Next, we iteratively guess depth(v,d), size(v,s), and branching(v,b) for $v \in V$, where b is the number of children of v, and d and s are the depth and size of the subtree with root v. We require $d \leq s$ and $1 \leq b \leq (\mathbb{V}(\ell(v)) - \mathsf{pre}(v))$. Moreover, if $d > 1$ then $b \geq 1$, and $b \geq 2$ for the root 1 unless $|V| \leq 2$. The rules for branching are:

```
branching(1, 1) :- non_hydrogen_atom_count(2).
1{ branching(1, 2..MAX) }1
    :- not branching(1, 1), symbol(1, E), element(E, _, VALENCE),
       MAX = #min{ N-1 : non_hydrogen_atom_count(N);
                   V-B : V=VALENCE, preset_bonds(1, B) }.
1{ branching(I, 1..MAX) }1
    :- symbol(I, E), I>1, element(E, _, VALENCE),
       MAX = #min{ S-D+1 : S=SIZE, D=DEPTH;
                     V-B : V=VALENCE, preset_bonds(I, B) },
       size(I, SIZE), SIZE >= DEPTH, depth(I, DEPTH), DEPTH > 1.
```

At this point, the used-up binding places due to multi_bonds at child vertices are captured in a fact postset_bonds(v, $post(v)$). Definition 2 (3) is equivalent to a check of $\mathsf{pre}(v) + \mathsf{post}(v) \leq \mathbb{V}(\ell(v))$ for each $v \in V$.

Next, we split the main chain evenly between the first two children of root 1, which have indices 2 and $2 + size(2)$. If the length is odd, the first child's depth is greater by 1:

```
depth(2, ((MAIN_CHAIN_LEN-1)+(MAIN_CHAIN_LEN-1)\2)/2)
    :- main_chain_len(MAIN_CHAIN_LEN), MAIN_CHAIN_LEN > 1.
depth(2+LEFT_SIZE, ((MAIN_CHAIN_LEN-1)-(MAIN_CHAIN_LEN-1)\2)/2)
    :- main_chain_len(MAIN_CHAIN_LEN), MAIN_CHAIN_LEN > 2,
       size(2, LEFT_SIZE).
```

In general, the depth of a first child is always set to its parent's depth minus 1. Depths for further children are chosen iteratively to be non-increasing.

```
16 depth(I+1, DEPTH-1)
17     :- depth(I, DEPTH), atom(I+1), branching(I, _), DEPTH > 1.
18 1{ depth(POS_2, 1..PREV_DEPTH) }1
19     :- branching(I, BRANCHING), BRANCHING > CHAIN,
20        depth(POS_1, PREV_DEPTH),
21        parent(I, CHILD_NR, POS_1), parent(I, CHILD_NR+1, POS_2).
```

The first child of a non-final vertex v is always $v + 1$ (line 22 below). Vertex ids for further children are chosen iteratively such that their left neighbor can reach its depth and the parent's size is not exceeded (lines 23–28). These choices also determine the `size` of each child (not shown).

```
22 parent(I, 0, I+1) :- branching(I, _), non_hydrogen_atom_count(N), I < N.
23 1{ parent(I, CHILD_NR, SUM+DEPTH..MAX_CHILD) }1
24     :- parent(I, CHILD_NR-1, SUM), depth(SUM, DEPTH), size(I, PARENT_S),
25        branching(I, BRANCHING), BRANCHING > CHILD_NR,
26        MAX_CHILD = #min{ N : non_hydrogen_atom_count(N);
27                          T : T=I+PARENT_S-BRANCHING+CHILD_NR },
28        MAX_CHILD >= SUM+DEPTH.
```

Next, we materialize the total order $\prec$ from Section 4.1 in a predicate `lt`. For graphs with cycles, we use the number of cycle markers per vertex as an additional ordering criterion instead of the (more costly) tree transformation of Section 4.2. The following constraints exclude cases that cannot be $\prec$-maximal, due to children traversed in $\prec$-increasing order (line 29) or choice of a non-optimal central vertex as root (line 30).

```
29 :- parent(I, CHILD_NR, I1), parent(I, CHILD_NR+1, I2), lt(I1, I2).
30 :- main_chain_len(MAIN_CHAIN_LEN), MAIN_CHAIN_LEN\2 = 0, lt(1, 2).
```

At this point, perfect symmetry breaking for acyclic graphs has been achieved. Cyclic graphs, however, can still have isomorphic representations, since the implementation (a) does not compare all possible choices of main chain, and (b) does not ensure that tree edges are obtained from longest extensions as in Definition 6. For (b), repeated longest path computations are impractical, but we can heuristically eliminate many non-optimal choices by excluding obvious violations.

