Symbolic Neural Generation with Applications to Lead Discovery in Drug Design

Ashwin Srinivasan [iD]   ASHWIN@GOA.BITS-PILANI.AC.IN
Department of CS & IS & APPCAIR, BITS Pilani, K.K. Birla Goa Campus, Goa, India

A. Baskar [iD]   ABASKAR@GOA.BITS-PILANI.AC.IN
Department of CS & IS, BITS Pilani, K.K. Birla Goa Campus, Goa, India

Tirtharaj Dash [iD]   TD522@CAM.AC.UK
Department of Biochemistry, University of Cambridge, UK

Michael Bain [iD]   M.BAIN@UNSW.EDU.AU
School of Computer Science and Engineering, University of New South Wales, Sydney, Australia

Sanjay Kumar Dey [iD]   SKDEY@ACBR.DU.AC.IN
Dr. B.R. Ambedkar Center for Biomedical Research, University of Delhi, New Delhi, India

Mainak Banerjee [iD]   MAINAK@GOA.BITS-PILANI.AC.IN
Department of Chemistry, BITS Pilani, K.K. Birla Goa Campus, Goa, India

Abstract

We investigate a relatively under-explored class of hybrid neurosymbolic models that integrate symbolic learning with neural reasoning to construct data generators meeting formal correctness criteria. In *Symbolic Neural Generators* (SNGs), symbolic learners examine logical specifications of feasible data from a small set of instances—sometimes just one. Each specification in turn constrains the conditional information supplied to a neural-based generator, which rejects any instance violating the symbolic specification. Like other neurosymbolic approaches, SNG exploits the complementary strengths of symbolic and neural methods. The outcome of an SNG is a triple (H, X, W), where H is a symbolic description of feasible instances constructed from data, X a set of generated new instances that satisfy the description, and W an associated weight. We introduce a semantics for such systems, based on the construction of appropriate *base* and *fibre* partially-ordered sets combined into an overall partial order, and outline a probabilistic extension relevant to practical applications. In this extension, SNGs result from searching over a weighted partial ordering. We implement an SNG combining a restricted form of Inductive Logic Programming (ILP) with a large language model (LLM) and evaluate it on early-stage drug design. Our main interest is the description and the set of potential inhibitor molecules generated by the SNG. On benchmark problems – where drug targets are well understood – SNG performance is statistically comparable to state-of-the-art methods. On exploratory problems with poorly understood targets, generated molecules exhibit binding affinities on par with leading clinical candidates. Experts further find the symbolic specifications useful as preliminary filters, with several generated molecules identified as viable for synthesis and wet-lab testing.

Keywords: Neurosymbolic Modelling, Symbolic Neural Generation, Inductive Logic Programming, LLMs, Drug Discovery

1 Introduction

Consider the following real problem:

> We are interested in generating a set of small molecules that can bind to a known protein. We know something about the protein: its amino-acid sequence, its role in causing some disease, an approximate section of the sequence where the small molecule should bind, and so on. We also know 5 small molecules (inhibitors) that are known to bind to this protein, of which 1 is known to have toxic side-effects. We also know, from our previous chemical knowledge, that to be easily synthesised we would like molecules that contain a particular scaffold, whose molecular weights are not too high or too low, predicted toxicity values are as low as possible, and predicted binding affinity to the protein is as high as possible. Additional constraints on the molecule are not known, since the 3 dimensional structure of the target site has only recently become available. Can we generate 10 additional possible inhibitors?

Conceptually, this can be seen as a special case of the general problem of identifying elements of a set:

$$\mathcal{X} = \{x : x \in \mathcal{U}, \Phi(x) \text{ is true}\}$$

where $\mathcal{U}$ is a set consisting of all possible instances of relevance – small molecules, for instance – and $\Phi(x)$ is a predicate that is true of the specific instances of interest. Given the remarkable abilities of large pre-trained models, if $\Phi(\cdot)$ is known, it is conceivable that we may indeed be able to generate molecules directly. Neural-generators are stochastic, and some of the generated molecules may not actually satisfy $\Phi(\cdot)$, but we can resort to some kind of rejection-sampling until we meet our requirement. The obvious difficulty is, of course, that for many problems, $\mathcal{U}$ can be very large (for example, there may be up to 10^{60} small molecules), and rejection-sampling can become quite inefficient. The real issue though is that for most complex real problems, we do not know Φ. Some options we could try are:

- We could consider fine-tuning an existing generative model with the data instances we already know. This runs into the difficulty that the data are often observations of phenomena that are rare, and therefore we usually have very few instances – in the order of 10s, rather than the several 100s or 1000s needed for effective fine-tuning. In the event, we will probably be left with a generator that is largely unaffected by the few data instances we have;

- We could consider prompt-engineering for an LLM. That is, assuming there exists some description p of the set $\mathcal{X}$ for some LLM λ, we attempt to find p by trial-and-error. There are two issues here: it is not clear that the text-based description p is any easier to identify than the formal description Φ; and it is not clear that it would be any more precise;

- We could consider exploiting the few-shot learning ability of an LLM (Brown et al., 2020a) by providing the data instances we know as part of the context information for the LLM. But, how do we then know whether the generated instances are indeed

from the set of interest? In-context learning usually works best within an iterative loop that updates the context with the result of some validation mechanism providing feedback. What would this be here? For specialised problems, human feedback may be both difficult to obtain and unlikely to be as helpful as with routine conversations.

The issue is this: with modern pre-trained large neural models capable of approximating a vast range of probability distributions, the difficulty is in finding verifiable ways of constraining them to 'focus'. However, this still begs the question of whether, *in principle*, the task can be achieved using a purely symbolic or purely connectionist approach ("unified" approaches in Hilario (2013); Hilario et al. (1994)).

In principle, the answer is "yes". For a symbolic generator, we would need to identify a symbolic description S to include probability values over the elements of $\mathcal{U}$: logic programs with a distributional semantics like Sato and Kameya (1997) or De Raedt et al. (2007) are examples of this. The symbolic description can then be directly used for generating instances in by sampling. The difficulty, however, is that symbolic representations are usually not fine-grained enough to capture very complex probability distributions; which are better approximated with the high-dimensional real-vector encodings (embeddings) used by neural networks. This is especially apparent with using language models for accessing complex conditional probability distributions.

What about a purely connectionist approach? Again, in principle, a neural network can encode a predicate such as $\Phi(\cdot)$. The difficulty here is that this encoding becomes increasingly more approximate if obtained from very few data samples (large language models have proved remarkably successful at identifying appropriate text-responses with very few examples; it is not clear as yet if this extends robustly to logical formulae). The lack of a precise, verifiable formal description from a neural model also presents a challenge. Of course, if human-understandability of the description is also required, then we face well-known difficulties with a neural encoding. Thus, the continued interest in hybrid systems is not for theoretical but for practical reasons. Hybrid systems can be engineered from existing modules; are easier to maintain, and are more amenable to controlled studies, especially if each component is largely concerned with a specific function.

In this paper, we propose using a symbolic component that examines possible approximations for $\Phi(\cdot)$ by constructing definitions for a predicate $\Sigma(\cdot)$. Each such definition is accompanied by a set of instances obtained from a neural generative model for which $\Sigma(\cdot)$ is true. Figure 1 shows the outcome we would like ideally, and what we would have to settle for in practice. This requires us to deal with a *dual-alignment* problem: we would like the symbolic component to find an S that aligns well with $\mathcal{X}$; and a neural generator that identifies an N that aligns well with S (denoted by the set X). This combination using a symbolic learner and a neural-based generator constitutes a *Symbolic Neural Generator*, or SNG.[1] Figure 1 shows what we would like ideally, and what often results in practice. In the taxonomy proposed in Hilario (2013), SNGs are a *hybrid* neurosymbolic[2] approach since they contain distinct neural and symbolic components.

---

1. We will sometimes use SNG to stand for the computation process of *symbolic neural generation*, with the context making it clear whether we mean a system or the process.

2. There have been several variations of this term, such as "neural-symbolic" (d'Avila Garcez et al., 2009). Current usage seems to have converged on the single word "neurosymbolic" (d'Avila Garcez and Lamb, 2023).
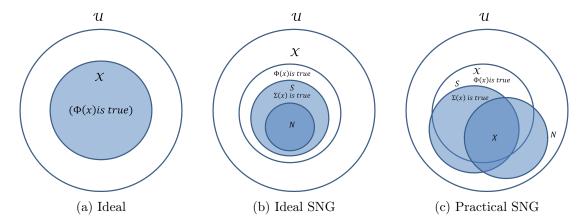
Figure 1: (a) Ideally, we would like to generate instances from the set of instances for which $\Phi(x)$ is true; (b) When $\Phi(\cdot)$ is not known, we approximate $\Phi(\cdot)$ by $\Sigma(\cdot)$, obtained using the hypothesis from a symbolic learner. We want to sample instances efficiently from S. N is the set of instances obtained from a neural-based generator. For an ideal SNG, $N \subseteq S \subseteq \mathcal{X}$; (c) In practice, the symbolic learner may not be perfect, and the neural-generator only has an approximate model of the conditional distribution. The set X is the set of instances generated that are in S.

We see the paper contributing in the following ways to research into neurosymbolic systems:

- We identify a class of hybrid neurosymbolic systems – called Symbolic Neural Generators, or SNGs – which draws on the strengths of symbolic learning and neural-based generative models. Potential symbolic descriptions are examined, and at the same time: (a) act as conditioning information for neural-based encodings of probability distributions; and (b) verify samples of new data instances generated by the neural component. The output of an SNG is a human-readable symbolic description and a set of instances consistent with that description.

- We provide a novel semantic framework for SNGs based on *poset semantics*. In this formulation, symbolic and neural components are jointly organized through a partial order that captures refinement relations among the individual components. The semantics is formalized using a categorical construction: the overall hybrid system is represented as a *Grothendieck construction* over an indexed family of partially-ordered sets. The construction yields a unified space linking symbolic and neural generation.

- We implement and test SNGs on the real-world problem of generating potential inhibitors for protein-targets. This constitutes the problem of 'lead-discovery', and realistic settings have the following charactestics. Given: (a) a few known inhibitors (in our main case study, only 5), and (b) some amount of prior biological and chemical knowledge; we want: (c) human-readable constraints on potential new inhibitors, and (d) proposals for potentially new inhibitors for the target. Our experiments show that SNG performs creditably on benchmarks from the literature, and – on an open problem – provides results that are understandable and interesting to a synthetic chemist and a structural biologist.

2 Background: Hybrid Neurosymbolic Modelling

The idea of neurosymbolic modelling for AI is not new; as has often been noted, the seminal paper of McCulloch and Pitts (1943) rests on the observation that "the activity of any neuron can be represented as a proposition". Thus arises an entire class of systems comprising a purely connectionist approach that approximate the patterns of categoric inference arising when using (often propositional, but not always) logical representations, as well as the patterns of plausible inference that arise when using probabilistic representations. There is now a very large literature on this kind of system – called a *unified* neurosymbolic system in Hilario (2013); Hilario et al. (1994) – and this forms a prominent part of the current landscape of neurosymbolic AI. We will not attempt an exhaustive review of unified neurosymbolic literature. For extensive treatments of that see: (d'Avila Garcez et al., 2009; Besold et al., 2017; d'Avila Garcez and Lamb, 2023; De Smet and De Raedt, 2025; Derkinderen et al., 2025) We note that while much of the work focuses on engineered solutions, we are now seeing the emergence of more abstract, mathematical perspectives on what represents a unified neurosymbolic system. Thus, Odense and d'Avila Garcez (2025) are concerned with identifying the conditions of semantic equivalence between architectures and representations; and De Smet and De Raedt (2025) provides the mathematical underpinning of a uniform inference mechanism for reasoning in systems containing both symbolic and neural components. Also under the category of unified systems, we do not present here any of the vast array of modelling and inference methods that continue to be developed using probability theory of continuous and discrete random variables and their associated distributions. Ultimately under the category of unified symbolic approaches, it is possible to provide a probabilistic semantics to (conditional and unconditional) generative models, irrespective of whether distributions are encoded by neural networks, probabilistic programs, graphical models or the like (we refer the reader to Murphy (2022), Chs. 20–28 for an extensive treatment) and to Marra et al. (2024) for the related area of statistical relational learning.

In this paper, we are concerned with the alternative form of *hybrid neurosymbolic systems* that contains distinct neural and symbolic components. It is helpful to categorise hybrid systems based on the the principal function of each component in the system (Fig. 2).[3] SNG as we propose it refers to a sub-class of systems in Category (B).

We are concerned with a restricted class of hybrid neurosymbolic systems whose task it is to 'generate' new data instances using (models for) conditional or joint probability distributions. In the rest of this section, we will highlight some recent relevant work that satisfies these criteria, in the categories shown in Fig. 2a(a).

