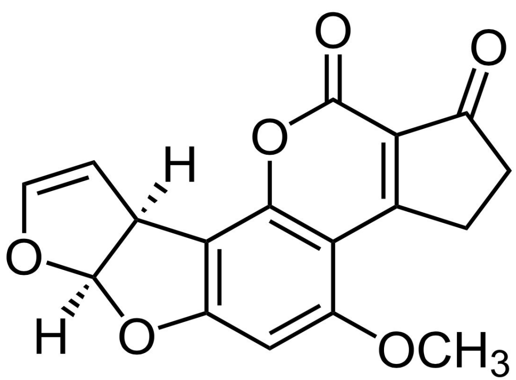
Grammar Variational Autoencoder (GVAE)
&
Syntax-Directed Variational Autoencoder For Structured Data (SD-VAE)

Prepared by: Qi He, Wei Zheng, Siyu Ji

Motivation

- Train generative models to construct more complex, discrete data types.
- Existing methods often produce invalid outputs.



Expression
$\mathtt{x}/1 + \mathtt{sin}(3) + \mathtt{sin}(\mathtt{x} * \mathtt{x})$
$1/2 + (\mathtt{x}) + \mathtt{sin}(\mathtt{x} * \mathtt{x})$
$\mathtt{x}/\mathtt{x} + (\mathtt{x}) + \mathtt{sin}(\mathtt{x} * \mathtt{x})$

```
v3=sin(v0);v8=exp(2);v9=v3-v8;v5=v0*v9;return:v5
v2=exp(v0);v7=v2*v0;v9=cos(v7);v8=cos(v9);return:v8
```

Introduction : GVAE & SD-VAE

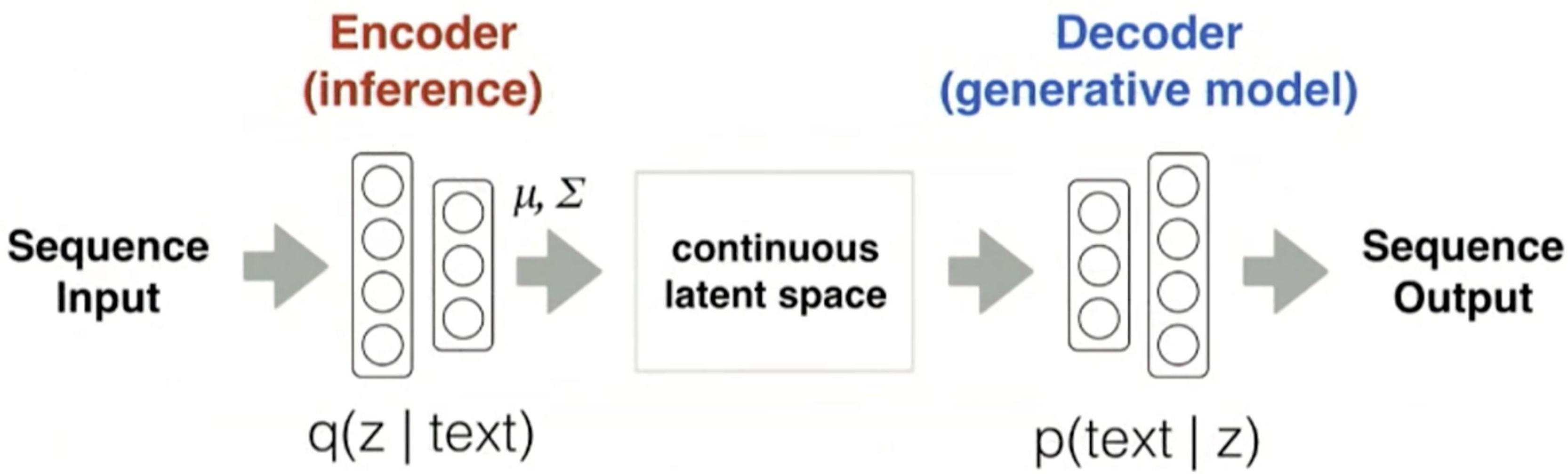
GVAE

- Learning syntactic rules to produce valid outputs
- Two different tasks : arithmetic expressions, molecules

SD-VAE

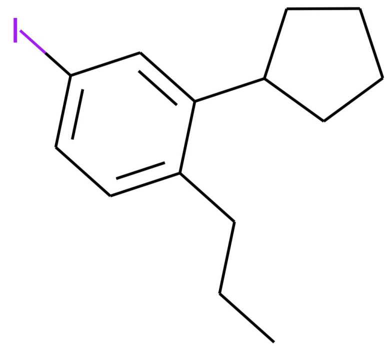
- Generate both syntactically and semantically correct data
- Efficient learning and inference
- Two different tasks: molecules generation, program generation

Variational Autoencoder for “text”

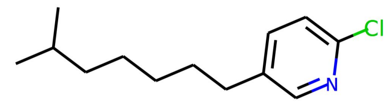


Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2015). Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Formal Languages



CCCc1ccc(I)cc1C1CCC-c1



CC(C)CCCCCc1ccc(Cl)nc1

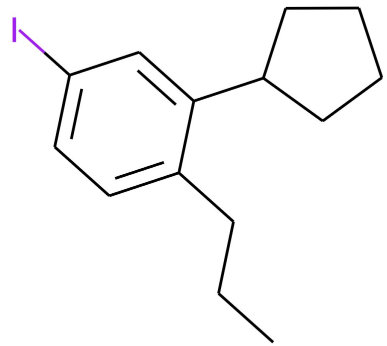
Challenges:

1. Formal Languages is very strict
2. Small changes in output leads to syntax error

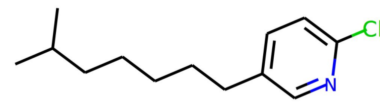
Opportunities:

