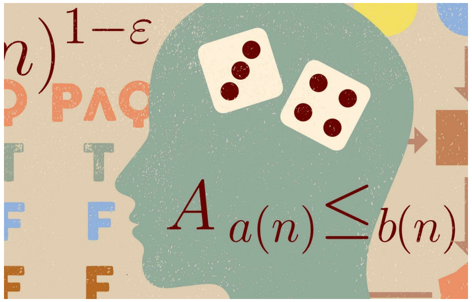
Efficient Counting in Boolean Circuits via Tree Automata

Marcelo Arenas
PUC & IMFD Chile and RelationalAI



Propositional logic: #DNF

Problem of counting the number of assignments that satisfy a propositional formula in DNF

- #DNF is #P-complete

#DNF admits a fully polynomial-time randomized approximation scheme (FPRAS)

Propositional logic: #DNF

There exists an algorithm $\mathcal{B}$ such that for every propositional formula φ and $\varepsilon \in (0,1)$:

$$\Pr\left((1-\varepsilon)\#\mathrm{DNF}(\varphi) \leq \mathcal{B}(\varphi,\varepsilon) \leq (1+\varepsilon)\#\mathrm{DNF}(\varphi)\right) \geq \frac{3}{4}$$

The number of steps to compute $\mathcal{B}(\varphi,\varepsilon)$ is bounded by $poly(|\varphi|, \frac{1}{\varepsilon})$

Propositional logic: #DNF

The existence of an FPRAS implies the existence of a randomized polynomial-time algorithm for (almost) uniform generation of solutions [JVV86]

Hence, we only focus on the problem of counting

Automata: #NFA

Problem of counting the number of strings of length n accepted by an NFA A

- #NFA is #P-complete since #DNF $\leq_{\text{par}}^{p}$ #NFA

#DNF $\leq^p_{\text{par}}$ #NFA

$$(p \wedge q \wedge \neg r) \vee (\neg p \wedge r \wedge s) \vee (q \wedge t \wedge \neg u)$$

#DNF $\leq^{p}_{\text{par}}$ #NFA

$$(p \land q \land \neg r) \lor (\neg p \land r \land s) \lor (q \land t \land \neg u)$$

p q r s t u

#DNF $\leq^{p}_{\text{par}}$ #NFA

$$(p \wedge q \wedge \neg r) \vee (\neg p \wedge r \wedge s) \vee (q \wedge t \wedge \neg u)$$

p q r s t u

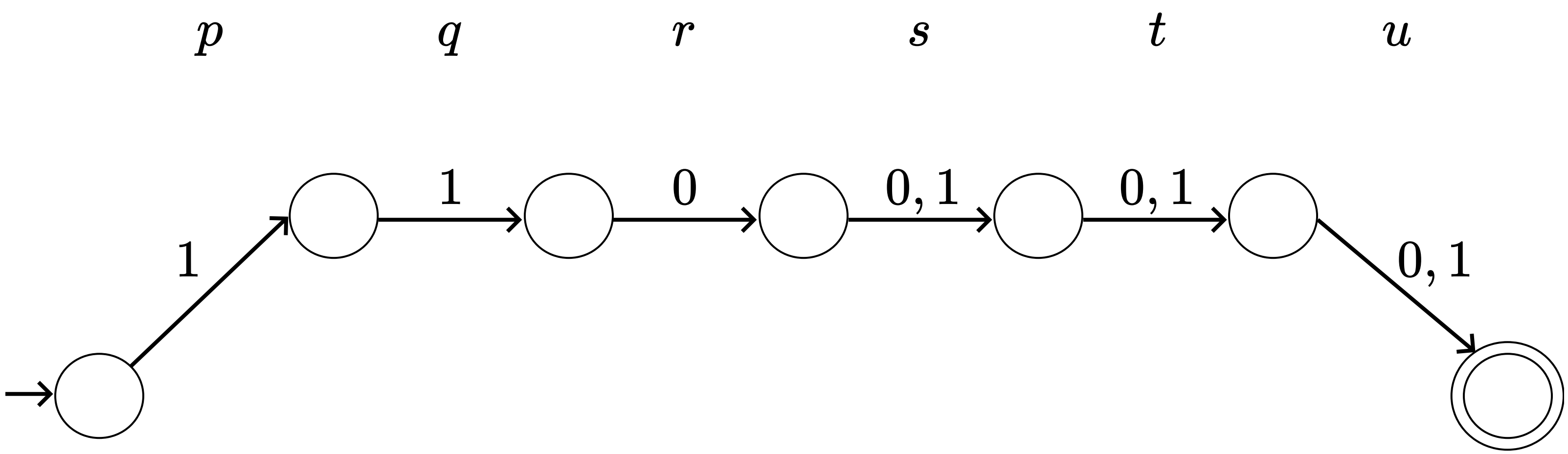
#DNF $\leq^{p}_{\text{par}}$ #NFA

$$(p \wedge q \wedge \neg r) \vee (\neg p \wedge r \wedge s) \vee (q \wedge t \wedge \neg u)$$

p q r s t u

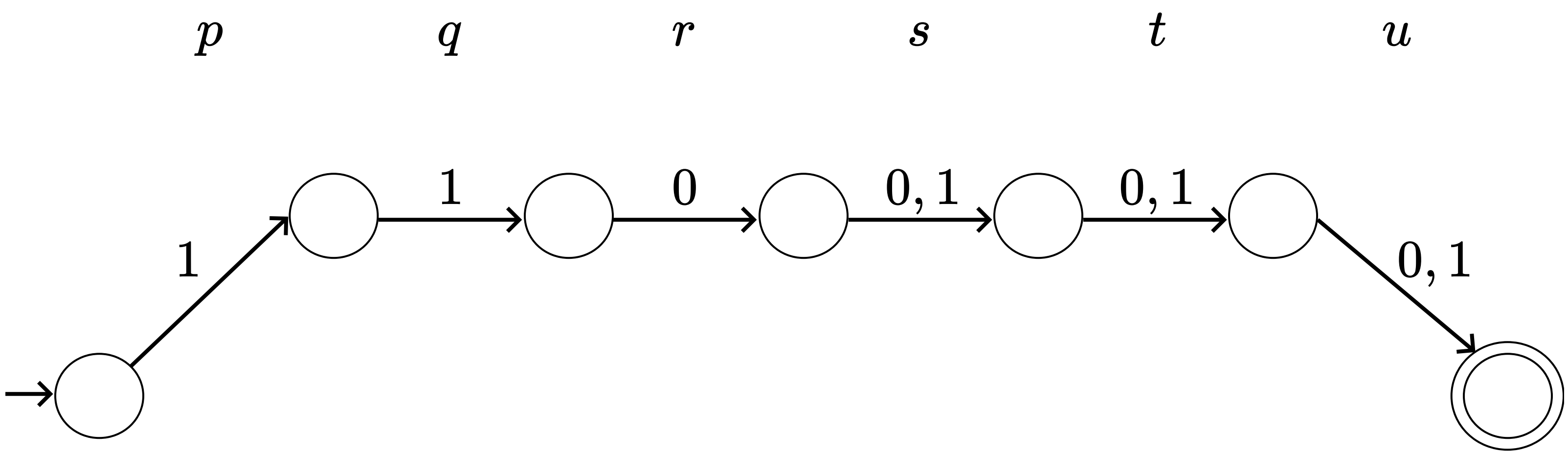
#DNF $\leq^p_{\text{par}}$ #NFA

$$(p \land q \land \neg r) \lor (\neg p \land r \land s) \lor (q \land t \land \neg u)$$



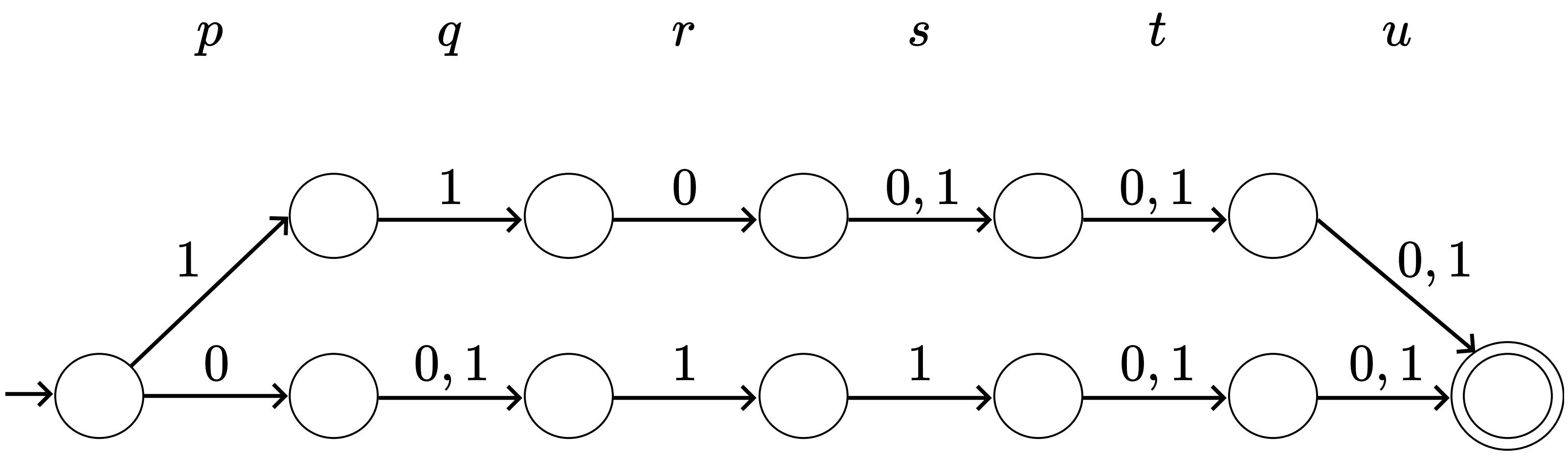
#DNF $\leq^p_{\text{par}}$ #NFA

$$(p \land q \land \neg r) \lor (\neg p \land r \land s) \lor (q \land t \land \neg u)$$



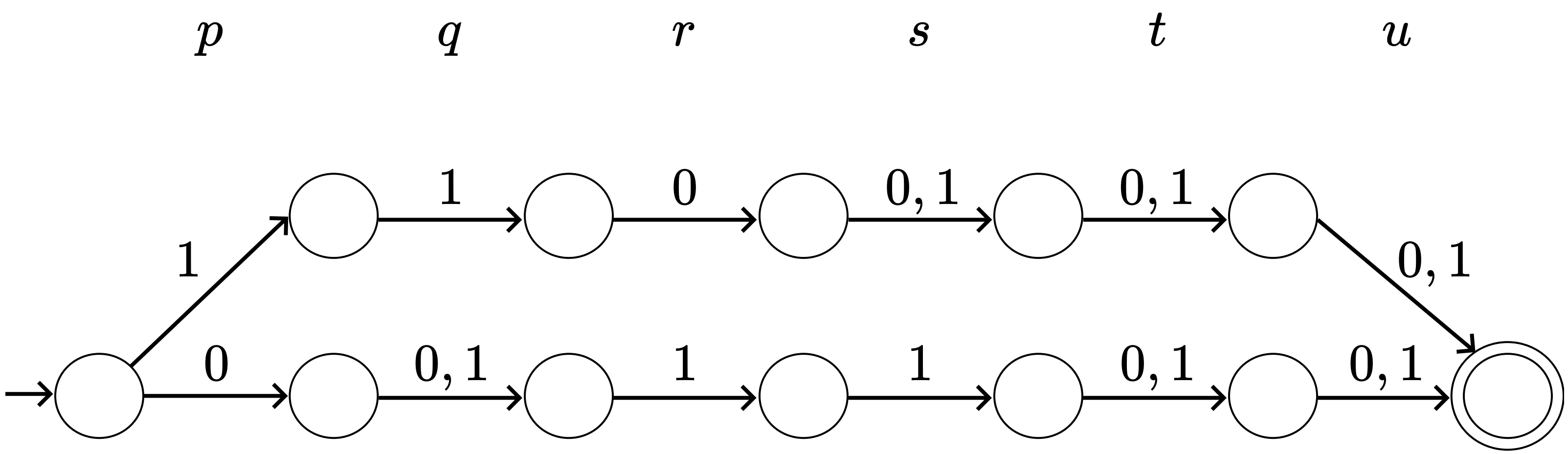
#DNF $\leq^{p}_{\text{par}}$ #NFA

$$(p \land q \land \neg r) \lor (\neg p \land r \land s) \lor (q \land t \land \neg u)$$



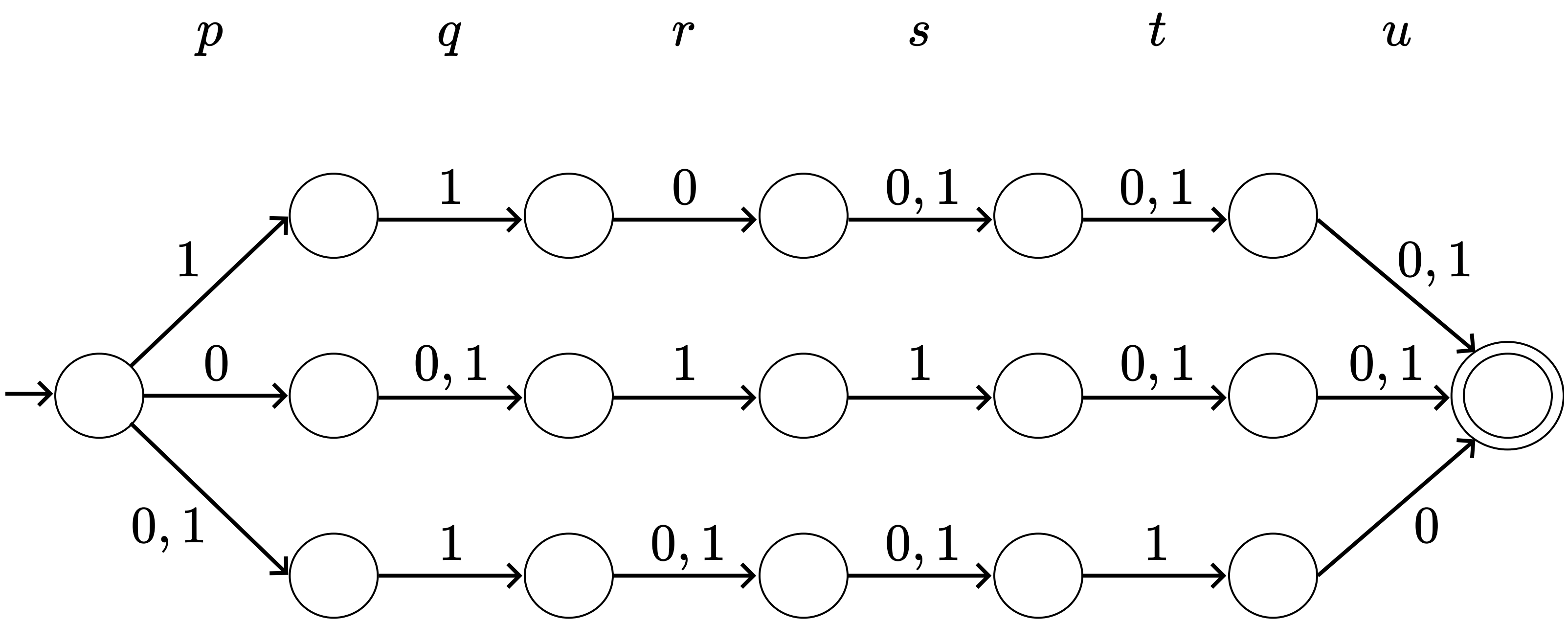
#DNF $\leq^p_{\text{par}}$ #NFA

$$(p \wedge q \wedge \neg r) \vee (\neg p \wedge r \wedge s) \vee (q \wedge t \wedge \neg u)$$