An emerging trend for generative models in Category A (Neural-Reasoning, Symbolic-Reasoning) is the use of pre-trained LLMs that invoke "tools" (Kautz, 2024) to verify correctness of instances generated by the LLM. In particular, such hybrid systems can invoke reasoning systems – theorem-provers, for example – implementing sound inference in symbolic logic (Cheng et al., 2025). For example, in a system like LINC (Olausson et al., 2024) an LLM is used to translate problem statements in natural language to expressions in first-order logic, and then attempts to find proofs using a theorem prover. LLMs with tools are able to

---

3. In practice, neither component does just one function or the other. For example, any non-trivial learning (induction) usually requires reasoning (deduction), and any sufficiently complex reasoning would usually be accompanied by some trial-and-error learning. Therefore, this categorisation is necessarily a simplification.

Figure 2: (a)

	Symbolic	
	Reason	*Learn*
Reason	(A)	(B)
Learn	(C)	(D)

Neural

(b)	
Category	Examples
(A)	Neural theorem proving; Plan verification;
	Safe neural controllers;
	Neural proof search
(B)	Grammar learning;
	Learning explanations for neural behaviour
	ILP with neural inference;
	SNG (this paper)
(C)	Schema learning with ontologies;
	Deep RL with symbolic constraints
	Semantic-loss based classification;
	Design generation with spatial constraints
(D)	Hybrid KG construction;
	Differentiable ILP;
	Symbolic concepts from neural representations
	Program synthesis with learned primitives

Figure 2: (a) A categorisation of hybrid neurosymbolic systems that consist of distinct Neural and Symbolic components. The primary role of each component is to *Learn* or to *Reason*; (b) Examples of neurosymbolic systems in each category.

exceed the formal reasoning capabilities of LLMs, which can struggle on logical reasoning tasks without additional training (Li et al., 2024; Qi et al., 2025). The AlphaGeometry2 system (Chervonyi et al., 2025) uses a language model to take in natural language statements of mathematical problems and generate formal expressions of problem facts in a domain-specific language for IMO geometry problems. A deductive database algorithm computes the closure of these facts, which is searched in parallel for solutions, with a language model used to generate the proofs. Closely related to the work here, the mechanism of language models with logical-feedback (LMLF) in Brahmavar et al. (2024) can be seen as an LLM equipped with a symbolic reasoner. The symbolic component performs two tasks: (a) using prior knowledge it progressively deduces new constraints that in turn modify the context of the LLM; and (b) it verifies that instances generated by the LLM satisfy the constraints. LLMs are used here for generation and to provide explanations in a natural or specialised language. The symbolic reasoner can also provide additional justification, by displaying the reasoning steps used to verify the output generated by the LLM.

While there is a relative paucity of generative hybrid systems in Category B (Neural-Reasoning, Symbolic-Learning), some special-purpose systems do exist. For example, building on earlier work in which pre-defined grammars were used (Kusner et al., 2017), a symbolic generative grammar defined on molecular hypergraphs was trained utilising probabilistic learning in Guo et al. (2022). The approach uses bottom-up grammar learning, where a molecule, represented as a hypergraph on the left-hand side of a production rule, generates a new molecule (by adding a hyperedge) on the right-hand side. A pre-trained neural network is used as a molecular feature selector and a probabilistic model is conditioned on the data using Monte Carlo sampling and gradient ascent to maximize score across molecular metrics. Molecules are then probabilistically sampled from this grammar, providing a data-efficient (i.e., it learns from a few molecules) and interpretable molecular generator.

More recently, a pre-trained multi-modal foundation model was used to guide the symbolic grammar learning, based on prompting and neural reasoning, which is then used to generate molecules (Sun et al., 2025). Although a symbolic theory is learned, both learning and reasoning are done by neural systems, and this recent work should be treated as a unified neurosymbolic system. Within robot planning, actions can be modelled as sets of predicates on the environment that represent pre-conditions or post-conditions. In Liang et al. (2025) an LLM generates predicates corresponding to a robot trajectory in the environment, where a plan either succeeded or failed. Generated predicates are converted to Python code, filtered against an API and used in a search for state abstractions. This corresponds to a form of program synthesis, in which a causal process world model for planning is learned using a probabilistic approach. This approach can be characterised as neural reasoning (generation) and symbolic (probabilistic) learning.

In contrast, there is more done on general-purpose generative hybrid approaches in Category C (Neural-Learning, Symbolic-Reasoning). For example, diffusion models are widely used to generate images, but can also be applied to generate discrete outputs such as language, or molecules as SMILES strings. However it is difficult to control generation to ensure certain requirements on the outputs are satisfied. A neurosymbolic approach to diffusion is in Christopher et al. (2025). To check for toxicity in molecule generation several black-box filters implemented in RDKit were selected. Their outputs are then used in constraints in a convex optimization solver which is integrated into the diffusion process. SPRING (Jacobson and Xue, 2025) is a related approach which incorporates a symbolic spatial reasoning module into diffusion-type models. A user-defined design language on spatial predicates and object relations enables greater control of generated designs.

Also in Category C is the important class of neural networks trained to generate molecules that are constrained by symbolic specifications, such as target properties of the molecule, or grammars or graph structures capturing chemical knowledge, e.g., Lim et al. (2018); Liu et al. (2018); Dash et al. (2021). This category also includes pre-trained LLMs fine-tuned with data expressed in molecular languages such as SMILES, and constrained to ensure correct generation with symbolic models (Zhang et al., 2025). A recent approach (Zhou et al., 2025) uses molecular analysis tools, some based on pre-trained statistical machine learning, to generate a range of relevant properties which are converted to text using natural language templates. These molecule-text pairs are added to the training set to fine-tune an LLM, which can then generate new molecules from text prompts with high probability of having the relevant properties.

Category D (Neural-Learning, Symbolic-Learning) represents, in some sense, the most complex category of hybrid neurosymbolic systems in Fig. 2a. Joint learning of neural and symbolic models is also a feature of Dai and Muggleton (2021): although neither model is generative, there is no reason in principle why meta-interpretive learning (the approach used in that paper) cannot be used to learn generative neural models guided by symbolic hypotheses. In principle, the unified framework of De Smet and De Raedt (2025) allows the joint learning of neural and symbolic models. It is unclear whether the related implementation DeepLog (Derkinderen et al., 2025) – not to be confused with the identically named symbolic learner described in Muggleton (2023) – has the operations required to achieve this, but we do not see any difficulty in principle.

3 Symbolic Neural Generation

Consider again the problem of identifying instances from the set:

$$\mathcal{X} = \{x : x \in \mathcal{U}, \Phi(x) \text{ is true }\}$$

with the condition that $\Phi(\cdot)$ is not known. However, we do have some instances of $\mathcal{U}$ for which membership in $\mathcal{X}$ (or otherwise) *is* known. There may also be some problem-specific background knowledge that may be of help. In symbolic neural generation, we approach this problem by using what we know to examine an approximation $\Sigma(\cdot)$ to $\Phi(\cdot)$ and using this to constrain the subsequent selection of instances using a neural-based sampler. This two-component approach can be seen as an instance of a hybrid system in Category B of the previous section. We will now attempt a clearer understanding of this kind of system.

3.1 Poset Semantics for SNGs

We present a simple semantics for hybrid neurosymbolic systems that perform symbolic neural generation. Later we suggest alternatives and extensions that apply to wider classes of hybrid neurosymbolic systems.

Let $\mathcal{U}$ denote a fixed finite universe. Let B be background knowledge and let $\mathcal{H}$ be a family of symbolic hypotheses. Assume each hypothesis H in $\mathcal{H}$ has a definition of a fixed unary predicate Σ which is defined over $\mathcal{U}$.

Definition 1 (Extension of H) *Let $H \in \mathcal{H}$. The semantic extension of H given B, denoted by $\mathrm{ext}(H|B)$, is the set $\{x : x \in \mathcal{U}, B \wedge H \models \Sigma(x)\}$. We will call this simply the extension of H and denote it by $\mathrm{ext}(H)$ if B is clear from the context.*

We use the constraints required by a *Grothendieck construction* to specify the structure of SNGs. For this, we first identify the *base poset* $(\mathcal{H}, \leq_{\mathcal{H}})$.

Definition 2 (Ordering over $\mathcal{H}$) *Let $B \in \mathcal{B}$ be background knowledge and $H_1, H_2 \in \mathcal{H}$ be hypotheses. We define $H_1 \geq_{\mathcal{H}} H_2$ as $\mathrm{ext}(H_1|B) \supseteq \mathrm{ext}(H_2|B)$.*

Proposition 1 $(\mathcal{H}, \geq_{\mathcal{H}})$ *is a partially ordered set.*

Proof Follows trivially from the reflexive, anti-symmetric and transitive properties of subset-inclusion. ∎

We will call $(\mathcal{H}, \geq_{\mathcal{H}})$ the *base* poset. For each $H \in \mathcal{H}$ we associate a *fibre-poset* which depends on the neural component. For the present, let the neural generator be a function $f_N : \mathcal{H} \times Z \to \mathcal{U}$ where Z is some latent space, and let

$$N(H) = \mathrm{im}(f_N(H)) = \{f_N(H, z) : z \in Z\} \subseteq \mathcal{U}$$

That is, $N(H)$ is a the set of all instances that can be generated in principle by the neural component using the hypothesis H.[4] Later we consider extensions that account for more subtle variants.

---

4. Thus, if the neural component is a stochastic model that defines a probability distribution P_N over $\mathcal{U}$, $N(H) = \{u \in \mathcal{U} : P_{N(H)}(u) > 0\}$.

Definition 3 (Fibred Poset of a base element) *For each $H \in \mathcal{H}$, we define a corresponding fibre-poset:*

$$F(H) = \left(2^{N(H) \cap \mathrm{ext}(H|B)}, \subseteq\right).$$

The fibre-posets of pairs of elements in the base lattice are related by the canonical inclusion (restriction) reindexing map for $H_1 \geq_{\mathcal{H}} H_2$:

$$\rho_{H_1,H_2} : F(H_1) \to F(H_2), \qquad \rho_{H_1,H_2}(S) = S \cap \mathrm{ext}(H_2 \mid B)$$

We note that whenever $H_1 \geq_{\mathcal{H}} H_2$ we have $\mathrm{ext}(H_2 \mid B) \subseteq \mathrm{ext}(H_1 \mid B)$, so these maps are monotone and satisfy the functoriality condition $\rho_{H_2,H_3} \circ \rho_{H_1,H_2} = \rho_{H_1,H_3}$.

Definition 4 (Total Poset) *Given a base-poset and fibre-posets associated with each element of the base-set, we define a total poset using the Grothendieck construction (Leinster, 2014):*

$$\mathcal{F} = \{(H, X) \mid H \in \mathcal{H}, \ X \subseteq N(H) \cap \mathrm{ext}(H \mid B)\},$$

equipped with the order

$$(H_1, X_1) \geq_{\mathcal{F}} (H_2, X_2) \quad \Longleftrightarrow \quad H_1 \geq_{\mathcal{H}} H_2 \text{ and } X_1 \supseteq X_2$$

A diagrammatic representation of these sets is shown in Fig. 3.
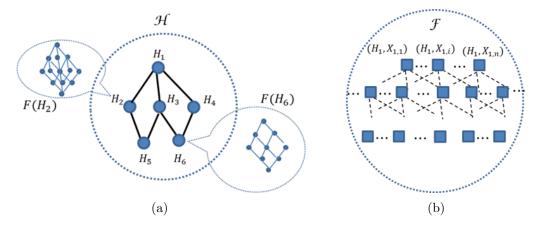

Figure 3: (a) Posets indexed by elements of a base poset $\mathcal{H}$. Each element H of the base poset is associated with a fibre-poset $F(H)$. (b) The Grothendieck construction combines the base poset and the fibre-posets to form a single total poset $\mathcal{F}$, that consists of pairs of elements. An element $(H, X) \in \mathcal{F}$ is such that $H \in \mathcal{H}$ and $X \in F(H)$.

How this is relevant: Elements of $\mathcal{F}$ are the candidate neural–symbolic pairs produced by the SNG system. The order combines symbolic generality ($\geq_{\mathcal{H}}$) and empirical support (X's), or *support-sets*, obtained from the neural component. The ordering $\geq_{\mathcal{F}}$ allows us to organise the space of possible neurosymbolic systems of this kind.

