Guiding Word Equation Solving using Graph Neural Networks

Parosh Aziz Abdulla,[1] Mohamed Faouzi Atig,[1]
Julie Cailler,[2,3] Chencheng Liang,[1] Philipp Rümmer[1,2]

[1]Uppsala University, Sweden
[2]University of Regensburg, Germany
[3]University of Lorraine & Inria, France

October 24, ATVA 2024

Word Equation (Example)

Word equation: $Xab = YaZ$

$X, Y,$ and Z are variables

a, b are letters

Solving a Word Equation (Example)

Word equation: $Xab = YaZ$

$X, Y,$ and Z are variables

a, b are letters

- Find assignments to variables that make the equality hold true (SAT)
- Prove there is no assignment to make the equality hold true (UNSAT)

Solving a Word Equation (Example)

Word equation: $Xab = YaZ$

$X, Y,$ and Z are variables

a, b are letters

$X = a, Y = a, Z = b$ satisfies the equation$(aab = aab)$

Word Equation Problem

- Important in modeling string constraints in verification tasks
  - E.g., Validate user inputs, ensuring correct string manipulations
- Difficult to solve
  - Decision procedures such as [1] and [2] has no implementation
  - Practical algorithms are incomplete

[1] Makanin, G.S.: The problem of solvability of equations in a free semigroup. Math. Sb. (N.S.) 103(145)(2(6)), 147–236 (1977)
[2] Plandowski, W.: Satisfiability of word equations with constants is in pspace. In: 40th Annual Symposium on Foundations of Computer Science, 495–500 (1999).

Word Equation Problem

$$AaAkjhhohhhkokookkkokkookkhkoohohCAYZjhBkjhkkkokhkkEkkoWQB = aABBBncYECaa$$

Variables: A, C, Y, Z, B, E, W, Q

Letters: a, k, j, h, o, n, c

- Difficult to solve (example)
  - Z3, cvc5, Ostrich, Z3-noodler, and Woorpje cannot solve it in 300 seconds

Contributions

- Define a set of calculus rules for word equation extended from [3]
- A framework applying the calculus and guided by Graph Neural Networks (GNNs)
- The experimental results show that the framework is competitive to leading string solvers for SAT problems

[3] Abdulla, P.A., Atig, M.F., Chen, Y.F., Holık, L., Rezine, A., Rummer, P., Stenman, J.: String constraints for verification. Computer Aided Verification, 150–166 (2014)

Calculus

- Based on the Nielsen transformation [4]
- Continuously simplify the first terms of both sides

[4] Nielsen, J.: Die Isomorphismen der allgemeinen, unendlichen Gruppe mit zwei Erzeugenden. Mathematische Annalen 78, 385–397 (1917).

Calculus (Example)

- Based on the Nielsen transformation [4]
- Continuously simplify the first terms of both sides

$$X \cdot u = a \cdot v$$

$$X = aY$$

$$Y \cdot u\,[X/aY] = v\,[X/aY]$$

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

Calculus

- May not terminate

$$R_1\ \frac{true}{\text{SAT}} \qquad R_2\ \frac{\epsilon = \epsilon \wedge \phi}{\phi} \qquad R_3\ \frac{X = \epsilon \wedge \phi}{\begin{array}{c}[X \mapsto \epsilon] \\ \phi\end{array}} \qquad R_4\ \frac{a \cdot u = \epsilon \wedge \phi}{\text{UNSAT}}$$

with $X \in \Gamma$ and $a \in \Sigma$.

(a) Simplification rules

$$R_5\ \frac{a \cdot u = a \cdot v \wedge \phi}{u = v \wedge \phi} \qquad R_6\ \frac{a \cdot u = b \cdot v \wedge \phi}{\text{UNSAT}}$$

with a, b two different letters from Σ.

(b) Letter-letter rules

$$R_7\ \frac{X \cdot u = a \cdot v \wedge \phi}{\begin{array}{c|c}[X \mapsto \epsilon] & [X \mapsto a \cdot X'] \\ u = a \cdot v \wedge \phi & X' \cdot u = v \wedge \phi\end{array}}$$

with X' a *fresh* element of Γ.

(c) Variable-letter rules

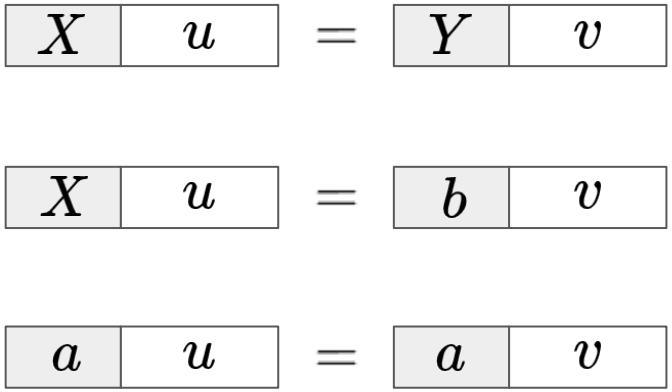
$$R_8\ \frac{X \cdot u = Y \cdot v \wedge \phi}{\begin{array}{c|c|c}[X \mapsto Y] & [X \mapsto Y \cdot Y'] & [Y \mapsto X \cdot X'] \\ u = v \wedge \phi & Y' \cdot u = v \wedge \phi & u = X' \cdot v \wedge \phi\end{array}}$$

with X', Y' *fresh* elements of Γ.

(d) Variable-variable rule

Calculus (Example)

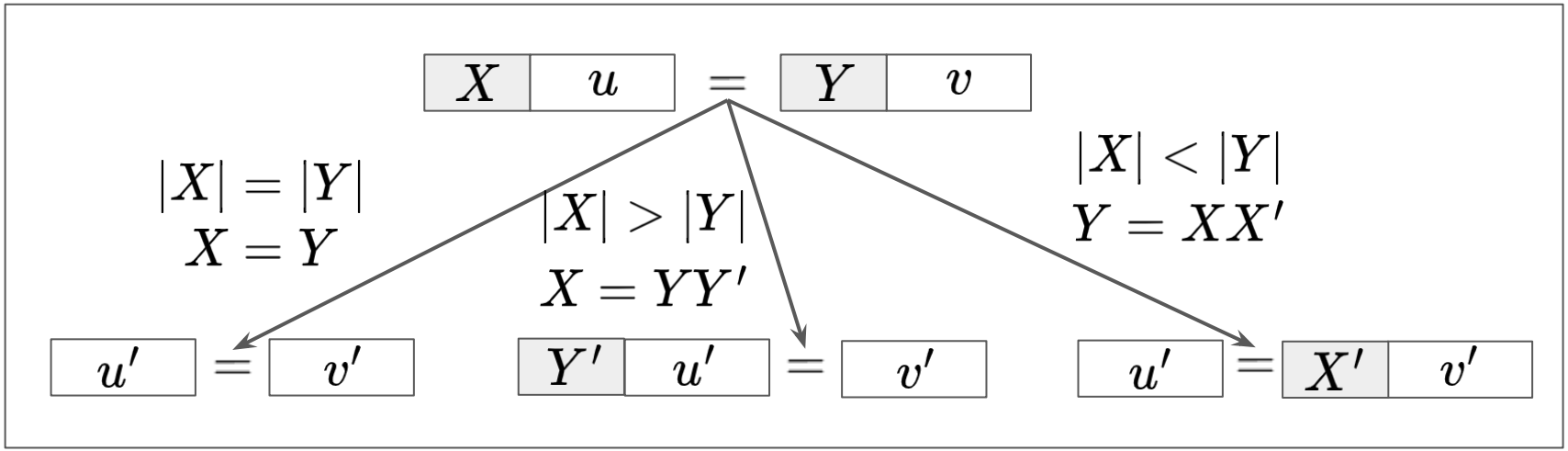
- The idea of the simplification
  - Variable - Variable
  - Variable - Letter
  - Letter - Letter



a, b are letters, u, v are terms, and X, Y are variables.

Calculus (Example)

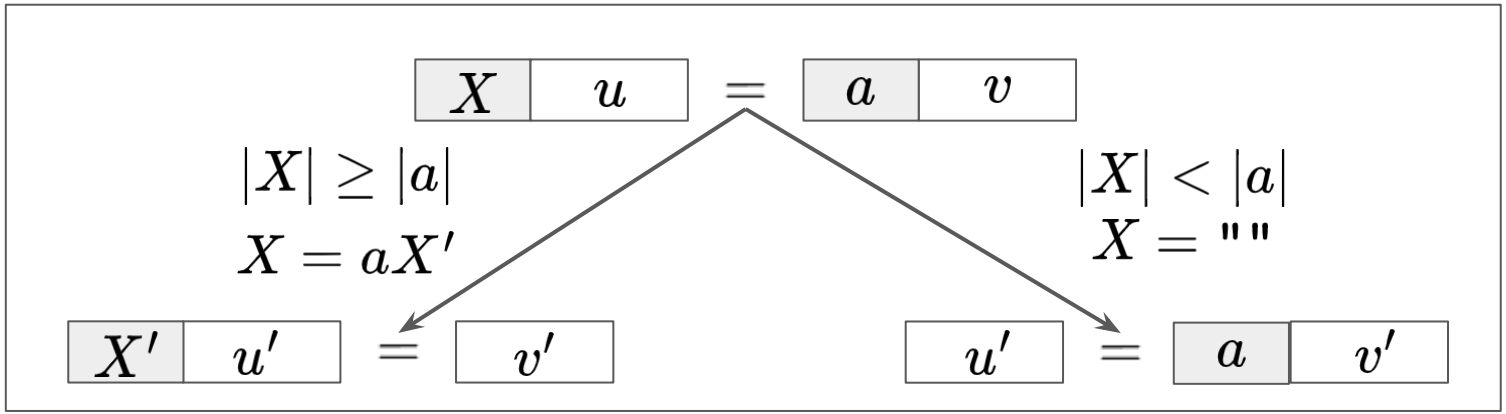
- First terms are Variable - Variable
  - Intuition: length



u, v, u', v' are terms, and X, Y, X', Y' are variables

Calculus (Example)

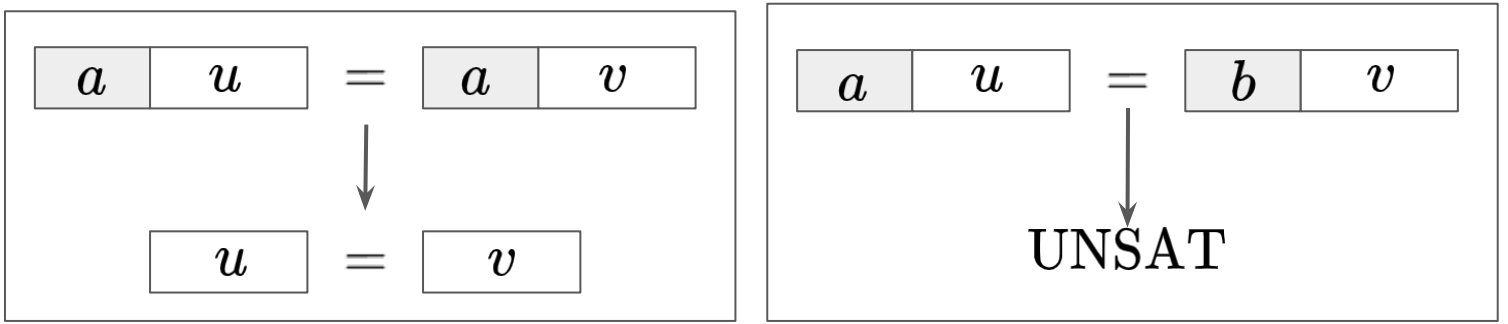
- First terms are Variable - Terminal
  - Intuition: length