1. Syntax is context free
2. Grammar is known and fixed
3. Parses are unique

Idea



CCCc1ccc(I)cc1C1CCC-c1

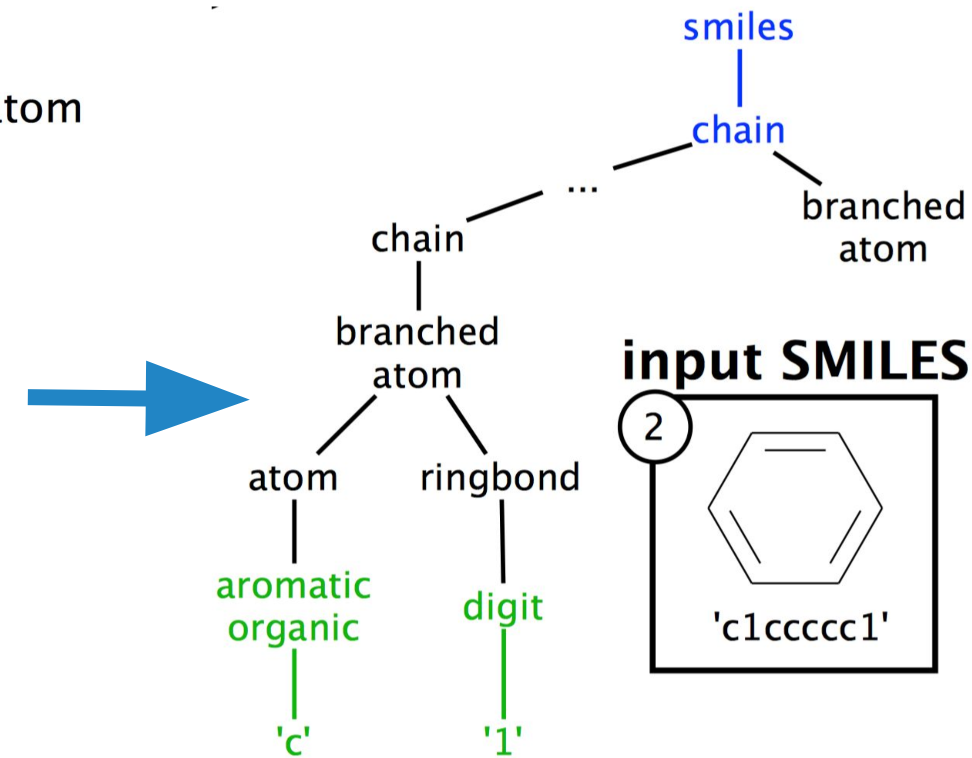


CC(C)CCCCCc1ccc(Cl)nc1

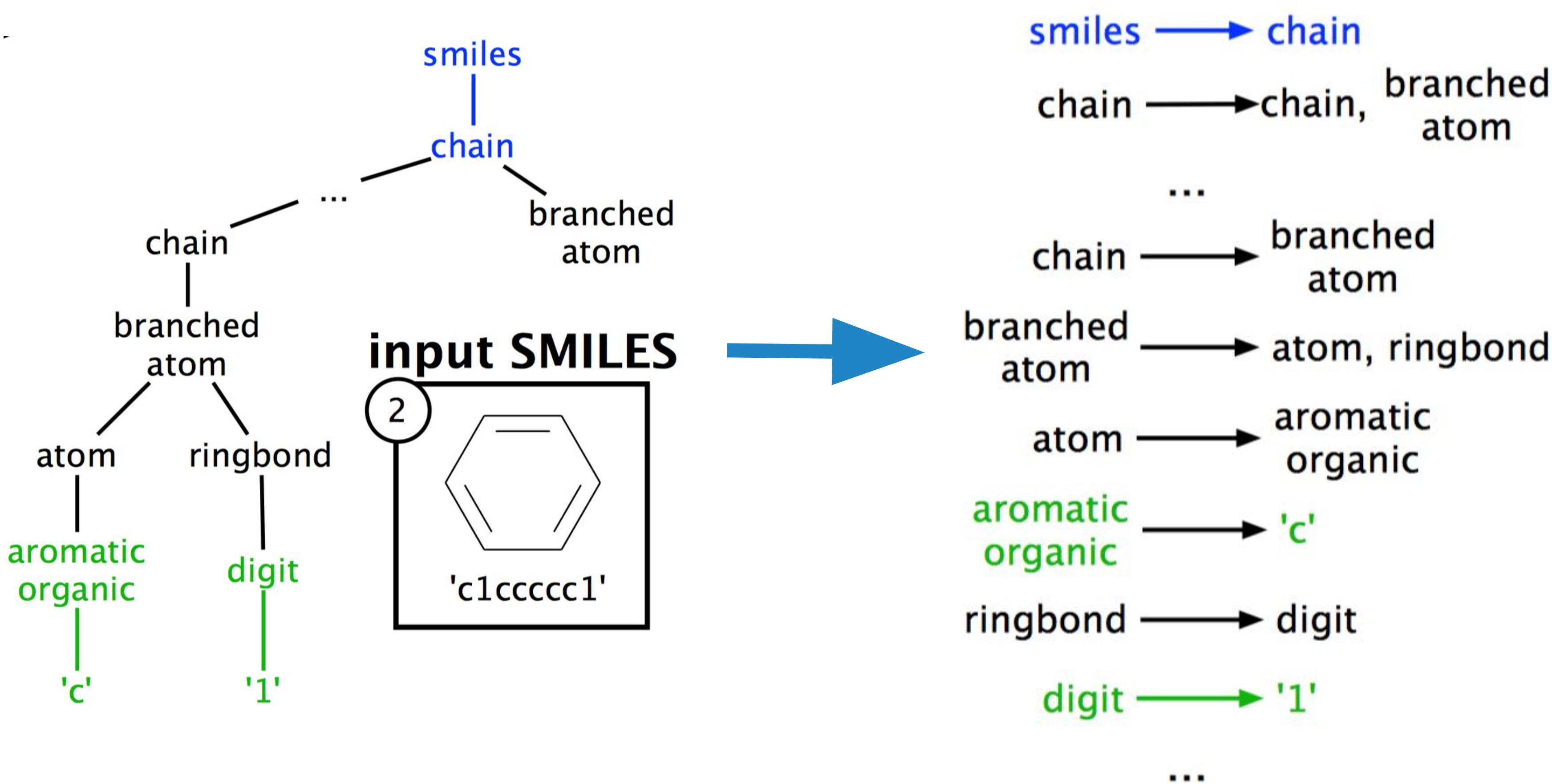
Generating string using the production rules in the grammar of the language

Encoding - form parse tree

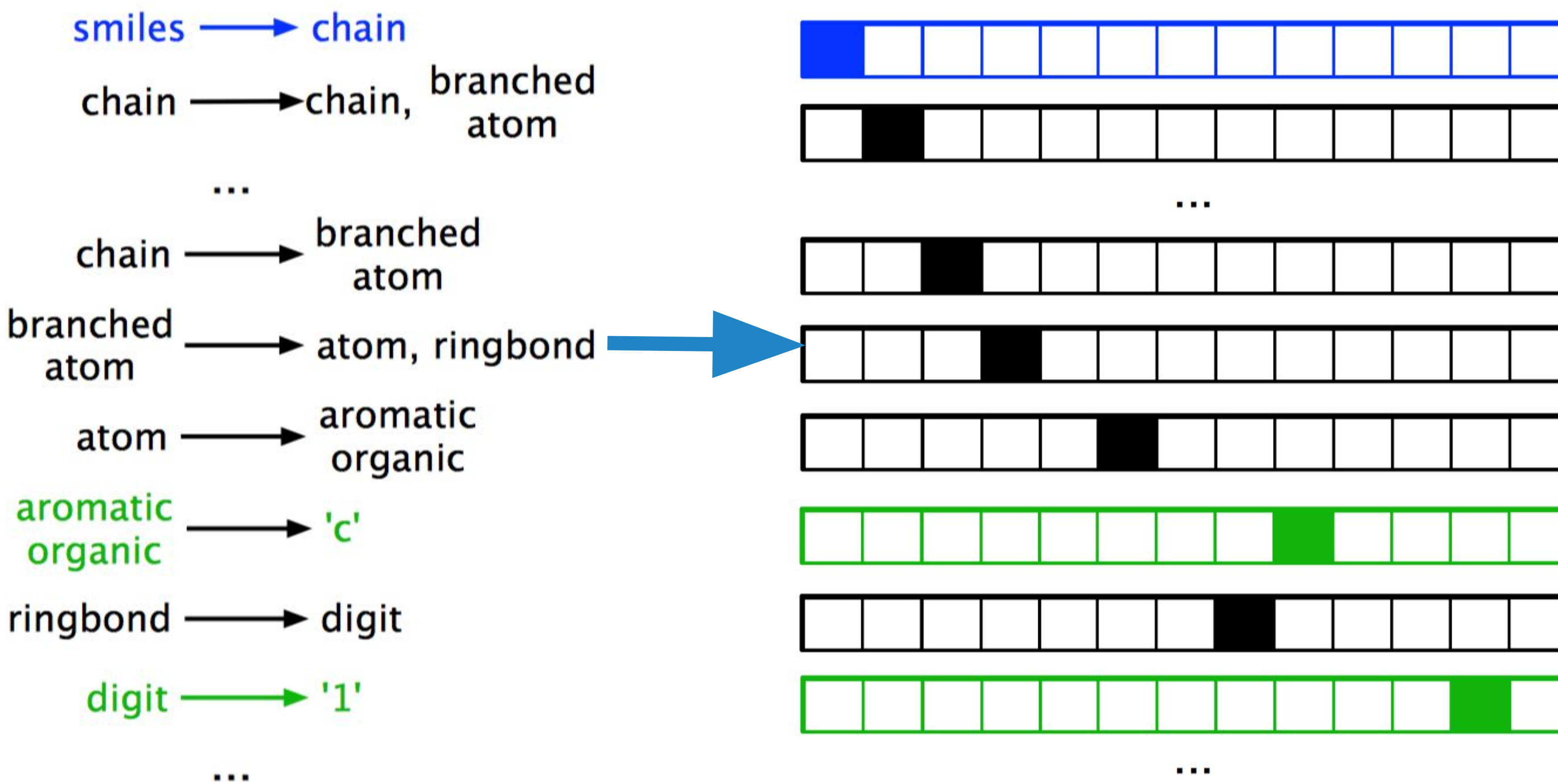
smiles → chain
chain → chain, branched atom
chain → branched atom
branched atom → atom, ringbond
branched atom → atom
atom → aromatic organic
atom → aliphatic organic
ringbond → digit
aromatic organic → 'c'
aliphatic organic → 'C'
aliphatic organic → 'N'
digit → '1'
digit → '2'



