When GNNs Met a Word Equations Solver: Learning to Rank Equations

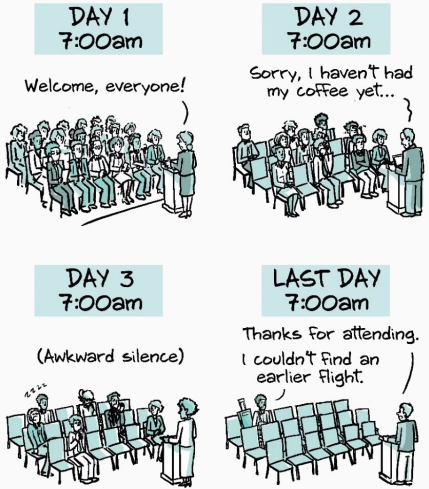
FroCoS 2025

Parosh Aziz Abdulla, Mohamed Faouzi Atig, Julie Cailler,
Chencheng Liang and Philipp Rümmer

VeriDis Team
University of Lorraine
CNRS, Inria, LORIA

October 1, 2025

THE CONFERENCE MORNING SESSION



Word Equation

Elements

- Empty string: ϵ
- Letters: a, b, ...
- Variables: X, Y, Z, ...

Word Equation

Elements

- Empty string: ϵ
- Letters: a, b, ...
- Variables: X, Y, Z, ...

$$Xabb = YaZ$$

Word Equation

Elements

- Empty string: ϵ
- Letters: a, b, ...
- Variables: X, Y, Z, ...

Solving a Word Equation

- SAT
- UNSAT

$$Xabb = YaZ$$

Word Equation

Elements

- Empty string: ϵ
- Letters: a, b, ...
- Variables: X, Y, Z, ...

Solving a Word Equation

- SAT
- UNSAT

$$Xabb = YaZ$$

$$X = a,\ Y = a,\ Z = bb$$

Word Equation

Elements

- Empty string: ϵ
- Letters: a, b, ...
- Variables: X, Y, Z, ...

Solving a Word Equation

- SAT
- UNSAT

$$aabb = aabb$$

$$X = a,\ Y = a,\ Z = bb$$

Together as One!

$\psi = \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n$ with φ_i a word equation

Together as One!

$$\psi = \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n \text{ with } \varphi_i \text{ a word equation}$$

ψ is

- SAT iff there exists one substitution that satisfies all the $\varphi_i \in \psi$
- UNSAT otherwise

Together as One!

$$\psi = \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n \text{ with } \varphi_i \text{ a word equation}$$

ψ is

- SAT iff there exists one substitution that satisfies all the $\varphi_i \in \psi$
- UNSAT otherwise

$$Xabb = YaZ \wedge XabbY = aXZa$$

Together as One!

$$\psi = \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n \text{ with } \varphi_i \text{ a word equation}$$

ψ is

- SAT iff there exists one substitution that satisfies all the $\varphi_i \in \psi$
- UNSAT otherwise

$$Xabb = YaZ \wedge XabbY = aXZa$$

$$X = a,\ Y = a,\ Z = bb$$

Together as One!

$$\psi = \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n \text{ with } \varphi_i \text{ a word equation}$$

ψ is

- SAT iff there exists one substitution that satisfies all the $\varphi_i \in \psi$
- UNSAT otherwise

$$aabb = aabb \wedge aabba = aabba$$

$$X = a,\ Y = a,\ Z = bb$$

Easy, right...?

$$AaAkjhhohhhkokookkkokkookkhkoohohCAYZjhBkjhkkkokhkkEkkoWQB$$
$$= aABBBncYECaa$$

- Variables: A, C, Y, Z, B, E, W, Q
- Letters: a, k, j, h, o, n, c

Easy, right...?

$$AaAkjhhohhhkokookkkokkookkhkoohohCAYZjhBkjhkkkokhkkEkkoWQB = aABBBncYECaa$$

- Variables: A, C, Y, Z, B, E, W, Q
- Letters: a, k, j, h, o, n, c

`Z3`, `cvc5`, `Ostrich`, `Z3-noodler`, and `Woorpje` cannot solve it in 300 seconds

Challenges

- Difficult to solve
  - ▷ Decidable
  - ▷ Complexity: NP-Hard and in PSPACE
  - ▷ Practical implementation are incomplete
- Important in modeling string constraints in verification tasks
  - ▷ User inputs, string manipulations, ...

Challenges

- Difficult to solve
  - ▷ Decidable
  - ▷ Complexity: NP-Hard and in PSPACE
  - ▷ Practical implementation are incomplete
- Important in modeling string constraints in verification tasks
  - ▷ User inputs, string manipulations, ...

Can machine learning help to solve conjunctive word equation problem?

Yet Another Talk about ML...

- Is it another “we used ML for [insert problem] and...”?

Yet Another Talk about ML...

- Is it another “we used ML for [insert problem] and...”?
  - ▷ Yes, but not only! (We also used our brains!)

Yet Another Talk about ML...

- Is it another “we used ML for [insert problem] and...”?
  - ▷ Yes, but not only! (We also used our brains!)
- Does it work, at least?

Yet Another Talk about ML...