u, v, u', v' are terms, a is a letter, and X, X' are variables

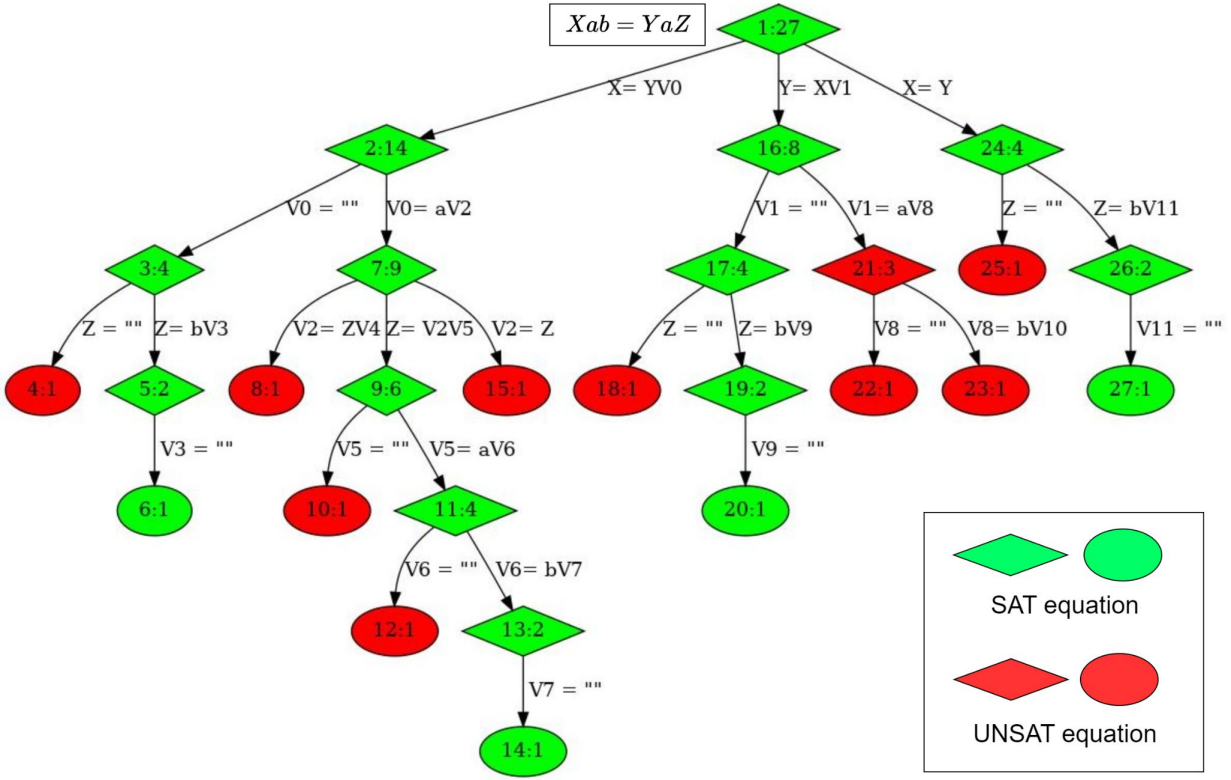
Calculus (Example)

- First terms are Letter- Letter

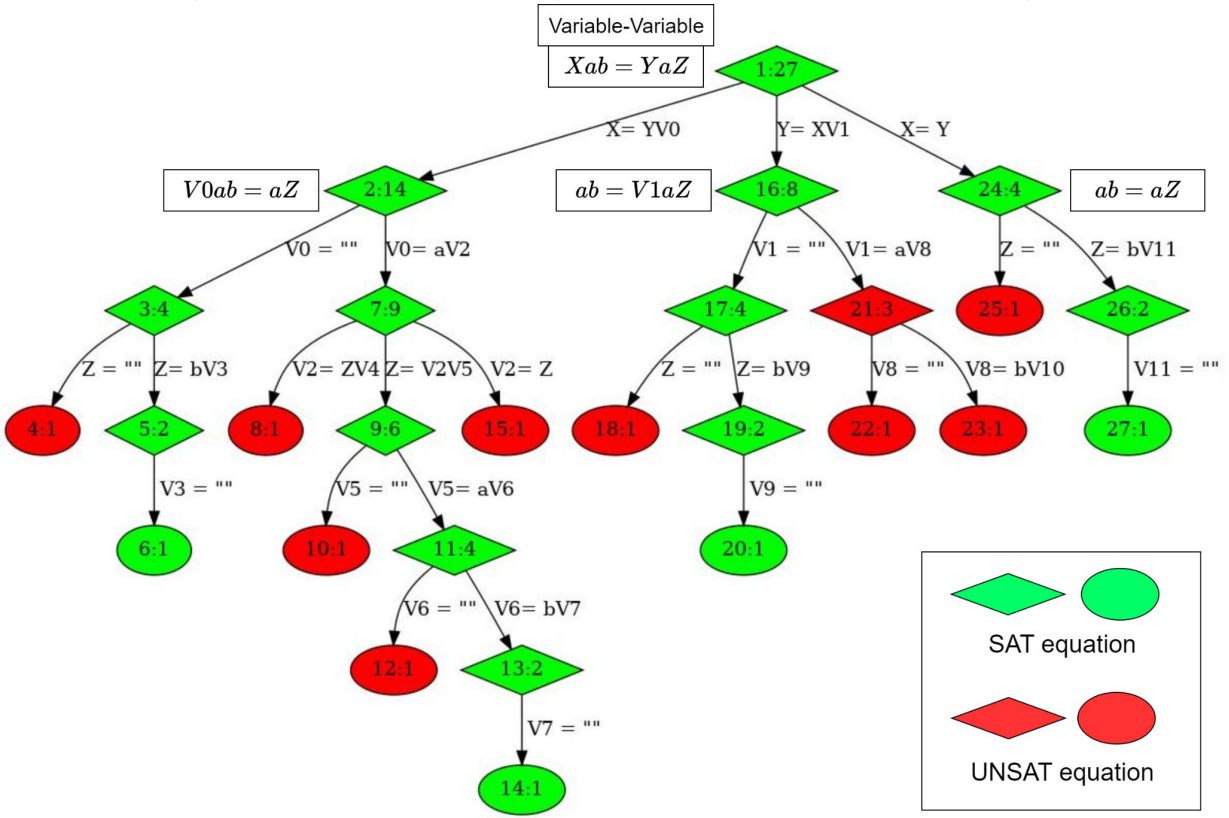


u, v are terms and a, b are letters

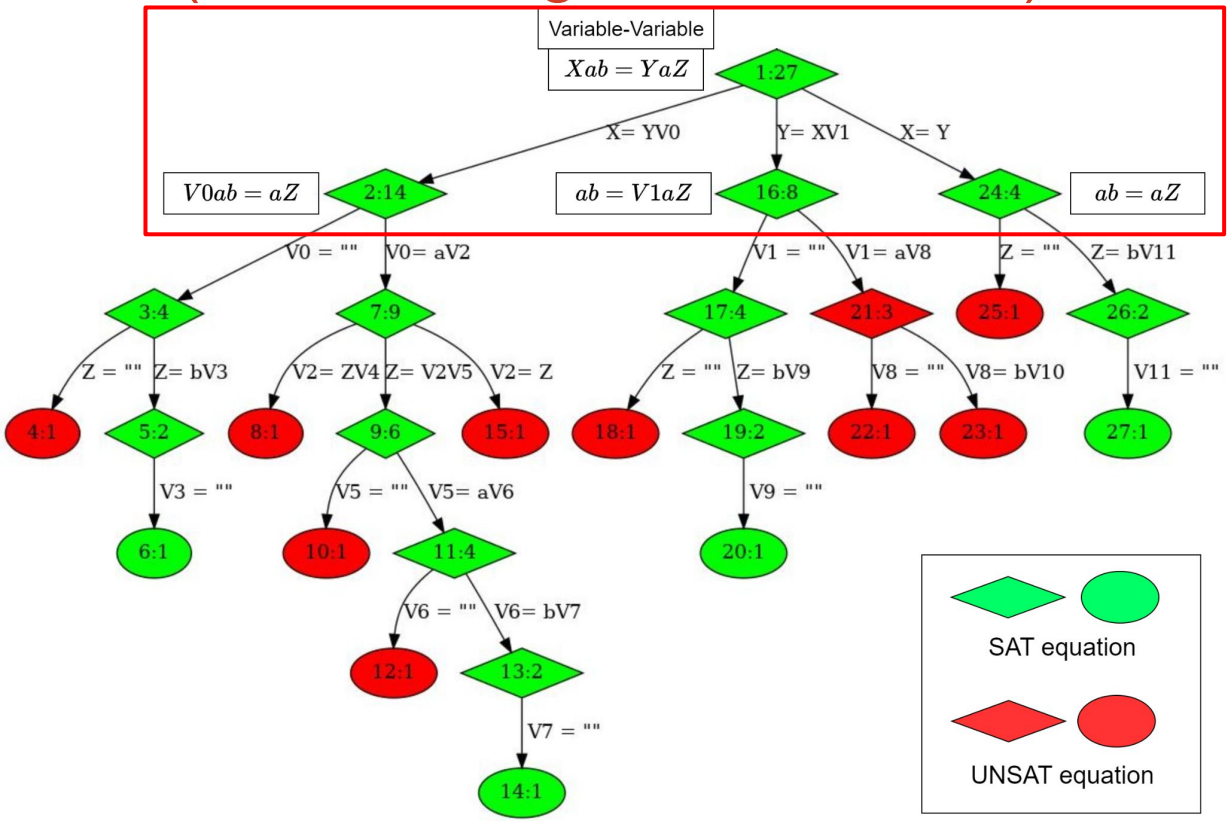
Split Algorithm (Constructing the Proof Tree)



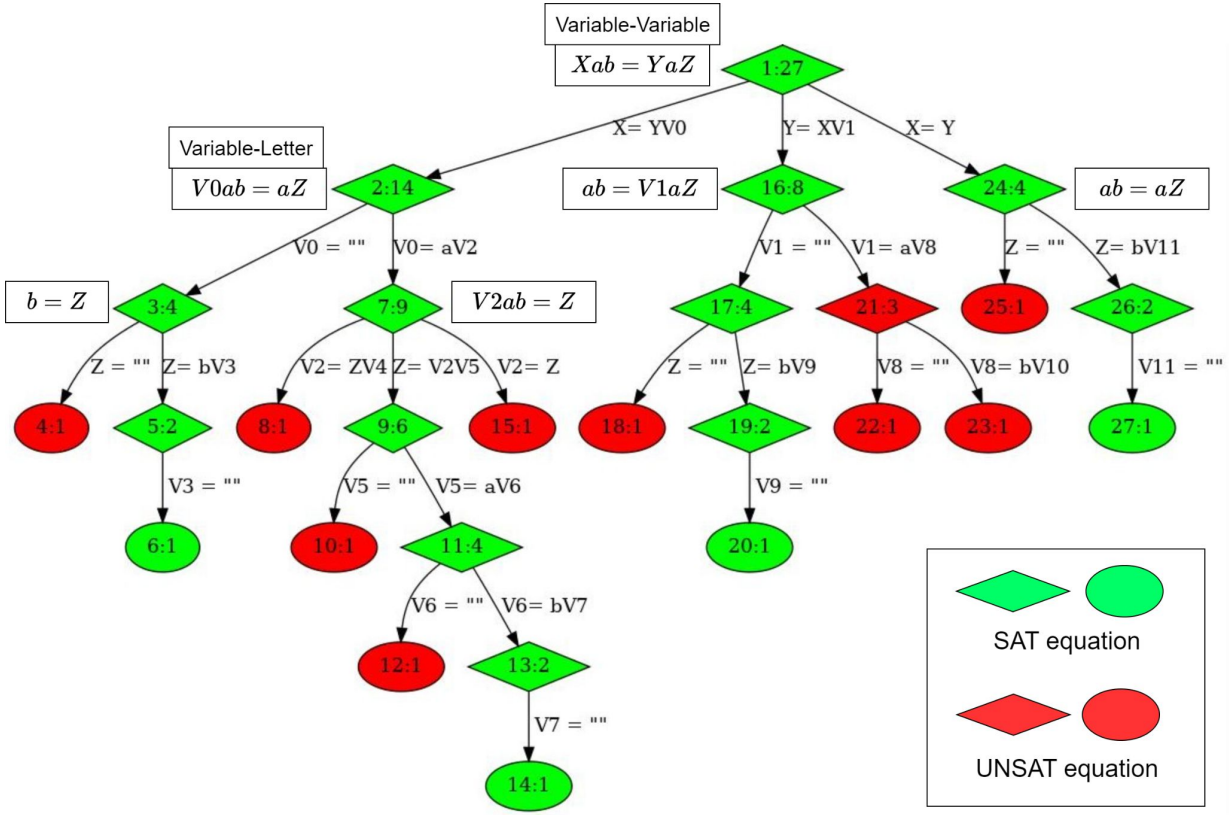
Split Algorithm (Constructing the Proof Tree)