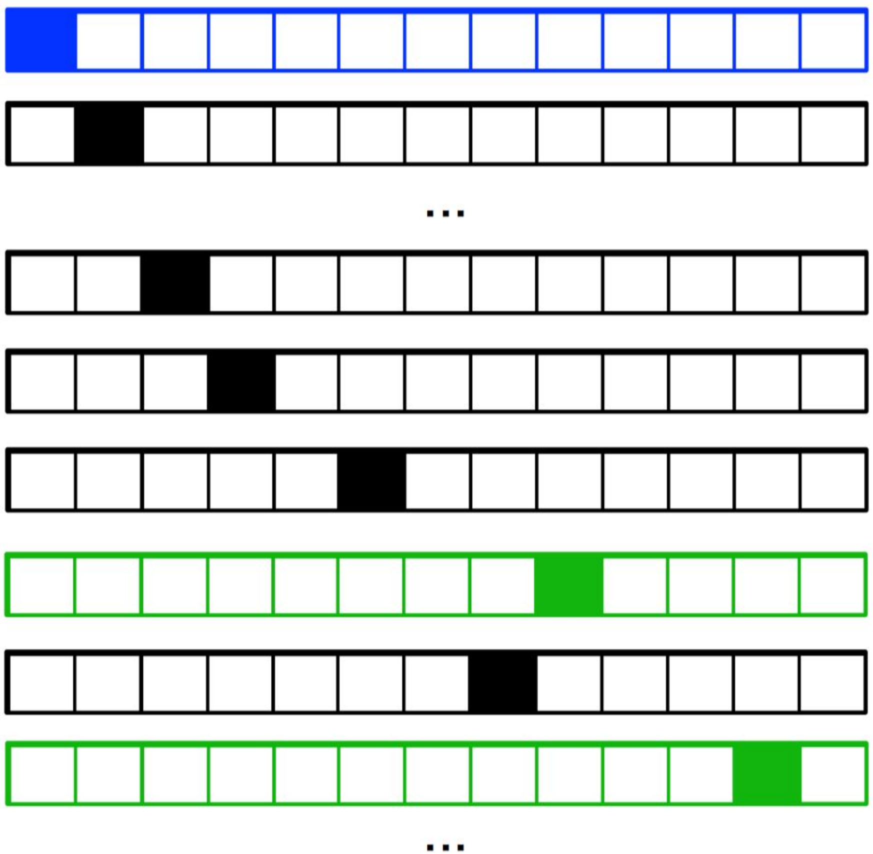
Encoding - extract rules

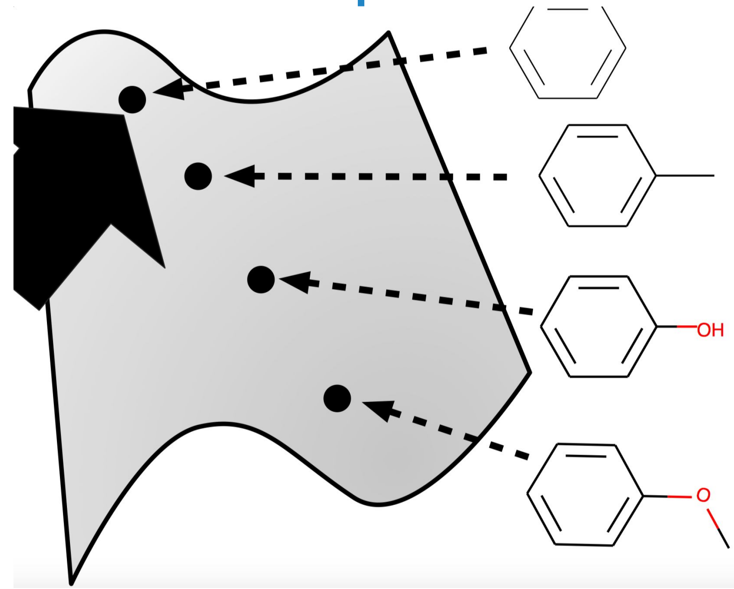


Encoding - Convert rules to one hot encoding



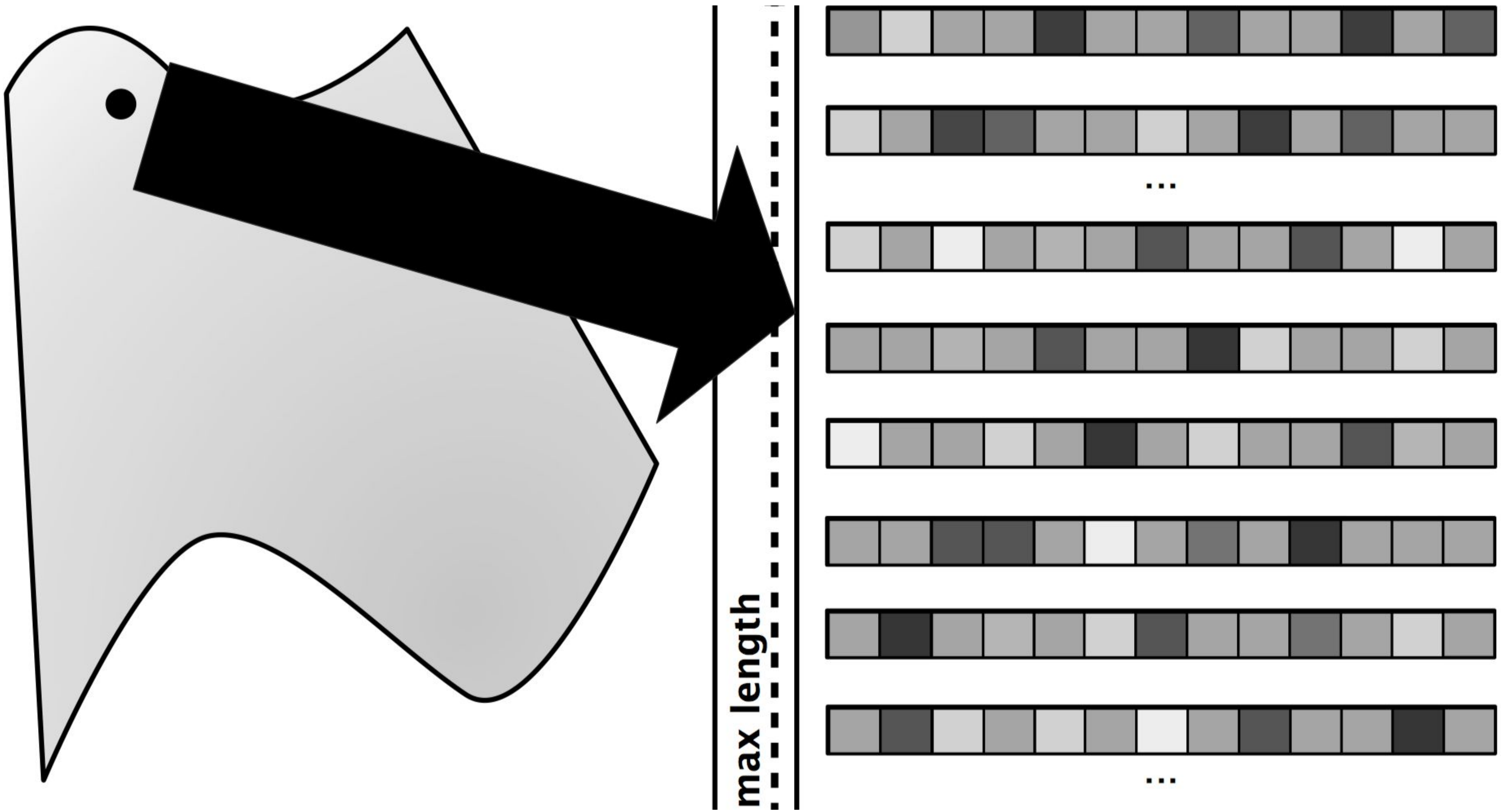