- Is it another “we used ML for [insert problem] and...”?
  - ▷ Yes, but not only! (We also used our brains!)
- Does it work, at least?
  - ▷ Enough to get accepted at FroCoS

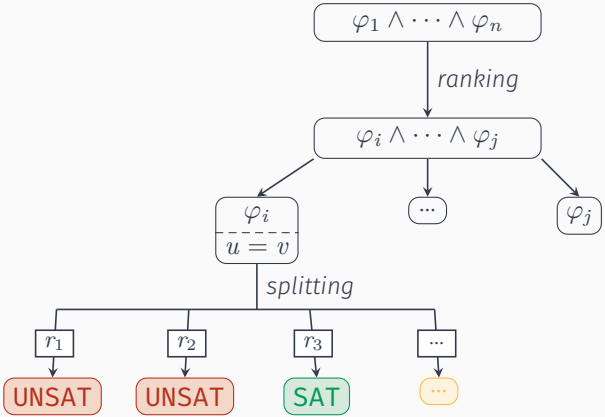
Yet Another Talk about ML...

- Is it another “we used ML for [insert problem] and...”?
  - ▷ Yes, but not only! (We also used our brains!)
- Does it work, at least?
  - ▷ Enough to get accepted at FroCoS
- Is human intelligence over now?

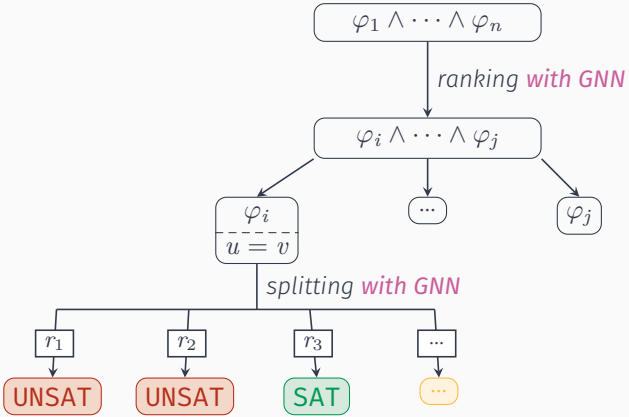
Yet Another Talk about ML...

- Is it another “we used ML for [insert problem] and...”?
  - ▷ Yes, but not only! (We also used our brains!)
- Does it work, at least?
  - ▷ Enough to get accepted at FroCoS
- Is human intelligence over now?
  - ▷ ~~It depends on the human~~
  - ▷ Obviously not!

Big Picture

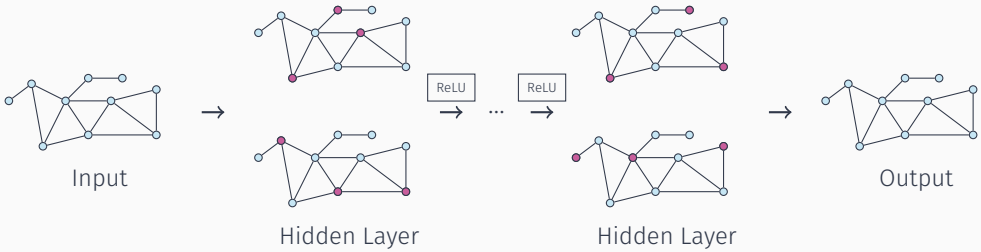

$\varphi_1 \wedge \cdots \wedge \varphi_n$
ranking
$\varphi_i \wedge \cdots \wedge \varphi_j$
φ_i
$u = v$
...
φ_j
splitting
r_1
r_2
r_3
...
UNSAT
UNSAT
SAT
...

Big Picture

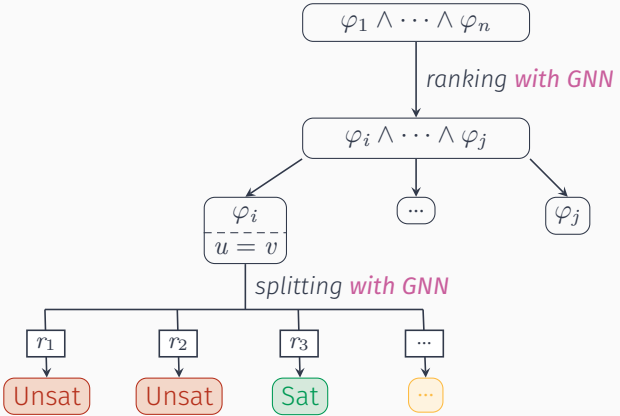

$\varphi_1 \wedge \cdots \wedge \varphi_n$
ranking with GNN
$\varphi_i \wedge \cdots \wedge \varphi_j$
φ_i
$u = v$
...
φ_j
splitting with GNN
r_1
r_2
r_3
...
UNSAT
UNSAT
SAT
...

Graph Neural Networks (GNN)

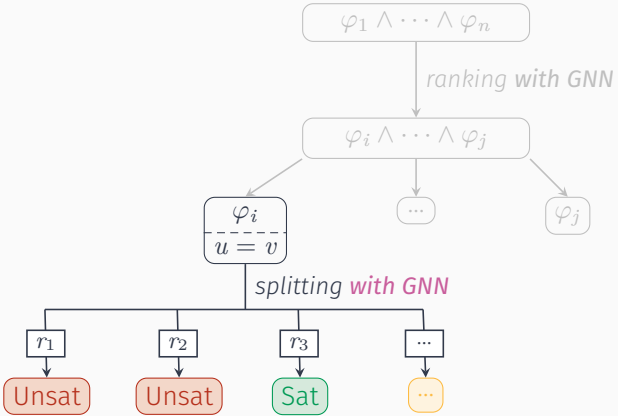
- Designed to work directly with graph-structured data
- Take graph as input, output node and graph representations
- Capture the structural features of graph



Big Picture


$\varphi_1 \wedge \cdots \wedge \varphi_n$
ranking with GNN
$\varphi_i \wedge \cdots \wedge \varphi_j$
φ_i
$u = v$
...
φ_j
splitting with GNN
r_1
r_2
r_3
...
Unsat
Unsat
Sat
...