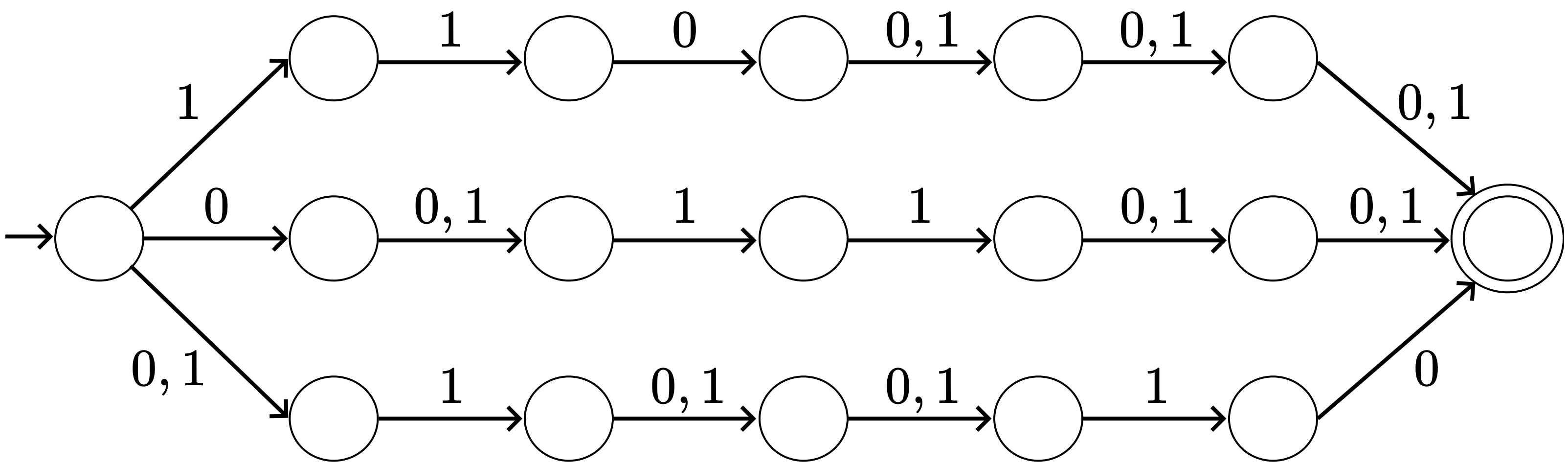
#DNF $\leq^p_{\text{par}}$ #NFA

$$(p \wedge q \wedge \neg r) \vee (\neg p \wedge r \wedge s) \vee (q \wedge t \wedge \neg u)$$



#DNF $\leq^p_{\text{par}}$ #NFA

$$(p \wedge q \wedge \neg r) \vee (\neg p \wedge r \wedge s) \vee (q \wedge t \wedge \neg u)$$



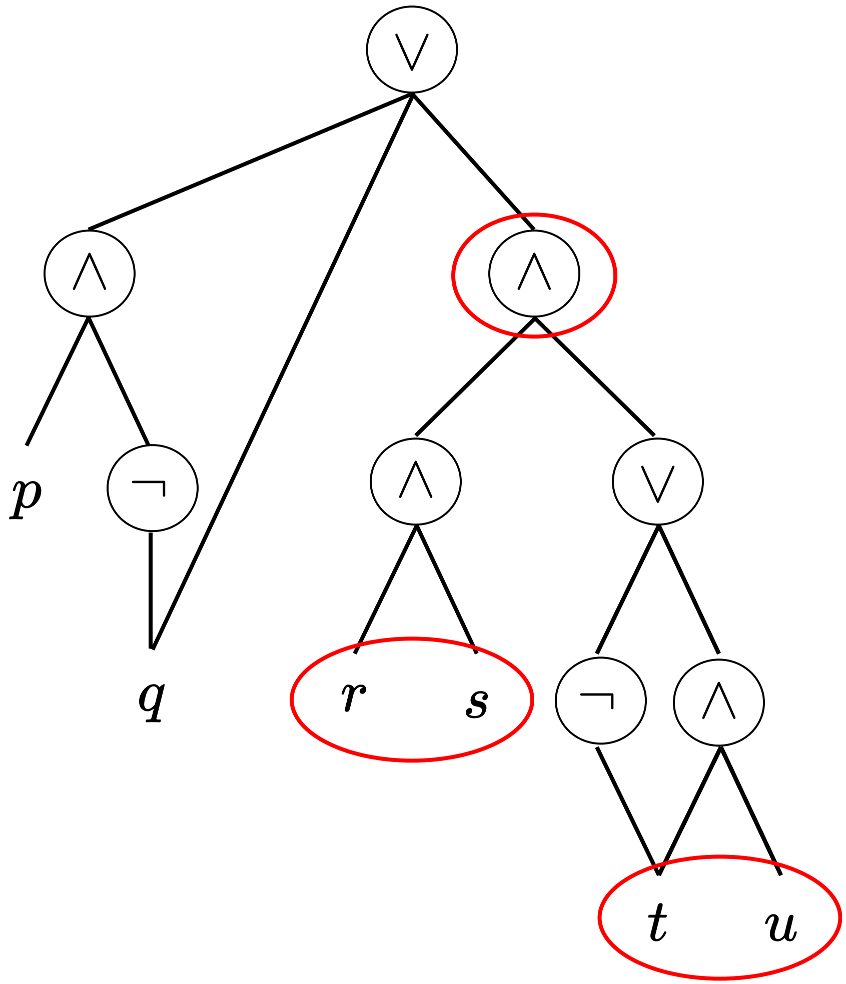
Approximating #NFA

Theorem [ACJR21a]: #NFA admits an FPRAS

From the previous reduction it is possible to obtain that #DNF admits an FPRAS

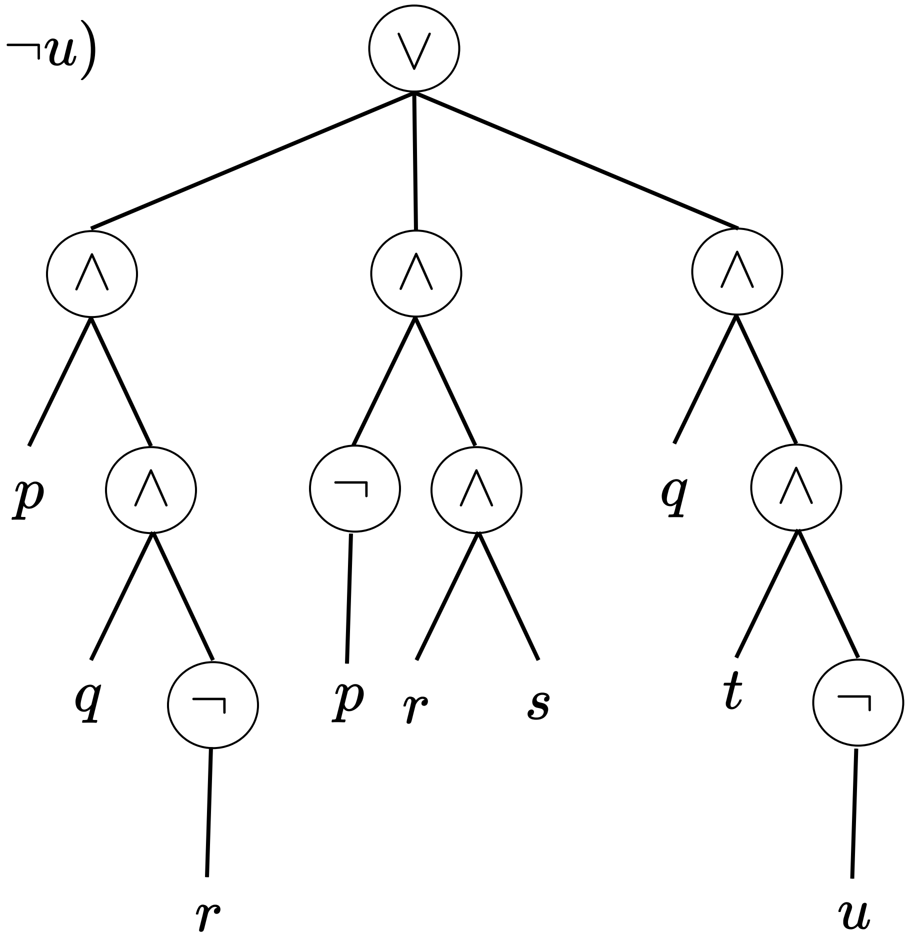
But this is a well-known result, can we obtain more?

Circuits: decomposable NNF (DNNF)



DNF and DNNF

$$(p \wedge q \wedge \neg r) \vee (\neg p \wedge r \wedge s) \vee (q \wedge t \wedge \neg u)$$



Can #DNNF be efficiently approximated by using #NFA?

#DNNF is #P-complete. An FPRAS for #DNNF can be obtained by proving that #DNNF $\leq^{p}_{\mathrm{par}}$ #NFA

- Or by considering another form of approximating preserving reduction

But it is not clear how to prove that #DNNF $\leq^{p}_{\mathrm{par}}$ #NFA

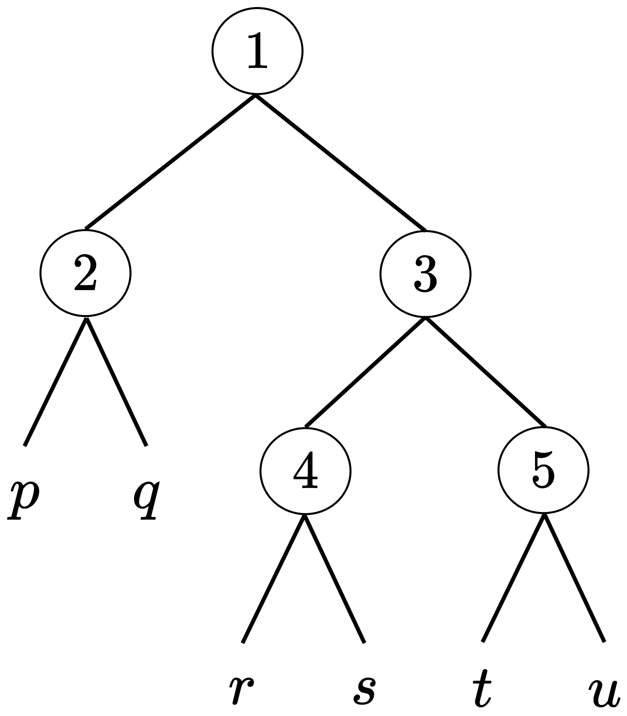
- Notice that DNNF is exponentially more succinct than DNF

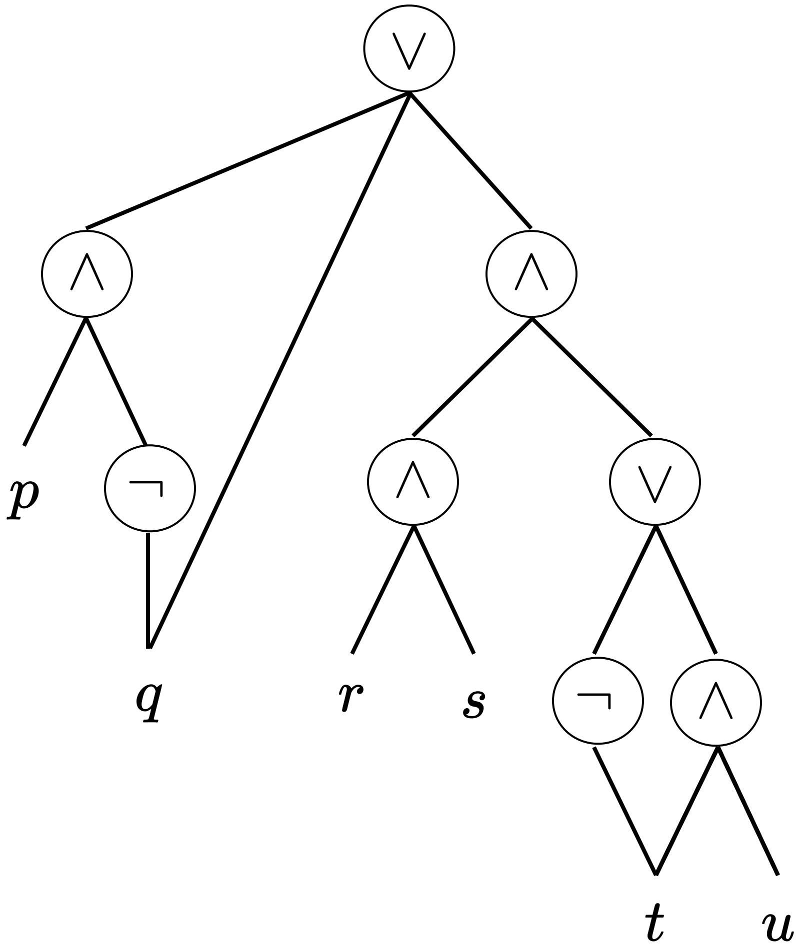
The goal of this talk

To show how automata can be used to prove that #DNNF admits an FPRAS for a natural and widely used fragment of DNNF