EXTENSION TO WEIGHTED POSETS

We now extend the deterministic setting used so far to account for two sources of uncertainty: (a) The hypotheses in the base may not all be equally likely, given data and background knowledge; (b) In practice the neural generator is stochastic and the support-sets are samples to which we can attach a probability (of obtaining the sample under the sampling distribution conditioned by H and B). In addition to the sets introduced, let us assume we are also given a set $E \subseteq \mathcal{U}$ (usually a set of known observations). Suppose we now wish to assign weights to each pair (H, X). There are two conceptually distinct sources of probability: (i) Plausibility of the output of the symbolic component, captured by a posterior $P(H \mid E, B)$; and (ii) Empirical or generative weight, denoted $w(H, X)$ arising from the stochastic aspect of the neural component. This reflects how much of the neural-generated data agrees with the symbolic hypothesis H which it supports. We formalise a simple probabilistic extension that combines these two sources of uncertainty.[5]

Definition 5 (Probabilistic Extension) *Let $\mathcal{H}$ be a hypothesis class, B background knowledge, and E denote initial data. Given a $H \in \mathcal{H}$ and an associated support-set $X \subseteq N(H) \cap \mathrm{ext}(H \mid B)$, let $w(H, X)$ be the empirical weight arising from the neural-generator and $P(H \mid E, B)$ be the posterior probability arising from the symbolic learner. We associate the following combined weight with (H, X):*

$$W(H, X) \;=\; P(H \mid E, B) \cdot w(H, X).$$

To allow the possibility of scoring functions that can act as proxies for the posterior probability, we will assume that $W(H, X) \in \mathbb{R}$ and replace $\mathcal{F}$ with the weighted space:

$$\mathcal{T} \;=\; \{\, (H, X, W) \mid H \in \mathcal{H}, \; X \subseteq N(H) \cap \mathrm{ext}(H \mid B), W \in \mathbb{R} \,\}.$$

We now have most of the pieces needed to describe what we mean by a symbolic neural generator. Additionally, we may know beforehand some elements of $\mathcal{U}$ for which $\Phi(\cdot)$ is true or false. These provided as pairs of subsets from $2^{\mathcal{U}} \times 2^{\mathcal{U}}$, which we denote by $\mathcal{E}$. Then, the fibred-poset constructions we have described allow us to specify a class of functions we call *symbolic neural generators.*

Definition 6 (Symbolic Neural Generator) *Let $\mathcal{U}, \mathcal{B}, \mathcal{E}, \mathcal{H}$ be as above. Let $(\mathcal{H}, \geq_{\mathcal{H}})$ be a partially ordered by extensions as in Defn. 2. Then each element of $\mathcal{H}$ is associated with a fibred-poset as in Defn. 3, which results in the total poset $\mathcal{F}$ arising from the Grothendieck construction in Defn. 4. Let $\mathcal{T}$ be a (weighted) fibred poset extending $\mathcal{F}$ (Defn. 5). Then a Symbolic Neural Generator is a function $SNG : \mathcal{E} \times \mathcal{B} \to \mathcal{T}$.*

The general principle of a symbolic base combined with neural-fibres can be applied to characterise systems in each of the cases (A)–(D) in Fig. 2a. The details are provided in the appendix A.1.

---

5. A "truly" probabilistic extension will require us to replace each deterministic fibre by a space of probability measures over the fibre. What we describe is a simpler version based on empirical estimation that attaches a scalar weight with each pair in the (deterministic) poset. This is sufficient for our purpose in the paper, namely, of selecting a hypothesis and support-set pair. If a more involved probabilistic reasoning is required over elements of the fibre-set, then the extension to a full-fledged probabilistic Grothendieck construction will be required.

3.2 Implementing SNGs

It is evident from Fig. 3(b) that an SNG implementation could be devised as a search through the space of pairs in $\mathcal{F}$, or the weighted version $\mathcal{T}$. In designing an implementation, we are required to address 3 questions: (i) How do we identify the H's; (ii) How do obtain the X's; and (iii) How do we search the space $\mathcal{T}$?

Of these, Question (ii) is the most pressing, since it is here that neural and symbolic components interact.[6]

Procedure 1 is a general-purpose implementation for addressing Question (ii) using a large language model (LLM) as a generator of instances from $\mathcal{U}$. The implementation uses the following subset of the extension of H:

Definition 7 (Non-vacuous extension of H) *Let $B \in \mathcal{B}$, and $H \in \mathcal{H}$. The nonvacuous extension of H given B is the set $\mathrm{nvext}(H|B) = \{x : x \in \mathcal{U}, (B \wedge H)(x) = true, (B \wedge H) \models \Sigma(x)\}$. We note $\mathrm{nvext}(H|B) \subseteq \mathrm{ext}(H|B)$.*

In the implementation, instances are checked for inclusion in $\mathrm{nvext}(H|B)$ instead of $\mathrm{ext}(H|B)$. This ensures that we do not accept any instance x that vacuously satisfy the requirement that $B \wedge H \models \Sigma(x)$. Gen employs rejection sampling with contextual updates to ensure that elements in X are in $\mathrm{nvext}(H|B)$.[7] It is reasonably common in the use of language models to include some data in the initial context (Brown et al., 2020b). For simplicity, we will assume these are provided in E.

The following proposition is evident enough from Step 8 in Procedure 1, but is nevertheless worth reinforcing and the proof is given in the appendix A.2:

Proposition 2 (Correctness of Gen) *The set M_n returned by Procedure 1 is an element of $F(H)$ and $w_n \in [0, 1]$.*

3.3 Demonstration of a simple SNG

A simple SNG results from adopting the following "decomposition" strategy: (a) First search the base poset to find good symbolic hypothesis H (this can be done with an ILP engine, for example); and (b) Use the H obtained in (a) and Procedure Gen to obtain a sample of instances from $\mathrm{ext}(H|B)$. Let us call this a *chain-processing* SNG, based on the sub-categorisation in Hilario (2013).

We demonstrate a chain-processing SNG using the chess endgame consisting of the White King, White Rook and Black King. Two problems for this endgame have been studied in the symbolic learning literature: learning rules for detecting illegal positions; and identifying the minimum number of moves to a win for the Rook's side assuming it is Black's turn to move (BTM). The first problem is the more widely studied, but is less interesting since it follows almost directly from the rules of the game. We will use the second problem, and specifically on identifying positions that White is 0 moves away from a win (one such example is shown

---

6. Question (i) is concerned with constructing symbolic hypothesis, given examples and background knowledge. Techniques developed in ILP are especially good at this. Question (iii) is about searching graphs efficiently, for which several alternatives are known.

7. We do not have to explicitly construct $\mathrm{nvext}(H|B)$ for this. Instead, we include x in S iff $B \wedge H \models \Sigma(x)$. This check can be performed by a model-checker (or in some restricted cases, by an inference engine).

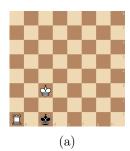**Procedure 1** `Gen`: An LLM-based implementation of the neural generator.

Input: L: an LLM capable of generating instances in a set conditional on the hypothesis H (see below); E: a subset of known data instances; B: background knowledge as provided to symbolic learner; H: a symbolic hypothesis for E given B; n: an upper-bound on the number of iterations; and s: an upper-bound on the number of samples to be drawn using the LLM; **Output**: a pair (w, X) where $w \in [0, 1]$ and $X \in F(H)$ $(= 2^{N(H) \cap \text{ext}(H|B)})$,

1: Let C_0 be the initial context containing a description of H and E
2: $M_0 = \emptyset$
3: $w_0 = 0$
4: $i := 1$
5: **while** $i \leq n$ **do**
6:    Let P_i be a prompt to generate x s.t. x in $\mathcal{U}$, and it completes the sentence $true : x$ given context C_{i-1}
7:    $S_i = Sample(s, L, P_i)$ //sample at most s instances from $\mathcal{U}$ using the LLM
8:    $D_i := \{(l, x) : x \in S_i \text{ and } l := (x \in \text{nvext}(H|B))\}$ // l is True or False
9:    Let context C_i be C_{i-1} updated with $true : x$ for $(true, x) \in D_i$ and $false : x$ for $(false, x)$ in D_i
10:   $M_i = \{x : (true, x) \in D_i\} \cup M_{i-1}$
11:   $i := i + 1$
12: **end while**
13: $w_n := \frac{|M_n|}{(s \times n)}$
14: **return** (w_n, M_n)

---

in Fig. 4(a)). That is, with Black to move it is checkmate and therefore won-for-white (WFW). These positions constitute about 0.1% of the entire dataset, and is therefore a rare event. The goal of the SNG will be to: (a) identify a symbolic hypothesis for WFW; and (b) generate instances for this rare event. A symbolic hypothesis has long been available in the literature, constructed by the GCWS extension to the ILP engine Golem (Bain et al., 2000) (see Fig. 4(b)).

Below we show a summary of iterations of `Gen` when the language model used is GPT-4o, and the maximum sample-size is set to 30. In the figure, "Without Symbolic" denotes there is no symbolic theory available; and "With Symbolic" refers to using the theory learned by the ILP engine (Fig. 4(b)). "0-shot" means no explicit examples of the concept WFW are provided within the hypothesis; and "5-shot" refers to providing 5 (positive) instances of WFW. From the results in Fig. 5 it is evident that the use of the symbolic model enables a progressive improvement in the conditional generation by the LLM. Given the small number of positive instances overall for the concept, it is unsurprising that LLM does not obtain any positive instances in the 0-shot samples without a symbolic theory.[8]

---

8. The results are with a sample-size of 30 on each iteration. For a uniform sampler, the probability of a positive instance in any single random draw is approximately $27/27000 \approx 0.001$. The chance of not obtaining at least 1 positive instance in a sample of 30 is $\approx 1 - (0.999)^{30} = 0.03$. It is unclear if this applies directly to language models, which have access to significant amounts of prior chess knowledge.

H :
Σ((WKF,WKR,WRF,WRR,BKF,BKR)) :-
    depth_of_win(0,WKF,WKR,WRF,WRR,BKF,BKR).

depth_of_win(0, c, 2, a, A, a, 2) :- not(ab3(0, c, 2, a, A, a, 2)).
depth_of_win(0, c, A, a, B, a, 1) :- not(ab2(0, c, A, a, B, a, 1)).
depth_of_win(0, A, 3, B, 1, A, 1) :- not(ab1(0, A, 3, B, 1, A, 1)).
ab1(0, A, 3, B, 1, A, 1) :- diff(A, B, d1).
ab2(0, c, A, a, 2, a, 1).
ab3(0, c, 2, a, A, a, 2) :- diff(2, A, d1).

(a)   (b)

Figure 4: (a) A position that is "won-for white" (WFW) with "black-to-move" (BTM). Here depth-of-win is zero, i.e., checkmate. There are 27 such positions out of a total of 28,056; (b) A symbolic hypothesis obtained using ILP (adapted from Bain et al. (2000)) (rewritten using Σ as required). In the context of this paper $\mathcal{U}$ consists of 6-tuples representing positions of the 3 pieces. The description is a Prolog-like syntax: variables start with upper-case, ":-" stands for $\leftarrow$, and "not" should be read as "not provable". The "diff/3" predicate is defined in the background knowledge and encodes file or rank differences. The "ab" predicates are new relations invented by the ILP system as it attempts to find a logical description for the depth-0 data instances. For example, in the position shown in (a), the invented "ab1" predicate ensures the white rook cannot immediately be taken by the black king. Σ should be understood as "WFW".

Iteration (i)	$|M_i|$			
	Without Symbolic		With Symbolic	
	0-shot	5-shot	0-shot	5-shot
1	0	8	17	23
2	0	23	30	30
3	0	25	30	30
4	0	25	30	30
5	0	27	30	30

Figure 5: Instances of WFW generated on each iteration of `Gen`. "Without Symbolic" represents the baseline of the LLM generating instances without any symbolic theory as part of the initial context or for verification (that is, $H = \emptyset$ in `Gen`). "With Symbolic" provides the WFW theory in Fig.4(b). "0-shot" means no examples are provided in E (and therefore are not part of the initial context), and "5-shot" means 5 WFR positions are provided in E.

What may surprise the reader however is that the generator appears to have generated 3 more instances – 30, instead of 27 – consistent with the symbolic theory. How is this possible, given that the symbolic description is stated in Bain et al. (2000) as a complete and correct recogniser for WFW. Closer study of the description in Bain et al. (2000) reveals that the symbolic description is only intended to be a complete and correct description of *legal* positions. The 3 additional positions generated are in fact all illegal positions. Thus, the

13

neural generator has generated unexpected instances that are consistent with the theory. This can be taken in 2 ways: (a) the verification step is only as good as the theory it is being verified against; and (b) the SNG can identify unanticipated instances that are consistent with the symbolic theory. This second aspect can be seen as a positive feature of the approach, especially in real-world problems such as the ones we consider later in the paper.

The reader could question the utility of the symbolic theory, on grounds that the LLM's performs well enough simply given a few initial examples (without symbolic theory, 5-shot). For this well-known synthetic dataset, this is indeed the case. However, even here we have still not generated all WFW instances after 5 iterations. For problems where the data instances are rare, missing even a few makes a difference (as in the real-world problems we consider next).

4 Application: SNGs for Lead Discovery in Drug-Design

We now turn to the real-world problem of generating small molecules subject to constraints imposed by a protein-target. For this, we will implement a *sub-processing* SNG (Hilario (2013)) which interleaves neural generation within the search for a symbolic hypothesis. For reasons of space, we do not provide a review of the role of small-molecule identification for early-stage drug design, and refer the reader to the literature (see, for example, Doytchinova (2022)). More specifically, and closely related to SNGs is the work in Abdel-Rehim et al. (2025). There, an LLM is used both to generate hypotheses and small molecules, and a robot-scientist is used to test the molecules proposed.[9]

A key requirement of any SNG is the the identification of the base poset $(\mathcal{H}, \geq_{\mathcal{H}})$. Here, we informally describe the hypothesis space for describing small molecules: a formal treatment is in Appendix B. Simply put, hypotheses for the problems considered here will encode interval-constraints on properties – which we will call *factors* – of molecules. An assignment of interval-values to factors will be called an *experiment*. Factors together with an experiment will be called a *factor-specification*. Hypotheses encode factor-specifications.