Big Picture


$\varphi_1 \wedge \cdots \wedge \varphi_n$
ranking with GNN
$\varphi_i \wedge \cdots \wedge \varphi_j$
φ_i
$u = v$
...
φ_j
splitting with GNN
r_1
r_2
r_3
...
Unsat
Unsat
Sat
...

Previously...[1]

Calculus

- Nielsen transformation
- Splitting rules, iterative deepening & backtracking
- Continuously simplify the first terms of both sides
- `DragonLi`

$$Xabb = YaZ$$

where a is a letter, u and v are terms, and X and Y are variables.

[1] Abdulla, Atig, Cailler, Liang and Rüemmer. *Guiding word equation solving using graph neural networks.* International Symposium on Automated Technology for Verification and Analysis (ATVA 2024).

Previously...[1]

Calculus

- Nielsen transformation
- Splitting rules, iterative deepening & backtracking
- Continuously simplify the first terms of both sides
- `DragonLi`

$$Xabb = YaZ$$
$$\downarrow r_1 \quad \downarrow r_2 \quad \downarrow r_3$$

where a is a letter, u and v are terms, and X and Y are variables.

[1] Abdulla, Atig, Cailler, Liang and Rüemmer. *Guiding word equation solving using graph neural networks.* International Symposium on Automated Technology for Verification and Analysis (ATVA 2024).

Previously...[1]

Calculus

- Nielsen transformation
- Splitting rules, iterative deepening & backtracking
- Continuously simplify the first terms of both sides
- `DragonLi`

$$Xabb = YaZ$$
$$\downarrow r_2\ X \mapsto Y$$

where a is a letter, u and v are terms, and X and Y are variables.

[1] Abdulla, Atig, Cailler, Liang and Rüemmer. *Guiding word equation solving using graph neural networks.* International Symposium on Automated Technology for Verification and Analysis (ATVA 2024).

Previously...[1]

Calculus

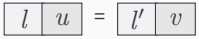
- Nielsen transformation
- Splitting rules, iterative deepening & backtracking
- Continuously simplify the first terms of both sides
- `DragonLi`

$$Xabb = YaZ$$
$$\downarrow r_2\ X \mapsto Y$$
$$abb = aZ$$

where a is a letter, u and v are terms, and X and Y are variables.

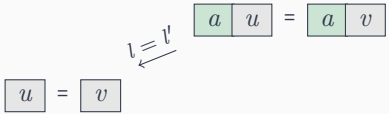
[1] Abdulla, Atig, Cailler, Liang and Rüemmer. *Guiding word equation solving using graph neural networks.* International Symposium on Automated Technology for Verification and Analysis (ATVA 2024).

Rules



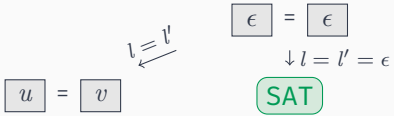
where l, l', a and b are a letters, u and v are terms, X and Y are variables, and X' and Y' are fresh variables.

Rules



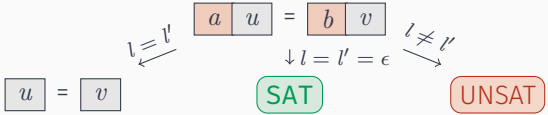
where l, l', a and b are a letters, u and v are terms, X and Y are variables, and X' and Y' are fresh variables.

Rules



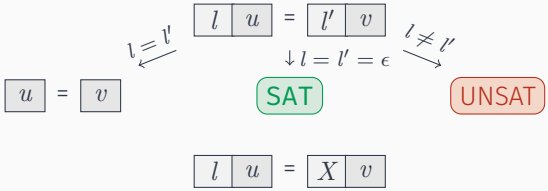
where l, l', a and b are a letters, u and v are terms, X and Y are variables, and X' and Y' are fresh variables.

Rules



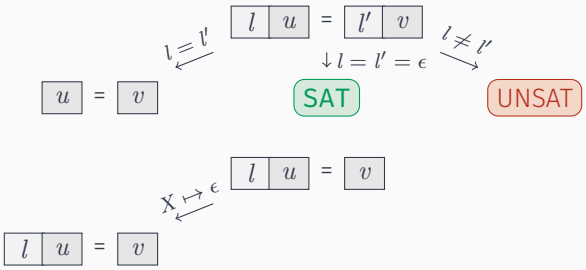
where l, l', a and b are a letters, u and v are terms, X and Y are variables, and X' and Y' are fresh variables.

Rules



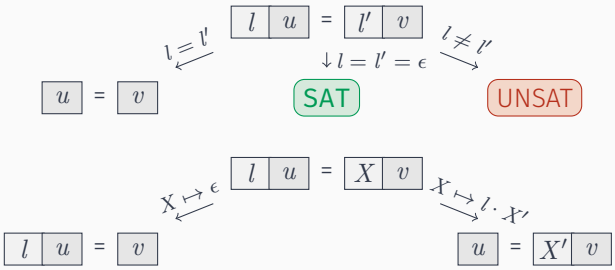
where l, l', a and b are a letters, u and v are terms, X and Y are variables, and X' and Y' are fresh variables.