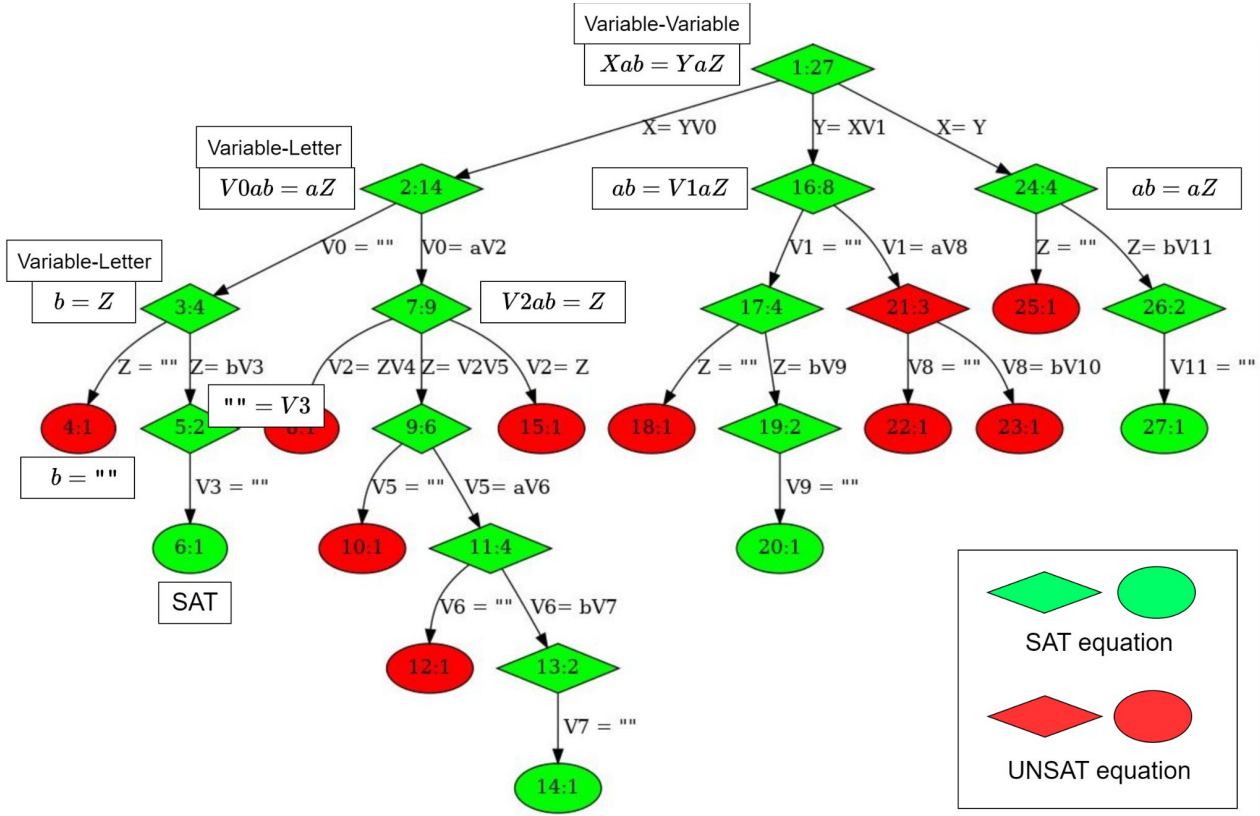
Split Algorithm (Constructing the Proof Tree)



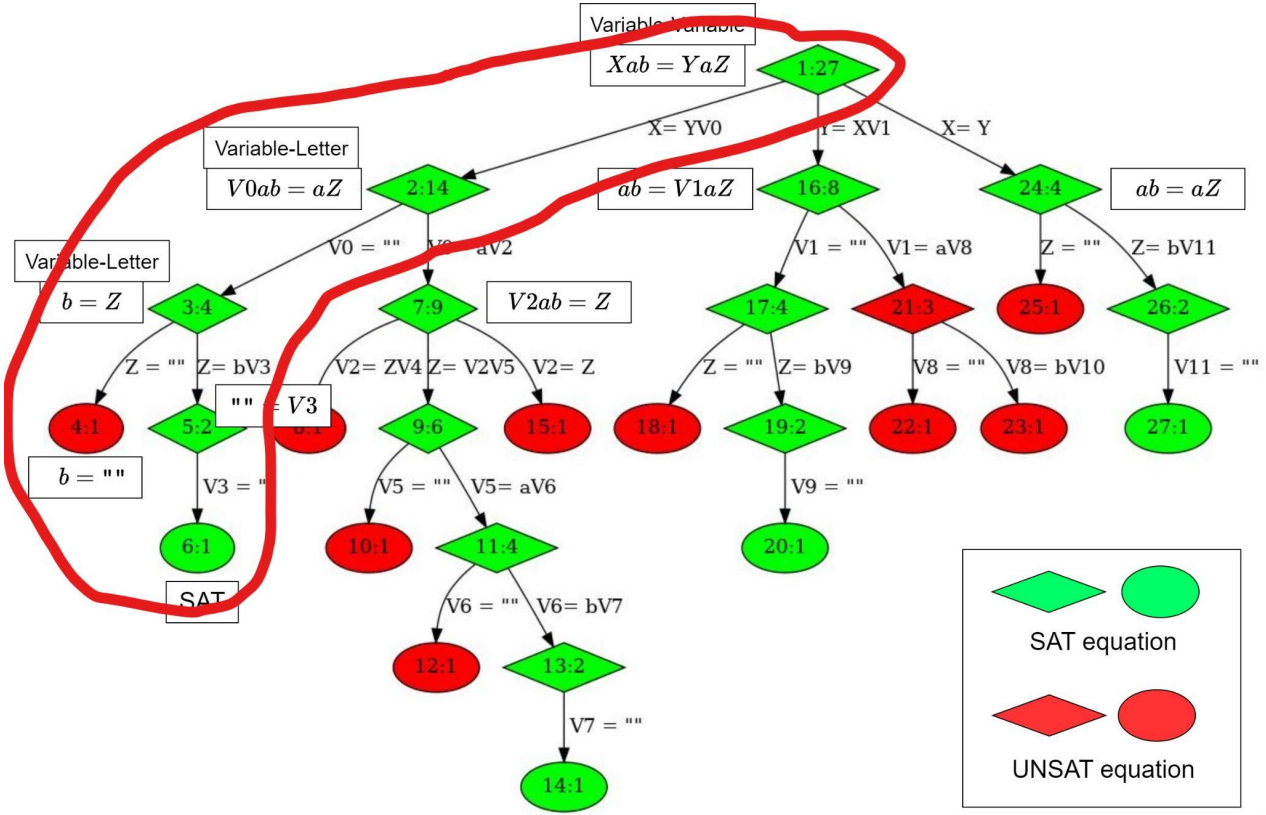
Split Algorithm (Constructing the Proof Tree)



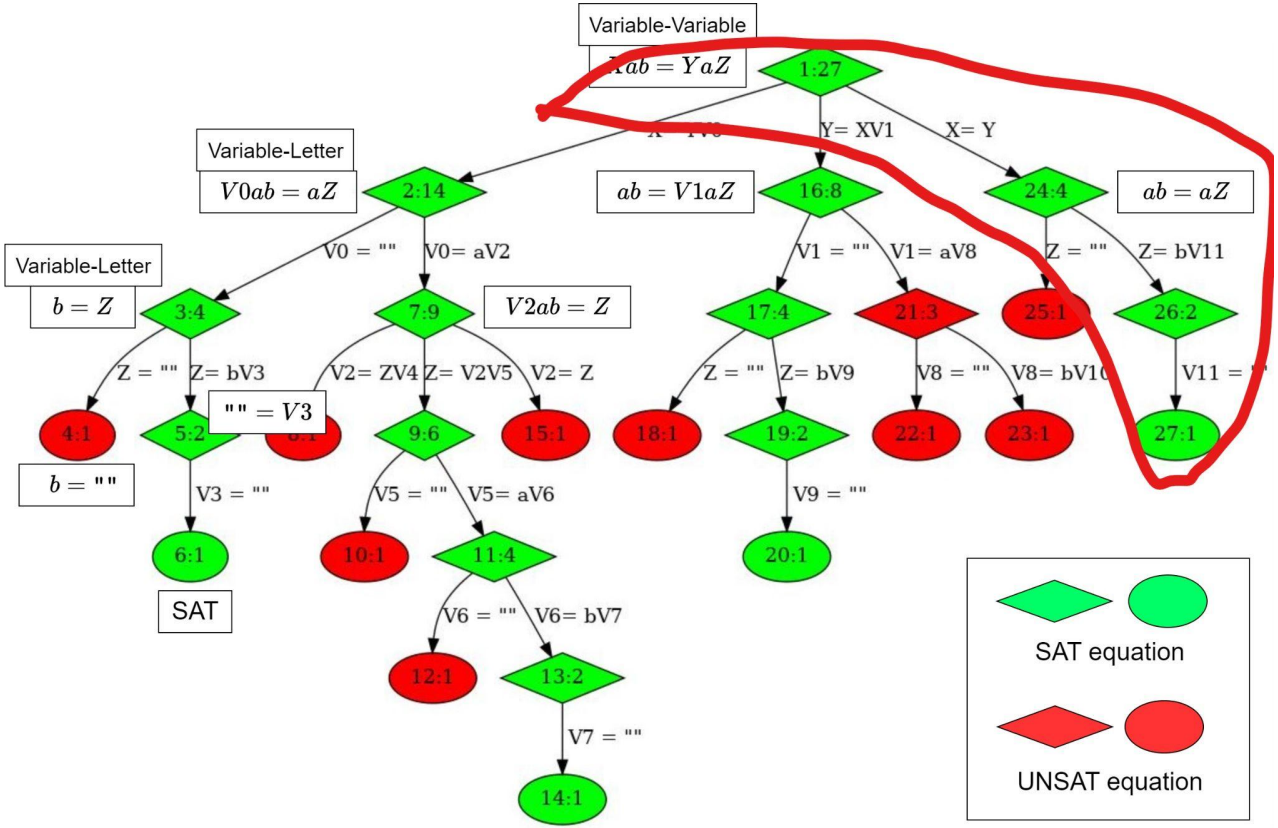
Split Algorithm (Constructing the Proof Tree)



Split Algorithm (A Path Leading to SAT)

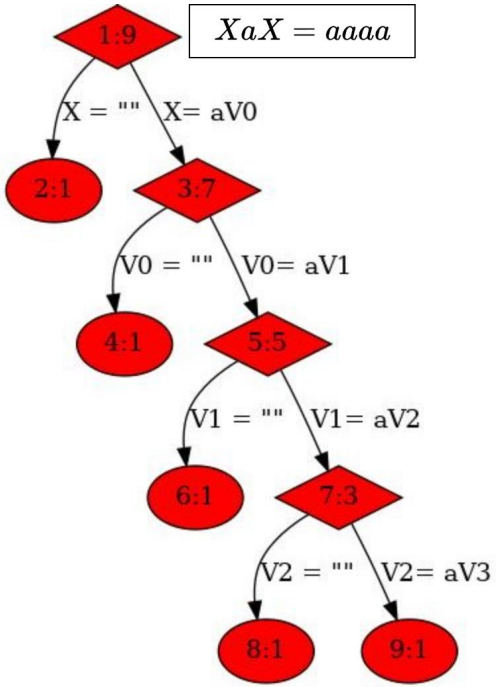


Split Algorithm (A Shortest Path Leading to SAT)