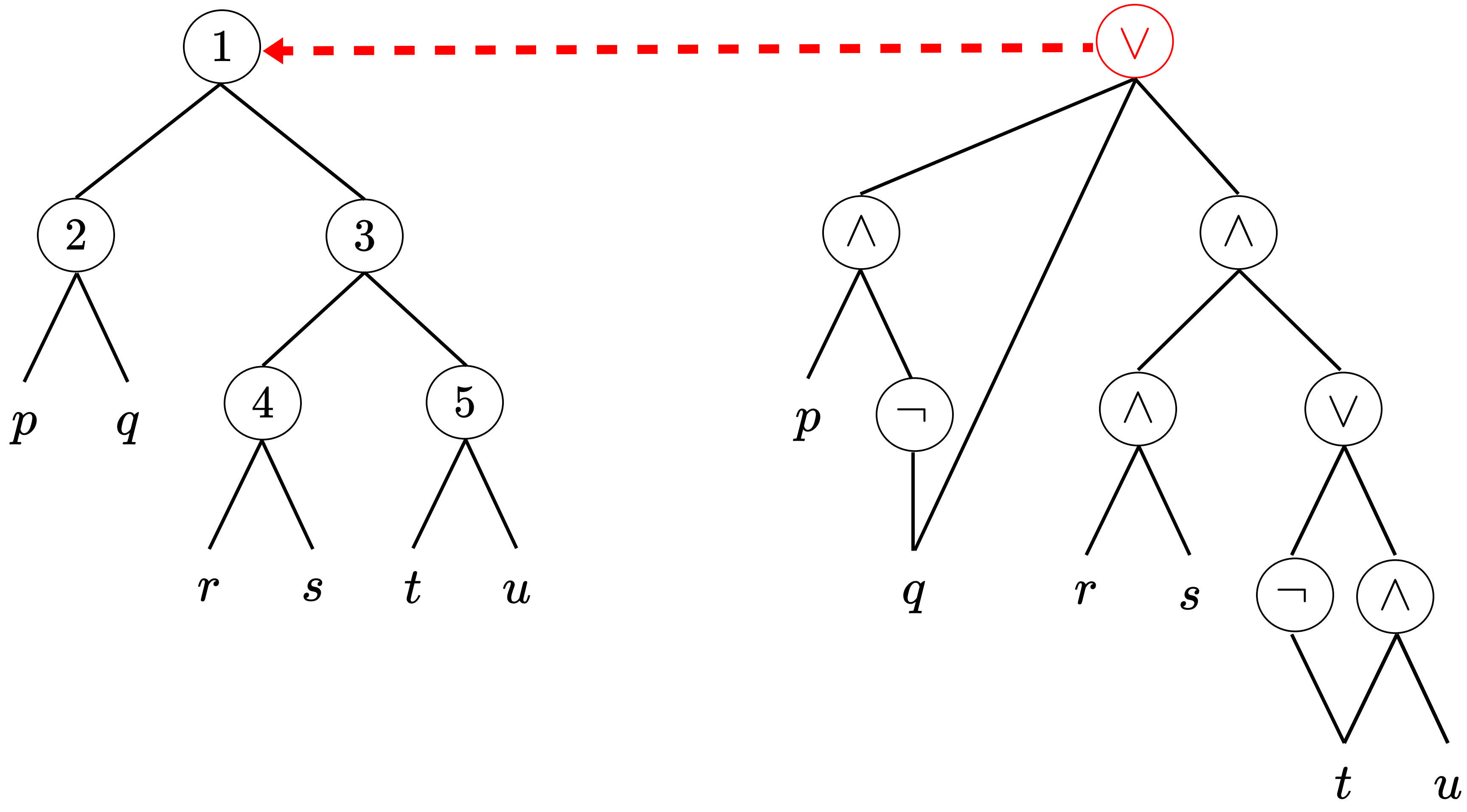
We focus on **structured DNNF**, and we consider the more powerful model of **tree automata**