Rules



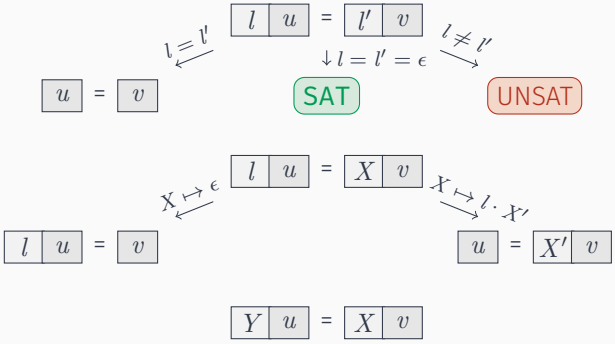
where l, l', a and b are a letters, u and v are terms, X and Y are variables, and X' and Y' are fresh variables.

Rules



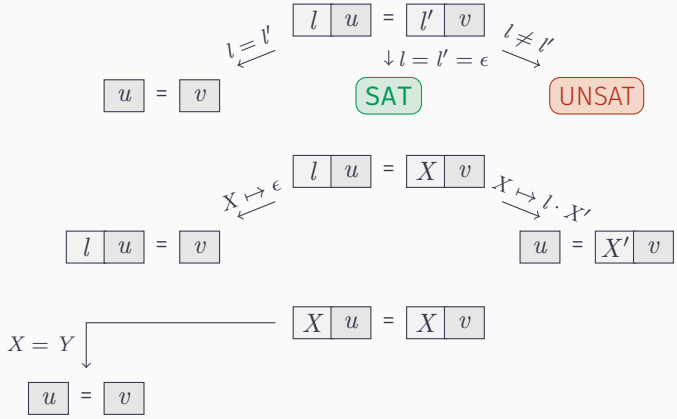
where l, l', a and b are a letters, u and v are terms, X and Y are variables, and X' and Y' are fresh variables.

Rules



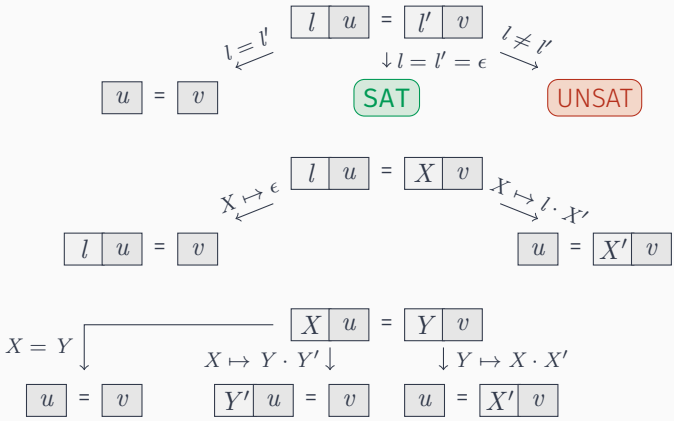
where l, l', a and b are a letters, u and v are terms, X and Y are variables, and X' and Y' are fresh variables.

Rules



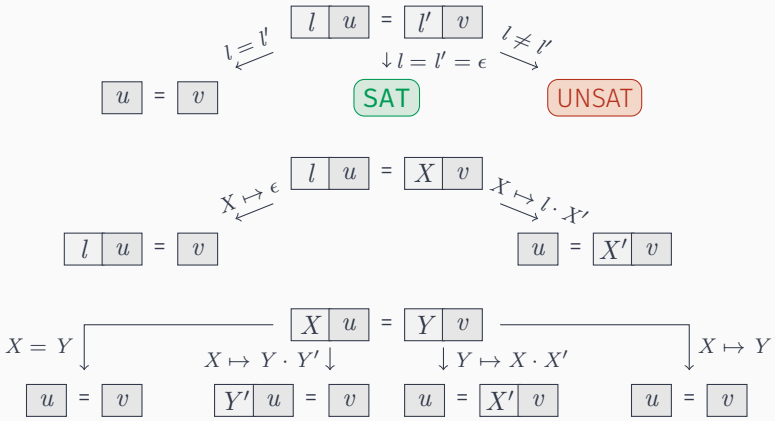
where l, l', a and b are a letters, u and v are terms, X and Y are variables, and X' and Y' are fresh variables.

Rules



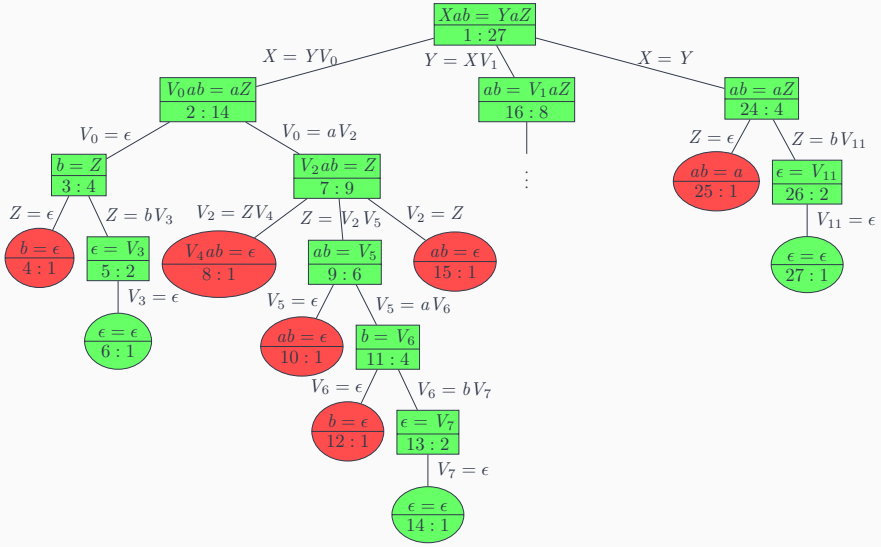
where l, l', a and b are a letters, u and v are terms, X and Y are variables, and X' and Y' are fresh variables.