Definition 8. *Let $G = \langle V, T \cup C, \ell, b\rangle$ be a tree representation with cycle edge $e = \{v_1, v_2\}$. Let $P = v_1 \dots v_2$ be the unique path in G that consists only of tree edges. Say that e is shortening, if an $e' \in P$ exists, s.t. $G' = \langle V, T' \cup C', \ell, b\rangle$ with $T' = (T \setminus \{e'\}) \cup \{e\}$ and $C' = (C \setminus \{e\}) \cup \{e'\}$ is deeper.*

Note that, for any G, $\min_{\prec}[G]_{\cong}$ cannot have shortening cycle edges. Hence, the symmetry-breaking remains correct when applying a heuristic to forbid them. The ASP implementation detects shortening cycle edges by a pattern-matching approach.

6 Experimental Evaluation

We evaluate our ASP implementation ("Genmol") for correctness, avoidance of redundant solutions, and runtime. All of our experiments were conducted on a mid-end server (2×QuadCore Intel Xeon 3.5GHz, 768GiB RAM, Linux NixOS 23.11) using Clingo v5.7.1 for ASP reasoning. Evaluation data, scripts, and results are online at `https://github.com/knowsys/eval-2024-asp-molecules`.

Evaluated Systems. The ASP-based core of our system Genmol consists of 174 rules (including 44 constraints).[5] As a gold standard, we use the existing commercial tool Molgen (`https://molgen.de`), which produces molecular graphs using a proprietary canonicalization approach. Moreover, we compare our approach to three ASP-based solutions: Naive is a direct ASP encoding[6] of Definition 2, which serves as a baseline; Graph refines Naive with graph-based symmetry-breaking by [1]; and BreakID is the system by [3], which adds symmetry-breaking constraints automatically to the grounding of Naive. Conceptually, BreakID is based on symmetry breaking for SAT [4].

For Graph, we adapt Definition 12 of [1], which applies to partitioned simple graphs G that are represented by their adjacency matrix $\mathcal{A}_G$:

$$\mathrm{sb}(G) = \bigwedge_{e\in\mathbb{E}} \bigwedge_{\substack{\ell(i)=\ell(j)=e,\\ i<j,\, j-i\neq 2}} \mathcal{A}_G[i] \preceq_{\{i,j\}} \mathcal{A}_G[j]. \tag{1}$$

Here, $\preceq_{\{i,j\}}$ denotes the lexicographic order comparing the ith and jth row of the adjacency matrix $\mathcal{A}_G$ of a molecular graph G, ignoring columns i and j. Graph representations that do not satisfy (1) are pruned. These constraints can be succinctly represented in ASP, and are appended to the naive implementation.[7]

Data set. For evaluation, we have extracted a dataset of molecules with molecular formulas and graph structures using the Wikidata SPARQL service [10]. We selected 19,287 chemical compounds with SMILES and an article on English Wikipedia (as a proxy for practical relevance). Due to performance constraints, we focus on the 8,980 compound subset of up to 17 atoms. Compounds with unconnected molecular graphs, atoms of non-standard valence, and subgroup elements were excluded, resulting in a dataset of 5,625 entries, of which we found 152 to have non-parsable SMILES.

Evaluation of Correctness. Given the complexity of the implementation, we also assess its correctness empirically. To this end, we augment our program with ASP rules that take an additional direct encoding of a molecular graph as input, and that check if the molecular graph found by Genmol is isomorphic to it. This allows us to determine if the given structures of molecules in our data set can be found in our tool. The validation graph structure is encoded in facts `required_bond`$(v_1, \ell(v_1), v_2, \ell(v_2), b(\{v_1, v_2\}))$ that were extracted from the SMILES representation in Wikidata.

Correctness experiments were measured with a timeout of 7 minutes. Out of 5,473 compounds, a matching molecular structure was found for 5,338, whereas 132 could not be processed within the timeout. For three compounds, *Sandalore* (Wikidata ID Q21099635), and *Eythrohydrobupropion* (Q113691142) as well as *Threodihydrobupropion* (Q72518680), the given structures could not be reproduced, which we traced back to errors in Wikidata that we have subsequently corrected.

[5] `https://github.com/knowsys/eval-2024-asp-molecules/blob/main/smiles.lp`
[6] `https://github.com/knowsys/eval-2024-asp-molecules/blob/main/naive.lp`
[7] `https://github.com/knowsys/eval-2024-asp-molecules/blob/main/lex.lp`

The evaluation therefore suggests that Genmol can find the correct molecular structures across a wide range of actual compounds. Timeouts occurred primarily for highly unsaturated, larger compounds (over 16 atoms), where millions of solutions exist.