SDNNF: structured DNNF

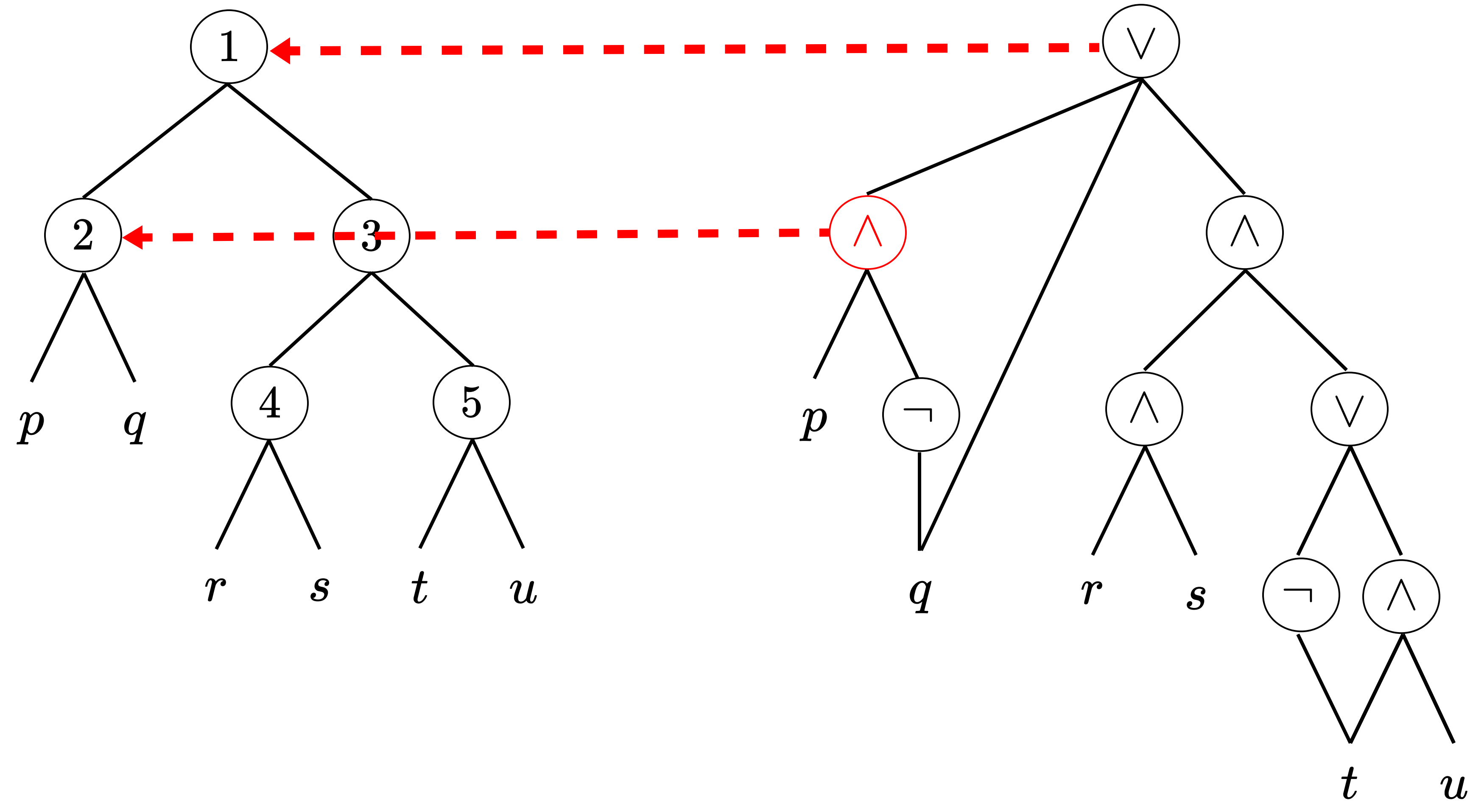




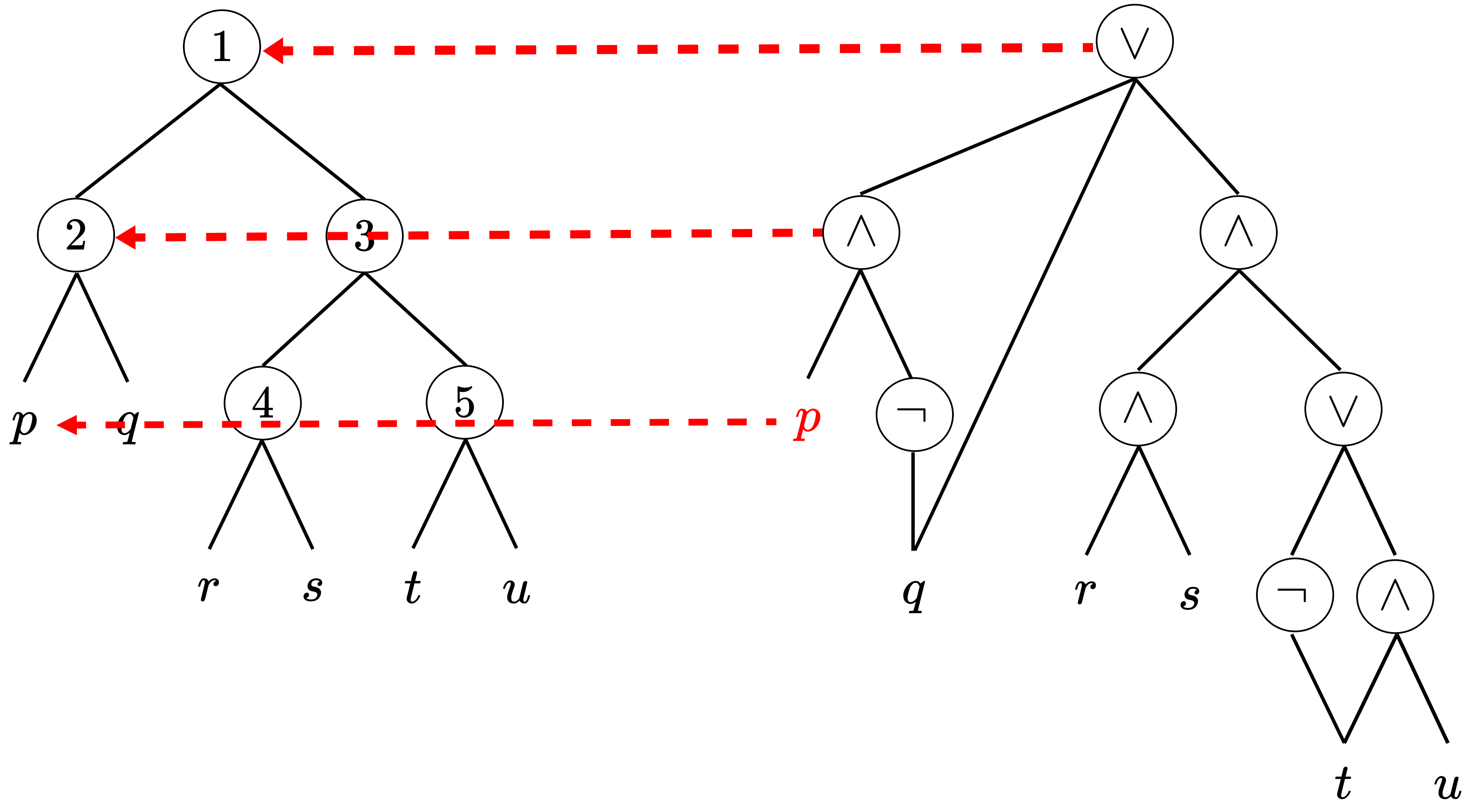
SDNNF: structured DNNF



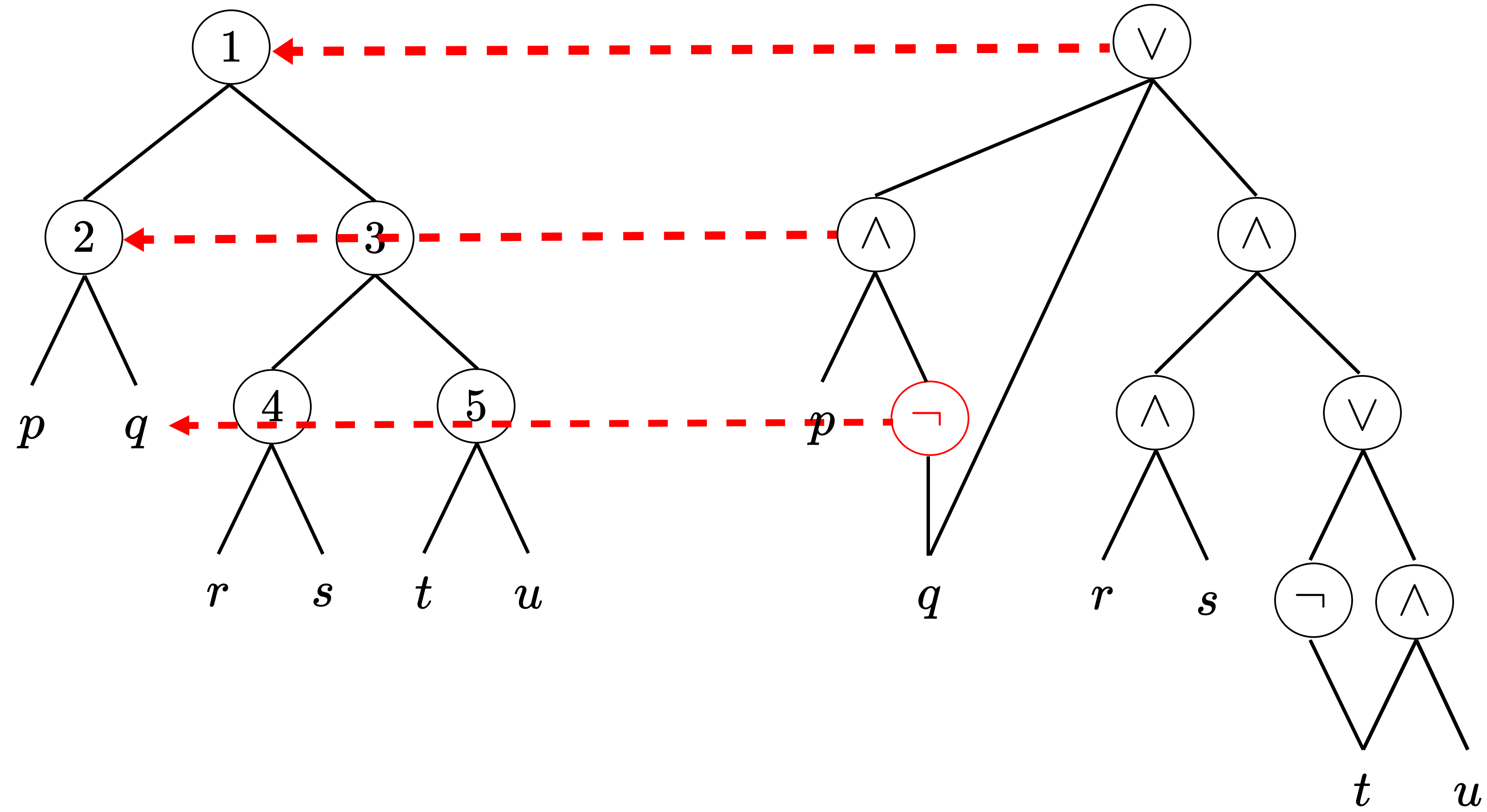
SDNNF: structured DNNF



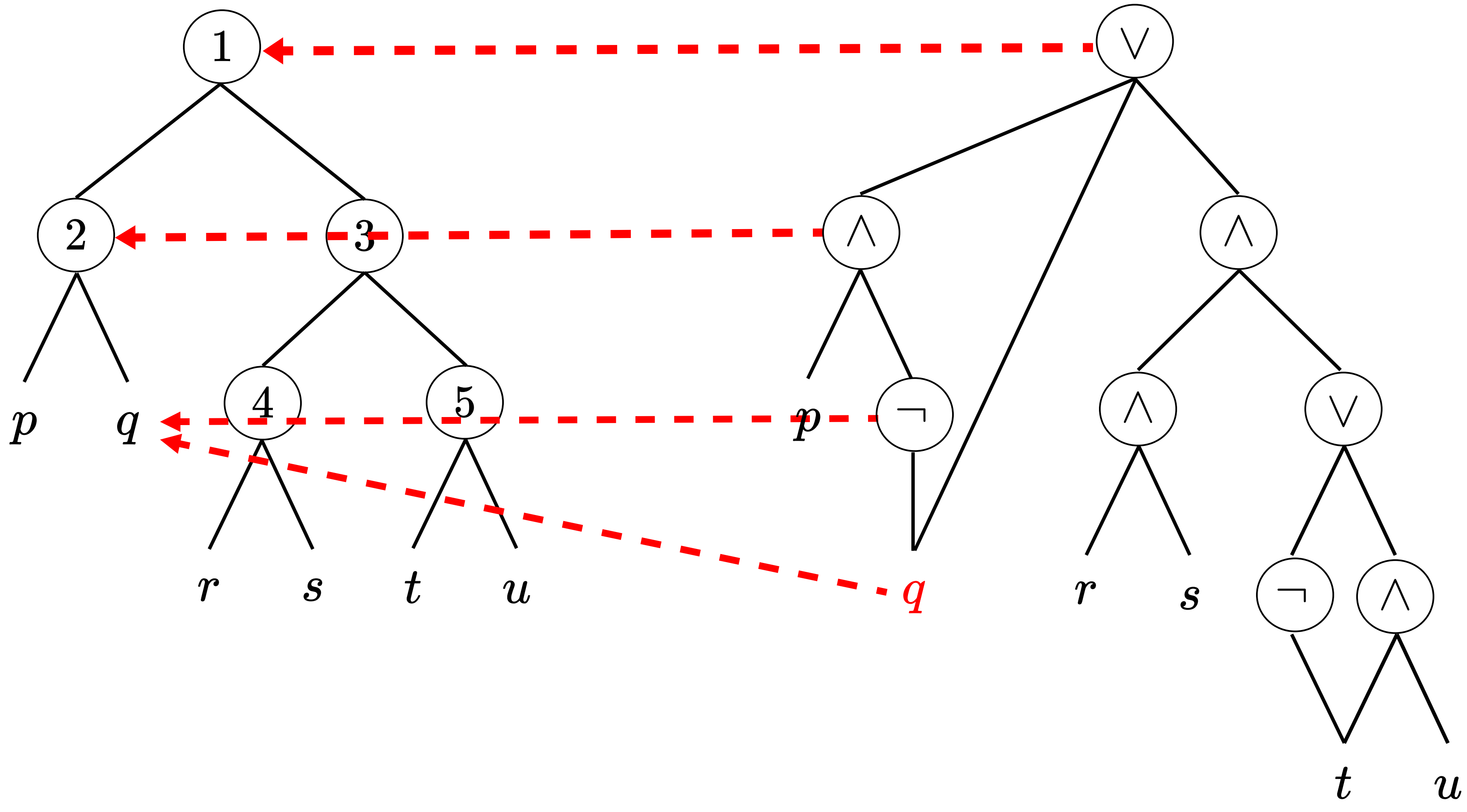
SDNNF: structured DNNF



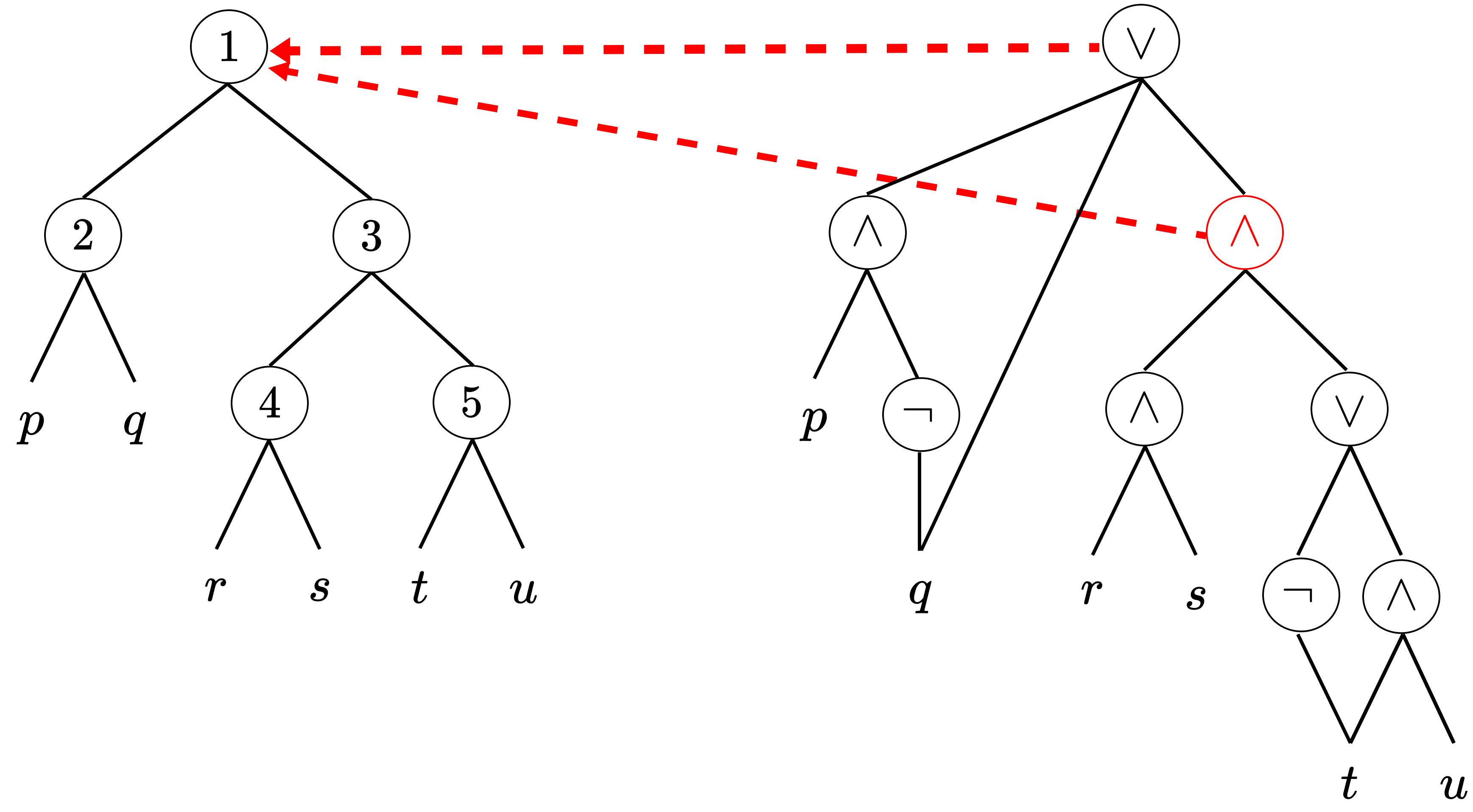
SDNNF: structured DNNF



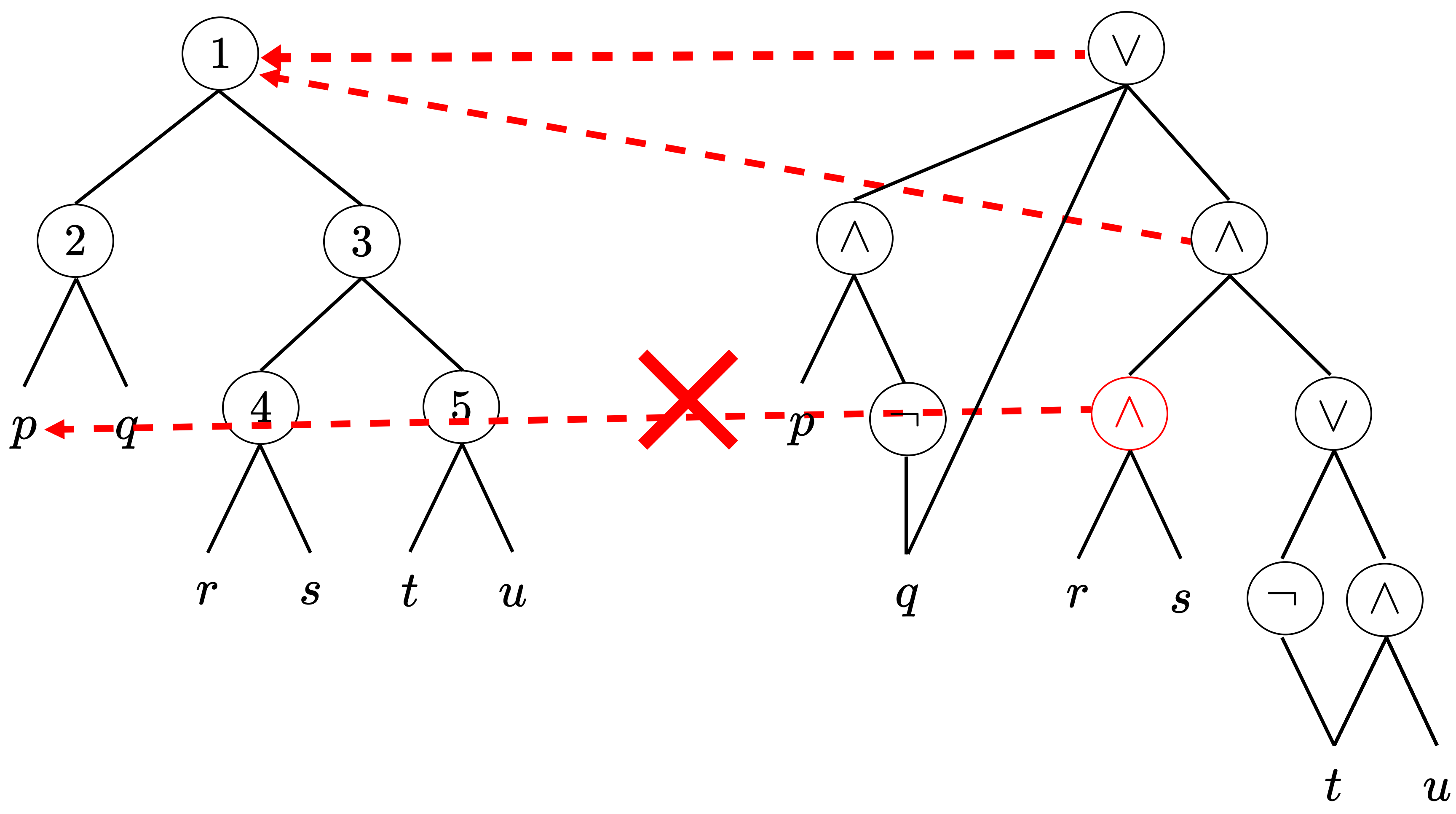
SDNNF: structured DNNF



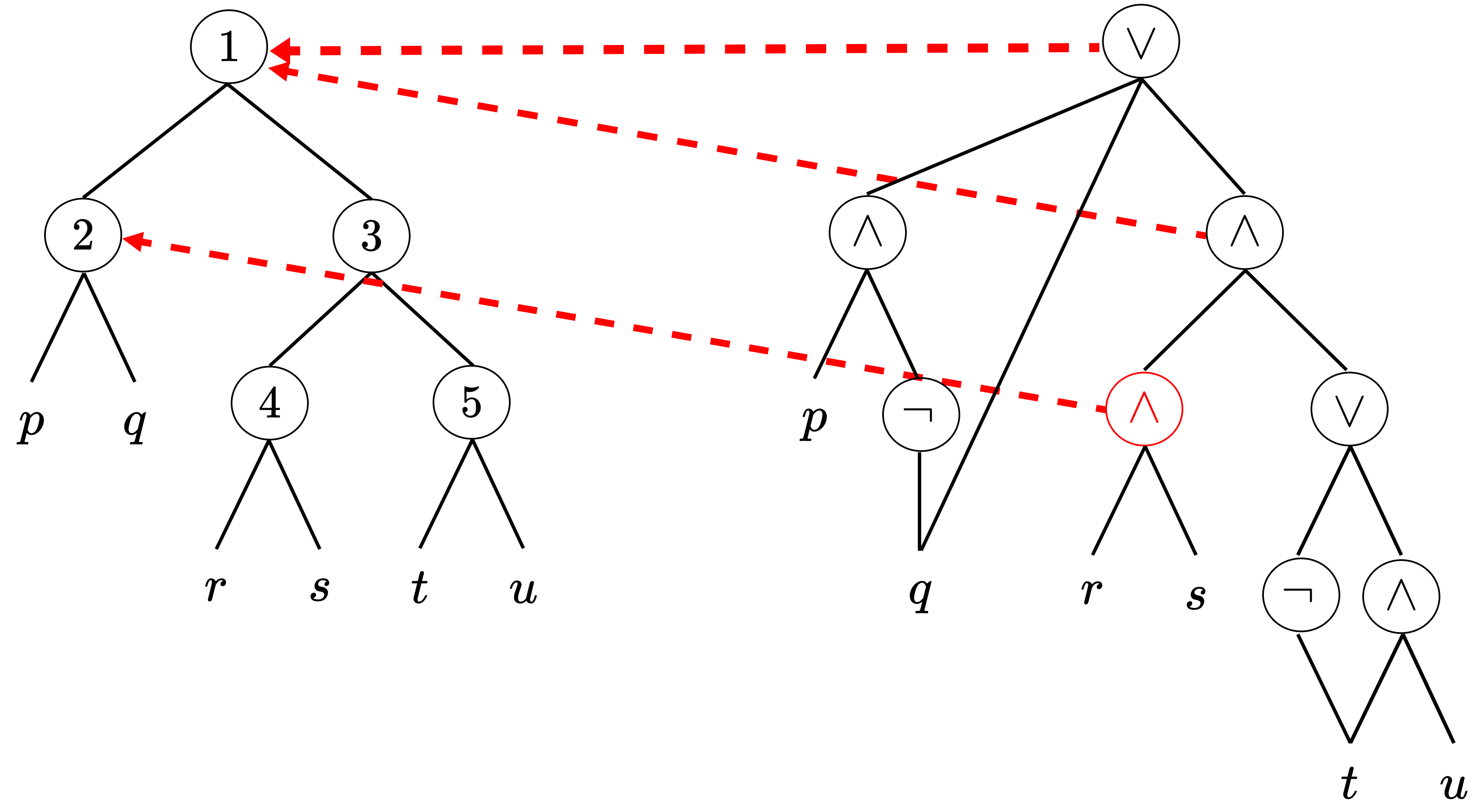
SDNNF: structured DNNF



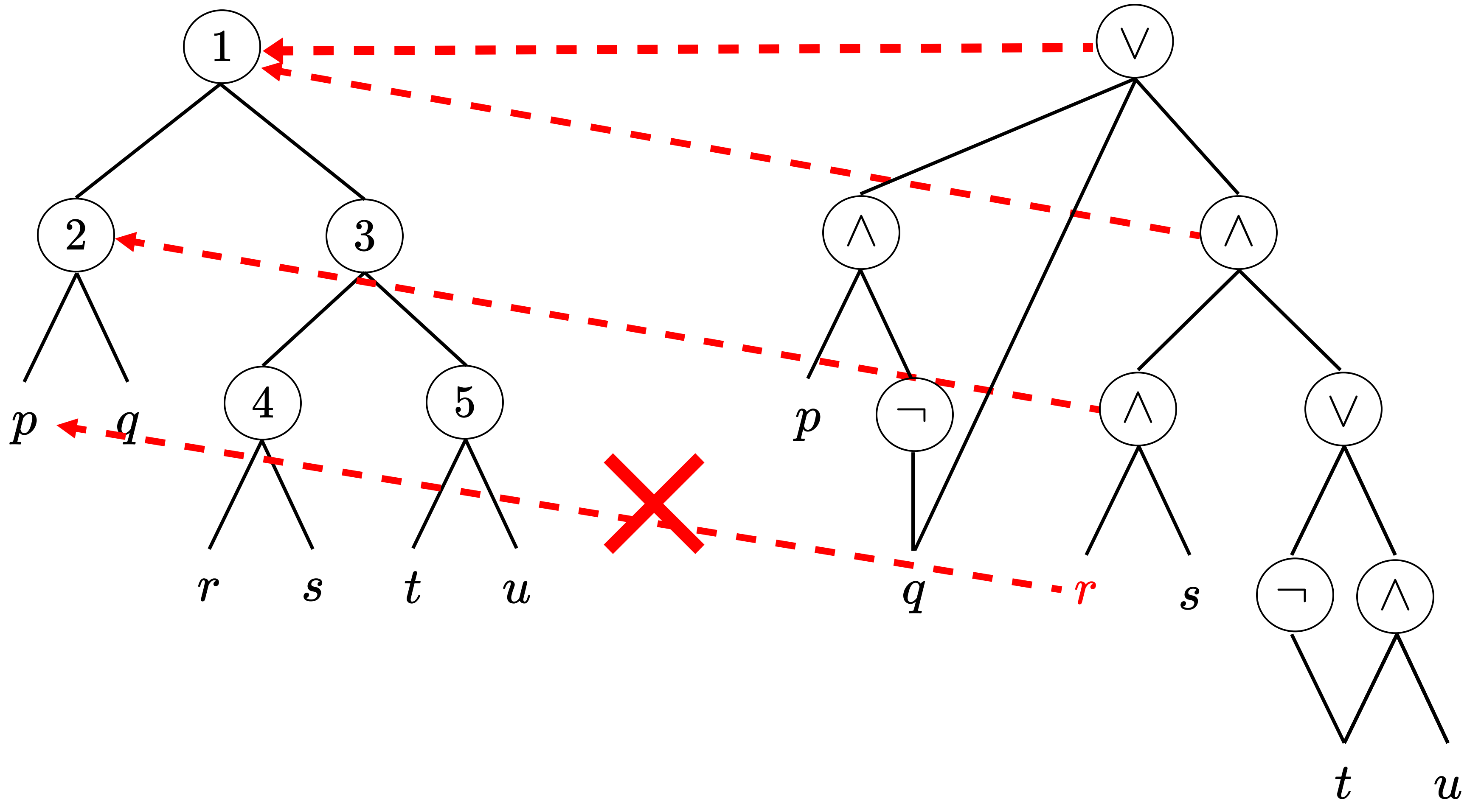
SDNNF: structured DNNF



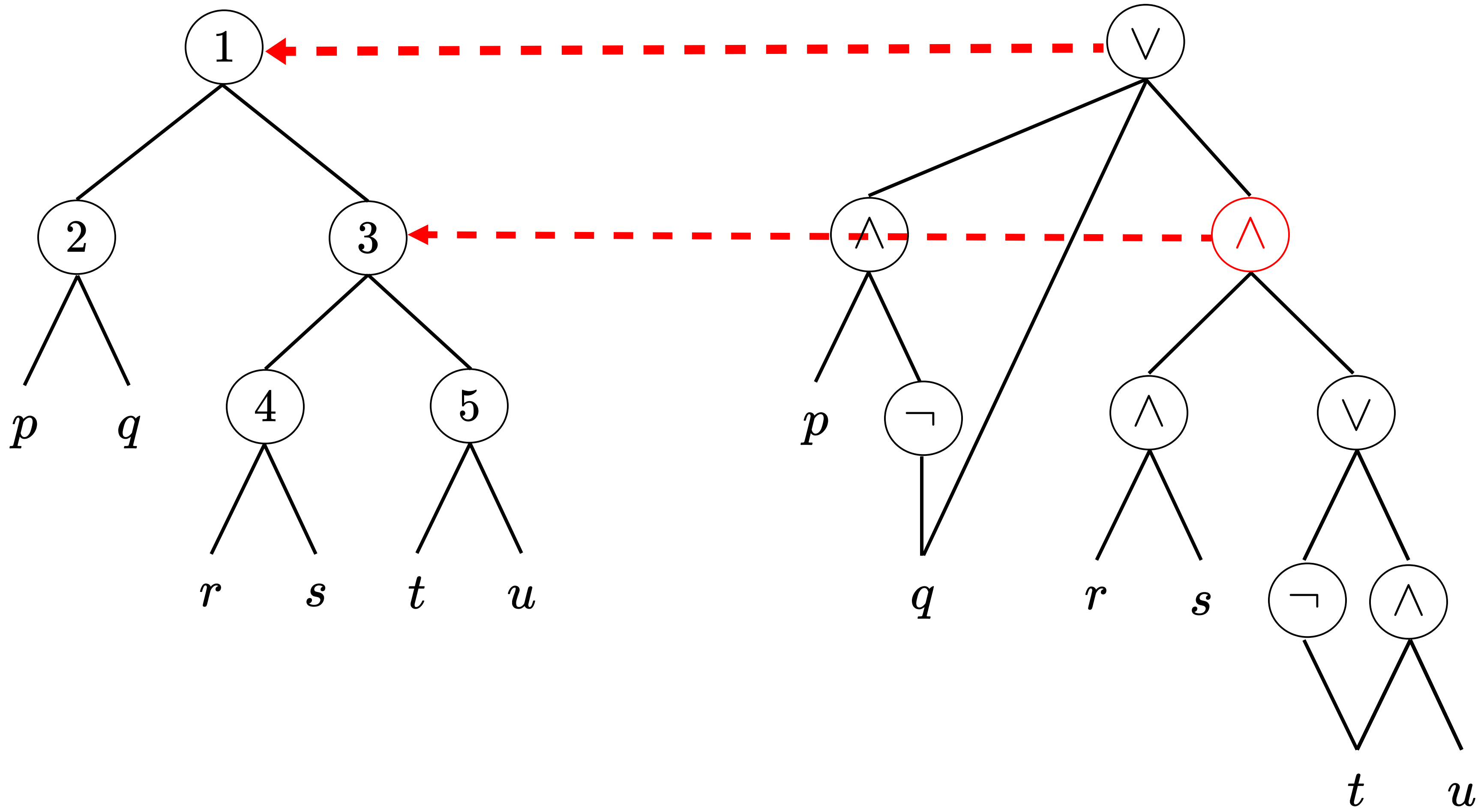
SDNNF: structured DNNF