Rules



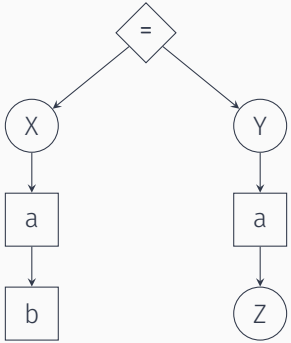
where l, l', a and b are a letters, u and v are terms, X and Y are variables, and X' and Y' are fresh variables.

Here Is a Proof...



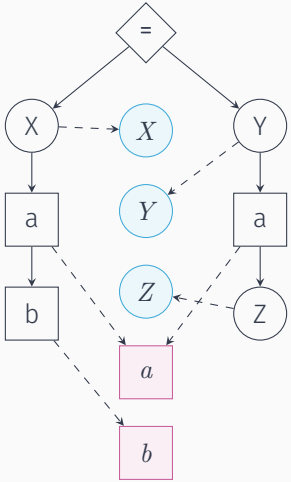
...and Here Is an Equation!

$$Xab = YaZ$$

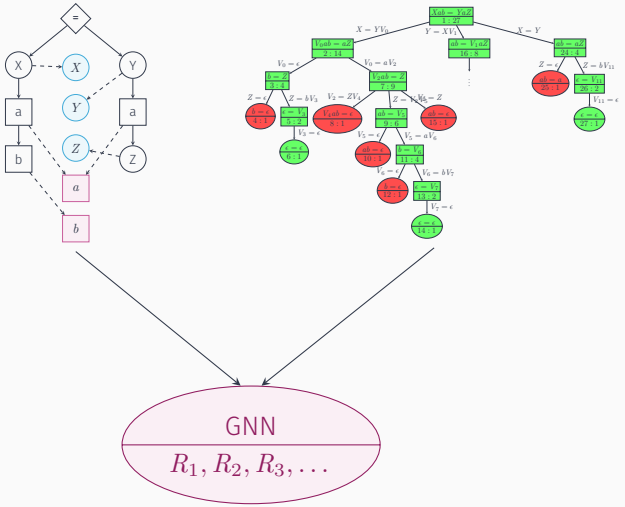


...and Here Is an Equation!

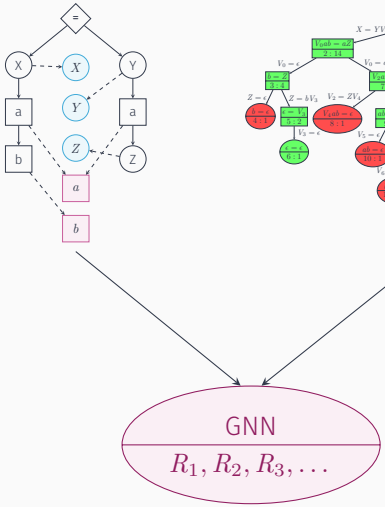
$$Xab = YaZ$$

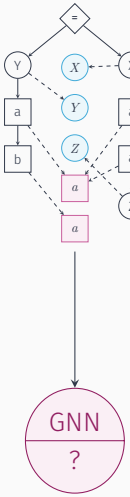


Better Together!



Better Together!





DragonLi, I Choose You!

- Python
- Focus on SAT
- Results comparable to state-of-the-art solvers

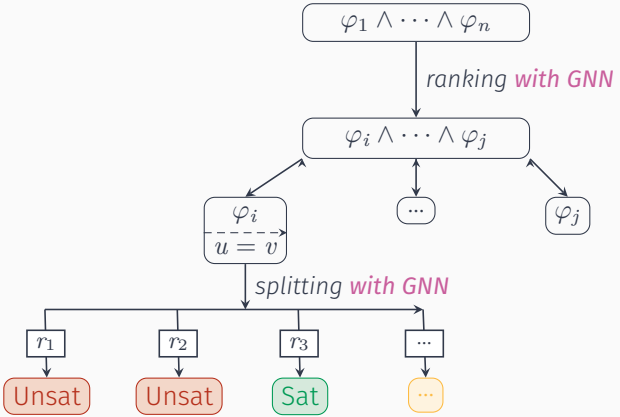
DragonLi, I Choose You!

- Python
- Focus on SAT
- Results comparable to state-of-the-art solvers

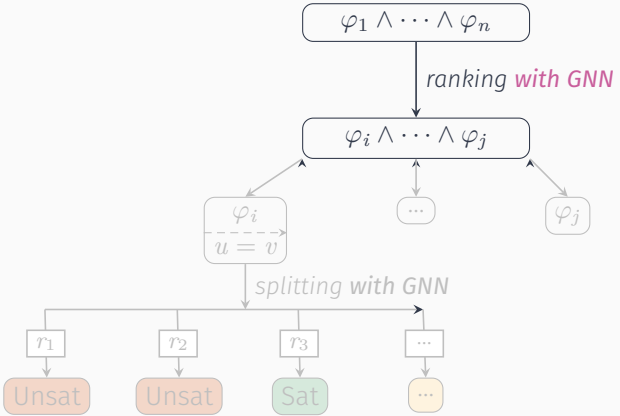
How to deal with more than one equation?