Encoding - map to latent space

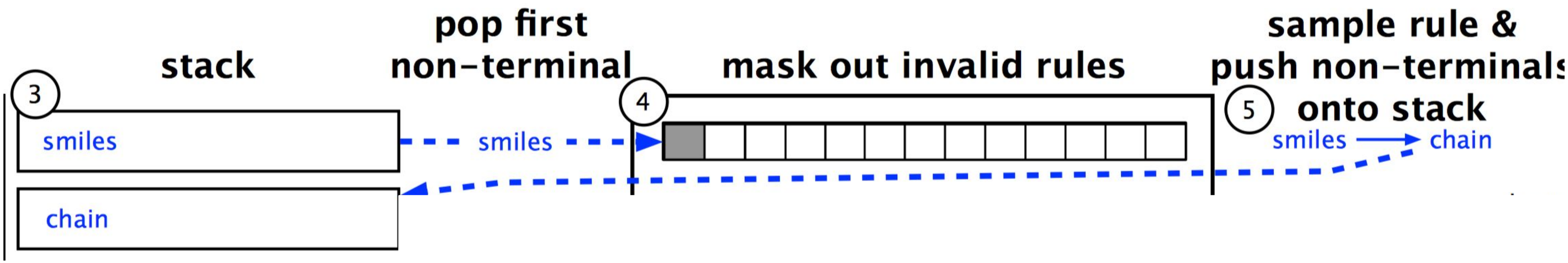




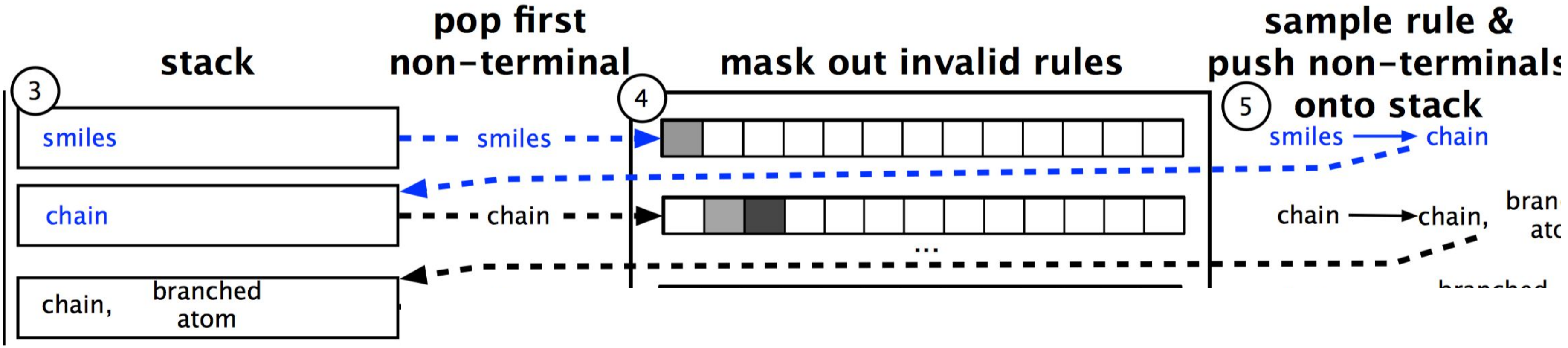
Was: One-hot characters
Now: One-hot production Rules