Example 1 (Factors, Experiments, Hypotheses) *Consider the factor-specification:*
(($MolWt$, $SynthSteps$, $Affinity$), ([200, 800], [4, 8], [6, 10])) specifies we have 3 factors – molecular weight, number of synthesis steps, and estimated binding affinity to the target – and an experiment that assigns MolWt to the interval [200, 800], SynthSteps to the interval [4, 8] and Affinity to the interval [8, 10]. Each such experiment is treated as a hypothesis – which we denote Hypothesis(($f_1, \ldots, f_n$), ($i_1, \ldots, i_n$)) – and represented as a clause. Thus Hypothesis (($MolWt$, $Affinity$), ([200, 800], [8, 10]) is the clause (here, $\mathcal{U}$ is the set of small molecules).

$$\forall x \in \mathcal{U} \ \Sigma(x) \leftarrow$$
$$MolWt(x) \in [200, 800] \ \wedge$$
$$SynthSteps(x) \in [4, 8] \ \wedge$$
$$Affinity(x) \in [8, 10]$$

Hypotheses are thus simple axis-parallel hyper-rectangles (akin to a categorical version of the probabilistic box-embedding introduced by Vilnis et al. (2018): a probability estimate will

---

9. We conjecture that the use of the symbolic learner as is done here may result in more reliable hypotheses, and require fewer experiments to be conducted by the robot.

be added later). We will assume that background knowledge B will specify how to compute the values of the factors.[10] Identifying a suitable SNG requires searching through potential (H, X) pairs (here X will be a set of molecules within $\text{ext}(H|B)$). For our experiments we use Procedure 2 that progressively examines (weighted) (H, X) pairs, in – as we will see shortly – a general-to-specific manner. For each H, a subset $X \subseteq \text{ext}(H|B)$ is obtained using the LLM-based implementation $\texttt{Gen}$. The procedure returns a triple (H, X, W) which contains the highest weight (H, X) pair encountered in the search.

Procedure 2 $\texttt{GenMol}$: implementing a greedy search over the space of symbolic descriptions and consistent sets of generated molecules

---

Input: L: an LLM; E: a sample of labelled and unlabelled molecules; B: background knowledge, that includes $(F, \boldsymbol{\Theta})$: a factor-specification with $F = (f_1, \ldots, f_n)$ and $\boldsymbol{\Theta} = ([\theta_1^-, \theta_1^+], \ldots, [\theta_n^-, \theta_n^+])$; s: an upper-bound on the number of samples in the search; k: an upper-bound on the number of steps in the search; l: an upper-bound on the number of LLM iterations; m: an upper-bound on the number of samples to be drawn by the LLM' and θ: a lower bound on the weight of an acceptable hypothesis

Output: $(H, X, W) \in \mathcal{T}$ where $H \in (\mathcal{H}, \geq_{\mathcal{H}})$ is a symbolic hypothesis; $X \in F(H)$ a set of molecules; and $W \in R$

1: $\mathbf{e}_0 = \boldsymbol{\Theta}$
2: $H_0 = Hypothesis(F, \mathbf{e}_0)$
3: $(w, M_0) = \texttt{Gen}(L, B, H_0, l, m)$ generate molecules in outermost hyper-rectangle
4: $q_0 = Q(H_0, E, B)$ see Appendix B.2 for Bayesian score
5: $w_0 = q_0 \times \mathbf{1}(w > 0)$
6: $i = 1$
7: $Done = ((w_0 < \theta) \vee (i > k))$
8: **while** $\neg Done$ **do**
9:    Let E_i be a random sample of s interval-vectors properly subsumed by $\mathbf{e}_{i-1}$
10:    Let $S = \{(w', \mathbf{e}, H, M) : \mathbf{e} \in E_k, \; H = Hypothesis(F, \mathbf{e}), \; q = Q(H, E, B), (w, M) = \texttt{Gen}(L, E, B, H, l, m), w' = q \times \mathbf{1}(w > 0)\}$
11:    Let $(W_i, \mathbf{e}_i, H_i, M_i) = \underset{(w, \mathbf{e}, H, M) \in S}{\text{argmax}} \; w$
12:    $Done = ((w_i < \theta) \vee (W_i < W_{i-1})$
13:    $i := i + 1$
14: **end while**
15: **return** $(H_{i-1}, M_{i-1}, W_{i-1})$

We note that $\texttt{Gen}$ is essentially a symbolic learner that repeatedly invokes a neural reasoner: it therefore exemplifies the *sub-processing* form of integration identified in Hilario (2013). We clarify the working of two aspects of the search by example. First, we highlight the hypotheses considered:

Example 2 (Nested Rectangles) *Suppose $\texttt{GenMol}$ is attempting to find a hypothesis given 2 factors $F = (Affinity, SynthesisSteps)$, with $\boldsymbol{\Theta} = \mathbf{e}_0 = ([5, 10], [4, 8])$. Then $\texttt{GenMol}$ starts*

---

10. Some computations, like $Affinity(\cdot)$, can be quite involved, requiring knowledge of the target-site, and the use of a molecular modelling program for estimating binding affinity of the molecule to this site.

with the constraint $(Affinity \in [5, 10]) \wedge (SynthesisSteps \in [4, 8])$, which corresponds to a rectangle R_0 in the Cartesian-space with Affinity and SynthesisSteps. Let the Q-value of the corresponding hypothesis be q_0. `SearchHyp` then randomly samples rectangles contained within R_0. Suppose the rectangle R_1, defined by $\mathbf{e}_1 = ([6, 10], [4, 6])$, and the corresponding hypothesis has the highest Q-value (q_1) of all the rectangles sampled. `SearchHyp` then iterates by sampling within R_1. The search procedure therefore identifies a sequence of nested rectangles.

Secondly, we note that `GenMol` is a randomised procedure. Sampling is done in Step 9 of `GenMol`. Here there are several options: the easiest to sample each dimension of the bounding hyper-rectangle independently, using a uniform distribution. Better sampling procedures exist (for example, Latin Hyper-rectangle Sampling (McKay et al., 2000), DIRECT (Jones, 2001), Bayesian sub-region sampling (Skilling, 2004) and so on). Alternatives to the simple greedy strategy of picking the best-scoring hyper-rectangle in Step 11 also clearly possible, by drawing an experiment from the distribution of w-scores.

Example 3 (Sampling (Hyper-)Rectangles) *Suppose `GenMol` is attempting to sample a rectangle bounded by $[x_1, x_2]$ and $[y_1, y_2]$. Examples of some strategies for sampling sub-rectangles are:*
Uniform orthogonal sampling. *(a) Select a pair of points $a = U(x_1, x_2)$ and $b = U(x_1, x_2)$ s.t. $x_1 < a < b < x_2$; (b) Select a pair of points $c = U(y_1, y_2)$ and $d = U(y_1, y_2)$ s.t. $y_1 < c < d < y_2$; and (c) The new rectangle is bounded by $[a, b]$ and $[c, d]$.*
Sampling with fixed upper- or lower-bounds. *(a) Select a point $a = U(x_1.x_2)$ s.t. $x_1 < a < x_2$; (b) Select a point $d = U(y_1, y_2)$ s.t. $y_1 < d < y_2$; and (c) The new rectangle is bounded by $[a, x_2]$ and $[y_1, d]$.*

We treat the choice of sampling method as an application-specific detail. The following properties will however hold for Procedure `GenMol`.

PROPERTIES OF GENMOL

Irrespective of the sampling strategy used, the following proposition holds for `GenMol` and the proof is given in the appendix B.3:

Proposition 3 *Let $(F, \cdot)$ be a factor-specification, B denote background knowledge. Let $\mathbf{e}_i$ ($1 \leq i \leq k$) be an experiment selected by `GenMol` on the ith iteration s.t. $\mathbf{e}_i$ is contained by $\mathbf{e}_{i-1}$. Let $H_i = Hypothesis(F, \mathbf{e}_i)$, and $H_{i-1} = Hypothesis(F, \mathbf{e}_{i-1})$. Then $H_{i-1} \models H_i$.*

It is straightforward to see that the hypotheses examined by `GenMol` are from the base poset $(\mathcal{H}, \geq_{\mathcal{H}})$.

Remark 1 ($H_{i-1} \geq_{\mathcal{H}} H_i$) *Let $H_{i-1}, H_i \in \mathcal{H}$ be hypotheses constructed by `GenMol` on iterations $i-1, i$. Let $\geq_{\mathcal{H}}$ be as defined in Defn. 2. Then $H_{i-1} \geq_{\mathcal{H}} H_i$. This follows from Prop. 3. If $H_{i-1} \models H_i$ then $B \wedge H_{i-1} \models B \wedge H_i$. It follows that $\text{ext}(H_{i-1}|B) \supseteq \text{ext}(H_i|B)$, and $H_{i-1} \geq_{\mathcal{H}} H_i$.*

16

Remark 2 (GenMol implements an SNG) *We note first that the symbolic hypotheses examined by* GenMol *from $\mathcal{H}$. Let (H_i, M_i, W_i) be the hypothesis, generated set of molecules and weight in* GenMol *on iteration i. By construction M_i is obtained using* Gen *with hypothesis $H_i \in \mathcal{H}$. By Prop. 2 $M_i \subseteq \text{ext}(H_i|B)$. From Step 10 W_i is either the Q-score or 0. Since Q-scores are in $\mathbb{R}$, $(H_i, M_i, W_i) \in \mathcal{T}$ (see Defn. 5 and* GenMol *can be taken to be an implementation of an SNG function defined in Defn. 6 (with some additional bounds on sample-sizes etc.)*

LIMITATIONS OF GenMol

We draw the attention of the reader to the following limitations of GenMol:

- Although GenMol returns an element from $\mathcal{T}$, it does not use the $\geq_{\mathcal{F}}$ ordering. Instead its search proceeds using only the base-poset ordering.

- GenMol assigns a 0 or 1 value for generator efficiency in Step 10 (through the use of the indicator function $\mathbf{1}(\cdot)$). This results in over-zealous pruning of the hypothesis space. A better estimate would be to use a measure based on the proportion $Q \times w$ as described in Sec.3.1.

- The search-strategy performs a greedy selection of a single element in Step 11. Again, while this reduces the size of the search-space, It is likely that a best-first strategy employing a priority-queue would be an improvement.

- GenMol inherits the inefficiency of Gen, which adopts rejection-sampling to ensure that data instances returned satisfy the the symbolic hypothesis.

4.1 Case Study 1: Well-Understood Targets with Many Ligands

We evaluate the performance of SNG in the controlled setting examined in (Brahmavar et al., 2024), with a known target-site, a large number of known inhibitors and non-inhibitors, and a single factor to be optimised (estimated binding affinity to the target-site).

PROBLEMS

Kinase Inhibitors. We conduct our controlled evaluations on 2 well-studied kinase inhibitors: (a) JAK2, with 4100 molecules provided with labels (3700 active); and (b) DRD2 (4070 molecules with labels, of which 3670 are active). These datasets are from the ChEMBL database (Gaulton et al., 2012), which are selected based on their IC_{50} values and docking scores. JAK2 inhibitors are drugs that inhibit the activity of the Janus kinase 2 (JAK2) enzyme, in turn affecting signalling pathways, especially to the cell nucleus. These pathways are critical for various immune response reactions and are used to develop drugs for autoimmune disorders like ulcerative colitis and rheumatoid arthritis. DRD2 (dopamine D2) inhibitors are drugs that block dopamine's ability to activate the DRD2 receptors. This reduces dopamine signalling, and is used to treat psychological disorders like schizophrenia.

We distinguish the following components of background knowledge:

(A) Specialists' knowledge. We require biological knowledge of the target site, or a proxy for the target size. We also need chemical knowledge of the relevant factors, the range of their values, and information (if any) of whether the factors need to be maximised or minimised. For this case study, we will use only 1 factor (estimated binding affinity).

(B) Factor-functions. These are definitions for computing the factors (like *Affinity*, *MolWt etc.*) for molecules. Usually this will also include procedures provided by some molecular modelling software (like RDKit).

ALGORITHMS AND MACHINES

All the experiments are conducted using a Linux (Ubuntu) based workstation with 64GB of main memory and 16-core Intel Xeon 3.10GHz processors. All the implementations are in Python3. We use `OpenAI` library (version 1.52.1) for sampling molecules from `GPT-4o` (Achiam et al., 2023), with temperature 0.7. We use `RDKit` (version 2024.09.3) (Bento et al., 2020) for computing molecular properties and `GNINA` (version 1.3) (McNutt et al., 2021) for computing docking scores (binding affinities) of molecules. Additional details of the experimental setup can be found in Appendix C. We have used PubChem Sketcher V2.4 for drawing the 2-D structures of the molecules shown in this paper.