How to improve on UNSAT problems?

Big Picture


$\varphi_1 \wedge \cdots \wedge \varphi_n$
ranking with GNN
$\varphi_i \wedge \cdots \wedge \varphi_j$
φ_i
$u = v$
...
φ_j
splitting with GNN
r_1
r_2
r_3
...
Unsat
Unsat
Sat
...

Big Picture


$\varphi_1 \wedge \cdots \wedge \varphi_n$
ranking with GNN
$\varphi_i \wedge \cdots \wedge \varphi_j$
φ_i
$u = v$
...
φ_j
splitting with GNN
r_1
r_2
r_3
...
Unsat
Unsat
Sat
...

Order Matters

$$AaAkjhhooohohCAYZjhBkjhkkkokhkkEk = aABBBncYECaa \land a = b$$

Order Matters

$$AaAkjhhooohohCAYZjhBkjhkkkokhkkEk = aABBBncYECaa \wedge a = b$$

- Left first: `Timeout`
- Right first: `UNSAT`

Order Matters

$$AaAkjhhooohohCAYZjhBkjhkkkokhkkEk = aABBBncYECaa \wedge a = b$$

- Left first: `Timeout`
- Right first: `UNSAT`
- Basic heuristic (+ length):
  1. $\epsilon = \epsilon$
  2. $\epsilon = u \cdot v$ or $u \cdot v = \epsilon$
  3. $a \cdot u = b \cdot v$ or $u \cdot a = v \cdot b$
  4. $a \cdot u = a \cdot v$
  5. Otherwise

Order Matters

$$AaAkjhhooohohCAYZjhBkjhkkkokhkkEk = aABBBncYECaa \wedge a = b$$

- Left first: `Timeout`
- Right first: `UNSAT`
- Basic heuristic (+ length):
  1. $\epsilon = \epsilon$
  2. $\epsilon = u \cdot v$ or $u \cdot v = \epsilon$
  3. $a \cdot u = b \cdot v$ or $u \cdot a = v \cdot b$
  4. $a \cdot u = a \cdot v$
  5. Otherwise

Can a GNN do better?

Cherry Picking

Minimal Unsatisfiable Set (MUS)

A set of equations U of a conjunctive word equation is a MUS iff the conjunction of U is unsatisfiable, and for all equations $e \in U$, the conjunction of $U \setminus \{e\}$ is satisfiable.

$$a = X \wedge b = Y \wedge aa = Xa \wedge aab = XaY \wedge X = b$$

Cherry Picking

Minimal Unsatisfiable Set (MUS)

A set of equations U of a conjunctive word equation is a MUS iff the conjunction of U is unsatisfiable, and for all equations $e \in U$, the conjunction of $U \setminus \{e\}$ is satisfiable.

$$a = X \wedge b = Y \wedge aa = Xa \wedge aab = XaY \wedge X = b$$

$$\{a = X, X = b\}$$

How to (m)Use Them?

Training Data

- Problems not solved by `DragonLi` are given to `Z3`, `Z3-Noodler`, `cvc5`, `Ostrich` and `Woorpje`
- Extraction of MUS

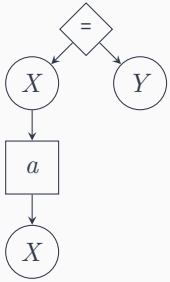
Labelling

For a conjunctive word equation $\phi = e_1 \wedge \cdots \wedge e_n$ and U its MUS:

$$\text{label}(e_i) = \begin{cases} 1 & \text{if } e_i \in U \text{ and } |e_i| = \min\left(\{|e| \mid e \in U\}\right) \\ 0 & \text{otherwise.} \end{cases}$$

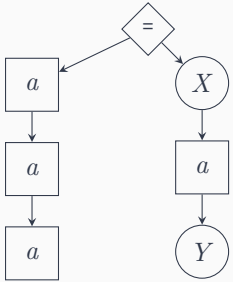
I Wouldn’t Count on It…

$XaX = Y$



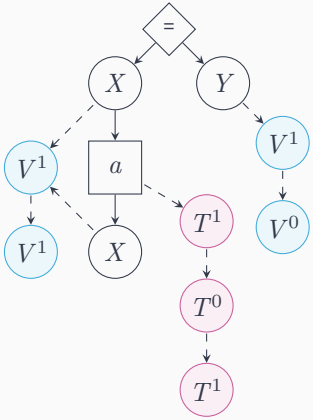
$\wedge$

$aaa = XaY$



I Wouldn't Count on It...