Decoding

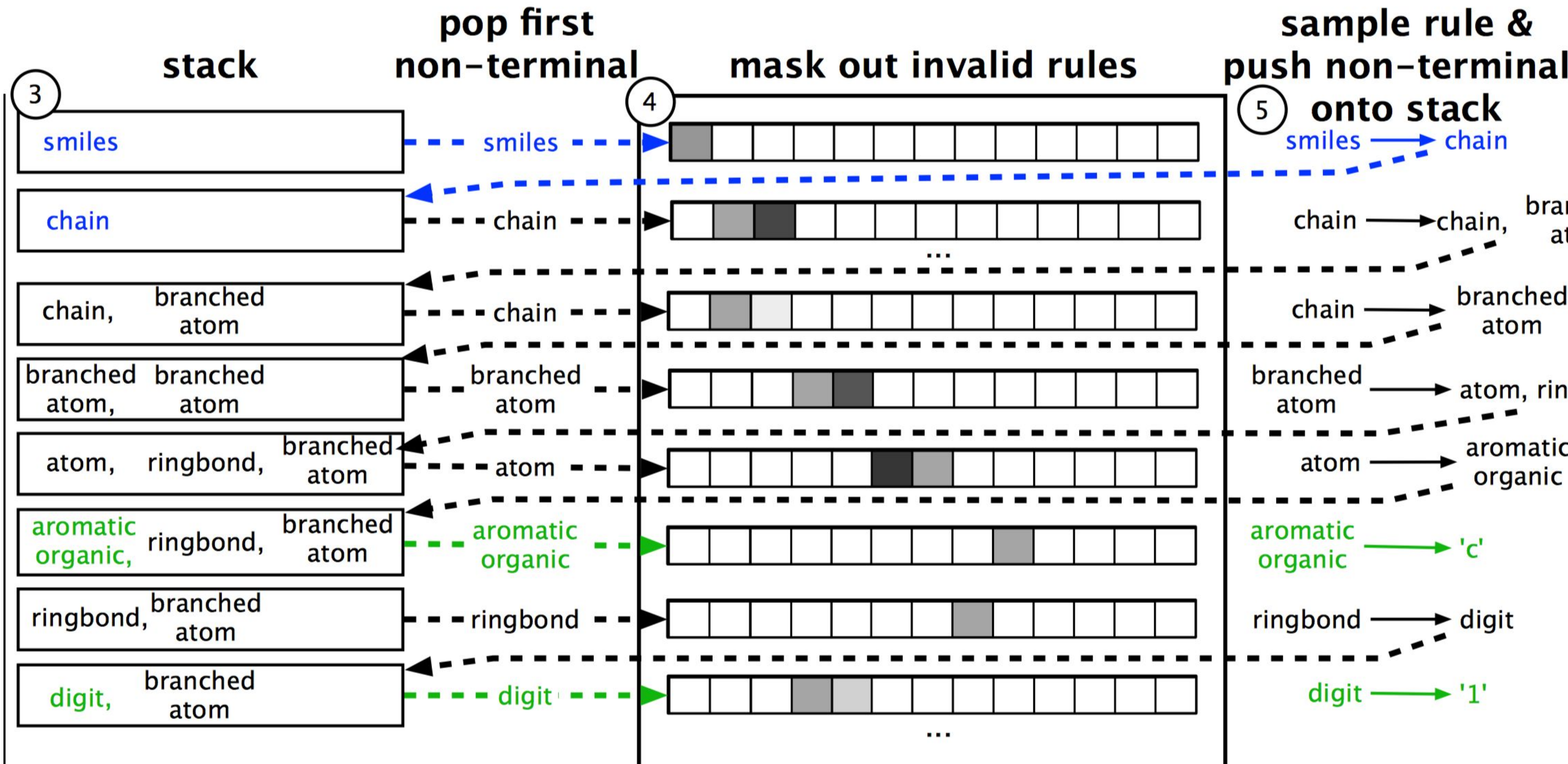




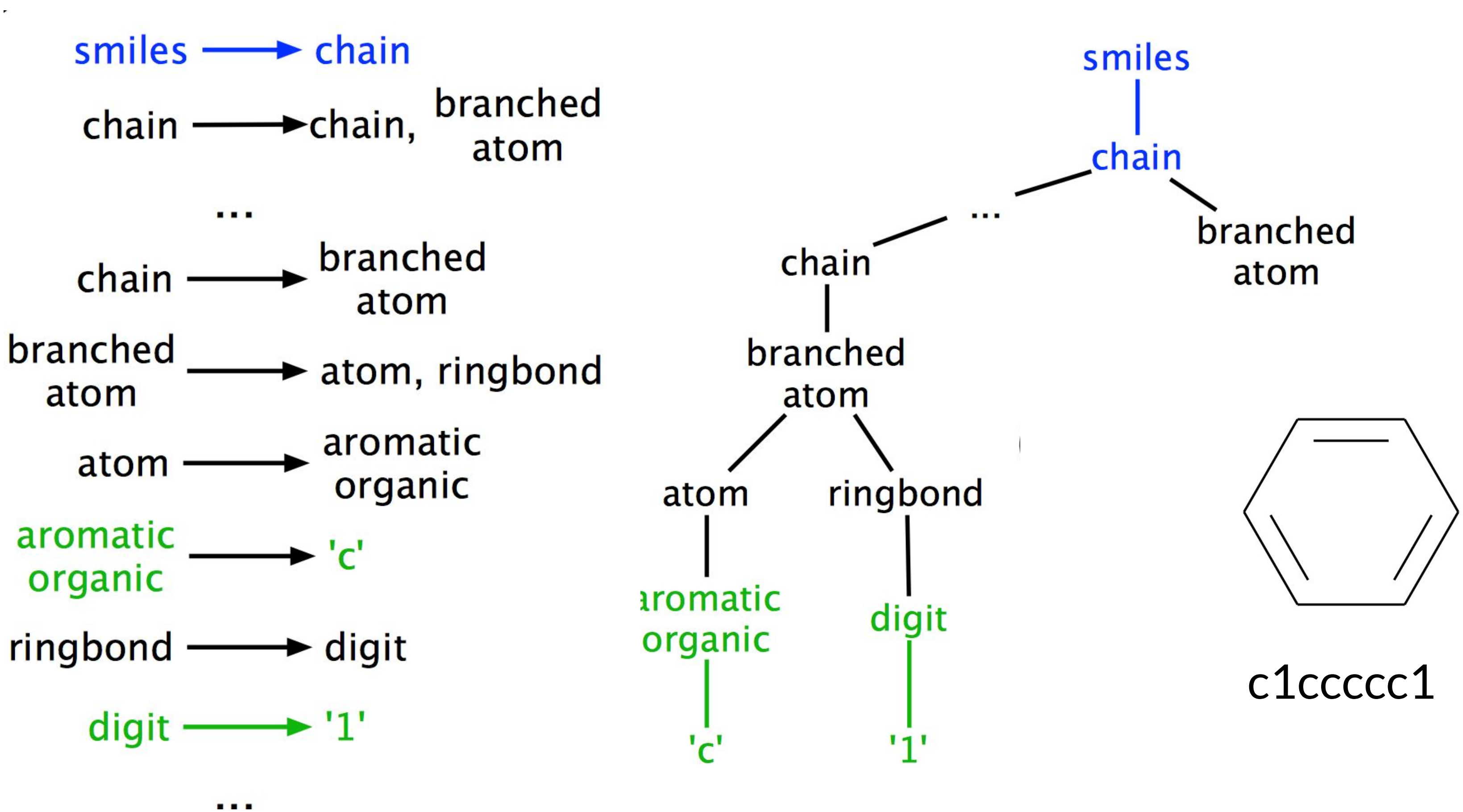
stack
pop first
non-terminal
mask out invalid rules
sample rule &
push non-terminals
onto stack
3
4
5
smiles
chain
smiles
smiles
chain



stack
pop first non−terminal
mask out invalid rules
sample rule & push non−terminals onto stack
3
4
5
smiles
smiles
smiles
chain
chain
chain
chain
chain,
chain, branched atom
...



stack
pop first non-terminal
mask out invalid rules
sample rule & push non-terminals onto stack
3
4
5
smiles
chain
chain, branched atom
branched atom, branched atom
atom, ringbond, branched atom
aromatic organic, ringbond, branched atom
ringbond, branched atom
digit, branched atom
smiles
chain
chain
branched atom
atom
aromatic organic
ringbond
digit
smiles → chain
chain → chain, bran atc
chain → branched atom
branched atom → atom, ring
atom → aromatic organic
aromatic organic → 'c'
ringbond → digit
digit → '1'


smiles → chain
chain → chain, branched atom
...
chain → branched atom
branched atom → atom, ringbond
atom → aromatic organic
aromatic organic → 'c'
ringbond → digit
digit → '1'
...
smiles
chain
...
branched atom
chain
branched atom
atom
ringbond
aromatic organic
digit
'c'
'1'
c1ccccc1