EXPERIMENTAL METHOD

Our method for experiments is straightforward and follows these steps:

1. Identify factor-set F and other bounds.

2. Obtain data instances consisting of positive, negative and unlabelled examples. Any positive example is taken to be feasible and any negative example is taken to be infeasible.

3. Using the background knowledge B described in Sec. 4.1, D, F and other bounds:

   (a) Obtain a set of molecules using `GenMol`;
   (b) Assess the quality of the molecules generated.

We refer the reader to Appendix C for additional details related to the method.

RESULTS

Figure 6 shows a comparison of SNG against the results tabulated for LMLF++ in Brahmavar et al. (2024). LMLF++ was the best performing variant in that paper, and was substantially better than previous benchmarks set by the use of a VAE-GNN combination and reinforcement-learning based methods. It is evident from the tabulation that SNG performs at least as well as LMLF++. It is also helpful if a lead generator proposes novel molecules. For the JAK problems, this is not easy, since the number of known inhibitors is

very large. Nevertheless, Fig. 7 suggests that the molecules generated by `GenMol` may still be quite novel (this is due to the prompt used for the LLM that attempts to generate molecules not in any known chemical database).

Problem	Known Inhibitors	`GenMol` (here)	LMLF++	VAE-GNN
JAK2	7.26 (0.64)	8.59 (0.23)	7.74 (0.30)	6.53 (1.18)
DRD2	6.86 (0.83)	7.53 (0.47)	7.66 (0.29)	-

Figure 6: Statistics of binding affinities (the higher the better) for molecules obtained from `GenMol` on benchmark datasets. The entries represent the mean values, with standard deviations shown in parentheses. We compare against recent results using LMLF++ (Brahmavar et al., 2024) and prior results using a VAE-GNN model (Dash et al., 2021).

Problem	TC
JAK2	0.14 (0.032)
DRD2	0.13 (0.034)

Figure 7: Potential novelty of LLM-generated molecules using `GPT-4o`. TC represents the Tanimoto coefficient, ranging from 0 to 1, where 1 signifies high similarity to known inhibitors, while 0 indicates complete structural dissimilarity. The entries represent the mean values, with standard deviations shown in parentheses.

4.2 Case Study 2: Less Understood Target with Few Ligands

We evaluate the performance of `GenMol` in an open-ended setting where the true target-site is not known precisely, and multiple factors have to be optimised (binding affinity, molecular weight and synthesis accessibility), and there are very few known inhibitors. Note: Background Knowledge; Algorithms and Machines; and Methods are the same as for Case Study 1 (Section 4.1).

PROBLEM

DBH Inhibitors. Human dopamine β-hydroxylase (DBH) is an enzyme that converts dopamine (DA) to norepinephrine (NA) and plays a pivotal role in regulating the concentration of NA, deficiency or overproduction of which causes several diseases related to the brain and the heart. This enzyme is thus of high therapeutic significance. The availability of the three-dimensional structure of DBH is expected to facilitate the identification of DBH active-site inhibitors. In the meantime, the crystal structure of a dimer of DBH has been determined, providing insights into its function and aiding in the design of inhibitors (Vendelboe et al., 2016). Specifically, we will use the *in silico* model of the dimer to generate small molecules with similar or better IC50 and KD values (in simulation) than at least one of the latest generation of DBH inhibitors. We will use as data the 5 known DBH inhibitors: Tropolone, Disulfiram, Nepicastat, Zamicastat, Etamicastat. The last three are shown in Fig. 8. Tropolone is a naturally occurring molecule, with known toxic effects. Disulfiram is a 1st generation molecule, and also with toxic side-effects. The last two molecules, Zamicastat

and Etamicastat are the latest generation of DBH inhibitors and are currently in double-blind human trials for hypertension. We focus on obtaining molecules with docking scores at least as good as Nepicastat, a 4th generation drug.
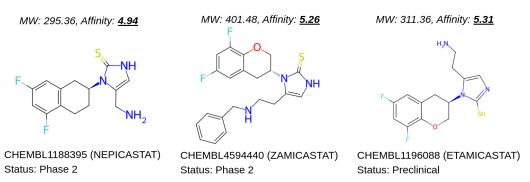

MW: 295.36, Affinity: **4.94**

MW: 401.48, Affinity: **5.26**

MW: 311.36, Affinity: **5.31**

CHEMBL1188395 (NEPICASTAT)
Status: Phase 2

CHEMBL4594440 (ZAMICASTAT)
Status: Phase 2

CHEMBL1196088 (ETAMICASTAT)
Status: Preclinical

Figure 8: Three known DBH inhibitors at different stages of their FDA approval status. Each compound is identified by their CHEMBL ID and name. 'MW' refers to molecular weight; 'Affinity' refers to the binding affinity predicted by GNINA software while docking the molecules to the DBH protein, *4zel*. The approval status of these compounds were noted as on 25 January 2025 from the ChEMBL online portal.

RESULTS

The exploratory problem concerns generating potential leads for DBH inhibition, given data on the structure and inhibitory values of 5 molecules on a proxy target to DBH. The inhibitory efficacy of a small number of molecules is known (see description of the problem above). We consider two kinds of exploratory experiments. First, the LLM is provided with the information about the known molecules and their structure: in LLM parlance, we are doing "few-shot learning" (correctly, we are using the LLM to draw from a distribution conditioned on the known molecules). We will call this "In-the-Box" exploration. Secondly, we do not give the LLM any information about known molecules ("zero-shot learning", or "Out-of-the-Box" exploration). The top-5 molecules obtained for each kind of experiment is shown in Fig. 9.

But are the molecules any good? Here are some assessments by specialists:

"In-the-Box" Exploration. The views of the specialists about Molecules 1–5 are as follows:

Structural Biologist. Molecules 1–4. These are (likely to be) good inhibitors of DBH since these molecules are structurally similar to dopamine and nepicastat. Another good feature of these molecules is that it carries Fluorenes as well. Molecule 5. Could be a better inhibitor of DBH since it is structurally similar to dopamine and nepicastat. It may work. It could be better than the four other molecules since it carries one oxygen in the aromatic ring. Another good feature of this molecule is that it also carries Fluorenes.
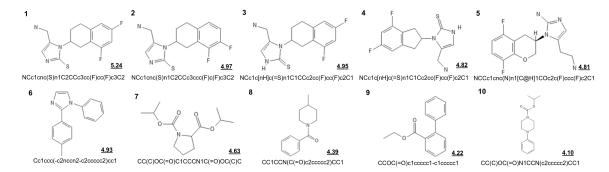
Figure 9: Potential inhibitors for DBH proposed by `GenMol`, along with their binding score to *4zel* protein. Molecules 1–5 are the top-5 molecules (ordered by estimated affinity) from "In-the-Box" exploration. Molecules 6–10 are from "Out-of-the-box" exploration. In the former the LLM used by `GenMol` has few-shot examples when it starts. In the later, no such information is provided, and LLM uses its underlying distribution over molecules are used to generate the molecules.

> **Synthetic Chemist.** All molecules can be synthesized, but is likely to be a long synthesis, and costs will be high.

> **"Out-the-Box" Exploration.** The views of the specialists about Molecules 6–10 are as follows:

> **Structural Biologist.** Molecule 6. This is a novel inhibitor of DBH where halogen is absent, OH/O is also absent. Not similar to dopamine or nepicastat. Thus, it can be an allosteric inhibitor of DBH. Molecules 7,8. This is also a novel inhibitor of DBH where halogen is absent. Since O is present twice, the affinity might be tighter. Aromatic rings are very dissimilar to dopamine or nepicastat. Thus, mode of inhibition is difficult to predict. Molecule 9. May not be a good inhibitor due to highly flexible sidechains. Molecule 10. This can be a better inhibitor than the first one since its aromatic rings are very interesting to bind in the active site of DBH. O= is present and thus the affinity might be tighter. Mode of inhibition should be competitive.

> **Synthetic Chemist.** All molecules can be synthesised, and synthesis is likely to be through a short route. The molecules may be commercially available.

The broad takeaways are these: (a) Unsurprisingly, In-the-Box exploration appears to yield molecules that are similar (but not the same) as existing inhibitors. In contrast, it is very interesting that Out-of-the-Box exploration appears to yield very different molecules to known inhibitors; (b) There are good biological reasons to expect Molecules 1–5 to bind to the target, and Molecules 6,7,8 and 10 to bind to the target. Of these, the biologist believes Molecules 6 and 10 to be especially interesting; and (c) On the synthesis side, all molecules appear to be synthesisable, but 6–10 appear to be more amenable to a short (and therefore, possibly cheaper) route.

4.3 Additional Experimental Observations

We now consider some questions that are of relevance to both case studies, and of wider interest to the uses of SNG:

Is symbolic learning useful? The motivation for SNG was that for certain kinds of problems, it is beneficial for symbolic learning to (a) "focus" the neural generator; and (b) act as a touchstone for human assessments of the generator. It is relevant to assess whether these aspects are reflected in the case-studies presented. Figure 10 shows the results from just using the LLM-based generator without a symbolic theory to constrain its output. The results suggest the symbolic hypothesis does appear to play a useful role. It is unclear to us why the variance with the use of symbolic theories is higher (we would have expected it to be lower).

Problem	Mean Affinity	
	Without Symbolic Learning	With Symbolic Learning
JAK2	4.53 (0.43)	7.72 (1.53)
DRD2	5.40 (0.60)	7.40 (0.61)
DBH	3.80 (0.37)	4.72 (0.30)

Figure 10: Estimated binding affinities of molecules generated without and with symbolic learning.

It is noteworthy that biologists and chemists examine molecules based on the symbolic description – in particular constraints on molecular weight and binding affinity. They are able to comment meaningfully on instances generated by the machine. These two findings, although qualitative, are important, as they indicate that both symbolic and neural descriptions are important and intelligible to human experts. Additionally, not included here for reasons of space, is a further round of 'molecule-exchange' between the chemist and the biologist, inspired by Molecules 1–5. The chemist proposed edited versions with shorter synthesis steps, and the biologist commented further on the biological suitability of those molecules.

Does the choice of LLM matter? We considered SNGs with 2 leading general-purpose LLMs: OpenAI's GPT-4o and Anthropic's Claude 3.5 Sonnet (Anthropic, 2024). Details of the experiments are in Sec. C.2 of the Appendix. The results show that neither LLM seems clearly better. It is also relevant to examine if such large, general-purpose LLMs are needed at all. We also examined the performance of SNG with a specialised, small language model (see the Appendix for details). Specifically, We implemented an SNG variant using 87M parameter GPT-2 model (entropy, 2023), The results show that general-purpose large models are better on the benchmark datasets of JAK2 and DRD2. But interestingly, for the DBH problem – where the target is less understood and known inhibitors are fewer – the smaller specialised model appears to perform better. This suggests the larger models may be performing better simply by virtue of having been exposed to the benchmark data during training.

Are the results sensitive to initial conditions/parameter settings? We again refer the reader to Sec. C.2 of the Appendix for detailed tabulations on the effect of initial conditions and parameter changes. We are specifically interested in the effect of the initial context provided to the LLM, and values for 2 parameters: the *sample size* used by the symbolic leaner and the LLM's *temperature*. On the benchmark data, the results appear unaffected by changes in the initial context. However, for the exploratory dataset, providing few-shot instances in the initial context does make a difference to the similarity to known data instances. On parameter settings, we find GPT-4o output to be the most sensitive to parameter changes. SNG with Claude and our specialised GPT-2 model were stable across changes in sample size and temperature.

5 Concluding Remarks

To date, the state of AI and ML has been in what Kuhn referred to as a *pre-paradigm* state, with two main competing schools of thought – symbolic and connectionist – offering different procedures, results and metaphysical principles. However, with startling results using connectionist techniques on problems of common interest to the competing schools, we may well be on the verge of consensus developing on what constitutes a Kuhnian notion of "normal science" at least in the machine-learning aspects of AI. Even so, there still appears to be a useful role for the tools and techniques developed by the symbolic school, especially in the use of ML for problems requiring the inclusion of prior knowledge, logical reasoning, correctness guarantees, and human-understandable explanations. For a class of problems with these characteristics, we this paper, we propose a neurosymbolic approach we have called symbolic neural generation, or SNG. SNG systems are characterised by the use of a (learned) symbolic description that constrain the generation of data by neural generators. A useful analogy is offered by the roles of specification and implementation in formal methods for software development. The symbolic model in SNG acts as the specification and the neural generator as the implementation. As is done in formal methods, we ensure that any output true of the implementation – instances generated – is also true of the specification (satisfies the constraints of the symbolic model). Unlike with classic formal methods, in SNG ML techniques play a role in obtaining both the specification and the implementation. We have proposed using the mathematical structure of fibered-posets by way of specifying the codomain of an SNG system. The paired elements comprising this set makes it a natural choice for hybrid neurosymbolic systems, where one element of the pair is of symbolic and the other of neural origin.