$XaX = Y$



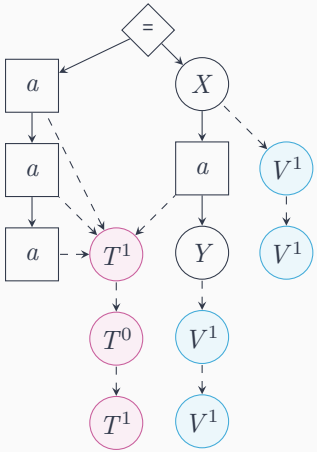
$\wedge$

$Occ(X) = 3$
$Enc(X) = 11$

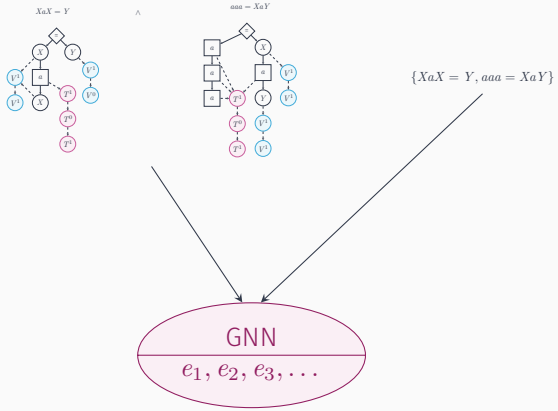
$Occ(Y) = 2$
$Enc(Y) = 10$

$Occ(a) = 5$
$Enc(a) = 101$

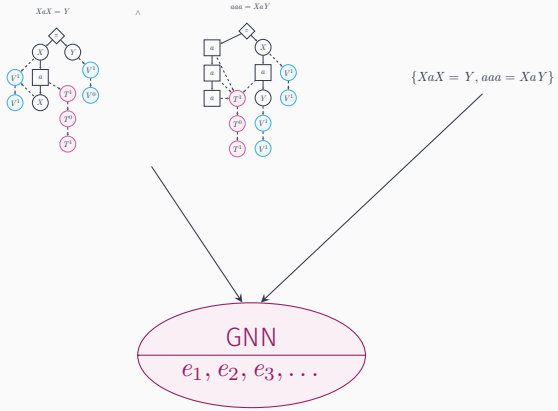
$aaa = XaY$

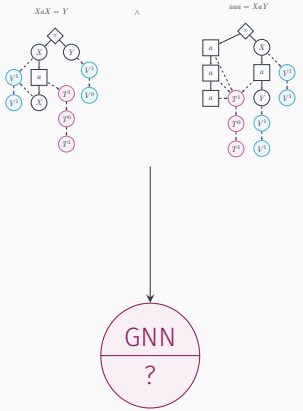


Better Together (bis repetita)



Better Together (bis repetita)





F.A.Q.

- Why a second GNN?

F.A.Q.

- Why a second GNN?
  - ▷ Different task
  - ▷ Varying size of the input
  - ▷ Focus on UNSAT

F.A.Q.

- Why a second GNN?
  - ▷ Different task
  - ▷ Varying size of the input
  - ▷ Focus on UNSAT
- Why this encoding?

F.A.Q.

- Why a second GNN?
  - ▷ Different task
  - ▷ Varying size of the input
  - ▷ Focus on UNSAT
- Why this encoding?
  - ▷ On big graph is inefficient
  - ▷ Unary: increase graph size
  - ▷ Ternary or more: blur structural distinction

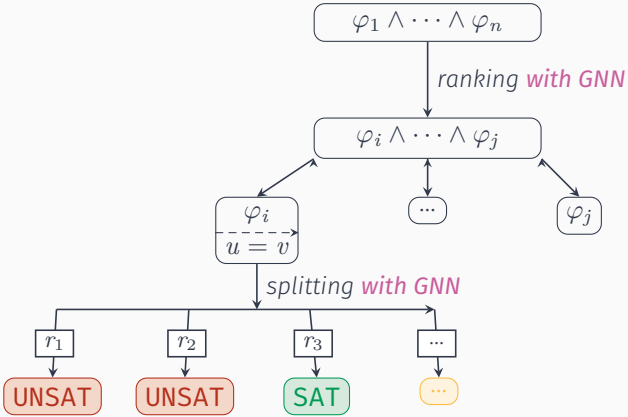
F.A.Q.

- Why a second GNN?
  - ▷ Different task
  - ▷ Varying size of the input
  - ▷ Focus on UNSAT
- Why this encoding?
  - ▷ On big graph is inefficient
  - ▷ Unary: increase graph size
  - ▷ Ternary or more: blur structural distinction
- When do you call the GNN?

F.A.Q.

- Why a second GNN?
  - ▷ Different task
  - ▷ Varying size of the input
  - ▷ Focus on UNSAT
- Why this encoding?
  - ▷ On big graph is inefficient
  - ▷ Unary: increase graph size
  - ▷ Ternary or more: blur structural distinction
- When do you call the GNN?
  - ▷ Once at the beginning with 5 equation of priority level 5
  - ▷ Limit overhead

Big Picture


$\varphi_1 \wedge \cdots \wedge \varphi_n$
ranking with GNN
$\varphi_i \wedge \cdots \wedge \varphi_j$
φ_i
$u = v$
...
φ_j
splitting with GNN
r_1
r_2
r_3
...
UNSAT
UNSAT
SAT
...

I Need Problems...

- Real-world benchmarks:
  - SMT-LIB and Zaligvinder
  - Remove length constraint, boolean operator, and regex
  - Too easy

I Need Problems...

- Real-world benchmarks:
  - ▷ SMT-LIB and Zaligvinder
  - ▷ Remove length constraint, boolean operator, and regex
  - ▷ Too easy
- Artificial benchmarks:
  - ▷ $s = s$
  - ▷ At most 60 letters (among a set T)
  - ▷ Replace n times substrings on both side by m variables
  - ▷ Between 0 and 100 equations

Linearity and Benchmarks

Linearity

A conjunctive word equation is called *linear* iff each variable appear **at most once** in each equation.

$X = a \wedge Y = b$ ✅ $X = a \wedge X = b$ ✅ $XaX = aaa \wedge YaX = YbZ$ ❌

Linearity and Benchmarks

Linearity

A conjunctive word equation is called *linear* iff each variable appear **at most once** in each equation.