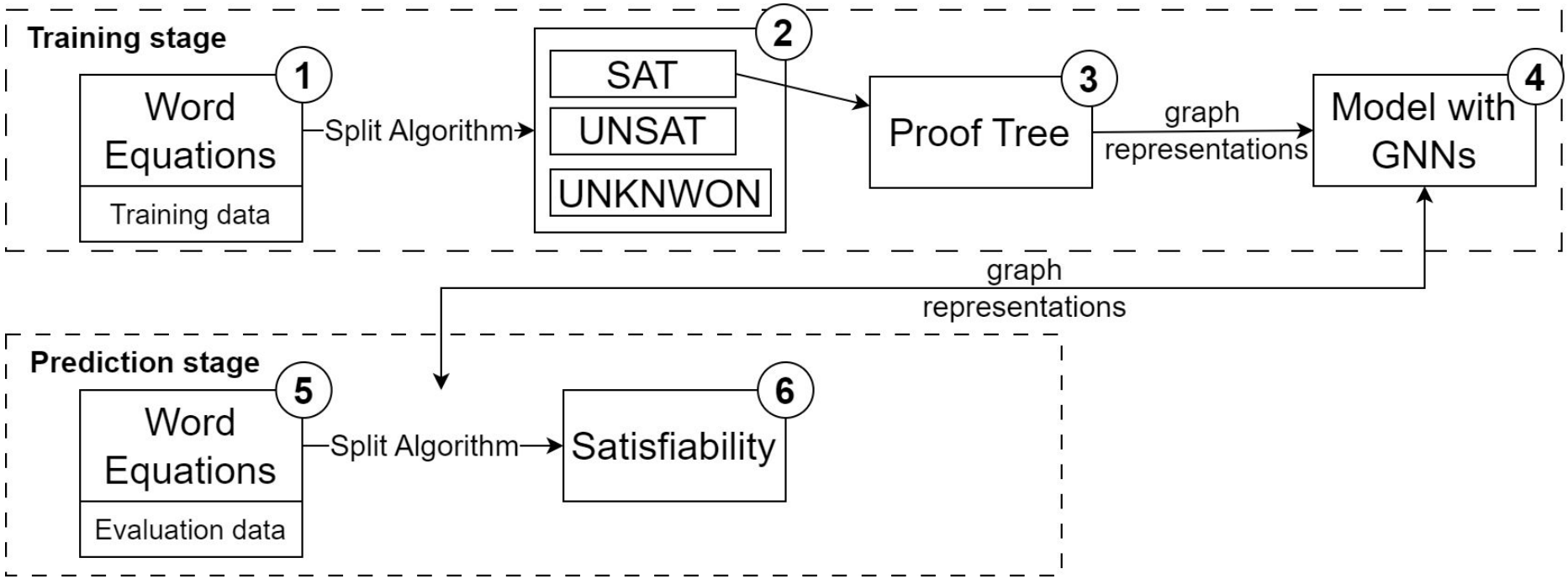


Split Algorithm (Proof Tree for UNSAT Problem)

- Need explore all paths to conclude UNSAT



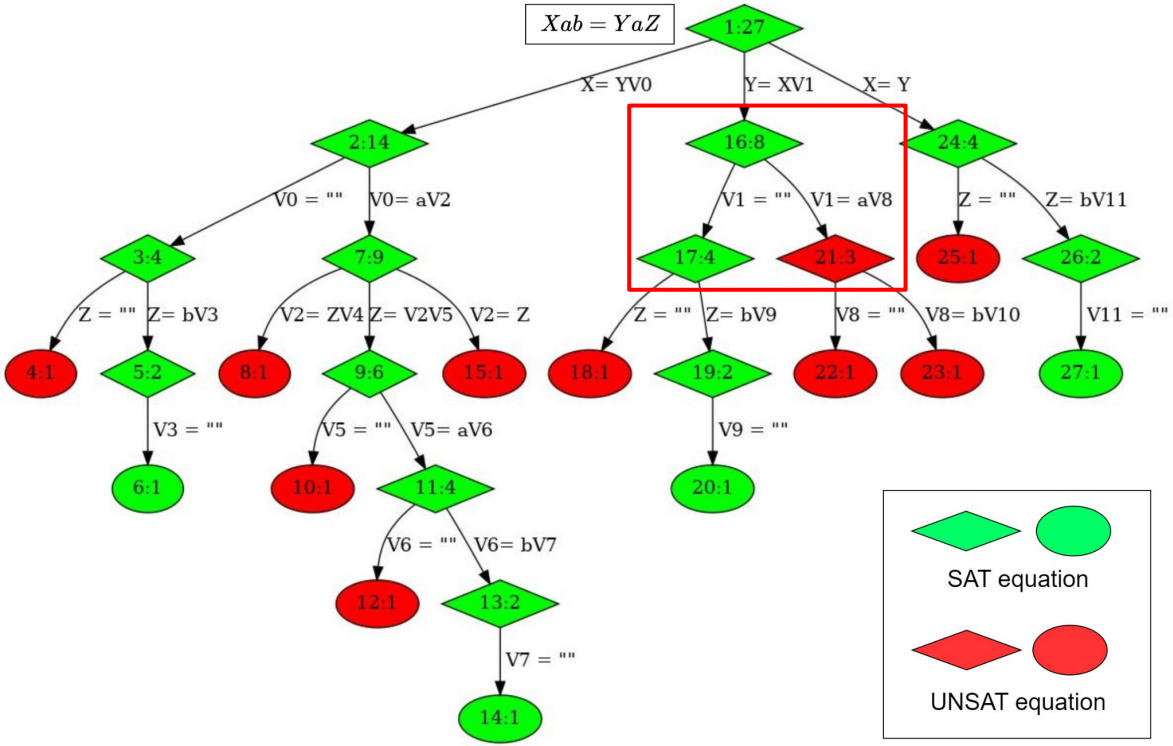
Working Pipeline (Data-driven Based Heuristic)



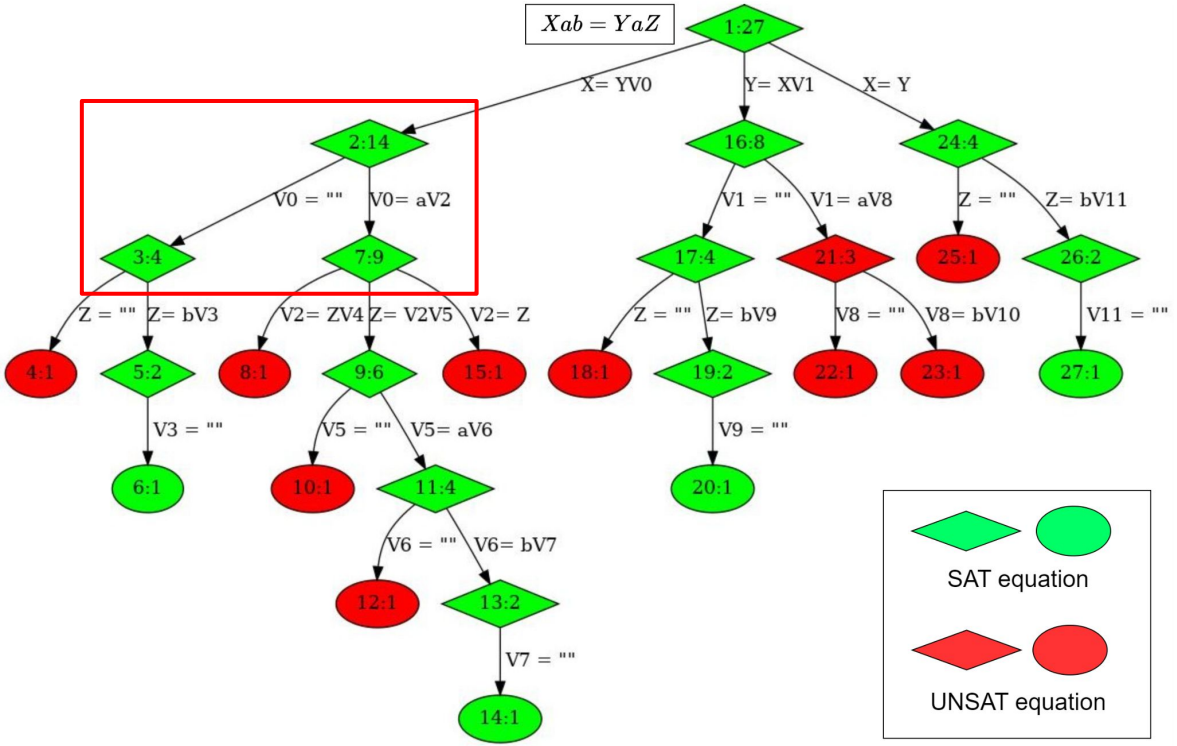
Graph Neural Networks (GNNs)

- A set of fully connected neural networks
- Take graph as input, output node and graph representations
- Can capture the structural features of graph

Label the Training Data at a Branch Point

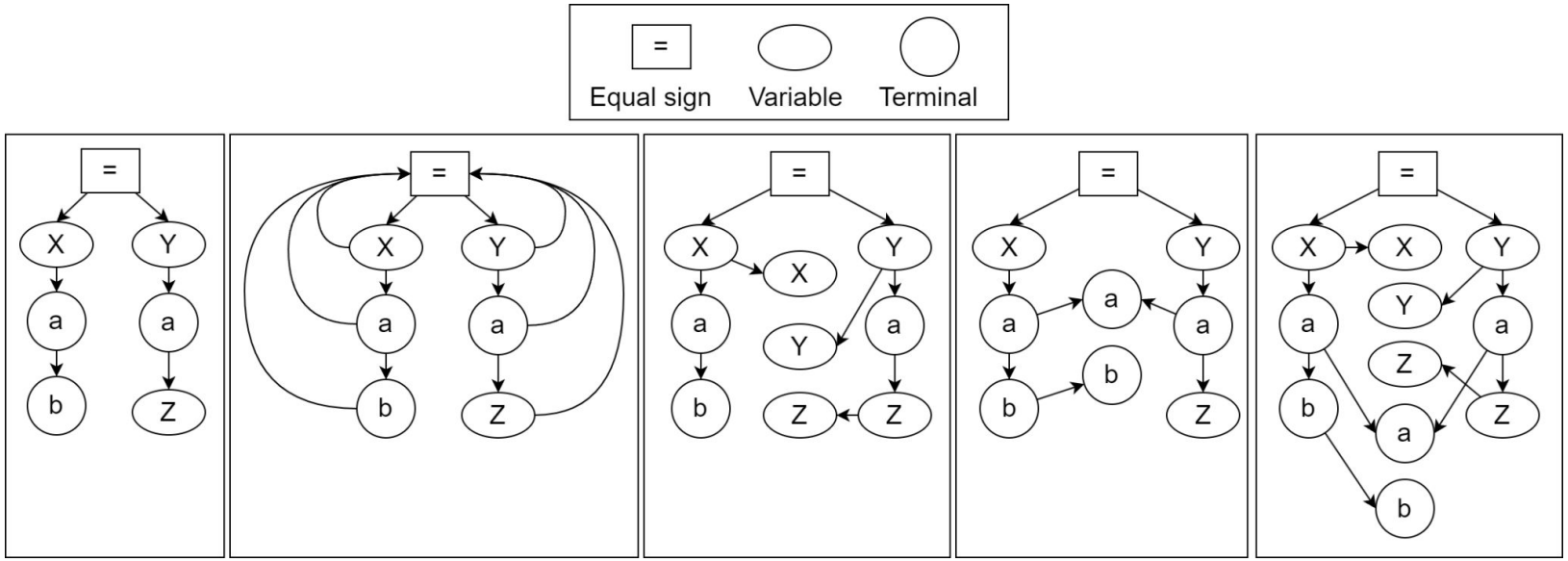


Label the Training Data at a Branch Point

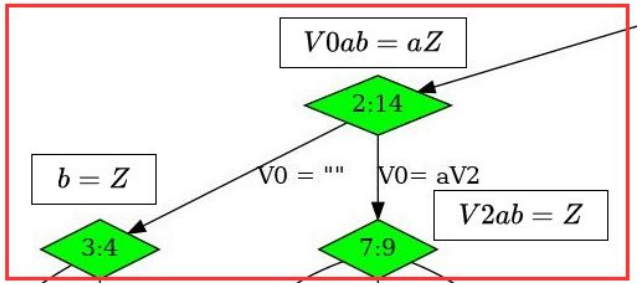


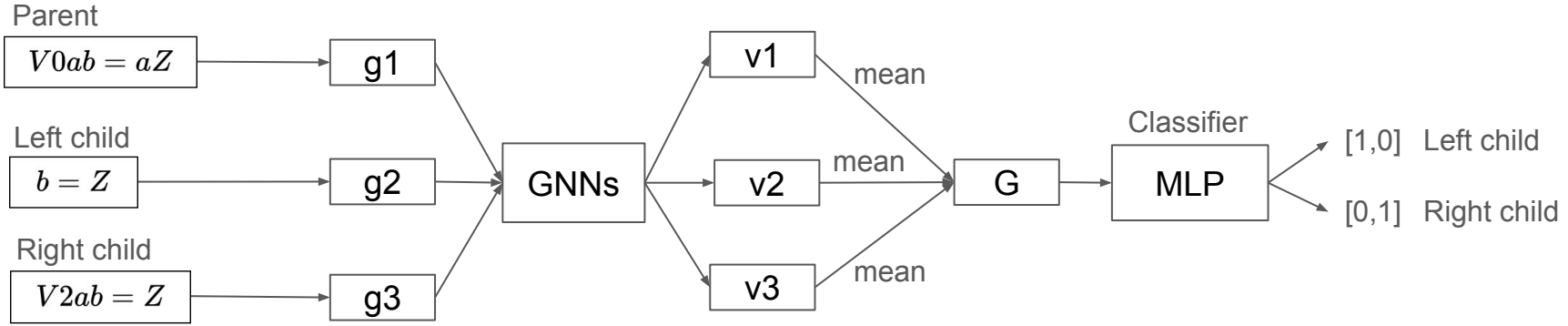
Graph Representations of Word Equation

Equation: $Xab = YaZ$



Model Structure





Benchmarks 1-4

1. Randomly generated SAT word equation

$$aksCwqeuafhkajshfweeta = aksfjdfbabDeuafhkajshBCDta$$

Benchmarks 1-4

1. Randomly generated SAT word equation
2. Word equation with a particular pattern [5]

$$X_n a X_n b X_{n-1} b X_{n-2} \cdots b X_1 = a X_n X_{n-1} X_{n-1} b X_{n-2} b \cdots b X_1 X_1 baa$$