GVAE vs CVAE

- Character VAE select any possible character
- Grammar VAE select **syntactically-valid** sequences
  - Stack
  - Mask operation

- CVAE and GVAE do not always produce **semantically-valid** sequence

Syntax and semantics check

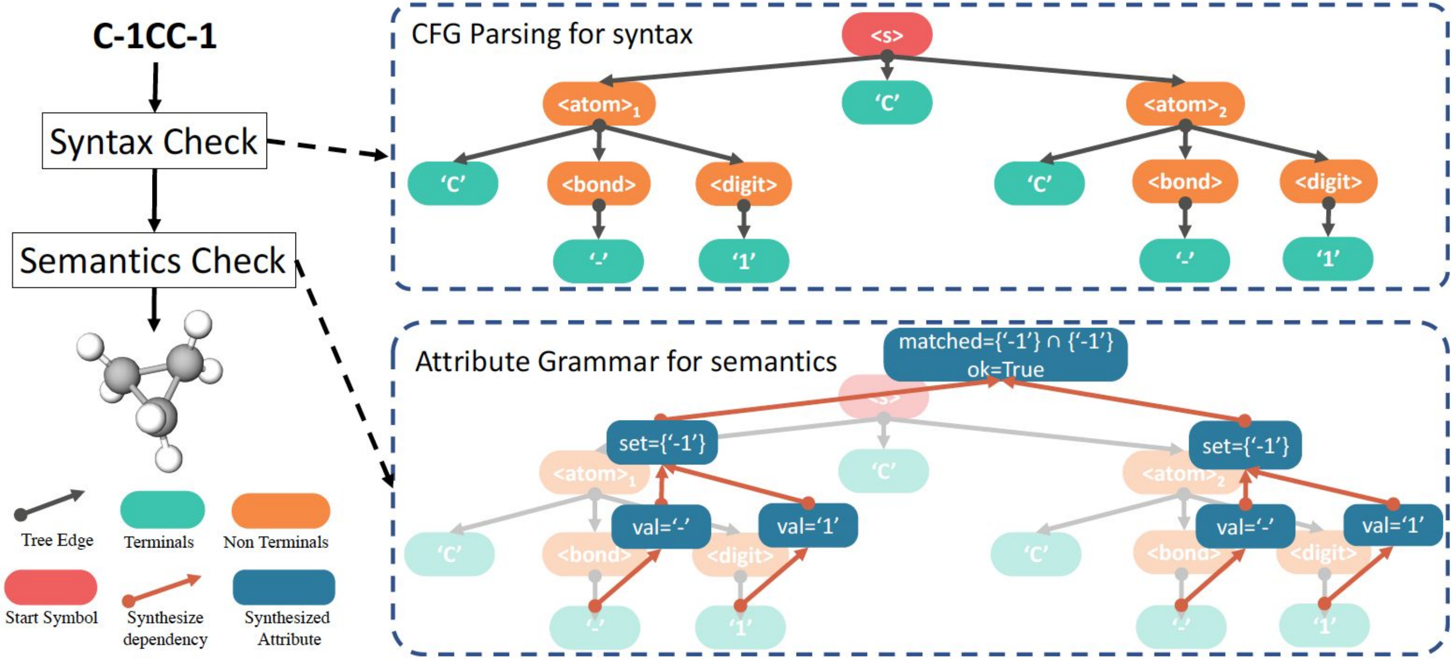


Figure 2: Bottom-up syntax and semantics check in compilers.

SD-VAE Structure

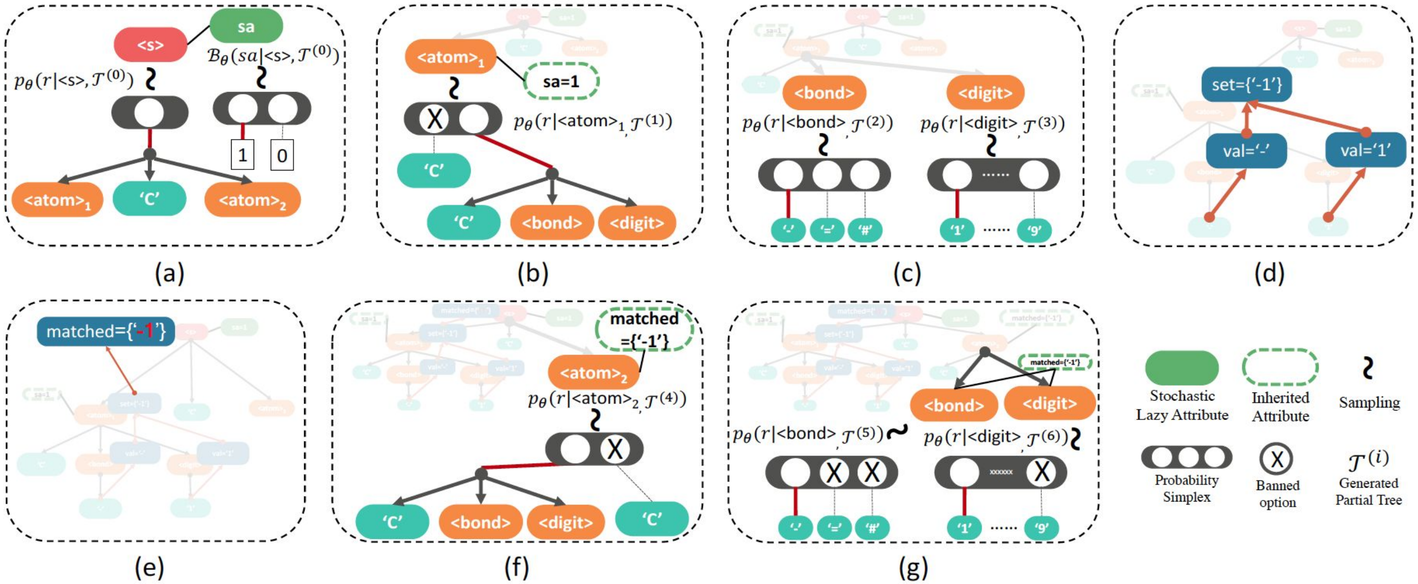


Figure 3: On-the-fly generative process of SD-VAE in order from (a) to (g). Steps: (a) stochastic generation of attribute; (b)(f)(g) constrained sampling with inherited attributes; (c) unconstrained sampling; (d) synthesized attribute calculation on generated subtree. (e) lazy evaluation of the attribute at root node.

Arithmetic expression

Given a set of 100,000 randomly generated univariate arithmetic expressions from the following grammar:

$$
\begin{aligned}
S &\rightarrow S \text{ `+' } T \mid S \text{ `*' } T \mid S \text{ `/' } T \mid T \\
T &\rightarrow \text{ `(' } S \text{ `)' } \mid \text{ `sin(' } S \text{ `)' } \mid \text{ `exp(' } S \text{ `)' } \\
T &\rightarrow \text{ `x' } \mid \text{ `1' } \mid \text{ `2' } \mid \text{ `3' }
\end{aligned}
$$

Limit the length to 15 production rules

Examples: sin(2), x/(3+1), 2 + x + sin(1/2), etc.