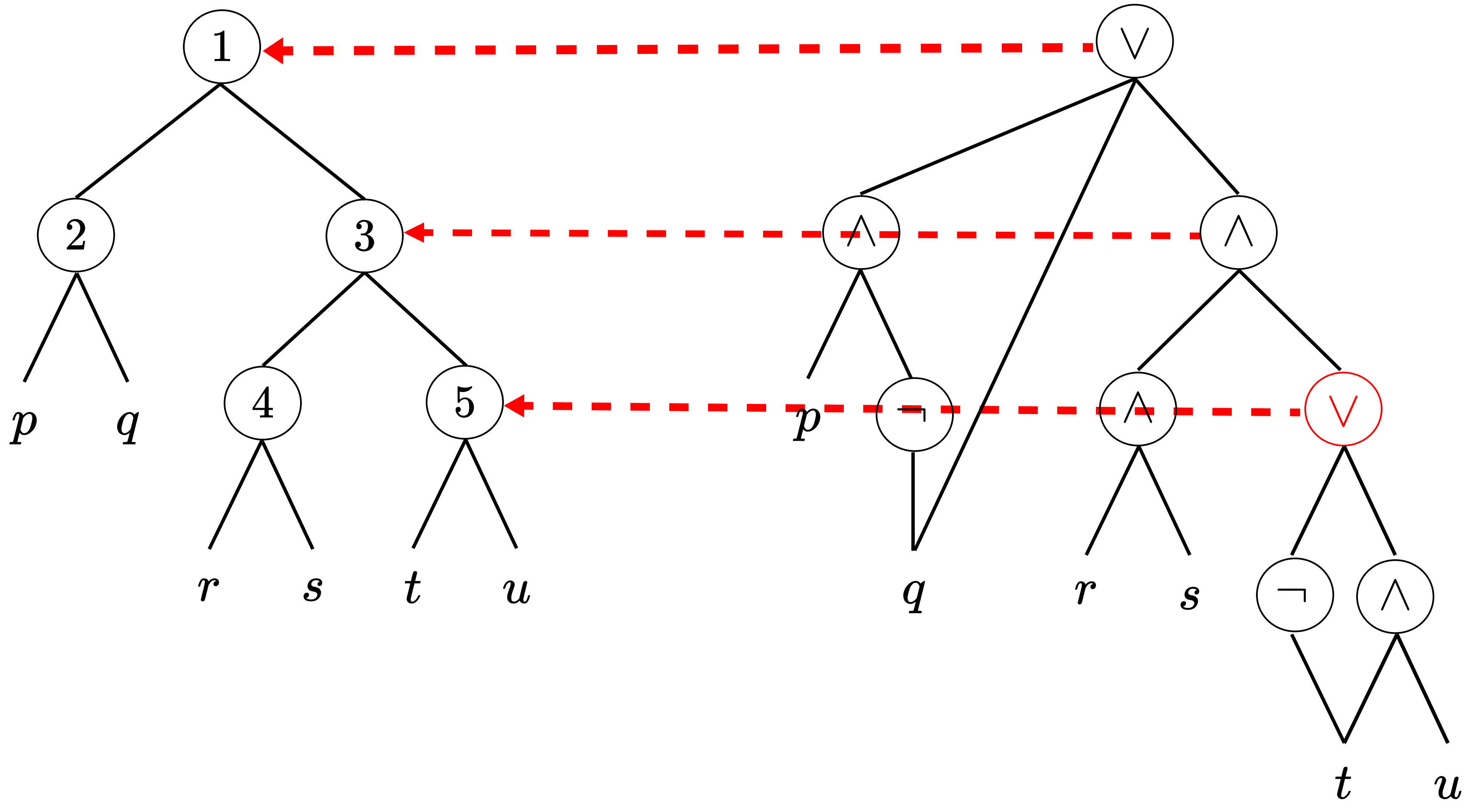
SDNNF: structured DNNF



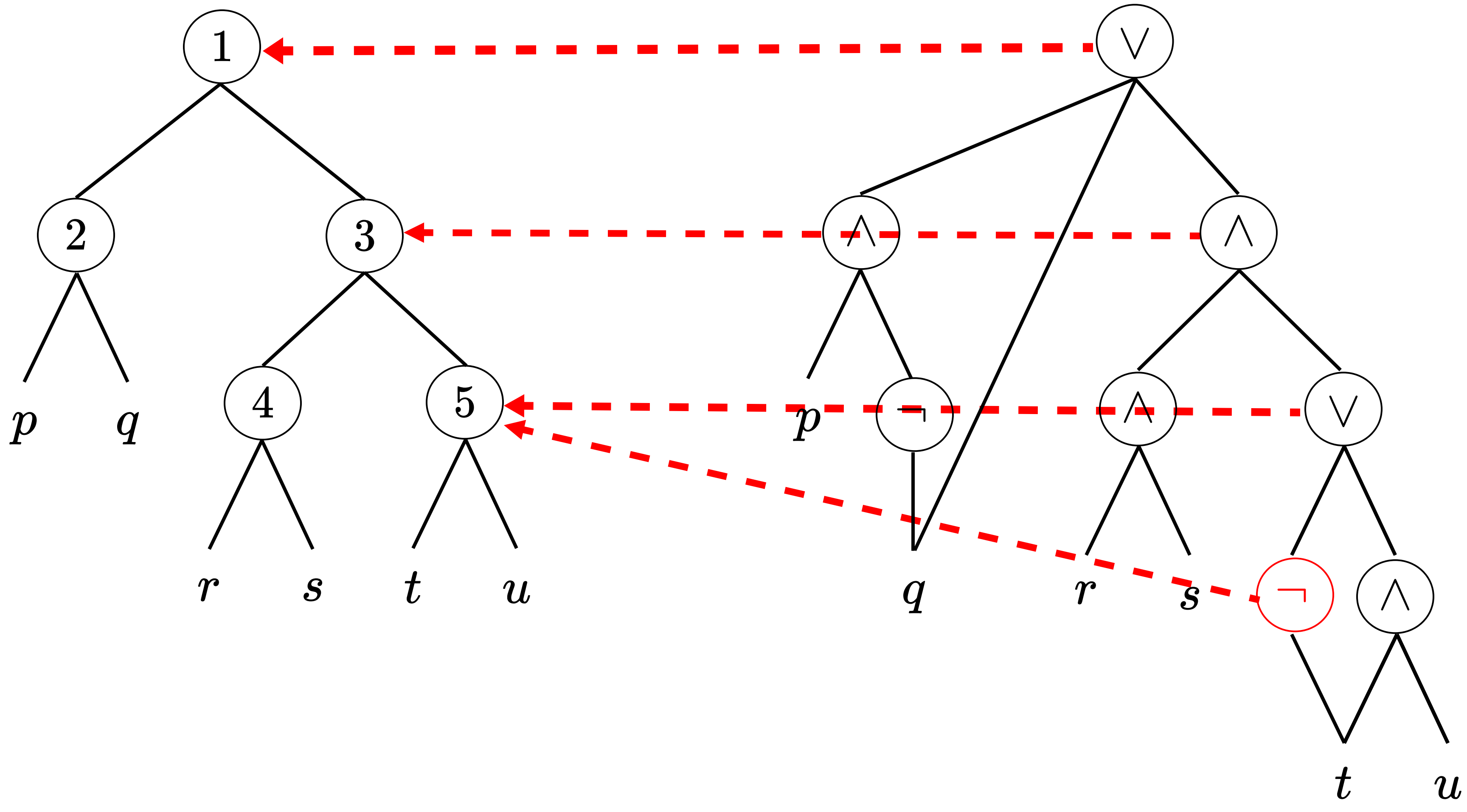
SDNNF: structured DNNF



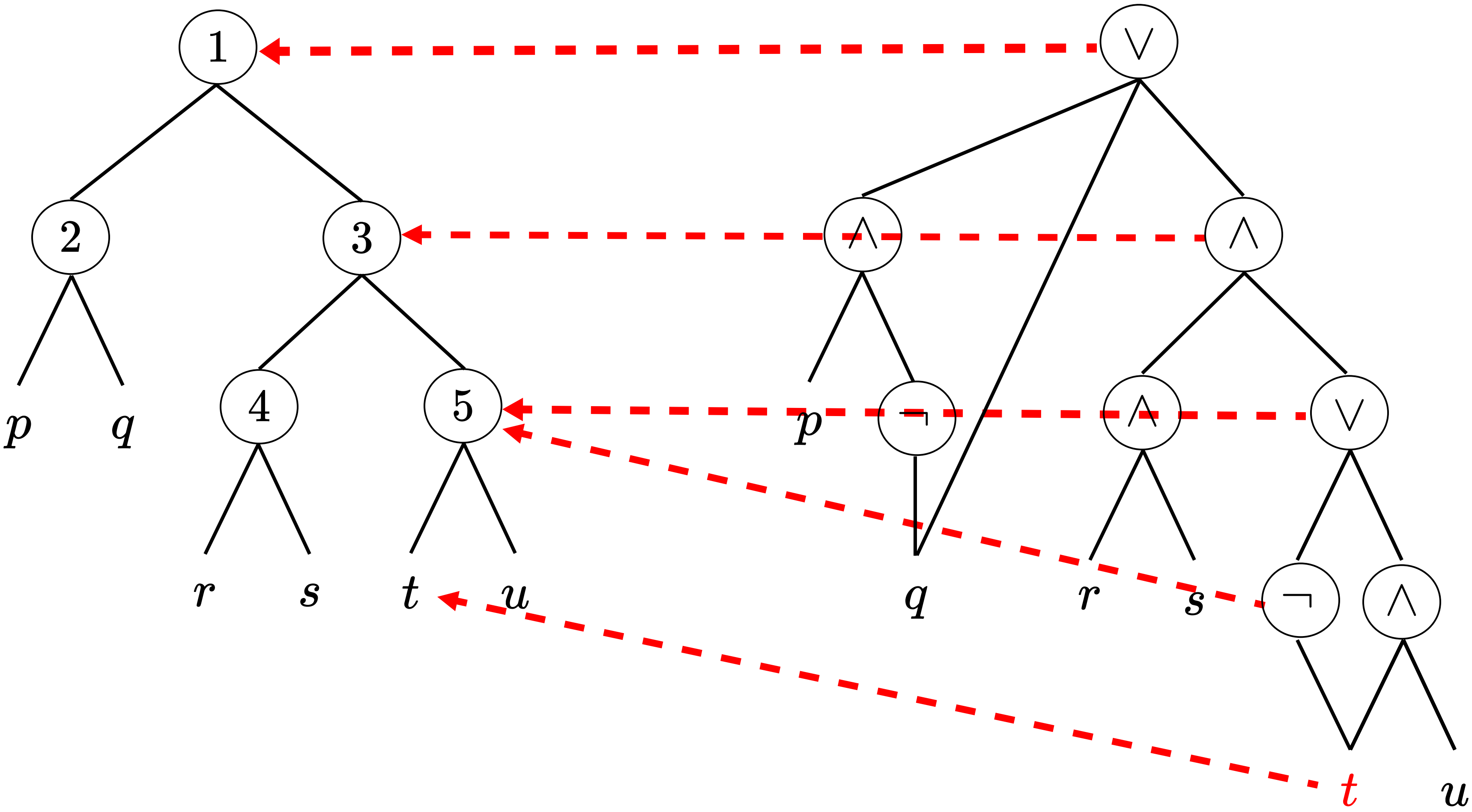
SDNNF: structured DNNF



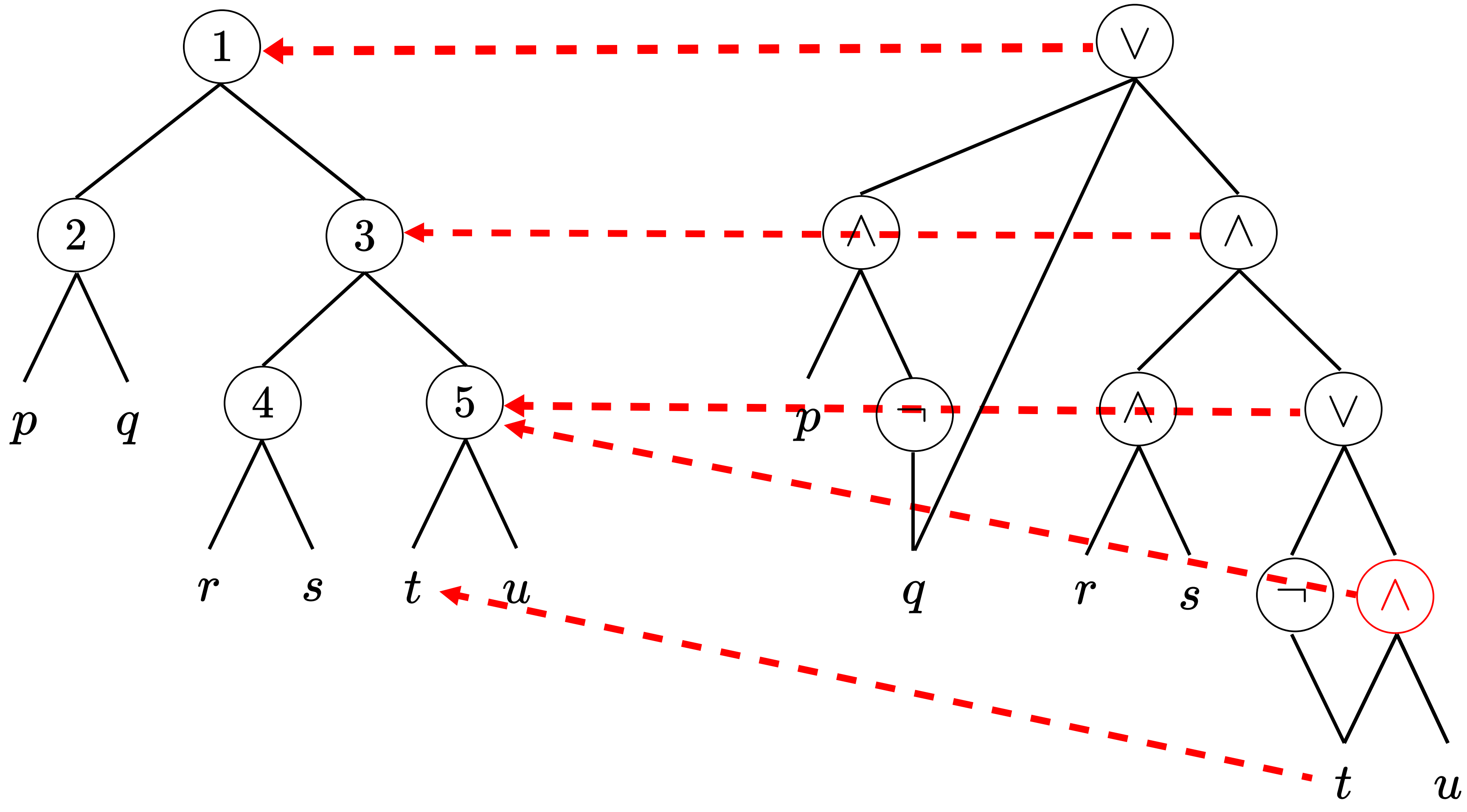
SDNNF: structured DNNF



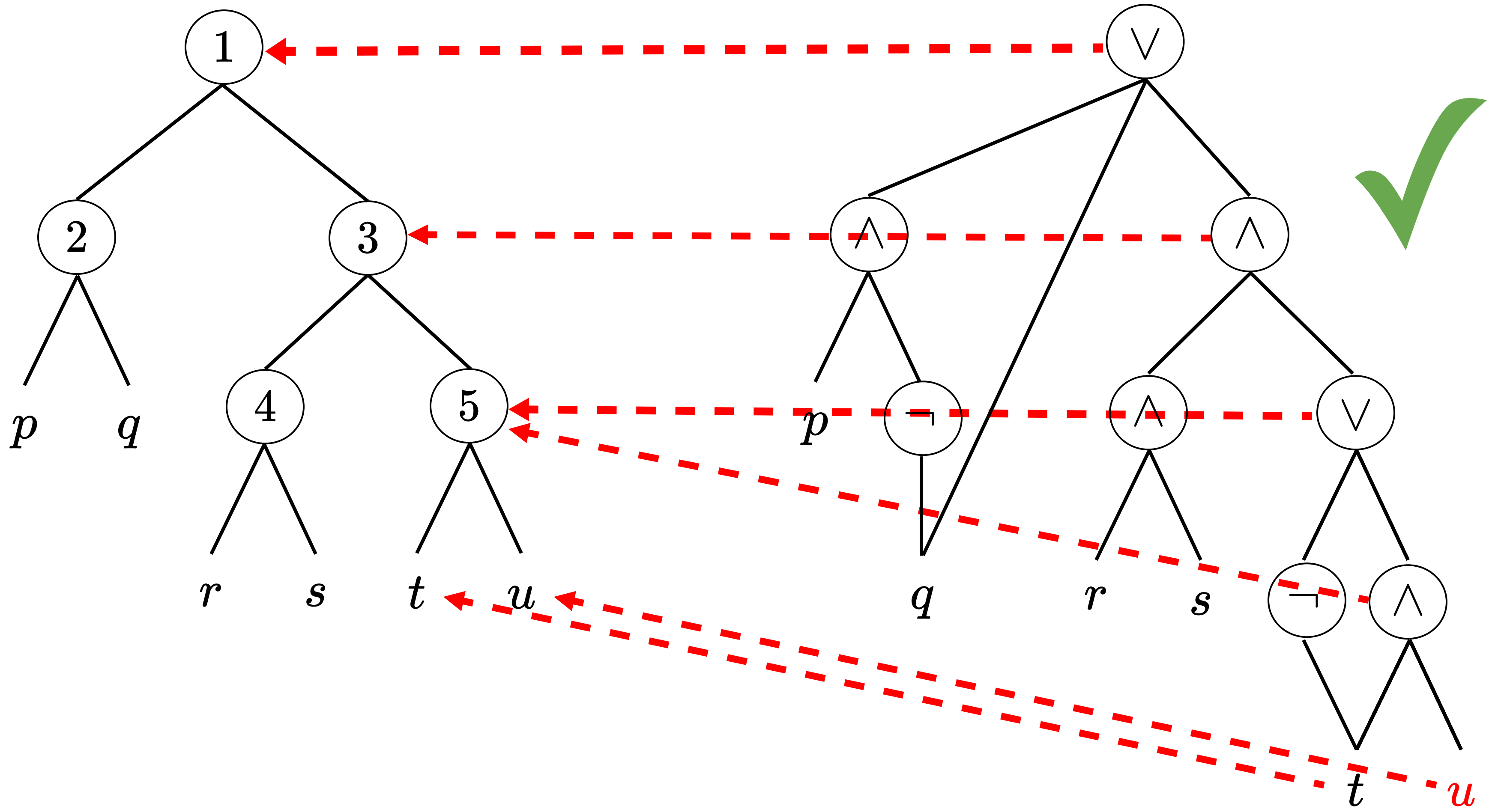
SDNNF: structured DNNF



SDNNF: structured DNNF



SDNNF: structured DNNF



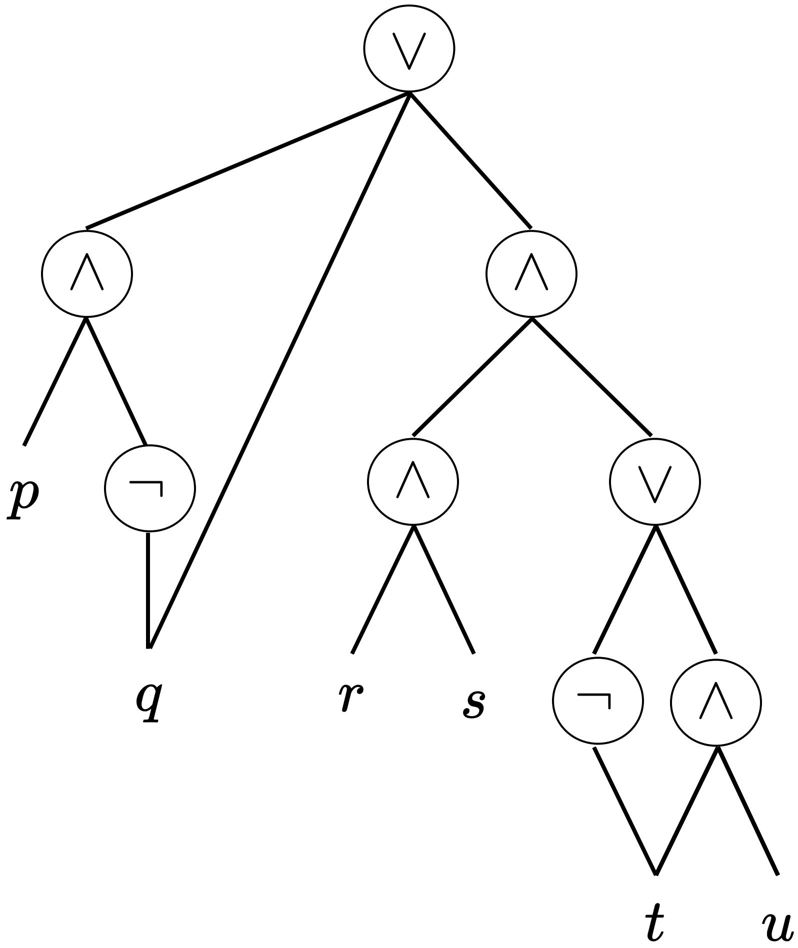
#SDNNF

Problem of counting the number of satisfying assignments of a SDNNF circuit

#SDNNF is #P-complete

- #DNF $\leq^p_{\text{par}}$ #SDNNF

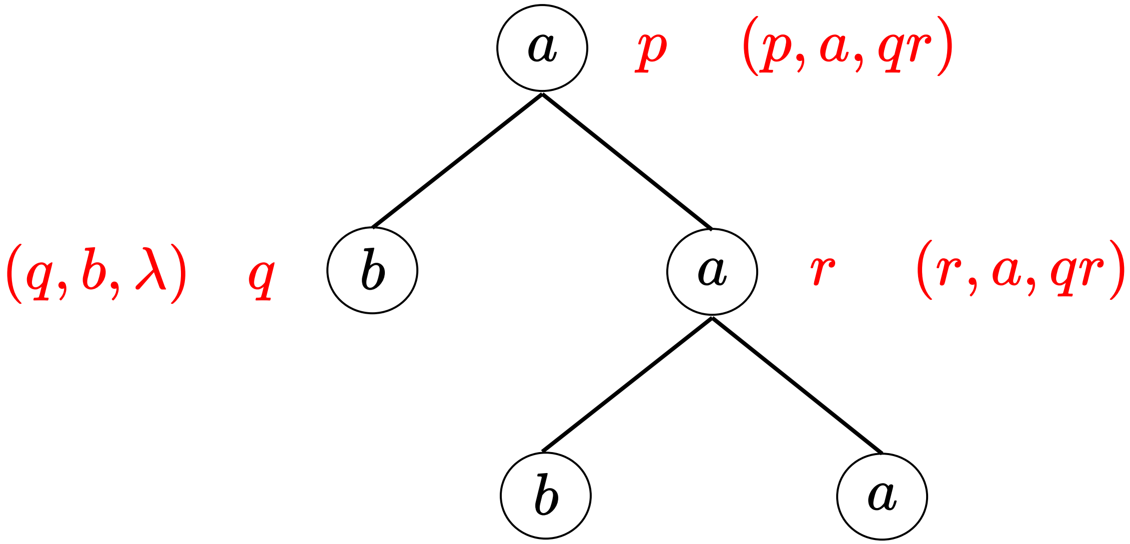