[5] Day, J.D., Ehlers, T., Kulczynski, M., Manea, F., Nowotka, D., Poulsen, D.B.: On solving word equations using SAT. Reachability Problems, 93–106. (2019)

Benchmarks 1-4

1. One randomly generated SAT word equation
2. One word equation with a particular pattern
3. Conjunctive word equations of Benchmark 1

$$kSY = WHXGk$$
$$\wedge\, vZX = vtntssemtvnetm$$
$$\wedge\, yffyyfFWVff = yffyyfyXLGZfU$$
$$\wedge\, NRQxgGI = xxgggFJTK$$

Benchmarks 1-4

1. One randomly generated SAT word equation
2. One word equation with a particular pattern
3. Conjunctive word equations of Benchmark 1
4. QF_S, QF_SLIA, and QF_SNLIA tracks of SMT-LIB without length constraints, regular expressions, and Boolean operators

$$AabcdBCefDEghF = BciabADGCFHE$$

$$abAaBabcdCdcDE = AbeFCcdbaBDcdGfg$$

Benchmarks 1-4 (Overview)

1. One randomly generated SAT word equation
2. One word equation with a particular pattern
3. Conjunctive word equations of Benchmark 1
4. QF_S, QF_SLIA, and QF_SNLIA tracks of SMT-LIB without length constraints, regular expressions, and Boolean operators

Table 1: Number of SAT (✓), UNSAT (×), UNKNOWN (∞), and evaluation (Eval) problems in the four benchmarks

Benchmark 1 Total: 3000				Benchmark 2 Total: 21000				Benchmark 3 Total: 41000				Benchmark 4 Total: 2310			
2000			Eval	20000			Eval	40000			Eval	1855			Eval
✓	×	∞		✓	×	∞		✓	×	∞		✓	×	∞	
1997	0	3	1000	1293	0	18707	1000	1449	1137	37414	1000	1673	16	166	455

Benchmarks 1-4 (Overview)

1. One randomly generated SAT word equation
2. One word equation with a particular pattern
3. Conjunctive word equations of Benchmark 1
4. QF_S, QF_SLIA, and QF_SNLIA tracks of SMT-LIB without length constraints, regular expressions, and Boolean operators

Table 1: Number of SAT (✓), UNSAT (×), UNKNOWN (∞), and evaluation (Eval) problems in the four benchmarks

Benchmark 1 Total: 3000				Benchmark 2 Total: 21000				Benchmark 3 Total: 41000				Benchmark 4 Total: 2310			
2000			Eval	20000			Eval	40000			Eval	1855			Eval
✓	×	∞		✓	×	∞		✓	×	∞		✓	×	∞	
1997	0	3	1000	1293	0	18707	1000	1449	1137	37414	1000	1673	16	166	455

Benchmarks 1-4 (Overview)

1. One randomly generated SAT word equation
2. One word equation with a particular pattern
3. Conjunctive word equations of Benchmark 1
4. QF_S, QF_SLIA, and QF_SNLIA tracks of SMT-LIB without length constraints, regular expressions, and Boolean operators

Table 1: Number of SAT (✓), UNSAT (×), UNKNOWN (∞), and evaluation (Eval) problems in the four benchmarks

Benchmark 1 Total: 3000				Benchmark 2 Total: 21000				Benchmark 3 Total: 41000				Benchmark 4 Total: 2310			
2000			Eval	20000			Eval	40000			Eval	1855			Eval
✓	×	∞		✓	×	∞		✓	×	∞		✓	×	∞	
1997	0	3	1000	1293	0	18707	1000	1449	1137	37414	1000	1673	16	166	455

Benchmarks 1-4 (Overview)

1. One randomly generated SAT word equation
2. One word equation with a particular pattern
3. Conjunctive word equations of Benchmark 1
4. QF_S, QF_SLIA, and QF_SNLIA tracks of SMT-LIB without length constraints, regular expressions, and Boolean operators

Table 1: Number of SAT (✓), UNSAT (×), UNKNOWN (∞), and evaluation (Eval) problems in the four benchmarks

Benchmark 1 Total: 3000				Benchmark 2 Total: 21000				Benchmark 3 Total: 41000				Benchmark 4 Total: 2310			
2000			Eval	20000			Eval	40000			Eval	1855			Eval
✓	×	∞		✓	×	∞		✓	×	∞		✓	×	∞	
1997	0	3	1000	1293	0	18707	1000	1449	1137	37414	1000	1673	16	166	455

Experimental Results (Benchmark 1)

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
1 (1000 SAT)	Fixed	999	-	0	777	0	4.1 (182.0)	- (-)	4.0 (169.0)	- (-)
	Random	996	-	0			4.2 (349.6)	- (-)	4.1 (269.8)	- (-)
	GNN	995	-	0			7.6 (215.7)	- (-)	7.0 (162.3)	- (-)
	cvc5	**1000**	-	0			0.1 (-)	- (-)	0.1 (-)	- (-)
	Ostrich	918	-	0			20.4 (-)	- (-)	19.6 (-)	- (-)
	Woorpje	967	-	0			1.6 (-)	- (-)	0.5 (-)	- (-)
	Z3	902	-	0			3.4 (-)	- (-)	2.4 (-)	- (-)
	Z3-Noodler	935	-	0			1.9 (-)	- (-)	1.1 (-)	- (-)

Experimental Results (Benchmark 1)

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
1 (1000 SAT)	Fixed	999	-	0	777	0	4.1 (182.0)	- (-)	4.0 (169.0)	- (-)
	Random	996	-	0			4.2 (349.6)	- (-)	4.1 (269.8)	- (-)
	GNN	995	-	0			7.6 (215.7)	- (-)	7.0 (162.3)	- (-)
	cvc5	**1000**	-	0			0.1 (-)	- (-)	0.1 (-)	- (-)
	Ostrich	918	-	0			20.4 (-)	- (-)	19.6 (-)	- (-)
	Woorpje	967	-	0			1.6 (-)	- (-)	0.5 (-)	- (-)
	Z3	902	-	0			3.4 (-)	- (-)	2.4 (-)	- (-)
	Z3-Noodler	935	-	0			1.9 (-)	- (-)	1.1 (-)	- (-)

Experimental Results (Benchmark 1)

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
1 (1000 SAT)	Fixed	999	-	0	777	0	4.1 (182.0)	- (-)	4.0 (169.0)	- (-)
	Random	996	-	0			4.2 (349.6)	- (-)	4.1 (269.8)	- (-)
	GNN	995	-	0			7.6 (215.7)	- (-)	7.0 (162.3)	- (-)
	cvc5	**1000**	-	0			0.1 (-)	- (-)	0.1 (-)	- (-)
	Ostrich	918	-	0			20.4 (-)	- (-)	19.6 (-)	- (-)
	Woorpje	967	-	0			1.6 (-)	- (-)	0.5 (-)	- (-)
	Z3	902	-	0			3.4 (-)	- (-)	2.4 (-)	- (-)
	Z3-Noodler	935	-	0			1.9 (-)	- (-)	1.1 (-)	- (-)

Experimental Results (Benchmark 1)

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
1 (1000 SAT)	Fixed	999	-	0	777	0	4.1 (182.0)	- (-)	4.0 (169.0)	- (-)
	Random	996	-	0			4.2 (349.6)	- (-)	4.1 (269.8)	- (-)
	GNN	995	-	0			7.6 (215.7)	- (-)	7.0 (162.3)	- (-)
	cvc5	**1000**	-	0			0.1 (-)	- (-)	0.1 (-)	- (-)
	Ostrich	918	-	0			20.4 (-)	- (-)	19.6 (-)	- (-)
	Woorpje	967	-	0			1.6 (-)	- (-)	0.5 (-)	- (-)
	Z3	902	-	0			3.4 (-)	- (-)	2.4 (-)	- (-)
	Z3-Noodler	935	-	0			1.9 (-)	- (-)	1.1 (-)	- (-)

Experimental Results (Benchmark 2)

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
2 (1000 in total)	Fixed	33	0	10	1	0	13.2 (1115.2)	- (-)	4.8 (7)	- (-)
	Random	41	0	6			11.7 (3879.5)	- (-)	4.2 (60)	- (-)
	GNN	**71**	0	27			46.0 (1813.5)	- (-)	5.1 (5)	- (-)
	cvc5	4	2	4			2.0 (-)	0.1 (-)	0.1 (-)	- (-)
	Ostrich	14	**43**	44			40.7 (-)	31.8 (-)	2.5 (-)	- (-)
	Woorpje	23	0	2			38.3 (-)	- (-)	0.1 (-)	- (-)
	Z3	6	0	2			0.1 (-)	- (-)	4.2 (-)	- (-)
	Z3-Noodler	19	0	0			45.8 (-)	- (-)	4.2 (-)	- (-)

Experimental Results (Benchmark 3)

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
3 (1000 in total)	Fixed	32	79	0	23	50	5.2 (1946.2)	65.8 (4227.0)	3.6 (57.0)	38.3 (796.6)
	Random	32	79	0			9.5 (3861.8)	65.0 (4227.0)	3.8 (61.7)	38.5 (796.6)
	GNN	32	65	0			214.3 (1471.2)	1471.2 (1471.2)	4.6 (63.7)	84.0 (796.6)
	cvc5	32	943	2			0.1 (-)	0.3 (-)	0.1 (-)	0.3 (-)
	Ostrich	27	926	0			5.8 (-)	4.7 (-)	4.6 (-)	4.5 (-)
	Woorpje	**34**	723	1			12.4 (-)	12.3 (-)	0.1 (-)	23.2 (-)
	Z3	26	**953**	10			5.6 (-)	0.5 (-)	4.7 (-)	0.1 (-)
	Z3-Noodler	28	926	0			22.7 (-)	0.3 (-)	8.9 (-)	0.1 (-)

Experimental Results (Benchmark 4)

Bench	Solver	Number of solved problems					Average solving time (split number)			
		SAT	UNS	UNI	CS	CU	SAT	UNS	CS	CU
4 (455 in total)	Fixed	416	6	0	403	2	5.1 (105.5)	17.7 (17119.5)	5.1 (51.0)	5.0 (246)
	Random	415	6	0			4.9 (61.3)	17.9 (17119.5)	4.9 (38.1)	4.4 (246)
	GNN	418	5	0			5.5 (118.3)	31.8 (5019.6)	5.3 (49.0)	8.8 (246)
	cvc5	406	34	0			0.1 (-)	0.1 (-)	0.1 (-)	0.1 (-)
	Ostrich	406	6	0			1.4 (-)	1.2 (-)	1.4 (-)	1.2 (-)
	Woorpje	**420**	2	0			0.2 (-)	3.6 (-)	0.2 (-)	3.6 (-)
	Z3	**420**	10	0			0.1 (-)	0.1 (-)	0.1 (-)	0.1 (-)
	Z3-Noodler	**420**	**35**	1			0.1 (-)	0.1 (-)	0.1 (-)	0.1 (-)

Conclusion

- Our framework simple but worked well
- Split algorithm is good at solving SAT word equation problems
- GNN guided version is good at solving problems with particular pattern

Future work

- Handle UNSAT problem more efficiently
- Extend to including regular expression

Thank you!