Evaluation of Symmetry Breaking. To assess to what extent our approximated implementation of canonical tree representations succeeds in avoiding redundant isomorphic solutions, we consider the smallest 1,750 distinct molecular formulas from our data set. We then computed molecular graph representations for all 1,750 cases for each of our evaluated systems, using Molgen as a gold standard to determine the actual number of distinct molecular graphs. The timeout for these experiments was 60 seconds. The

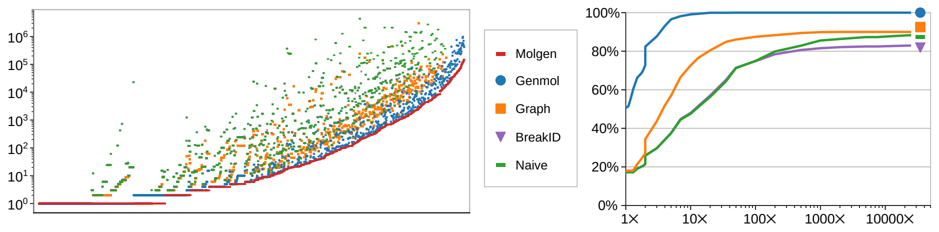


Fig. 3: Number of models for each compound in the data set (left) and ratio of compounds with model counts within a factor of the gold standard (right)

number of returned solutions are shown in Figure 3 (left), with samples sorted by their number of distinct graphs according to Molgen. As expected, Molgen is a lower bound, and in particular no implementation finds fewer representations (which would be a concern for correctness), while Naive is an upper bound. As expected, no ASP tool achieves perfect canonization of results, but the difference between the number of solutions and the optimum vary significantly. In particular, BreakID rarely improves over Naive (just 24 such cases exist), though it does cause one third more timeouts.

For Naive, some samples led to over 20,000 times more models than Molgen, whereas the largest such factor was just above 39 for Genmol (for C_8H_2). Figure 3 (right) shows the ratio of samples with model counts within a certain factor of the gold standard. For example, the values at 10 show the ratios of samples for which at most ten times as many models were computed than in Molgen: this is 99% for Genmol, 72% for Graph, and 48% for BreakID and Naive. All ratios refer to the same total, so the curves converge to the ratio of cases solved within the timeout. Their starting point marks the ratio with exact model counts: 51% for Genmol and 17% – 18% for the others.

We conclude that symmetry breaking in Genmol, though not perfect, performs very well in comparison to generic approaches. In absolute terms, the results might be close enough to the optimum to remove remaining redundancies in a post-processing step.

Performance and Scalability To assess the runtime of our approach, we conduct experiments with series of uniformly created molecular formulas of increasing size. We consider two patterns: formulas of the form $C_nH_{2n+2}O$ belong to tree-shaped molecules (such as ethanol with SMILES `OCC`), whereas formulas of the form $C_nH_{2n}O$ require one cycle (like oxetane, `C1COC1`) or double bond (like acetone, `CC(=O)C`). We use a timeout of 10min for all tools except Molgen, whose free version is limited to 1min runtime. All runs are repeated five times and the median is reported.

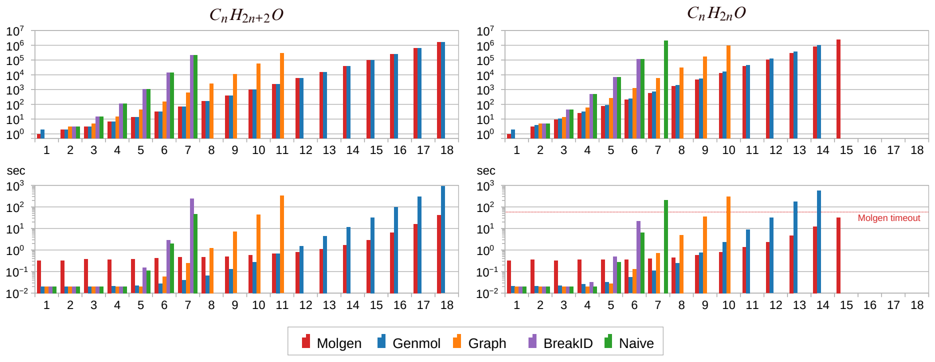


Fig. 4: Model numbers (top) and runtimes (bottom) for molecules of increasing size

The results are shown in Figure 4. As before, Molgen serves as a gold standard. As seen in the graphs on the top, the number of distinct molecular structures grows exponentially, and this optimum is closely tracked by Genmol (perfectly for the tree-shaped case). Graph reduces model counts too, albeit less effectively, whereas BreakID does not achieve any improvements over Naive in these structures.