Our goal here: to show that #SDNNF admits an FPRAS



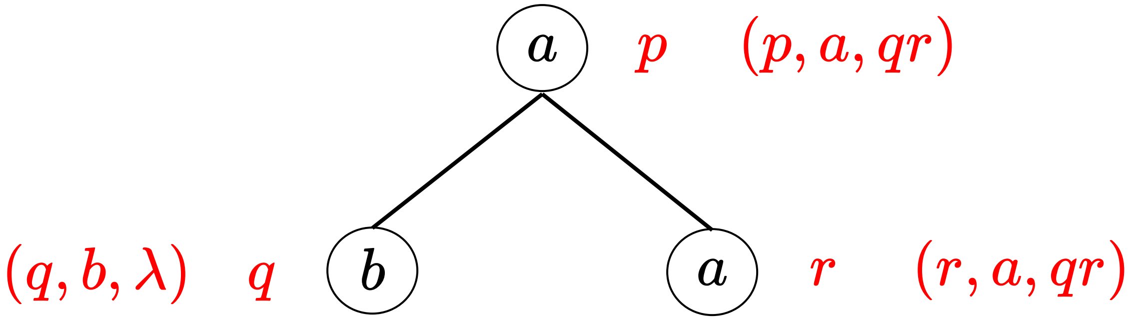
The main ingredient in the solution: Tree automata

This is the right representation for the problem of counting the number of assignments satisfying a structured DNNF circuit

Tree automata (TA)



Tree automata (TA)



Top-down tree automata: (Q, Σ, Δ, I)

- $Q = \{p, q, r\}$ is the set of states
- $\Sigma = \{a, b\}$ is the alphabet
- $I = \{p\}$ is the set of initial states
- $\Delta = \{(p, a, qr), (q, b, \lambda), (r, a, qr)\}$ is the transition relation

Tree automata: parity