Q & A

Benchmark 1

- One randomly generated SAT word equation
  - Construct a word equation with only terminals and SAT
  - Repeatedly replace substrings by fresh variables

aksfjdfbaboiwqeuafhkajshfweeta = aksfjdfbaboiwqeuafhkajshfweeta

aksAaboiwqeuafhkajshfweeta = aksfjdfbaboiwqeuafhkajshBCDta

aksAaboiwqeuafhkajshfweeta = aksfjdfbaboiwqeuafhkajshBCDta

aksCwqeuafhkajshfweeta = aksfjdfbabDeuafhkajshBCDta

Benchmark 2

- Generate a word equation with following pattern [4]:

$$X_n a X_n b X_{n-1} b X_{n-2} \cdots b X_1 = a X_n X_{n-1} X_{n-1} b X_{n-2} b \cdots b X_1 X_1 baa$$

- Replacing each terminal b in with a randomly generated string:

$$X_n a X_n kjlYU X_{n-1} AhdCdjals X_{n-2} \cdots sadVad X_1$$
$$= a X_n X_{n-1} X_{n-1} JKG X_{n-2} BCSa \cdots hki X_1 X_1 HIaa$$

[4] Day, J.D., Ehlers, T., Kulczynski, M., Manea, F., Nowotka, D., Poulsen, D.B.: On solving word equations using SAT. In: Filiot, E., Jungers, R., Potapov, I. (eds.) Reachability Problems. pp. 93–106. Springer International Publishing, Cham (2019)

Benchmark 3

- Conjoining 1-100 word equations from Benchmark 1

$$kSY = WHXGk$$
$$\wedge\, vZX = vtntssemtvnetm$$
$$\wedge\, yffyyfFWVff = yffyyfyXLGZfU$$
$$\wedge\, NRQxgGI = xxgggFJTK$$

Benchmark 4

- Extracted from the non-incremental QF_S, QF_SLIA, and QF_SNLIA tracks of SMT-LIB by removing their length constraints, regular expressions, and Boolean operators

$$AabcdBCefDEghF = BciabADGCFHE$$

$$abAaBabcdCdcDE = AbeFCcdbaBDcdGfg$$

Experimental Results

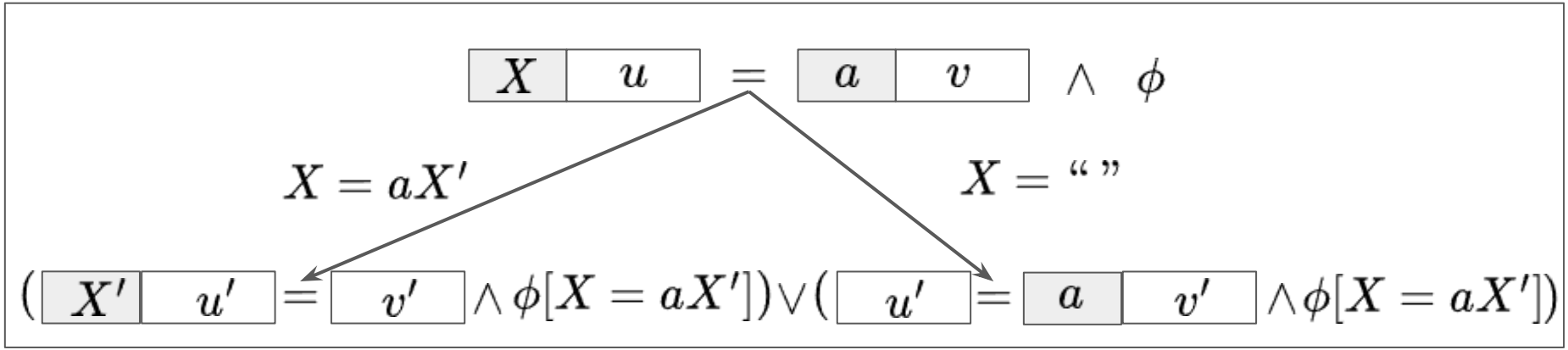
GT	Benchmark 1									Benchmark 2								
	BT_1			BT_2			BT_3			BT_1			BT_2			BT_3		
	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3
G1	991	**1000**	996	997	999	999	584	627	624	57	48	44	53	45	40	3	3	3
G2	998	1000	1000	998	1000	998	584	627	624	49	46	41	53	40	50	3	3	3
G3	997	1000	998	998	1000	999	584	627	624	53	43	49	65	46	55	3	3	3
G4	985	999	997	984	999	995	584	627	624	65	52	38	59	54	44	3	3	3
G5	995	1000	999	995	1000	995	584	627	624	64	46	44	**71**	54	50	3	3	3
GT	Benchmark 3									Benchmark 4								
	BT_1			BT_2			BT_3			BT_1			BT_2			BT_3		
	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3
G1	32	32	31	32	32	32	14	16	16	413	415	413	417	417	416	400	404	402
G2	34	32	32	34	32	34	13	16	16	416	416	416	418	418	417	400	402	403
G3	32	32	32	31	32	33	13	16	16	416	416	417	418	417	415	400	402	404
G4	**35**	32	32	34	32	32	14	16	16	414	413	415	417	417	416	400	403	403
G5	31	31	31	32	31	32	14	16	16	415	416	414	**418**	417	416	400	402	402

Table 3: Detailed results for number of solved problems of DragonLi in terms of five graph representations (G1 to G5 represent Graph 1 to 5 in Section 4.1), three strategies to apply GNN back to the algorithm (S1, S2, S3 in Section 4.4), and three backtracking strategies (BT_1, BT_2, BT_3 in Section 3.3). The column GT denotes graph type.

GT	Benchmark 1									Benchmark 2								
	BT_1			BT_2			BT_3			BT_1			BT_2			BT_3		
	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3
G1	991	**1000**	996	997	999	999	584	627	624	57	48	44	53	45	40	3	3	3
G2	998	1000	1000	998	1000	998	584	627	624	49	46	41	53	40	50	3	3	3
G3	997	1000	998	998	1000	999	584	627	624	53	43	49	65	46	55	3	3	3
G4	985	999	997	984	999	995	584	627	624	65	52	38	59	54	44	3	3	3
G5	995	1000	999	995	1000	995	584	627	624	64	46	44	**71**	54	50	3	3	3
GT	Benchmark 3									Benchmark 4								
	BT_1			BT_2			BT_3			BT_1			BT_2			BT_3		
	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3
G1	32	32	31	32	32	32	14	16	16	413	415	413	417	417	416	400	404	402
G2	34	32	32	34	32	34	13	16	16	416	416	416	418	418	417	400	402	403
G3	32	32	32	31	32	33	13	16	16	416	416	417	418	417	415	400	402	404
G4	**35**	32	32	34	32	32	14	16	16	414	413	415	417	417	416	400	403	403
G5	31	31	31	32	31	32	14	16	16	415	416	414	**418**	417	416	400	402	402

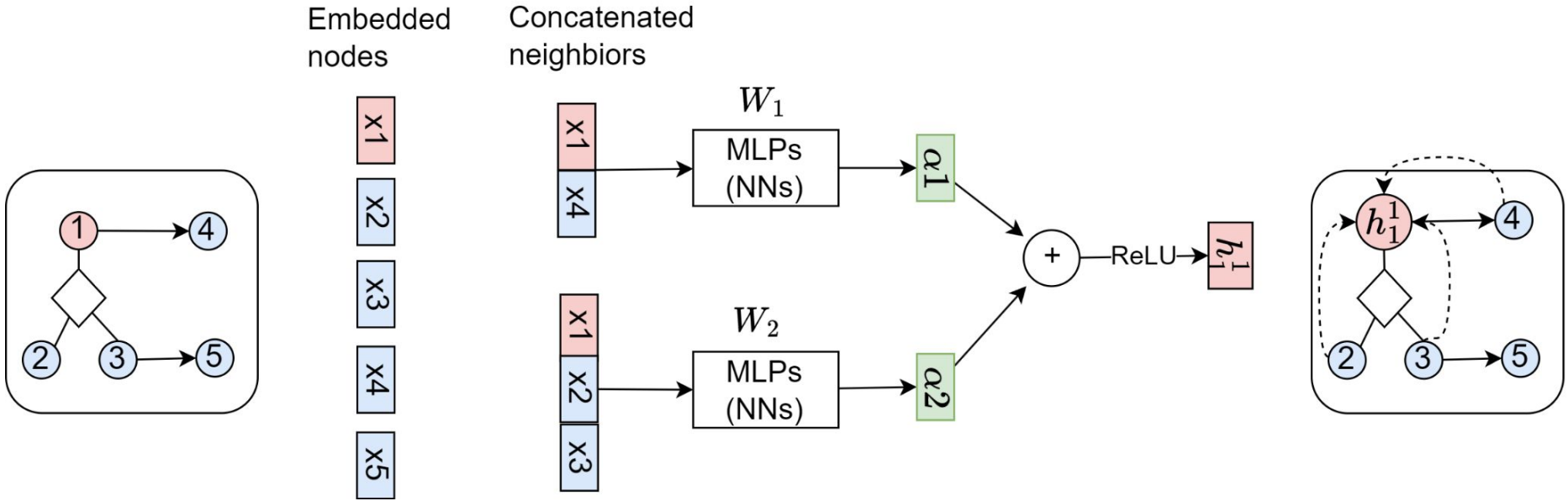
Proof rules (Extend to Conjunctive Word Equations)

- First terms are Variable - Terminal (remove this slide)

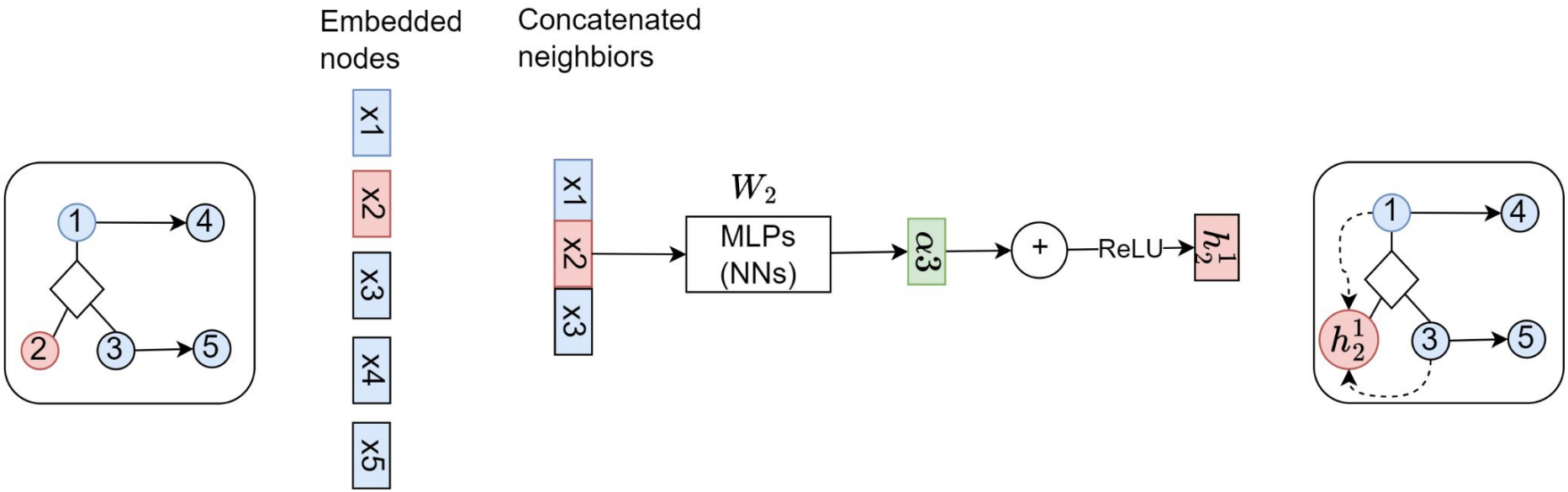


where ϕ is conjunctive word equations

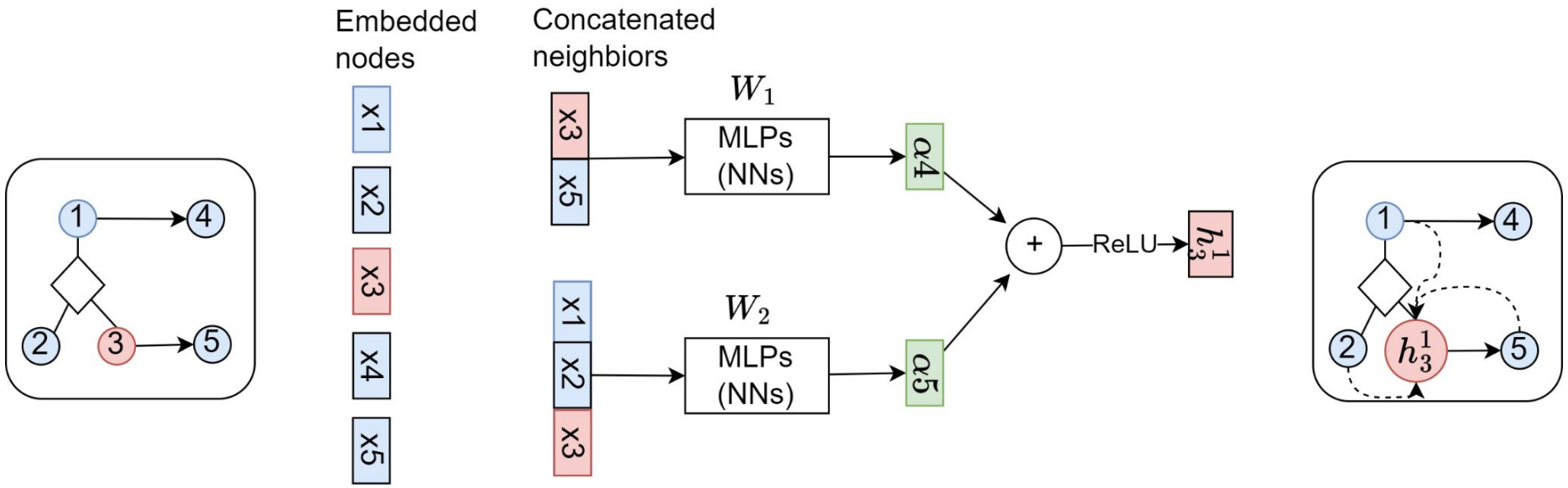
MUSHyperNet Framework (GNN model):