Train both CVAE and GVAE to learn a latent space

Smoothness

Interpolation between two arithmetic expressions

Bowman et al. (2016)

- Encode two equations
- Perform Linear interpolation in the latent space

Intermediate point which does not decode to valid equations

Character VAE	Grammar VAE
3*x+exp(3)+exp(1)	**3*x+exp(3)+exp(1)**
2*2+exp(3)+exp(1)	3*x+exp(3)+exp(1)
3*1+exp(3)+exp(2)	3*x+exp(x)+exp(1/2)
2*1+exp3)+exp(2)	2*x+exp(x)+exp(1/2)
2*3+(x)+exp(x*3)	2*x+(x)+exp(1*x)
2*x+(2)+exp(x*3)	2*x+(x)+exp(x*x)
2*x+(1)+exp(x*x)	**2*x+(1)+exp(x*x)**

Expression best fits the dataset

1000 input values x linearly-spaced between [-10,10]

True function: 1/3 + x + sin(x*x)

5 iterations of batch Bayesian optimization using Expected Improvement (EI)

Average across 10 repetitions of the process

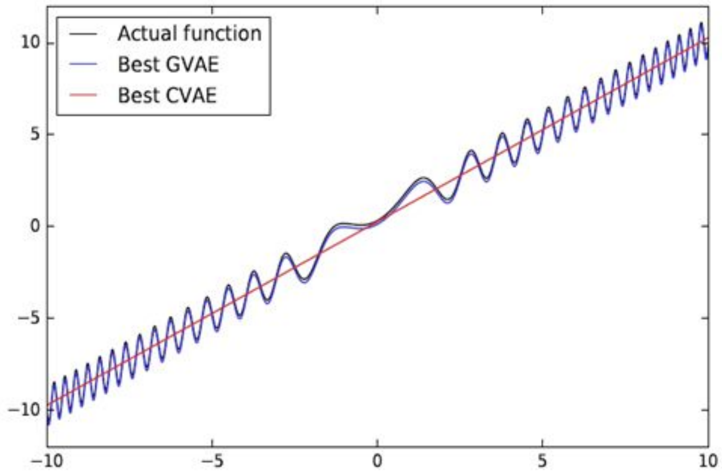
Method	Frac. valid	Avg. score
GVAE	**0.99±0.01**	**3.47 ±0.24**
CVAE	0.86±0.06	4.75±0.25

*Use log(1+MSE) to measure best fit.

Expression best fits the dataset

True function: 1/3 + x + sin(x*x)

Method	#	Expression	Score
GVAE	1	$x/1 + \sin(3) + \sin(x*x)$	**0.04**
	2	$1/2 + (x) + \sin(x*x)$	**0.10**
	3	$x/x + (x) + \sin(x*x)$	**0.37**
CVAE	1	$x*1 + \sin(3) + \sin(3/1)$	0.39
	2	$x*1 + \sin(1) + \sin(2*3)$	0.40
	3	$x + 1 + \sin(3) + \sin(3+1)$	0.40



Program Semantics

The programs are represented as a list of statements.

Each statement is an atomic arithmetic operation on variables.

V3=sin(V0);V8=exp(2);V9=V3-V8;V5=V0∗V9;return:V5

Program Semantics:

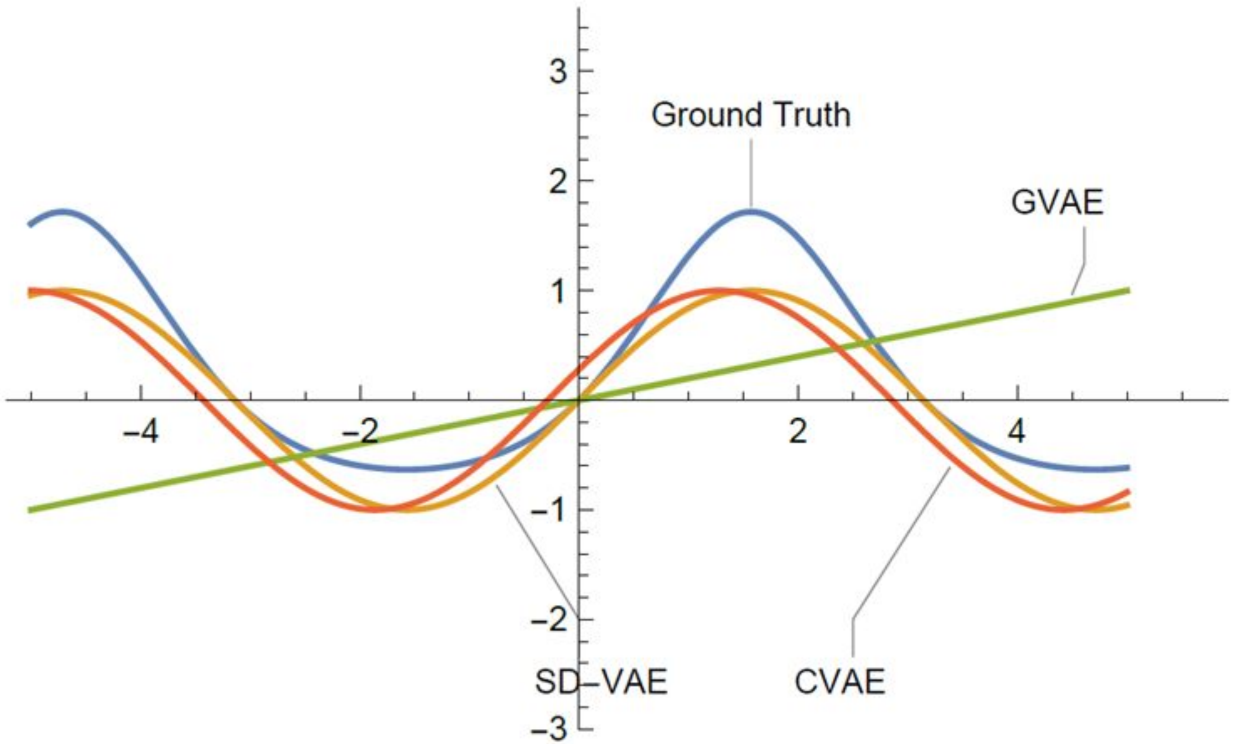
1. Variables should be defined before use.
2. Program must return a variable.
3. Number of statements should be less than 10.