In the paper, a practical motivation for SNG is provided by the need to accelerate the identification of 'leads' in early stage drug-design. Leads are small molecules capable of binding to a target protein, and satisfying some physico-chemical constraints. The specific area where ML could help is in cases where the structure of target site is not well understood, and there are as yet very few small molecules that have been shown to be good inhibitors. From a ML standpoint, this constitutes an ideal situation for symbolic learning. We also want to be able to generate entire molecules that can be examined critically by synthetic chemists and structural biologists. This generation requires a complex probability model that can adequately represent the molecular patterns of interest, and also efficient mechanisms for sampling from the distribution. Neural generators – especially those based on language

models – appear to be especially well-suited for this. motivation for developing SNG. We have provided results on benchmark problems that show SNG to be comparable to state-of-the-art methods. However, it is the exploratory study that we think is substantially more relevance to the area of early-stage drug-design. Specifically, it shows: (a) the use of very small numbers of data instances (5, in this case); (b) and (b) some of the results from the generator – especially in the "Out of the Box" mode – may be biologically novel and can be synthesised. While the output has been made intelligible to the specialist through the use of an LLM, controlling and verifying the LLM's output has been made possible through the use of the symbolic model.

There are several ways in which the work here can be improved and extended. Immediately of relevance are improvements to the `GenMol` algorithm. We have already listed a number of limitations of the procedure: addressing each one will improve both the quality of solutions returned, and the efficiency of the approach. Additional studies of the "out-of-the-box" kind will also help establish SNG as a useful tool for early-stage drug design. Further exploration is also needed on the symbolic side. The use of probabilistic symbolic learning for example, may be needed for problems for which background knowledge is uncertain or missing. Nothing in the conceptual framework of SNG systems requires the use of categoric logical descriptions. More broadly, SNG systems are of relevance not just to molecule-generation, but for any problem requiring the generation of data that needs verification either formally or by a person, but for which we do not already have an formal description (generation of system's behaviour, subject to the constraints of symbolically-learned digital twins; planners where a symbolic model is used to constrain and verify plans generated by an LLM; and symbolic models constraining the generation of experiments for active learning are some examples). Finally, we believe the conceptual structure of fiber posets are naturally applicable to the development of other kinds of hybrid neurosymbolic systems. For example, when developing neural-predictors with symbolic-explainers, it is evident that we are dealing with pairs of models. Whether there exist fiber poset constructions for such hybrid models needs further exploration.

Code and Data Availability

Acknowledgements

References

Abbi Abdel-Rehim, Hector Zenil, Oghenejokpeme Orhobor, Marie Fisher, Ross J. Collins, Elizabeth Bourne, Gareth W. Fearnley, Emma Tate, Holly Smith, Larisa N. Soldatova, and Ross King. Scientific hypothesis generation by large language models: laboratory validation in breast cancer treatment. *J. R. Soc. Interface*, 22, 2025.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Anthropic. Claude 3.5 sonnet: Frontier intelligence at $2\times$ the speed. https://www.anthropic.com/news/claude-3-5-sonnet, Jun 2024.

Michael Bain, Stephen Muggleton, and Ashwin Srinivasan. Generalising Closed World Specialisation: A Chess End Game Application. Technical report, CiteSeerX / University of New South Wales Technical Report, 2000. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.5624. Unpublished technical report, available online.

A Patrícia Bento, Anne Hersey, Eloy Félix, Greg Landrum, Anna Gaulton, Francis Atkinson, Louisa J Bellis, Marleen De Veij, and Andrew R Leach. An open source chemical structure curation pipeline using rdkit. *Journal of Cheminformatics*, 12:1–16, 2020.

T. Besold, A. d'Avila Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kühnberger, L. Lamb, D. Lowd, P. Lima, L. de Penning, G. Pinkas, H. Poon, and G. Zaverucha. Neural-Symbolic Learning and Reasoning: A Survey and Interpretation. *arXiv preprint arXiv:1711.03902v1*, 2017. URL https://arxiv.org/pdf/arXiv:1711.03902v1.pdf.

Shreyas Bhat Brahmavar, Ashwin Srinivasan, Tirtharaj Dash, Sowmya Ramaswamy Krishnan, Lovekesh Vig, Arijit Roy, and Raviprasad Aduri. Generating novel leads for drug discovery using llms with logical feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 21–29, 2024.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, et al. Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165*, 2020a.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020b.

Fengxiang Cheng, Haoxuan Li, Fenrong Liu, Robert van Rooij, Kun Zhang, and Zhouchen Lin. Empowering LLMs with Logical Reasoning: A Comprehensive Survey. *arXiv preprint arXiv:2502.15652*, 2025.

Y. Chervonyi, T. Trinh, M. Olšák, X. Yang, H. Nguyen, M. Menegali, J. Jung, V. Verma, Q. Le, and T. Luong. Gold-medalist Performance in Solving Olympiad Geometry with AlphaGeometry2. *arXiv preprint arXiv:2502.03544*, 2025. URL https://arxiv.org/abs/2502.03544.

Jacob K. Christopher, Michael Cardei, Jinhao Liang, and Ferdinando Fioretto. Neuro-symbolic generative diffusion models for physically grounded, robust, and safe generation, 2025. URL https://arxiv.org/abs/2506.01121.

Wang-Zhou Dai and Stephen H. Muggleton. Abductive knowledge induction from raw data. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 1845–1851. ijcai.org, 2021. doi: 10.24963/ijcai.2021/254. URL https://doi.org/10.24963/ijcai.2021/254.

Tirtharaj Dash, Ashwin Srinivasan, Lovekesh Vig, and Arijit Roy. Using domain-knowledge to assist lead discovery in early-stage drug design. In *International Conference on Inductive Logic Programming*, pages 78–94. Springer, 2021.

A. d'Avila Garcez, L. Lamb, and D. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Springer, 2009.

Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. ProbLog: A probabilistic Prolog and its application in link discovery. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2462–2467, 2007.

L. De Smet and L. De Raedt. Defining Neurosymbolic AI. *arXiv preprint arXiv:2507.11127*, 2025. URL https://arxiv.org/abs/2507.11127.

Vincent Derkinderen, Robin Manhaeve, Rik Adriaensen, Lucas Van Praet, Lennert De Smet, Giuseppe Marra, and Luc De Raedt. The deeplog neurosymbolic machine, 2025. URL https://arxiv.org/abs/2508.13697.

I. Doytchinova. Drug design-past, present, future. *Molecules*, 23:27(5):1496, 2022.

A. d'Avila Garcez and L. Lamb. Neurosymbolic AI: The 3rd Wave. *Artificial Intelligence Review*, 56(11):12387–12406, 2023. doi: 10.1007/s10462-023-10448-w.

entropy. GPT2 Zinc 87M. https://huggingface.co/entropy/gpt2_zinc_87m, 2023. GPT-2 model (87M parameters) trained on 480M SMILES from the ZINC database, MIT license.

Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1): D1100–D1107, 2012.

M. Guo, V. Thost, B. Li, P. Das, J. Chen, and W. Matusik. Data-efficient Graph Grammar Learning for Molecular Generation. *arXiv preprint arXiv:2203.08031*, 2022. URL https://arxiv.org/abs/2203.08031.

M. Hilario, C. Pellegrini, and F. Alexandre. Modular Integration of Connectionist and Symbolic Processing in Knowledge-Based Systems. In *International Symposium on Integrating Knowledge and Neural Heuristics*, pages 123–132, 1994.

Melanie Hilario. An overview of strategies for neurosymbolic integration. *Connectionist-Symbolic Integration*, pages 13–35, 2013.

John J Irwin, Khanh G Tang, Jennifer Young, Chinzorig Dandarchuluun, Benjamin R Wong, Munkhzul Khurelbaatar, Yurii S Moroz, John Mayfield, and Roger A Sayle. Zinc20—a free ultralarge-scale chemical database for ligand discovery. *Journal of chemical information and modeling*, 60(12):6065–6073, 2020.

M. Jacobson and Y. Xue. Integrating symbolic reasoning into neural generative models for design generation. *Artificial Intelligence*, 339(104257), 2025.

Donald R Jones. Direct global optimization algorithm. *Encyclopedia of optimization*, pages 431–440, 2001.

H. Kautz. Tools Are All You Need. 2024.

M. Kusner, B. Paige, and J. Hernández-Lobato. Grammar Variational Autoencoder. In *Proc. International Conference on Machine Learning*, pages 1945–1954, 2017.

Tom Leinster. *Basic Category Theory*, volume 143 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 2014. ISBN 978-1107044241. See Ch. 8 for the Grothendieck construction.

Z. Li, Z. Zhou, Y. Yao, Y.-F. Li, C. Cao, F. Yang, X. Zhang, and X. Ma. Neuro-Symbolic Data Generation for Math Reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pages 23488–23515, 2024.

Y. Liang, D. Nguyen, C. Yang, T. Li, J. Tenenbaum, C. Rasmussen, A. Weller, Z. Tavares, T. Silver, and K. Ellis. ExoPredicator: Learning Abstract Models of Dynamic Worlds for Robot Planning. *arXiv preprint arXiv:2509.26255*, 2025.

Jaechang Lim, Seongok Ryu, Jin Woo Kim, and Woo Youn Kim. Molecular generative model based on conditional variational autoencoder for de novo molecular design. *Journal of cheminformatics*, 10:1–9, 2018.

Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems*, 31, 2018.

G. Marra, S. Dumančić, R. Manhaeve, and L. De Raedt. From statistical relational to neurosymbolic artificial intelligence: A survey. *Artificial Intelligence*, 328(104062), 2024.

Eric McCreath and Arun Sharma. LIME: A system for learning relations. In *International conference on algorithmic learning theory*, pages 336–374. Springer, 1998.

W. McCulloch and W. Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathemetical Biophysics*, 5:115–133, 1943.

Michael D McKay, Richard J Beckman, and William J Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

Andrew T McNutt, Paul Francoeur, Rishal Aggarwal, Tomohide Masuda, Rocco Meli, Matthew Ragoza, Jocelyn Sunseri, and David Ryan Koes. Gnina 1.0: molecular docking with deep learning. *Journal of cheminformatics*, 13(1):43, 2021.

Stephen Muggleton. Learning from positive data. In *International conference on inductive logic programming*, pages 358–376. Springer, 1996.

Stephen Muggleton. Hypothesizing an algorithm from one example: the role of specificity. *Philosophical Transactions of the Royal Society A*, 381(2251):20220046, 2023.

Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. The MIT Press, London, 2022. Volume 2.

S. Odense and A. d'Avila Garcez. A semantic framework for neurosymbolic computation. *Artficial Intelligence*, 340(104273), 2025.

T. Olausson, A. Gu, B. Lipkin, C. Zhang, A. Solar-Lezama, J. Tenenbaum, and R. Levy. LINC: A Neurosymbolic Approach for Logical Reasoning by Combining Language Models with First-Order Logic Provers. *arXiv preprint arXiv:2310.15164v2*, 2024. URL https://arxiv.org/abs/2310.15164v2.

C. Qi, R. Ma, B. Li, H. Du, B. Hui, J. Wu, Y. Laili, and C. He. Large Language Models Meet Symbolic Provers for Logical Reasoning Evaluation. *arXiv preprint arXiv:2502.06563*, 2025. URL https://arxiv.org/abs/2502.06563.

Taisuke Sato and Yoshitaka Kameya. Prism: a language for symbolic-statistical modeling. In *IJCAI*, volume 97, pages 1330–1339, 1997.

John Skilling. Nested sampling. *Bayesian inference and maximum entropy methods in science and engineering*, 735:395–405, 2004.

M. Sun, W. Yuan, G. Liu, W. Matusik, and J. Chen. Foundation Molecular Grammar: Multi-Modal Foundation Models Induce Interpretable Molecular Graph Languages. *arXiv preprint arXiv:2505.22948*, 2025.

Trine V Vendelboe, Pernille Harris, Yuguang Zhao, Thomas S Walter, Karl Harlos, Kamel El Omari, and Hans EM Christensen. The crystal structure of human dopamine β-hydroxylase at 2.9 å resolution. *Science advances*, 2(4):e1500980, 2016.

Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew Mccallum. Probabilistic embedding of knowledge graphs with box lattice measures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 263–272, 2018.

Qiang Zhang, Keyan Ding, Tianwen Lv, Xinda Wang, Qingyu Yin, Yiwen Zhang, Jing Yu, Yuhao Wang, Xiaotong Li, Zhuoyi Xiang, et al. Scientific Large Language Models: A Survey on Biological & Chemical Domains. *ACM Computing Surveys*, 57(6):161, 2025.

P. Zhou, J. Wang, C. Li, Z. Wang, Y. Liu, S. Sun, J. Lin, L. Wei, X. Cai, H. Lai, W. Liu, L. Wang, Y. Liu, and X. Zen. Instruction multi-constraint molecular generation using a teacher-student large language model. *BMC Biology*, 23(105), 2025. doi: 10.1186/s12915-025-02200-3. URL https://doi.org/10.1186/s12915-025-02200-3.

Appendix A. Additional conceptual details for SNG

A.1 Note on extending other hybrid systems:

The general principle of a symbolic base combined with neural-fibres can be applied to characterise systems in each of the cases (A)–(D) in Fig. 2a.