We would like to check whether a tree labeled with $\{a, b\}$ has an even number of nodes with label a

Tree automata: (Q, Σ, Δ, I)

- $Q = \{e, o\}$
- $\Sigma = \{a, b\}$
- $I = \{e\}$
- $\Delta = \{(e, a, eo), (e, a, oe), (e, b, ee), (e, b, oo), \dots,$

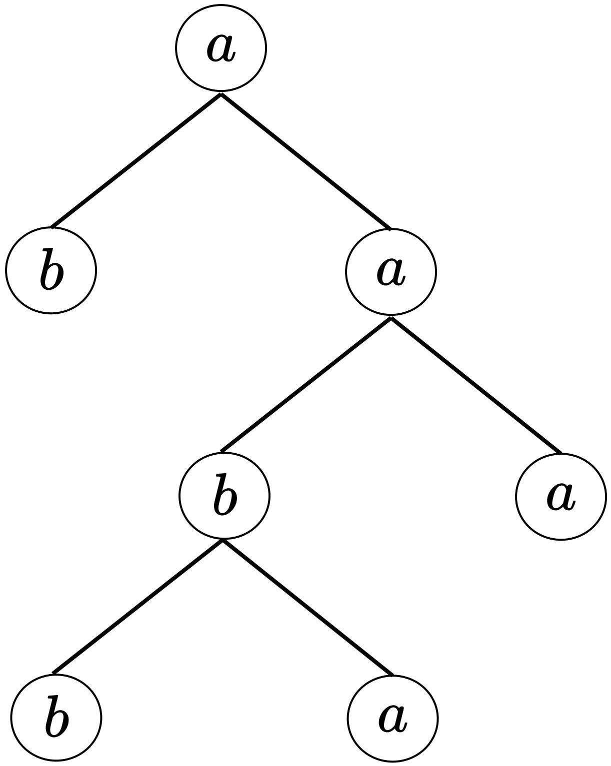
Tree automata: parity

We would like to check whether a tree labeled with $\{a, b\}$ has an even number of nodes with label a

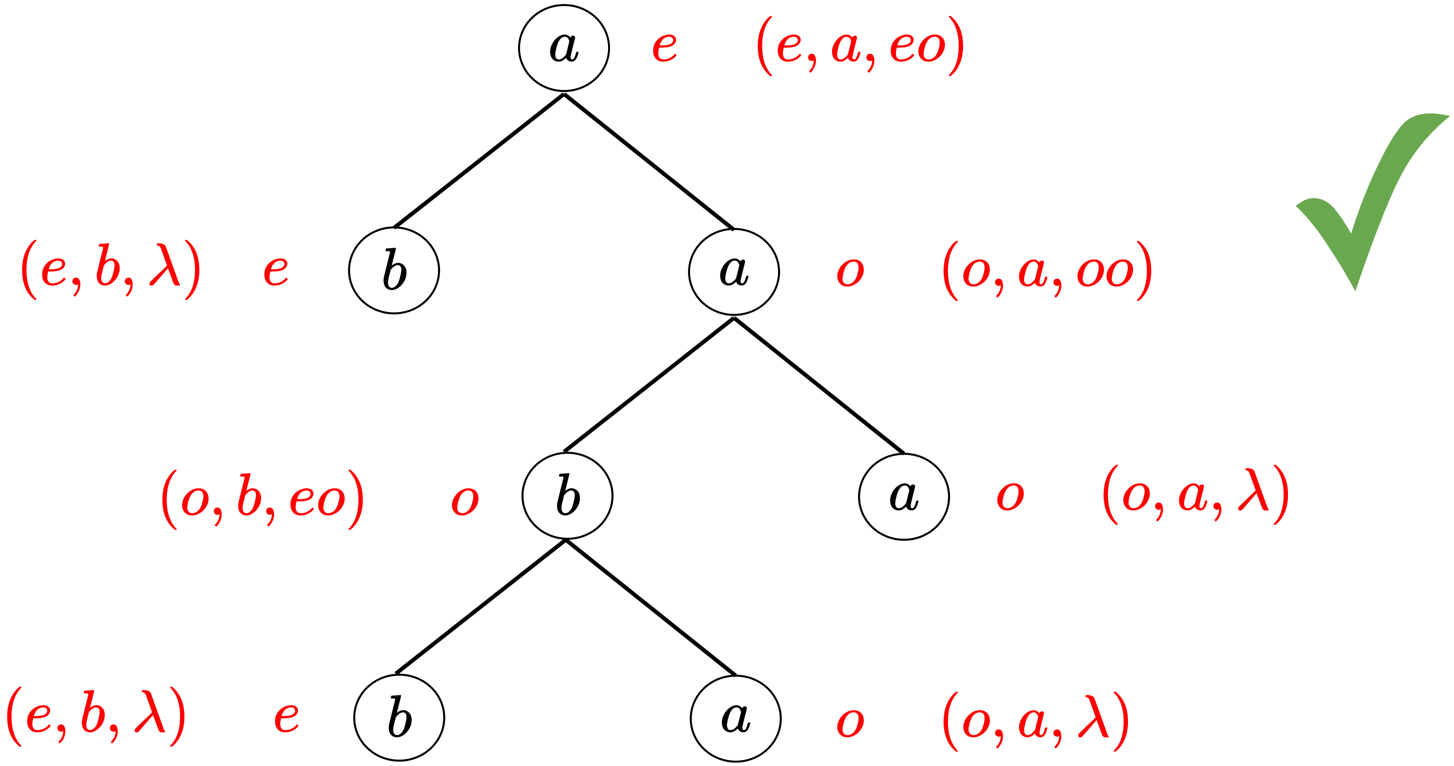
Tree automata: (Q, Σ, Δ, I)

- $Q = \{e, o\}$
- $\Sigma = \{a, b\}$
- $I = \{e\}$
- $\Delta = \{(e, a, eo), (e, a, oe), (e, b, ee), (e, b, oo), \ldots, (e, b, \lambda), (o, a, \lambda)\}$