As expected, the runtimes indicate similarly exponential behavior as inputs grow, but the point at which computation times exceed the timeout is different for each tool. Molgen achieves the best scalability overall, whereas Genmol is most scalable among the ASP-based approaches. BreakID is even slower than Naive, largely due to longer solving times, whereas the preprocessing of the grounding had no notable impact.

7 Conclusion

Motivated by the practical problem of interpreting results in mass spectrometry, we developed an ASP-based approach for enumerating molecular structures based on partial information about their chemical composition. We focused on molecular formulas as inputs, but our prototype Genmol can also use additional signals, such as known molecular fragments. Indeed, one of the main strengths of an ASP-based solution is that it is very simple to refine the search space with additional constraints. Our extensive evaluation showed that our approach improves upon direct ASP-based solutions and existing optimizations by several orders of magnitude, regarding both performance and conciseness, bringing it into the range of optimized commercial implementations that (presumably) lack the flexibility of ASP.

In general, we believe that our conceptual work towards canonical graph representations and their efficient realization in ASP is relevant beyond our motivating application scenario. Most of our ideas can be readily generalized to other graph-related search problems, and may therefore be of interest to ASP practitioners. Moreover, our real-world evaluation data can be a valuable benchmark for further research on automated symmetry-breaking in ASP. In the future, we hope that our approach can be augmented with (possibly ASP-based) post-processing that eliminates remaining redundancies.

Acknowledgments. This work is supported by Deutsche Forschungsgemeinschaft in project number 389792660 (TRR 248, CPEC), by the Bundesministerium für Bildung und Forschung under European ITEA project 01IS21084 (InnoSale), in the Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI), and by BMBF and DAAD in project 57616814 (SECAI).

Disclosure of Interests The authors have no further competing interests to declare.

References

1. Codish, M., Miller, A., Prosser, P., Stuckey, P.J.: Constraints for symmetry breaking in graph representation. Constraints **24**(1), 1–24 (2019). `https://doi.org/10.1007/s10601-018-9294-5`
2. De Hoffmann, E., Stroobant, V.: Mass spectrometry: principles and applications. John Wiley & Sons (2007)
3. Devriendt, J., Bogaerts, B.: BreakID: Static symmetry breaking for ASP (system description). CoRR **abs/1608.08447** (2016), `http://arxiv.org/abs/1608.08447`
4. Devriendt, J., Bogaerts, B., Bruynooghe, M., Denecker, M.: Improved static symmetry breaking for SAT. In: Proc. 19th Int. Conf. Theory and Applications of Satisfiability Testing (SAT'16). LNCS, vol. 9710, pp. 104–122. Springer (2016). `https://doi.org/10.1007/978-3-319-40970-2_8`
5. Devriendt, J., Bogaerts, B., Bruynooghe, M., Denecker, M.: On local domain symmetry for model expansion. Theory Pract. Log. Program. **16**(5-6), 636–652 (2016). `https://doi.org/10.1017/S1471068416000508`
6. Drescher, C., Tifrea, O., Walsh, T.: Symmetry-breaking answer set solving. AI Communications **24**(2), 177–194 (2011). `https://doi.org/10.3233/AIC-2011-0495`
7. Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., Schneider, M.: Potassco: the Potsdam answer set solving collection. AI Commun. **24**(2), 107–124 (2011). `https://doi.org/10.3233/AIC-2011-0491`
8. Gugisch, R., Kerber, A., Kohnert, A., Laue, R., Meringer, M., Rücker, C., Wassermann, A.: MOLGEN 5.0, a molecular structure generator. In: Advances in Mathematical Chemistry and Applications, chap. 6, pp. 113–138. Elsevier (2015)
9. Khaled, T., Benhamou, B.: Symmetry breaking in a new stable model search method. In: LPAR-22 Workshop and Short Paper Proceedings. Kalpa Publications in Computing, vol. 9, pp. 58–74. EasyChair (2018). `https://doi.org/10.29007/1l5r`
10. Malyshev, S., Krötzsch, M., González, L., Gonsior, J., Bielefeldt, A.: Getting the most out of Wikidata: Semantic technology usage in Wikipedia's knowledge graph. In: Proc. 17th Int. Semantic Web Conf. (ISWC'18). LNCS, vol. 11137, pp. 376–394 (2018)
11. Tarzariol, A., Gebser, M., Schekotihin, K., Law, M.: Learning to break symmetries for efficient optimization in answer set programming. In: Proc. 35th AAAI Conf. on Artificial Intelligence (AAAI'23). pp. 6541–6549. AAAI Press (2023). `https://doi.org/10.1609/aaai.v37i5.25804`
12. Weininger, D.: SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. J. Chem. Inf. Comput. Sci. **28**(1), 31–36 (1988). `https://doi.org/10.1021/ci00057a005`