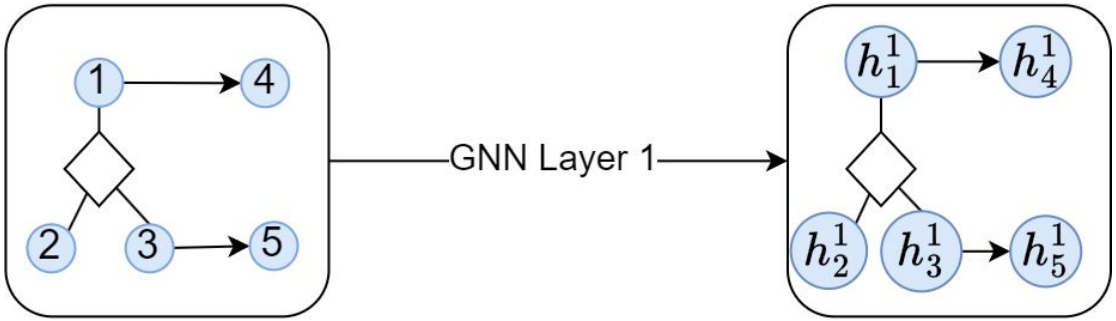
MUSHyperNet Framework (GNN model):



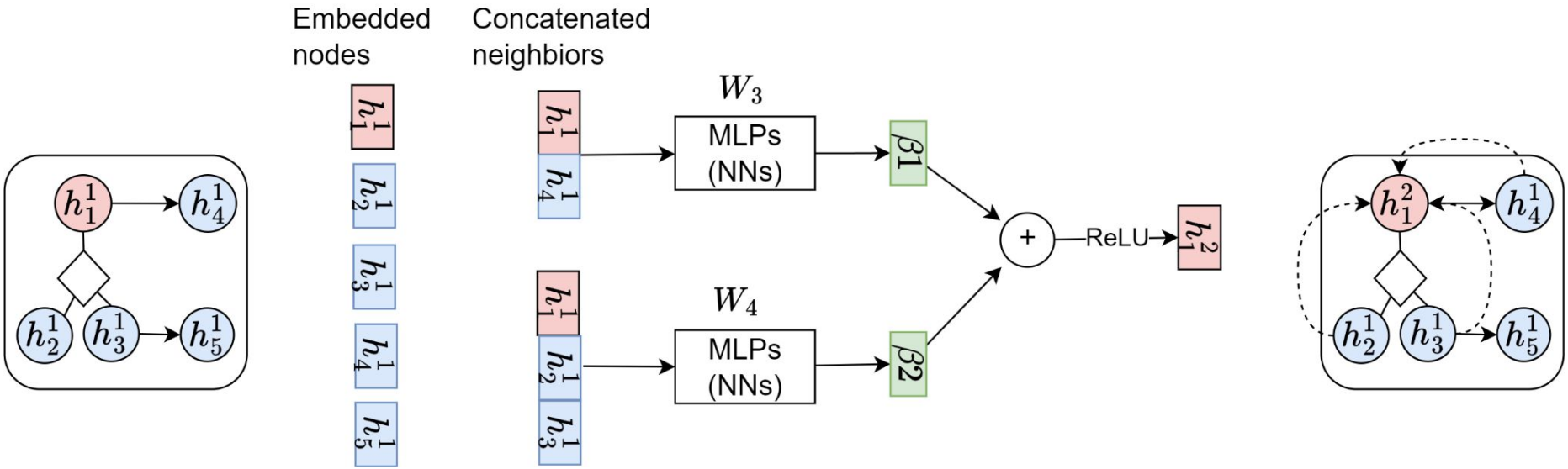
MUSHyperNet Framework (GNN model):



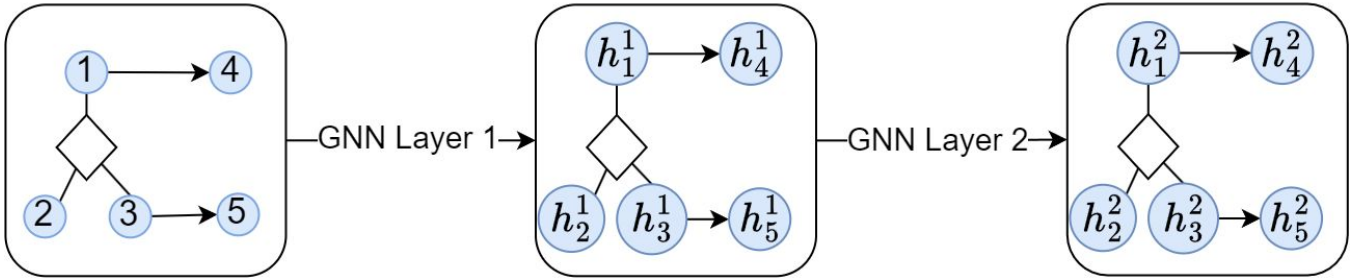
MUSHyperNet Framework (GNN model):



MUSHyperNet Framework (GNN model):



MUSHyperNet Framework (GNN model):



Conjunctive Word Equations

- Transform conjunctive word equations to a single word equation

Conjunctive of word equations: $\bigwedge_{i=1}^{n} w_i^l = w_i^r$

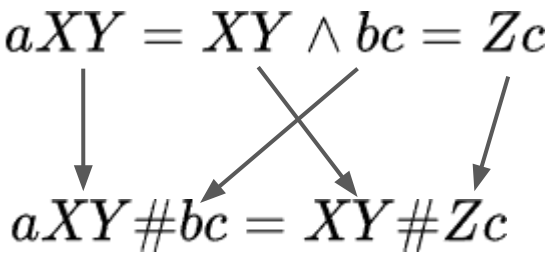
$w_1^l \# w_2^l \# \ldots \# w_n^l = w_1^r \# w_2^r \# \ldots \# w_n^r$

where w_i^l and w_i^r are strings of left and right hand side of a equation.

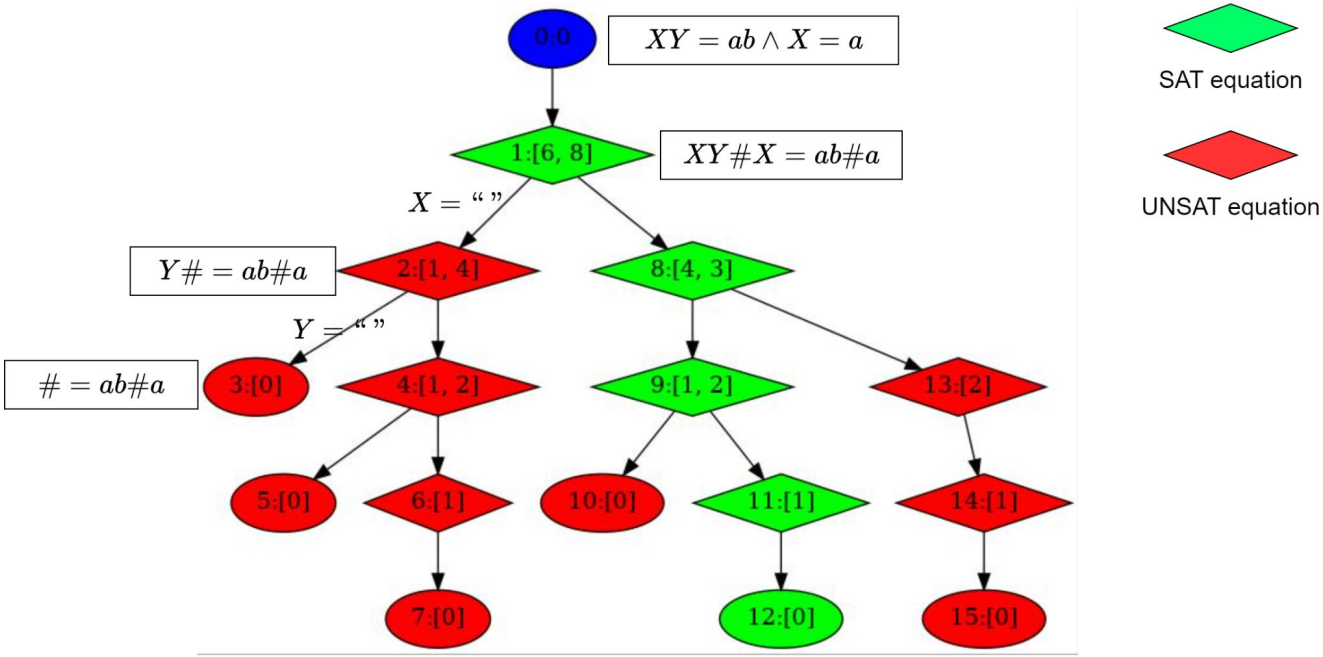
The $\#$ is a special symbol and not in a terminal set.

Conjunctive Word Equations

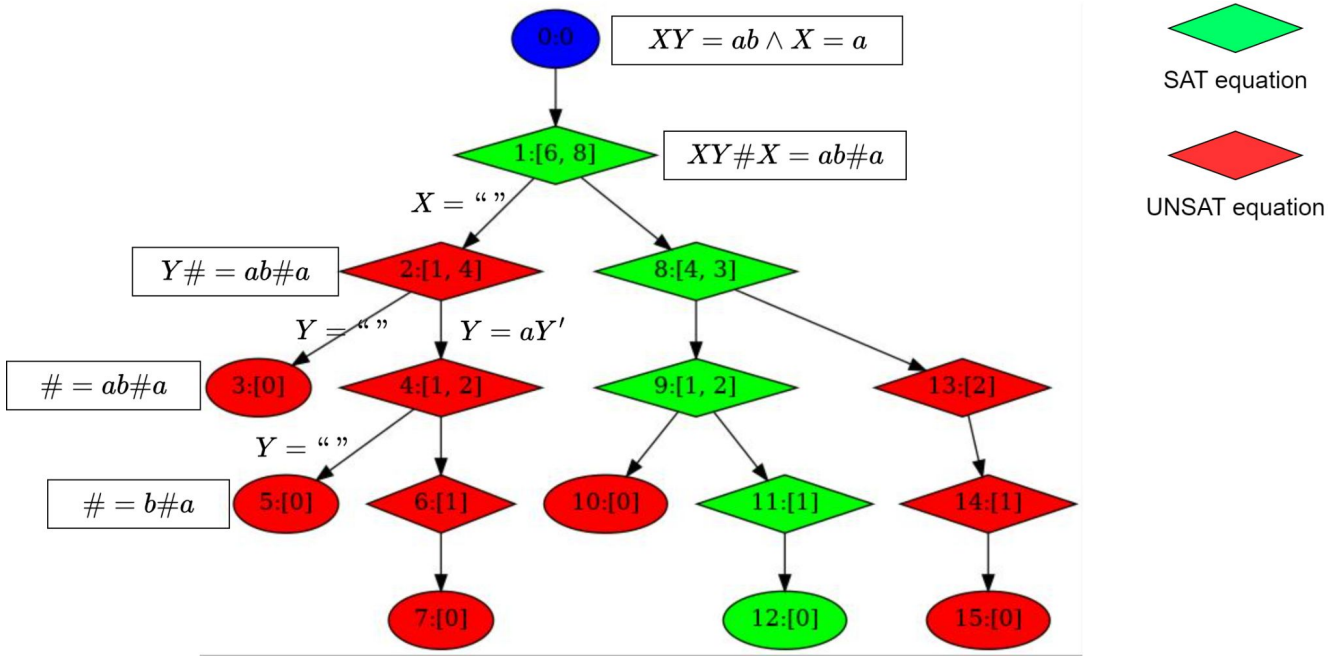
- Encode conjunctive word equations to a single word equation