Example 4 (Hybrid systems with a symbolic base) *Let us assume the following:*

- *A fixed universal set $\mathcal{U}$ (instances, embeddings, predictions, programs etc).*

- *Background knowledge B;*

- *A symbolic component that can enumerate elements from a set of symbolic hypotheses $\mathcal{H}$; and*

- *A neural component that – given any element from $H \in \mathcal{H}$ – can generate elements from $\mathcal{U}$ that are consistent with H.*

Additionally, the set $\mathcal{H}$ is assumed to a partially ordered set, with ordering $\geq_{\mathcal{H}}$. Associated with each element H in the ordering is a fibre-poset $F(H)$ that is obtained from the neural component. Each element of the base set is combined with elements from the fibre-sets to give a partially ordered set $\mathcal{F}$ of hybrid systems. Example hybrid systems in the categorisation in Fig. 2(a) that can be characterised in this way are:

Case	Symbolic	Neural	Base	Fibre	$\mathcal{F}$
A	Reasoning	Reasoning	Hypotheses ordered by entailment	Neural reasoners consistent with the symbolic entailments	Pairs ordered by stronger symbolic entailment and higher neural agreement
B	Learning	Reasoning	Hypotheses ordered by specialization	Neural approximators of the symbolic extensions	Pairs ordered by more specific hypotheses and more confident neural outputs
C	Reasoning	Learning	Hypotheses ordered by entailment	Neural representations trained to preserve symbolic reasoning patterns	Pairs ordered by stronger entailment and representational inclusion
D	Learning	Learning	Hypotheses ordered by generality	Neural learners minimizing a loss defined by the symbolic hypothesis	Pairs ordered by more general hypotheses and lower training loss

This example shows how the setting of a symbolic base with neural fibres can be used to characterise other hybrid systems where the symbolic component has the primary role. Further variants – of less relevance to this paper – but of wider applicability refer to: (a) Hybrid neurosymbolic systems where the neural component has the primary role (Grothendieck constructions with a neural base and symbolic fibres); and (b) Hybrid systems where both neural and symbolic components are equally important. This results in Grothendieck constructions over a base set consisting of pairs, with the fibre sets representing a hybrid semantics.[11]

---

11. This applies when the system is co-trained, or outputs are coupled. The ordering over $\mathcal{F}$ can then be very flexible, not just the obvious product orders, but also problem-specific orders.

A.2 Correctness of Procedure Gen

Proposition 2 *The set M_n returned by Procedure 1 is an element of $F(H)$ and $w_n \in [0,1]$.*

Proof Gen executes the loop (Steps 5–12) n times. We claim the following is loop invariant:

$$M_{i-1} \subseteq N(H) \cap \text{ext}(H|B) \text{ and } |M_{i-1}| \leq s \times (i-1)$$

At the start of the iteration ($i = 1$), $M_{i-1} = \emptyset$ and the invariant is trivially true. Assume the invariant holds at the start of the kth iteration. That is, $M_{k-1} \subseteq N(H) \cap \text{nvext}(H|B)$ and $|M_{k-1} \in [0,1]$. On the kth iteration: (i) $S_k \subseteq N(H)$; and (ii) In Step 10 $M_k = Z_k \cup M_{k-1}$, where $Z_k = \{x : x \in \mathcal{U}, (true, x) \in D_k\}$. Since $(true, x) \in D_k$ iff $x \in S_k \cap \text{nvext}(H|B)$. So $Z_k \subseteq N(H) \cap \text{ext}(H|B)$. Therefore $M_k \subseteq N(H) \cap \text{ext}(H|B)$. Also since $|D_k| \leq s$, $|M_k| \leq s + |M_{k-1}|$. Since $|M_{k-1}| \leq s \times (k-1)$, it follows that $|M_k| \leq s \times k$. The loop variable i is incremented to $k+1$, and at the start of the next iteration, clearly $M_{i-1} = M_k$ which is a subset of $\text{ext}(H|B)$ and has at most $s \times k$ elements. The procedure clearly terminates since i is bounded by n, and the procedure returns the set M_n which is a subset of $\text{ext}(H|B)$ and $w_n \in [0,1]$. ∎

Appendix B. Algorithmic Details: Application to Lead Discvovery

B.1 Setting up the base poset

We first introduce some definitions that are helpful in clarifying both the set of hypotheses $\mathcal{H}$ and the ordering over that set.

Definition 8 (Interval-Vectors) *An n-dimensional interval-vector $\mathbf{v} = ([a_1, b_1], \ldots, [a_n, b_n])$ is an element of $(\mathbb{R} \times \mathbb{R})^n$ where $a_i \leq b_i$ for $i \in \{1, \ldots, n\}$.*

We will sometimes denote $(\mathbb{R} \times \mathbb{R})$ as $\mathcal{I}$ and $(\mathbb{R} \times \mathbb{R})^n$ as $\mathcal{I}^n$. The set $\mathcal{I}^n$ is therefore the set of n-dimensional hyper-rectangles, and an interval-vector is a hyper-rectangle.
The following definition is useful later.

Definition 9 (Interval-Vector Containment) *Given interval-vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{I}^n$ If $(\mathbf{v}_2[1] \subseteq \mathbf{v}_1[1]) \wedge \cdots \wedge (\mathbf{v}_2[n] \subseteq \mathbf{v}_1[n])$ then we will say $\mathbf{v}_2$ is contained by $\mathbf{v}_1$ (resply. $\mathbf{v}_1$ contains $\mathbf{v}_2$) We denote this by $\mathbf{v}_2 \sqsubseteq \mathbf{v}_1$ (resply. $\mathbf{v}_1 \sqsupseteq \mathbf{v}_2$). If there exists at least one $j \in \{1, \ldots, n\}$ s.t. $\mathbf{v}_2[j] \subset \mathbf{v}_1[j]$, we will say $\mathbf{v}_2$ is properly contained by $\mathbf{v}_1$ (resply. $\mathbf{v}_1$ properly contains $\mathbf{v}_2$). We denote this by $\mathbf{v}_2 \sqsubset \mathbf{v}_1$ (resply. $\mathbf{v}_1 \sqsupset \mathbf{v}_2$).*

Clearly, if $\mathbf{v}_1 \sqsubset \mathbf{v}_2$ then $\mathbf{v}_1 \sqsubseteq \mathbf{v}_2$.

Definition 10 (Factors) *Let $\mathcal{U}$ be a set of instances. A factor is a function $f : \mathcal{U} \to \mathbb{R}$.*

Definition 11 (Factor Specification) *Let $F = (f_1, \ldots, f_n)$ be a sequence of factors. A factor specification is the pair (F, Θ) and $\Theta = ([f_1^-, f_1^+], \ldots, [f_n^-, f_n^+]) \in \mathcal{I}^n$. For $i \in \{1, \ldots, n\}$, $[f_i^-, f_i^+]$ is the range of values for the factor f_i.*

A factor-specification allows us to define the notion of an *experiment*:

Definition 12 (Experiment) *Let $\mathcal{U}$ be a set of instances. Let $((f_1, \ldots, f_n), \boldsymbol{\Theta})$ be a factor-specification. An experiment $\mathbf{e}$ given the factor-specification, or simply an experiment, is an interval-vector in $\mathcal{I}^n$ s.t. $\boldsymbol{\Theta}$ subsumes $\mathbf{e}$.*

Experiments specify conjunctive logical constraints on factors. We adopt the terminology in Inductive Logic Programming (ILP) and each experiment will be associated with a *hypothesis*.

Definition 13 (Hypothesis) *Let $\mathcal{U}$ be a set of instances, $(F, \boldsymbol{\Theta})$ be the factor specification where $F = (f_1, \ldots, f_n)$. Let $\mathbf{e}$ be an experiment given $(F, \boldsymbol{\Theta})$. Then a hypothesis given an experiment is the clause:*

$Hypothesis((f_1, \ldots, f_n), (i_1, \ldots, i_n)):$
$\forall x (\Sigma(x) \leftarrow$
$\qquad x \in \mathcal{U} \wedge$
$\qquad (f_1(x) \in i_1) \wedge \cdots \wedge (f_n(x) \in i_n))$

The hypothesis space $\mathcal{H}$ is the set of such hypotheses. The ordering on this set will be same as introduced in Defn. řefdef:pohyp, namely pointwise inclusion of extensions.

B.2 Bayesian Scoring of Hypotheses

Definition 14 (McCreath's Q-Heuristic) *Let $\mathcal{U}$ denote the set of all instances. Let h be a hypothesis as defined in Defn. 13. Let E^+ denote a set of positive examples and E^- denote a set of negative examples s.t. $(|E^+| \geq 1$ and $|E^-| \geq 0$. Let $D = E^+ \cup E^-$. Let $ext(h|B) = \{x : x \in \mathcal{X}, B \wedge h \models Feasible(x)\}$; and for any $S \subseteq \mathcal{U}$, $\theta(S) = \frac{|S|}{|\mathcal{X}|}$. Let ϵ be the probability that an instance is randomly assigned to E^+ (resply. E^-). Let B denote background knowledge, $TP(H|B, D) = \{e : e \in E^+, e \in ext(H|B)\}$; $TN(H|B, D) = \{e : \neg e \in E^-, B \wedge H \wedge e \notin ext(h|B)\}$; and $FPN(H|B, D) = D \setminus (TP(H|B, D) \cup TN(H|B, D))$. Then, dropping the inclusion of B, D for convenience, the fixed-example model in McCreath and Sharma (1998) defines the quality of a hypothesis as:*

$$Q(H) = \log(P(H))$$
$$+ |TP(H)| \log\left(\frac{1-\epsilon}{\theta(\text{ext}(H))} + \epsilon\right)$$
$$+ |TN(H| \log\left(\frac{1-\epsilon}{1 - \theta(\text{ext}(h))} + \epsilon\right)$$
$$+ |FPN(H| \log(\epsilon).$$

For the special case of $\epsilon = 0$, the quality of a hypothesis in the fixed-example setting simplifies to:
$$Q(H) = \log(P(H)) + |E^+| \log \frac{1}{\theta(\text{ext}(h))} + |E^-| \log \frac{1}{1 - \theta(\text{ext}(h))}.$$

In (McCreath and Sharma, 1998) it is shown that maximising $Q(|HB, D)$ maximises the Bayesian posterior $P(H|B, D)$, along other theoretical results including a proof of (probabilistic) convergence to a target concept. Assuming the entailment relation $\models$ can be checked, the practical difficulties in using the Q-heuristic are in obtaining the values for $\theta(\text{ext}(H))$ and $P(H)$. We note the following:

We will need the following to be able to use the Q-heuristic here:

a. In order to obtain the sets TP, TN and FPN we will require B to contain all the definitions needed to evaluate the constraint in the hypothesis (that is, B will need to contain definitions for the $f_i(\cdot)$).

b. By definition, $\text{ext}(h)$ is the set of feasible instances as defined in $Defn.1$. We can estimate $\theta(ext(h))$ on a random sample $X \subset \mathcal{X}$ as follows: Let $S = \{x : x \in X, \Phi(x) \text{ is } true\}$. Then the (maximum-likelihood) estimate of $\theta(\text{ext}(h))$ is $\hat{\theta}(ext(h)) = \frac{|S|}{|X|}$. [12]

Remark 3 *These results follow straightforwardly from the definition in Defn. 14:*

Positive-only data. *Let $E^+ \neq \emptyset$ and $E^- = \emptyset$. Let $\epsilon = 0$. If $P(H_1) = P(H_2)$, then $(Q(H_1) > Q(H_2))$ iff $(\text{ext}(h_1) < \text{ext}(h_2))$.*

Negative-only data. *Let $E^- \neq \emptyset$ and $E^+ = \emptyset$. Let $\epsilon = 0$. If $P(H_1) = P(H_2)$, then $(Q(H_1) > Q(H_2))$ iff $(\text{ext}(H_1) > \text{ext}(H_2))$.*

That is, in the noise-free case, with equal prior probabilities, and positive data only, more specific hypotheses will be preferred; and with negative data only, more general hypotheses will be preferred.