CVAE
v6=cos(7);v8=exp(9);v2=v8*v0;v9=v2/v6;return:v9
v8=cos(3);v7=exp(7);v5=v7*v0;v9=v9/v6;return:v9
v4=cos(3);v8=exp(3);v2=v2*v0;v9=v8/v6;return:v9
v6=cos(3);v8=sin(3);v5=v4*1;v5=v3/v4;return:v9
v9=cos(1);v7=sin(1);v3=v1*5;v9=v9+v4;return:v9
v6=cos(1);v3=sin(10;;v9=8*v8;v7=v2/v2;return:v9
v5=exp(v0;v4=sin(v0);v3=8*v1;v7=v3/v2;return:v9
v5=exp(v0);v1=sin(1);v5=2*v3;v7=v3+v8;return:v7
v4=exp(v0);v1=v7-8;v9=8*v3;v7=v3+v8;return:v7
v4=exp(v0);v9=v6-8;v6=2*v5;v7=v3+v8;return:v7
v6=exp(v0);v8=v6-4;v4=4*v8;v7=v4+v8;return:v7

SD-VAE
v6=cos(7);v8=exp(9);v2=v8*v0;v9=v2/v6;return:v9
v6=cos(7);v8=exp(9);v2=v8*v0;v9=v2/v6;return:v9
v6=cos(7);v8=exp(9);v3=v8*v0;v9=v3/v8;return:v9
v6=cos(7);v8=v6/9;v1=7*v0;v7=v6/v1;return:v7
v6=cos(7);v8=v6/9;v1=7*v6;v7=v6+v1;return:v7
v6=cos(7);v8=v6/9;v1=7*v8;v7=v6+v8;return:v7
v6=exp(v0);v8=v6/2;v9=6*v8;v7=v9+v9;return:v7
v6=exp(v0);v8=v6-4;v9=6*v8;v7=v9+v8;return:v7
v6=exp(v0);v8=v6-4;v9=6*v6;v7=v9+v8;return:v7
v6=exp(v0);v8=v6-4;v4=4*v6;v7=v4+v8;return:v7
v6=exp(v0);v8=v6-4;v4=4*v8;v7=v4+v8;return:v7

GVAE
v6=cos(7);v8=exp(9);v2=v8*v0;v9=v2/v6;return:v9
v3=cos(8);v6=exp(9);v6=v8*v0;v9=v2/v6;return:v9
v3=cos(8);v6=2/8;v6=v5*v9;v5=v8v5;return:v5
v3=cos(6);v6=2/9;v6=v5+v5;v5=v1+v6;return:v5
v5=cos(6);v1=2/9;v6=v3+v2;v2=v5-v6;return:v2
v5=sin(5);v3=v1/9;v6=v3-v3;v2=v7-v6;return:v2
v1=sin(1);v5=v5/2;v6=v2-v5;v2=v0-v6;return:v2
v1=sin(1);v7=v8/2;v8=v7/v9;v4=v4-v8;return:v4
v8=sin(1);v2=v8/2;v8=v0/v9;v4=v4-v8;return:v4
v6=exp(v0);v2=v6-4;v8=v0*v1;v7=v4+v8;return:v7
v6=exp(v0);v8=v6-4;v4=4*v8;v7=v4+v8;return:v7

Finding program



Finding program

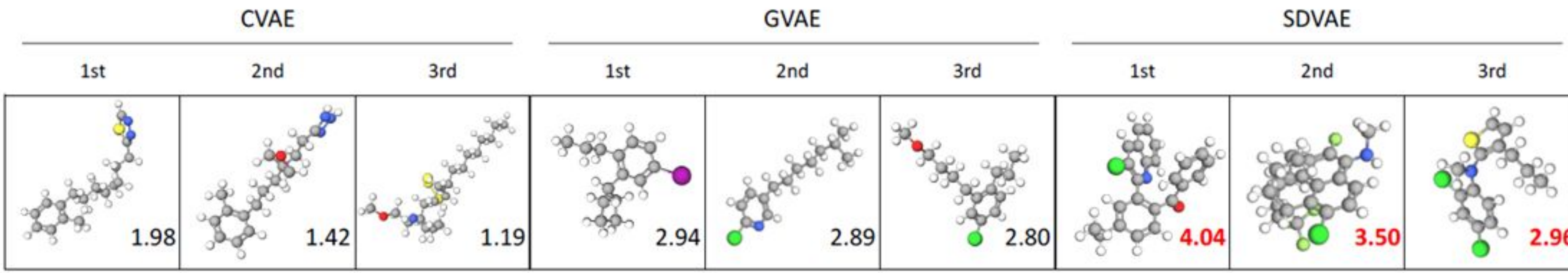
Method	Program	Score
CVAE	`v7=5+v0;v5=cos(v7);return:v5`	0.1742
	`v2=1-v0;v9=cos(v2);return:v9`	0.2889
	`v5=4+v0;v3=cos(v5);return:v3`	0.3043
GVAE	`v3=1/5;v9=-1;v1=v0*v3;return:v3`	0.5454
	`v2=1/5;v9=-1;v7=v2+v2;return:v7`	0.5497
	`v2=1/5;v5=-v2;v9=v5*v5;return:v9`	0.5749
SD-VAE	`v6=sin(v0);v5=exp(3);v4=v0*v6;return:v6`	**0.1206**
	`v5=6+v0;v6=sin(v5);return:v6`	**0.1436**
	`v6=sin(v0);v4=sin(v6);v5=cos(v4);v9=2/v4;return:v4`	**0.1456**
Ground Truth	`v1=sin(v0);v2=exp(v1);v3=v2-1;return:v3`	—

*The distance is measured by log(1+MSE).

Molecules

- Training data: 250,000 SMILES strings randomly selected from ZINC database
- Goal: maximize the water-octanol partition coefficient (logP)
- Consider a penalized logP score that takes into account ring size and synthetic accessibility.

Best molecules by each method



Molecule Reconstruction

- Start with 5000 true molecules from a hold-out set
- Encode each molecule 10 times and decode each encoding 100 times
- 1000 decoded molecules for each of the 5000 input molecules
- Get percentage of molecules reconstructed out of the 5,000,000 attempts.

	Program		Zinc SMILES	
Methods	**Reconstruction %***	**Valid Prior %**	**Reconstruction %**	**Valid Prior %**
SD-VAE	**96.46 (99.90, 99.12, 90.37)**	**100.00**	**76.2**	**43.5**
GVAE	71.83 (96.30, 77.28, 41.90)	2.96	53.7	7.2
CVAE	13.79 (40.46, 0.87, 0.02)	0.02	44.6	0.7

Prior Validity

- Sample 1000 latent points from the prior distribution p(z) = N(0,I)
- Decode each point 500 times
- Test if the decoded SMILES strings correspond to valid molecules.

	Program		Zinc SMILES	
Methods	**Reconstruction %***	**Valid Prior %**	**Reconstruction %**	**Valid Prior %**
SD-VAE	**96.46 (99.90, 99.12, 90.37)**	**100.00**	**76.2**	**43.5**
GVAE	71.83 (96.30, 77.28, 41.90)	2.96	53.7	7.2
CVAE	13.79 (40.46, 0.87, 0.02)	0.02	44.6	0.7

Predictive performance

Objective	Method	Expressions
LL	GVAE	**-1.320±0.001**
	CVAE	-1.397±0.003
RMSE	GVAE	**0.884 ±0.002**
	CVAE	0.975±0.004

	Program		Zinc	
Method	LL	RMSE	LL	RMSE
CVAE	-4.943 ± 0.058	3.757 ± 0.026	-1.812 ± 0.004	1.504 ± 0.006
GVAE	-4.140 ± 0.038	3.378 ± 0.020	-1.739 ± 0.004	1.404 ± 0.006
SD-VAE	**-3.754 ± 0.045**	**3.185 ± 0.025**	**-1.697 ± 0.015**	**1.366 ± 0.023**

Thanks!

Any questions?