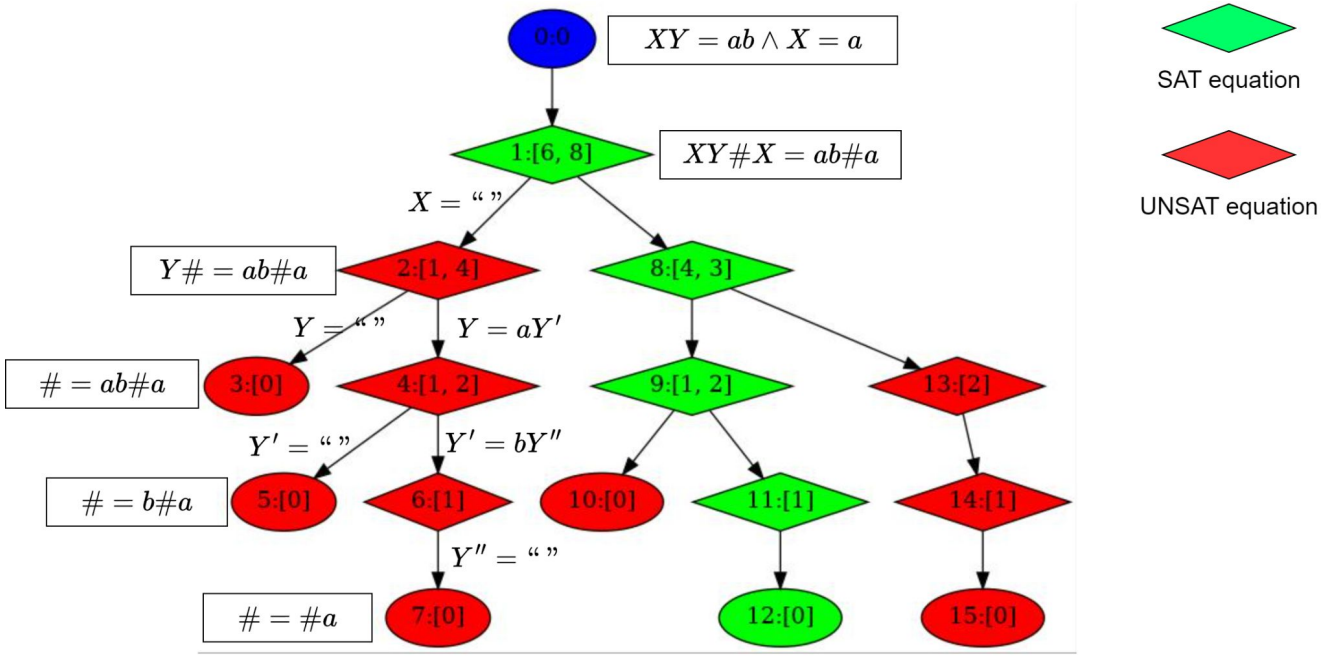
Split Algorithm (Constructing the Proof Tree)



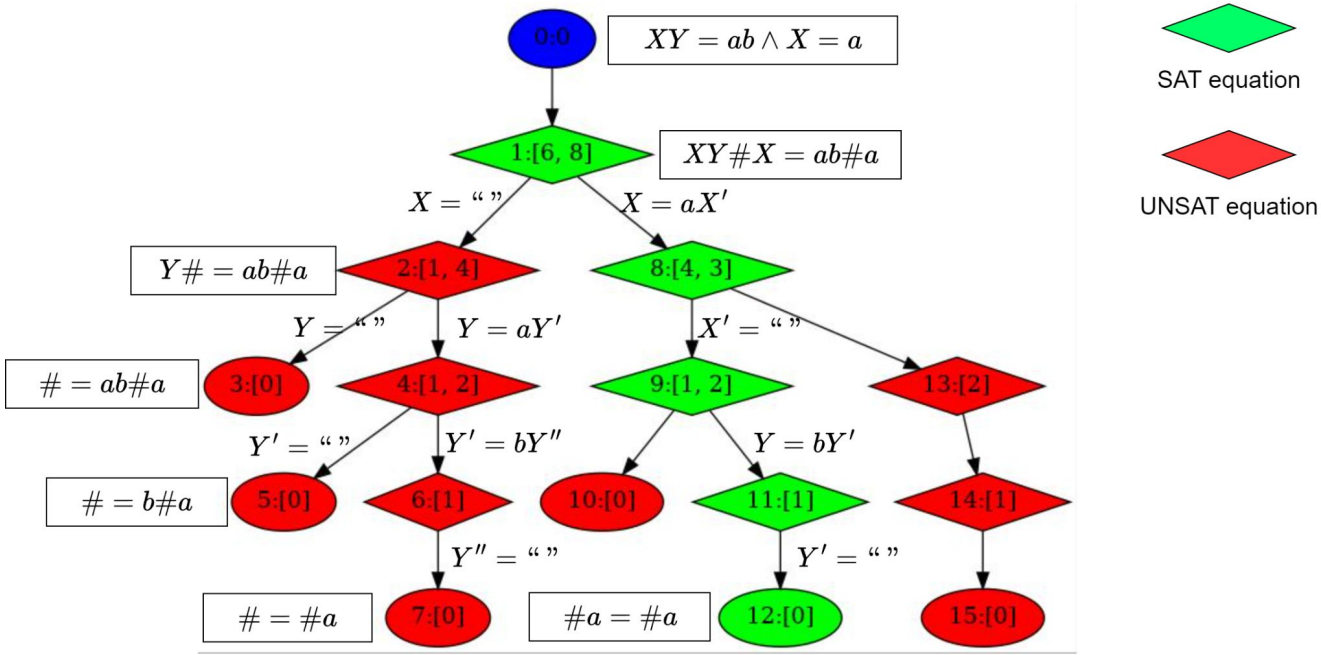
Split Algorithm (Constructing the Proof Tree)



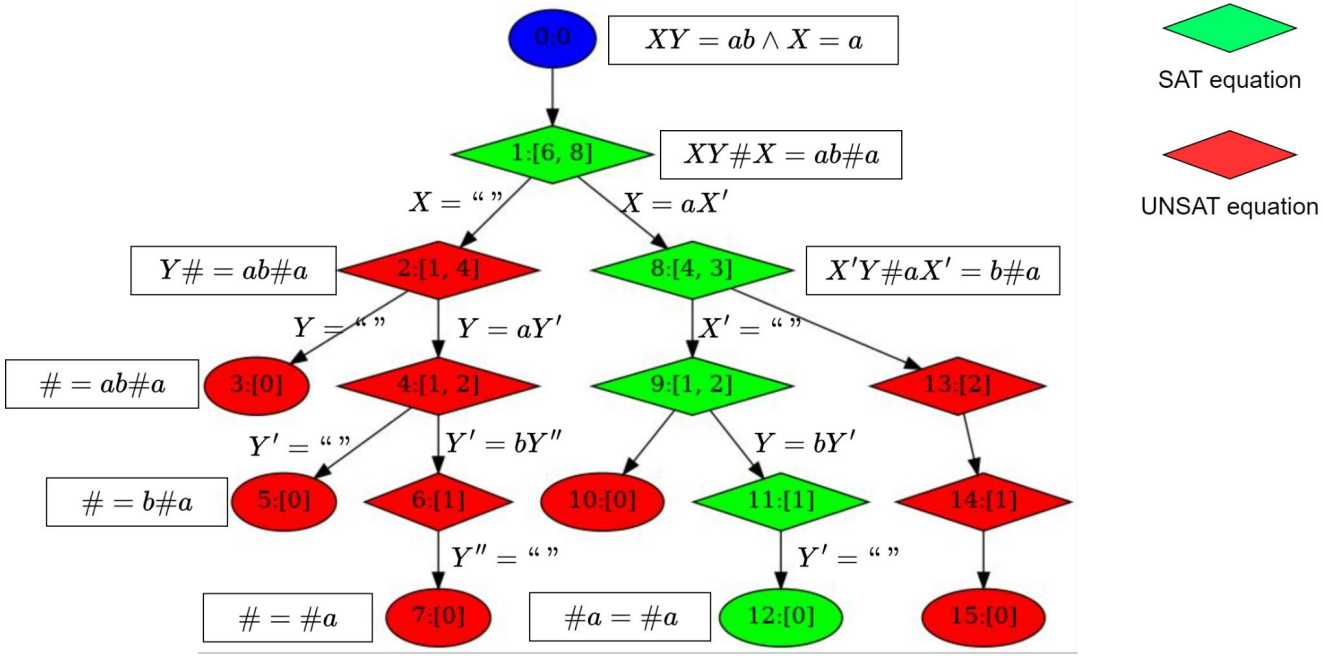
Split Algorithm (Constructing the Proof Tree)



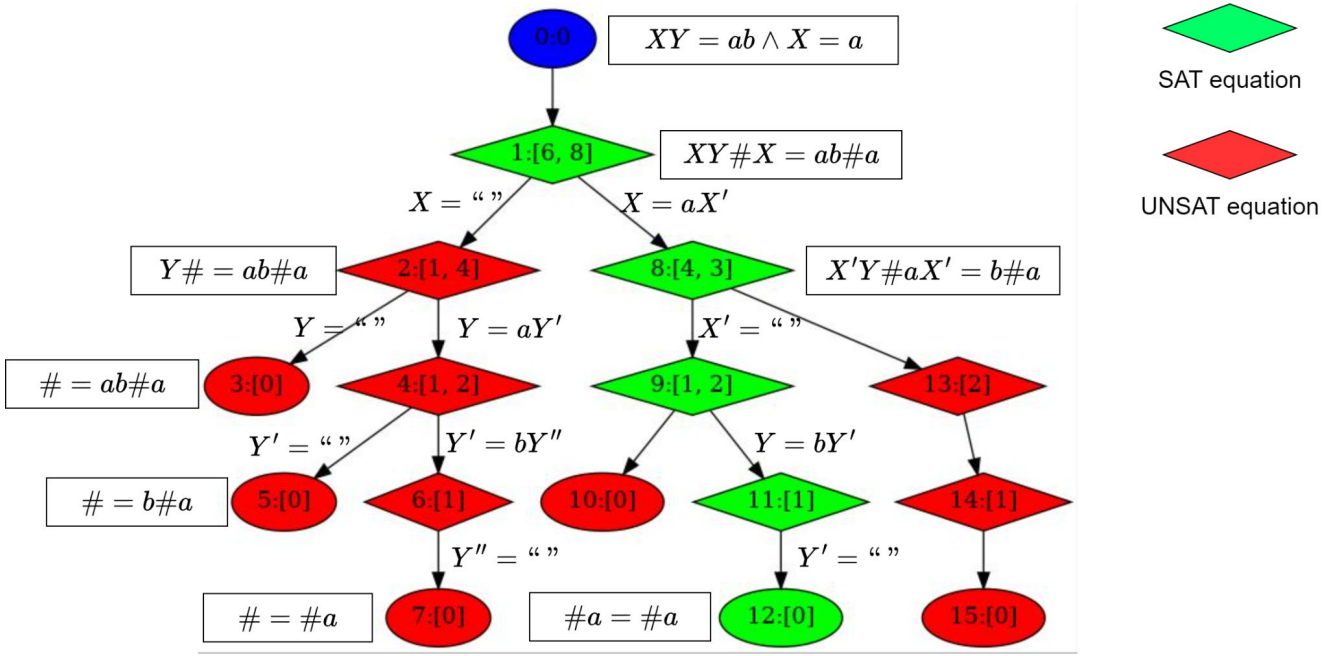
Split Algorithm (Constructing the Proof Tree)



Split Algorithm (Constructing the Proof Tree)



Split Algorithm (Constructing the Proof Tree)



Benchmarks (merge with previous slide)

Table 1: Number of SAT (✓), UNSAT (×), UNKNOWN (∞), and evaluation (Eval) problems in the four benchmarks

Benchmark 1 Total: 3000				Benchmark 2 Total: 21000				Benchmark 3 Total: 41000				Benchmark 4 Total: 2310			
2000			Eval	20000			Eval	40000			Eval	1855			Eval
✓	×	∞		✓	×	∞		✓	×	∞		✓	×	∞	
1997	0	3	1000	1293	0	18707	1000	1449	1137	37414	1000	1673	16	166	455