B.3 Property of the procedure `GenMol`

Proposition 3 *Let $(F, \cdot)$ be a factor-specification, B denote background knowledge. Let $\mathbf{e}_i$ ($1 \leq i \leq k$) be an experiment selected by `GenMol` on the ith iteration s.t. $\mathbf{e}_i$ is contained by $\mathbf{e}_{i-1}$. Let $H_i = Hypothesis(F, \mathbf{e}_i)$, and $H_{i-1} = Hypothesis(F, \mathbf{e}_{i-1})$. Then $H_{i-1} \models H_i$.*

Proof First we observe that $H_i(x) : (\Sigma(x) \leftarrow (x \in \mathcal{U} \wedge C_i(x))$ and $H_{i-1}(x) : (\Sigma(x) \leftarrow (x \in \mathcal{U}) \wedge C_{i-1}(x))$. It is easy to see that since by construction $\mathbf{e_{i-1}}$ contains $\mathbf{e_i}$, $\forall x(C_i(x) \models C_{i-1}(x))$. Now suppose $H_{i-1} \not\models H_i$. That is, there exists some $a \in \mathcal{U}$ s.t. $H_{i-1}(a)$ is true and $H_i(a)$ is false. Since $H_i(a)$ is false, $\Sigma(a)$ is false and $C_i(a)$ is true. Since $C_i(x) \models C_{i-1}(x)$ for all x, and $C_i(a)$ is true, therefore $C_{i-1}(a)$ is true. Since H_{i-1} is assumed true, and $C_{i-1}(a)$ is true, then $\Sigma(a)$ is true, which is a contradiction. So for all $a \in \mathcal{U}$, whenever H_{i-1} is true then H_i is also true. ∎

---

12. A sample-based estimate is also used in McCreath and Sharma (1998) and Muggleton (1996).

Appendix C. Additional Experimental Details: Case Studies

C.1 Method

The following additional details are relevant to the experimental method:

- In the controlled (Validate) experiments, we are only attempting to optimise one factor, namely: docking score, which is indicative of binding affinity. This is in line with what was done in (Brahmavar et al., 2024). For the open-ended (Explore) experiments, we will extend this to include: number of synthesis steps, and estimated yield per step.

- The factor-set specification also requires identifying minimum and maximum for the initial search space. For all experiments, we use: $\{affinity : [3, 10], molwt : [200, 700], SAS : [0, 7.0]\}$, where *affinity* is the predicted affinity from GNINA software, *molwt* is the molecular weight, and *SAS* denotes synthesis accessibility score.

- For the controlled experiments, we use the known inhibitors and non-inhibitors as the dataset D. For the open-ended experiments we use 5 known inhibitors of DBH and 5 randomly sampled molecules from ChEMBL as non-inhibitors. Estimating the Q-heuristic requires a sample of unlabelled molecules. For this we use a randomly drawn set of 1000 molecules from the ChEMBL database.

- The description of `SearchHyp` does not specify a sampling method for obtaining subsumed hyper-rectangles. We use an approach based on Latin Squares Hyper-Rectangle Sampling (LHRS: (McKay et al., 2000)). Some additional prior information may be available that may be used to modify the basic LHRS approach: (a) If we know beforehand that a factor is to maximised (for example, binding affinity), then we do not sample points from the upper-end of the range for the factor. Thus, subsuming rectangles are obtaining by only random placements of the lower-end; (b) Similarly, if we know beforehand that we want to minimise a factor, then only the upper-threshold is sampled. If nothing is known then a standard LHRS approach is adopted.

- For all experiments, we use a value of $s = 10$; and $n = 10$ for `GenMol` and we sample 100 molecules when it returns the optimal hypothesis. The set of feasible molecules during search and in the final generation are considered for evaluation.

- For molecule generation from LLM, we use model's chat-completion API with a maximum output length of $128 \times$ `max_samples` tokens, a temperature of 0.7 to balance diversity and determinism. The prompt content was provided via the `messages` parameter (see Appendix C for details).

- We assess the results of controlled experiments by examining the range and median docking scores of the molecules generated, and compare those to those generated by the LMLF procedure in (Brahmavar et al., 2024). For the open-ended experiments, we obtain the statistics on docking scores, and compare them to those of the latest generation of molecules used for DBH inhibition (see Sec. 4.2). In addition, we also provide an assessment of the molecules by an expert synthetic chemist.

- For each target problem, we assess the novelty of the generated molecules by using the average Tanimoto (or Jaccard) coefficient to the database of known inhibitors.

C.2 Results

The following details refer to the questions in Sec. 4.3.

The choice of LLM. Figure 11 compares the performance of `GenMol` when implemented with two different general-purpose LLMs: OpenAI's GPT-4o and Anthropic's Claude 3.5 Sonnet (Anthropic, 2024). Both these `GenMol` variants are evaluated under identical parameter settings and prompts to ensure a fair comparison. Results are shown for both the zero early context ($|C_0| = 0$) and early context ($|C_0| = 5$) settings across the three benchmark protein targets. Overall, the two models yield comparable mean binding affinities, with neither showing a clear advantage from the inclusion of early context. This similarity in performance between the two general-purpose LLMs raises the question whether a domain-specific LLM might offer a potential advantage. We examine this next.

We implement a `GenMol` variant using 87M parameter variant of GPT-2 that has been trained on a large corpus of about 480 million molecules (as SMILES strings) from the ZINC database (entropy, 2023), which provides extensive coverage of chemical space and encodes general chemical grammar and structure-property relationships. We refer this model here as Molecule-GPT2. This model is expected to generate syntactically valid and chemically diverse molecules due to its broad exposure to molecular representations. However, because the training data does not contain information specific to the protein targets considered here, the model may be lacking the ability to exploit target-specific binding patterns that are otherwise known to the general-purpose LLMs through their enormous training corpus of research articles and large-scale studies. Nevertheless, Molecule-GPT2 produces competitive predicted affinities for JAK2 and DRD2, which are probably very highly studied proteins and for these substantial domain-knowledge is available from the literature. Interestingly, it outperforms general-purpose LLMs for DBH, a relatively lesser studied target and additional target-specific knowledge has to be inferred from known inhibitor molecules.

Although these results indicate that a general-purpose LLM is more suited for target-specific lead generation in the manner done in this study, a more thorough investigation is warranted. Furthermore, while prompting was used directly to generate molecules from the general-purpose LLMs, the same approach was not straightforward for the domain-specific LLM (Molecule-GPT2), primarily because it was not trained in the same manner as GPT-4o or Claude 3.5 Sonnet. For each protein target, we first had to construct a small set of structural scaffolds by examining known inhibitor molecules, and then use these scaffolds as prefixes for GPT2-based molecule generation (see Appendix C for more details on this). This additional preprocessing step may have contributed to the lower performance observed for Molecule-GPT2 relative to the general-purpose LLMs.

The effect of parameters. We consider 3 parameters: (a) Initial context provided to the LLM; (b) sample-sizes used by the symbolic learner; and (c) LLM temperature. Fig. 13 shows the changes in mean binding affinity with the initial context ("few-shots") provided to the LLM. The results suggest that there is little difference in predicted affinity, although there is a large change in the Tanimoto coefficient for the DBH data.

Figures 14–15 shows the effect of varying (in turn): sample-size in `GenMol`; and LLM "temperature". `GenMol` with GPT-4o shows statistically significant difference in predicted binding affinities, with two different sample sizes ($s = 10, 20$) for the JAK2 and DBH proteins. Although having a high sample size ($s = 20$) may result in better and more

Problem	$|C_0|$	GPT-4o	Claude 3.5 Sonnet
JAK2	0	7.72 (1.53)	8.41 (0.25)
	5	8.59 (0.23)	8.43 (0.22)
DRD2	0	7.40 (0.61)	7.44 (0.48)
	5	7.53 (0.47)	7.43 (0.42)
DBH	0	4.72 (0.30)	4.67 (0.63)
	5	4.23 (0.41)	4.53 (0.58)

Figure 11: Comparison of mean (standard deviation) predicted binding affinities for molecules generated by SNG using two LLMs: OpenAI's GPT-4o and Anthropic's Claude 3.5 Sonnet. Both models are evaluated without early context ($|C_0| = 0$) and with early context ($|C_0| = 5$) settings across three protein targets (JAK2, DRD2, DBH). For both LLMs, the temperature parameter was fixed at 0.7.

Problem	GPT-4o	Claude 3.5 Sonnet	Molecule-GPT2
JAK2	7.72 (1.53)	8.41 (0.25)	6.50 (0.60)
DRD2	7.40 (0.61)	7.44 (0.48)	6.99 (0.61)
DBH	4.72 (0.30)	4.67 (0.63)	5.74 (0.41)

Figure 12: Comparison of mean (standard deviation) predicted binding affinities for molecules generated by SNG variants (with zero early context) and a 87M parameter GPT2 model trained with 480M molecules from the ZINC database.

Problem	$|C_0|$	Affinity	TC
JAK2	0	7.72 (1.53)	0.14 (0.032)
	5	8.59 (0.23)	0.15 (0.014)
DRD2	0	7.40 (0.61)	0.13 (0.034)
	5	7.53 (0.47)	0.13 (0.033)
DBH	0	4.72 (0.30)	0.24 (0.056)
	5	4.23 (0.41)	0.08 (0.022)

Figure 13: Statistics of binding affinities and novelty of LLM-generated molecules using `GPT-4o`. The entries represent the mean values, with standard deviations shown in parentheses. C_0 denotes a set of known inhibitor of the target protein, provided as an early context to the LLM during prompting (see later). Predicted affinity refers to the predicted binding affinity from `GNINA` software.

consistent binding affinities, this was not observed for DRD2 protein. Figure 15 shows the effect of sampling temperature on predicted binding affinities for molecules generated by `GenMol` using GPT-4o and Claude 3.5 Sonnet. For GPT-4o, temperature changes produce statistically significant differences in mean affinity scores, whereas Molecule-GPT2 exhibits minimal variation across temperatures and statistically significant difference in number of molecules generated by `GenMol` and the mean affinity. We performed a similar investigation

using Claude 3.5 Sonnet and observe that it remains largely stable across temperatures, with no statistically significant variation in mean affinity.
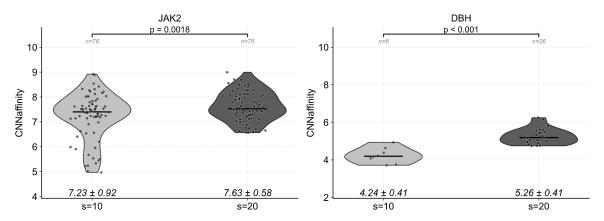

Figure 14: Effect of the parameter s on `GenMol`'s performance in generating molecules for JAK2 and DBH proteins. The LLM used here is GPT-4o.
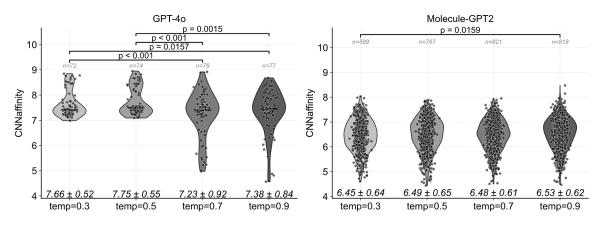

Figure 15: Effect of the temperature parameter while generating molecules using `GenMol` with a general-purpose LLM, GPT-4o (left) and a domain-specific LLM, Molecule-GPT2 (right). p-value is computed using Welch's t-test comparing the mean CNNaffinity between two temperature settings.

C.3 Prompts

We distinguish between 2 types of prompts for the API calls to the LLM (in this paper, `GPT`):

- System prompt: We use this prompt to guide the model's behaviour and responses. It sets the overall instructions for the model, such as defining its role and the syntactic format in which it should respond. In this work, we use this as: "You are a scientist specialising in chemistry and drug design. Your task is to generate valid SMILES strings as a comma-separated list inside square brackets. Return the response as plain text

without any formatting, backticks, or explanations. The response must be formatted exactly as follows: [SMILES1, SMILES2, ...]. Avoid any extra text or explanations."

- User prompt: This is the input provided by the user, containing the actual query constructed in manner described below. The LLM generates responses based on this input while considering the instructions set by the system prompt. There are two kinds of user prompts based on whether a set of inhibitors are shown to the LLM during search and generation or not. For revealing known inhibitors, we use: "Generate up to s novel valid molecules similar to the following positive molecules: [...]". Otherwise, the prompt is simply "Generate up to s novel valid molecules". We also allow feasible molecules generated in GenMol to be used as "context". In this case, we use it as a part of the user prompt as: "Additionally, consider these previously generated feasible molecules: [...]."

C.4 Domain-specific LLM: Molecule-GPT2

We implemented a variant of GenMol using the publicly available GPT2 model hosted on Hugging Face (entropy, 2023). This model, which we call Molecule-GPT2, is based on the GPT2 architecture and has approximately 87 million parameters and trained with about 480 million molecules (in SMILES representation) from the ZINC database (Irwin et al., 2020). Unlike general-purpose LLMs such as GPT-4o and Claude 3.5 Sonnet, Molecule-GPT2 is not instruction-tuned and does not natively support natural language prompting for molecule generation. Therefore, we employed a scaffold-based prefixing strategy to condition the model's outputs on specific protein targets. For each target protein, we first identified a small set of representative molecular scaffolds by extracting common substructures from known inhibitors. In all our experiments, we restrict to top 5 scaffolds, with each scaffold of maximum length 8. These scaffolds, encoded in SMILES format, were then used as fixed prefixes during autoregressive generation with the GPT2 model. We performed molecule generation using HuggingFace's model.generate() API with a maximum output length of 120 tokens, nucleus sampling with $p = 0.9$, temperature $= 0.8$, and stochastic sampling enabled. For each scaffold, we generated multiple candidate molecules for downstream binding affinity evaluation for GenMol. Although this approach may not be the optimal one, it does ensure that the generated molecules were chemically valid and retained structural motifs relevant to the target protein.