$X = a \wedge Y = b$ ✅ $X = a \wedge X = b$ ✅ $XaX = aaa \wedge YaX = YbZ$ ❌

- **A1** : fresh variable, $T = 6$, $n \in [0, 5]$, $m \in [0, 5]$ (*linear*)
- **A2** : A1 + $T = 26$, $n \in [0, 16]$, and $m = 1$ (*linear*)
- **B** : A1 + No fresh variable (*non-linear*)
- **C** : $X_n a X_n b X_{n-1} \cdots b X_1 = a X_n X_{n-1} X_{n-1} b \cdots X_1 X_1 baa$ + $b \mapsto A_1$ (*highly non-linear*)

Experimental Setup

- `DragonLi`: basic, random and GNN-based
- `Z3`, `Z3-Noodler`, `cvc5`, `Ostrich`, and `Woopje`
- 300 seconds timeout
- 2 Intel Xeon E5 2630 v4 at 2.20 GHz/core and 128GB memory
- GNNs trained on NVIDIA A100 GPUs
- 60 000 problems per benchmark for training
- 1000 problems per benchmark for experiments

Results — A1

Solver	Number of solved problems					Average solving time (split number)			
	SAT	UNSAT	UNI	CS	CU	SAT	UNSAT	CS	CU
DragonLi	**24**	955	0	13	678	5.6 (244.8)	6.5 (1085.3)	5.0 (94.4)	5.7 (126.3)
Random-DragonLi	22	944	0			5.6 (198.8)	6.3 (932.6)	5.6 (137.6)	5.7 (180.5)
GNN-DragonLi	**24**	**961**	0			6.1 (164.7)	7.5 (1974.8)	6.1 (96.4)	6.3 (**60.5**)
cvc5	**24**	952	1			0.5	0.6	0.1	0.3
Z3	17	960	0			8.7	0.4	1.1	0.1
Z3-Noodler	22	939	2			5.7	0.3	4.8	0.1
Ostrich	17	931	0			15.0	5.5	8.0	4.7
Woorpje	23	744	0			3.0	12.5	0.1	12.2

Results — A2

Solver	Number of solved problems					Average solving time (split number)			
	SAT	UNSAT	UNI	CS	CU	SAT	UNSAT	CS	CU
DragonLi	59	824	0	3	0	8.5 (4233.4)	11.8 (1231.3)	4.7 (27.3)	-
Random-DragonLi	44	806	1			24.7 (29779.6)	6.2 (210.9)	4.6 (27.3)	-
GNN-DragonLi	59	836	4			8.4 (1330.6)	11.6 (1074.1)	5.9 (27.3)	-
cvc5	**67**	142	15			0.6	56.0	0.1	-
Z3	8	**870**	10			1.1	0.6	0.1	-
Z3-Noodler	22	7	1			15.4	3.8	0.4	-
Ostrich	13	18	2			24.8	38.8	8.6	-
Woorpje	0	0	0	-	-	-	-	-	-

Results — B

Solver	Number of solved problems					Average solving time (split number)			
	SAT	UNSAT	UNI	CS	CU	SAT	UNSAT	CS	CU
DragonLi	11	805	0	4	294	4.9 (62.5)	5.2 (81.5)	4.9 (29.2)	5.3 (82.4)
Random-DragonLi	10	894	0			5.0 (58.7)	5.8 (295.2)	5.0 (27.25)	5.2 (73.1)
GNN-DragonLi	11	821	0			6.5 (65.1)	6.8 (70.0)	6.5 (28.25)	6.8 (**60.2**)
cvc5	12	915	0			0.1	0.6	0.1	0.7
Z3	11	859	3			0.1	0.2	0.1	0.1
Z3-Noodler	**24**	911	1			4.9	0.4	1.3	0.4
Ostrich	12	**917**	2			6.9	3.7	3.3	4.2
Woorpje	19	330	1			29.5	6.0	0.2	5.0
	3	62	0	1		65.0	28.4	0.2	223.1

Results — C

Solver	Number of solved problems					Average solving time (split number)			
	SAT	UNSAT	UNI	CS	CU	SAT	UNSAT	CS	CU
DragonLi	2	0	0	-	-	5.1 (85.5)	-	-	-
Random-DragonLi	2	0	0	-	-	5.0 (85.5)	-	-	-
GNN-DragonLi	-	-	-	-	-	-	-	-	-
cvc5	0	**909**	17	-	1	-	46.9	-	17.3
Z3	1	821	12	1		0.8	1.7	0.8	0.1
Z3-Noodler	**7**	657	4	1		0.2	94.1	0.1	1.0
Ostrich	0	61	0	-		-	77.2	-	27.1
Woorpje	3	62	0	1		65.0	28.4	0.2	223.1

“Why Is Your Work Good?”

- `DragonLi` performs well on linear problems
- Fewer splits
- No improvement on non-linear problems
- Need for dedicated strategies (equation length, letter count, …)
- Re-implementation can offer improvements
- Increasing training dataset increases the number of problems solved

This is the End!

Contributions

- Extension of `DragonLi` to handle conjunctive word equations
- Multi-classification task to handle variable number of inputs
- MUS & new graph representation
- Good results on linear problems

What's Next?

- Optimize GNN overhead
- Call timing
- Length constraint and regular expressions

Thank you! ☺

https://github.com/ChenchengLiang/DragonLi