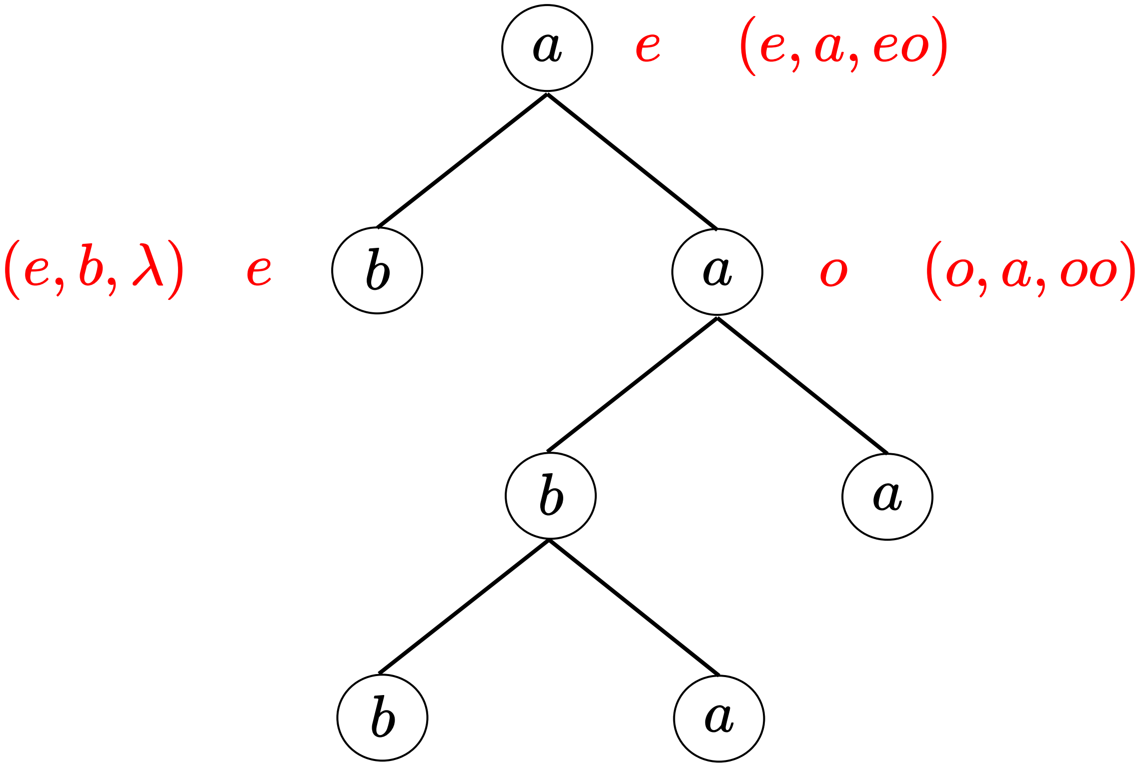
Tree automata: parity



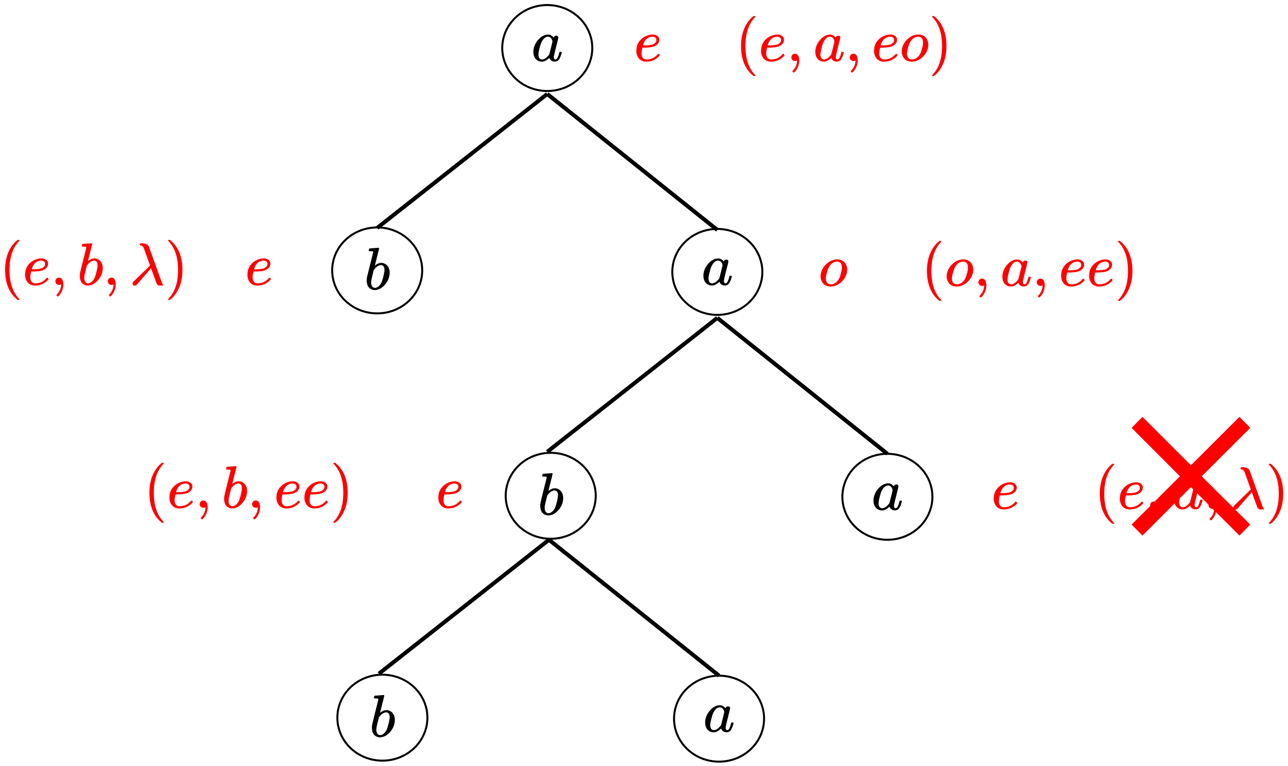
Tree automata: parity



Tree automata: parity



Tree automata: parity



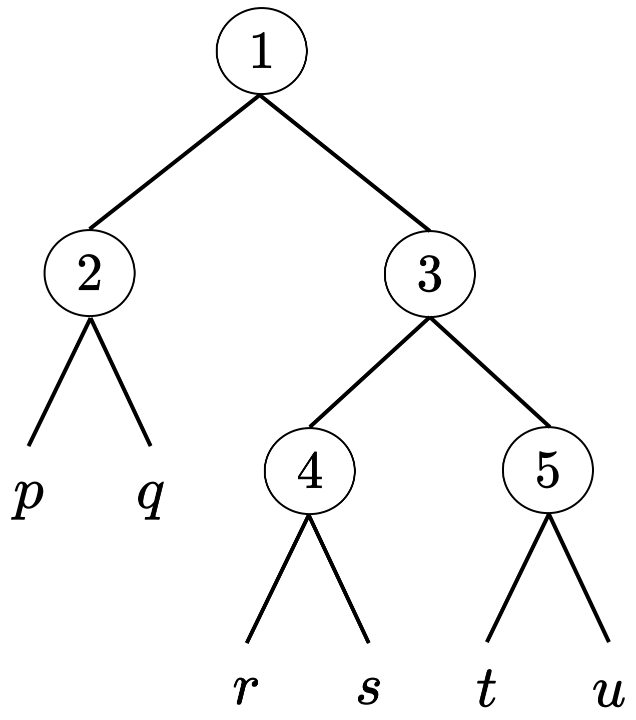
The problem #TA

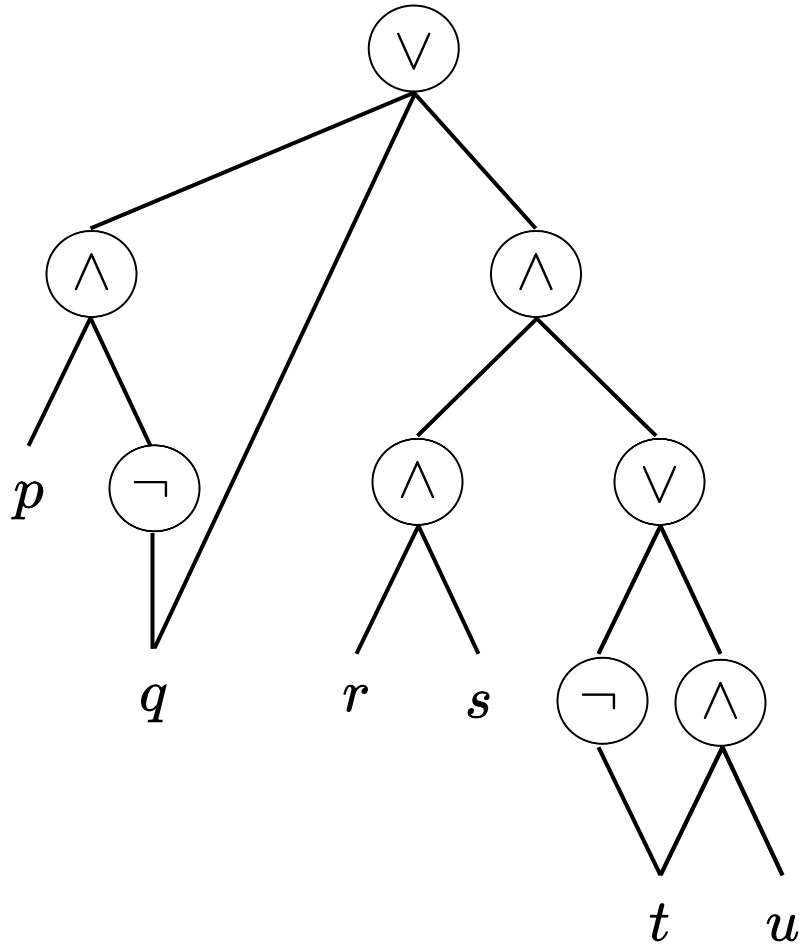
Input: A tree automaton T and a number n (given in unary)

Output: Number of trees t such that t is accepted by T and the number of nodes of t is n

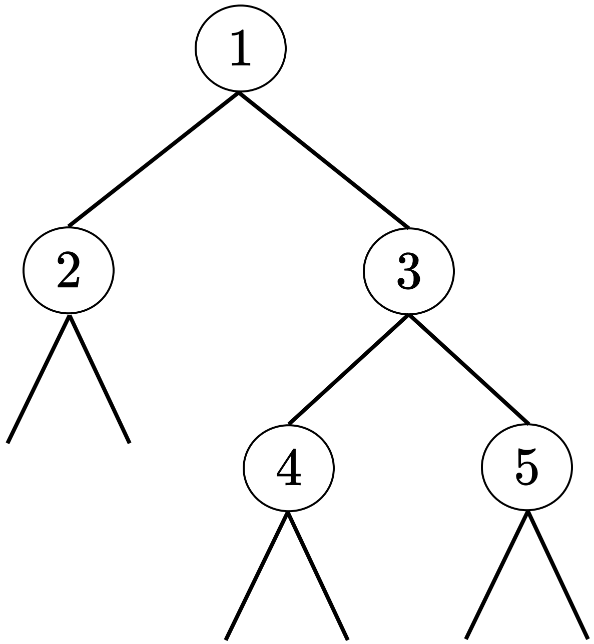
Theorem [ACJR21b]: #TA admits an FPRAS

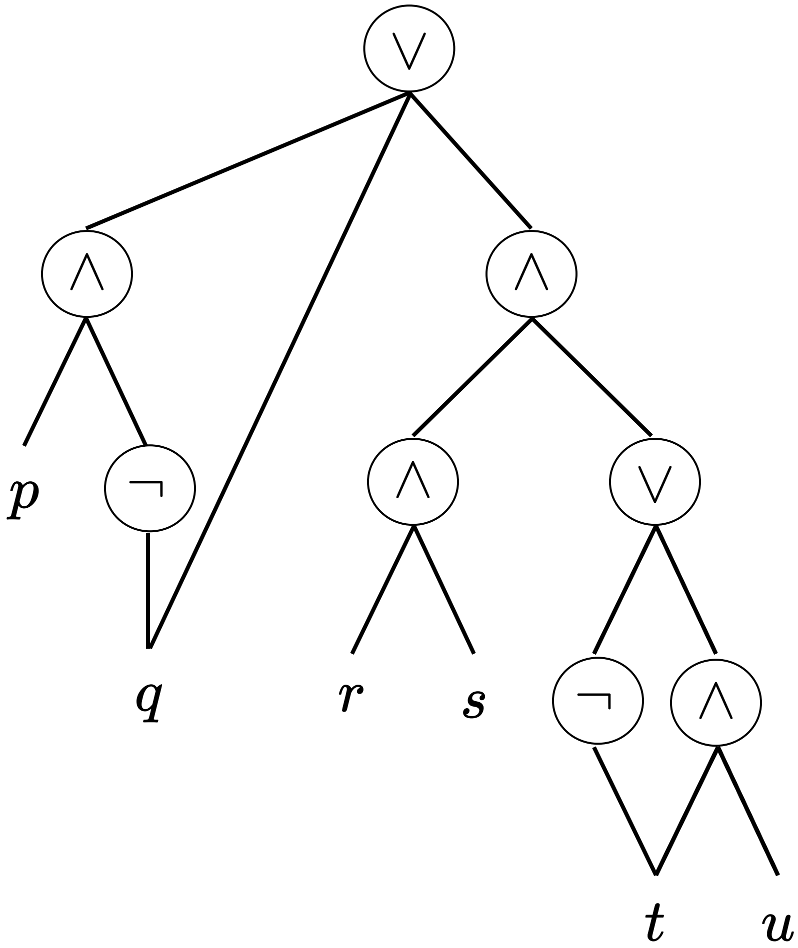
#SDNNF $\leq^p_{\mathrm{par}}$ #TA



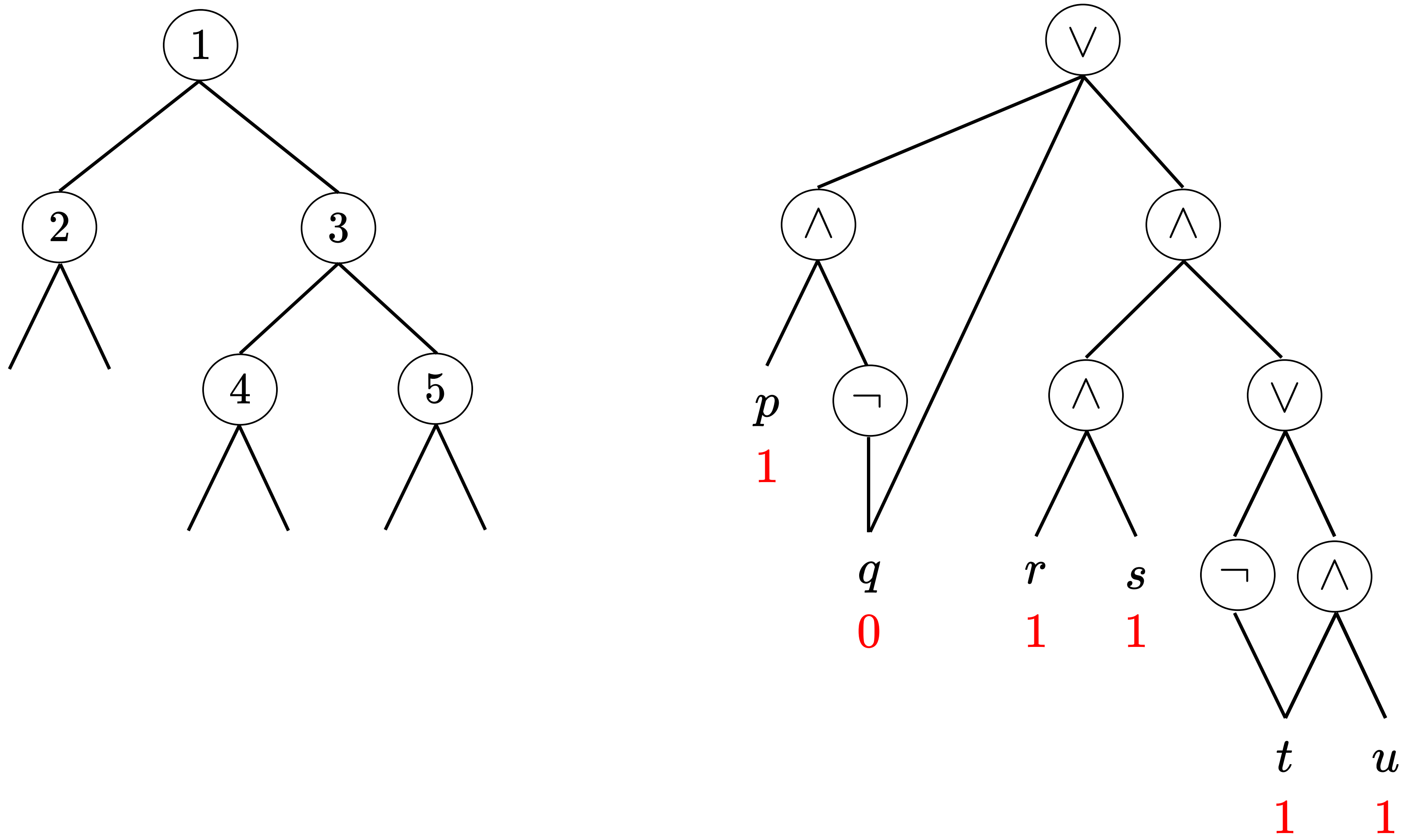


#SDNNF $\leq^{p}_{\mathrm{par}}$ #TA

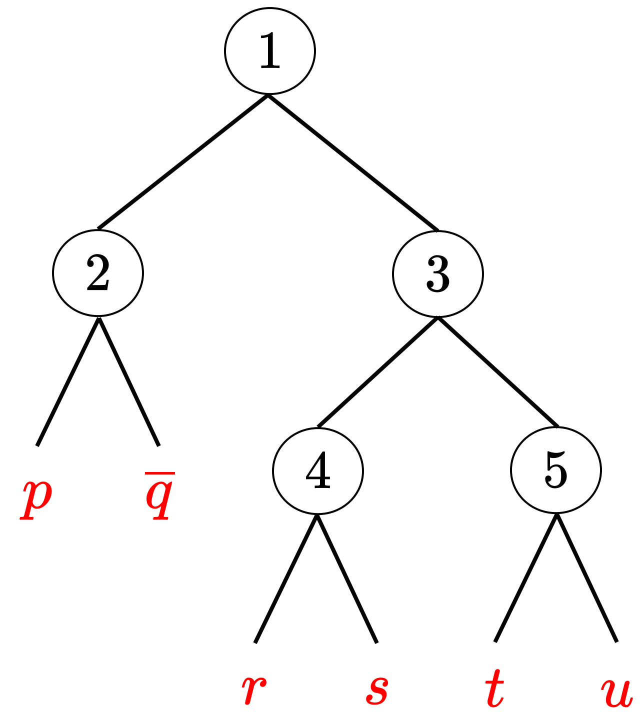


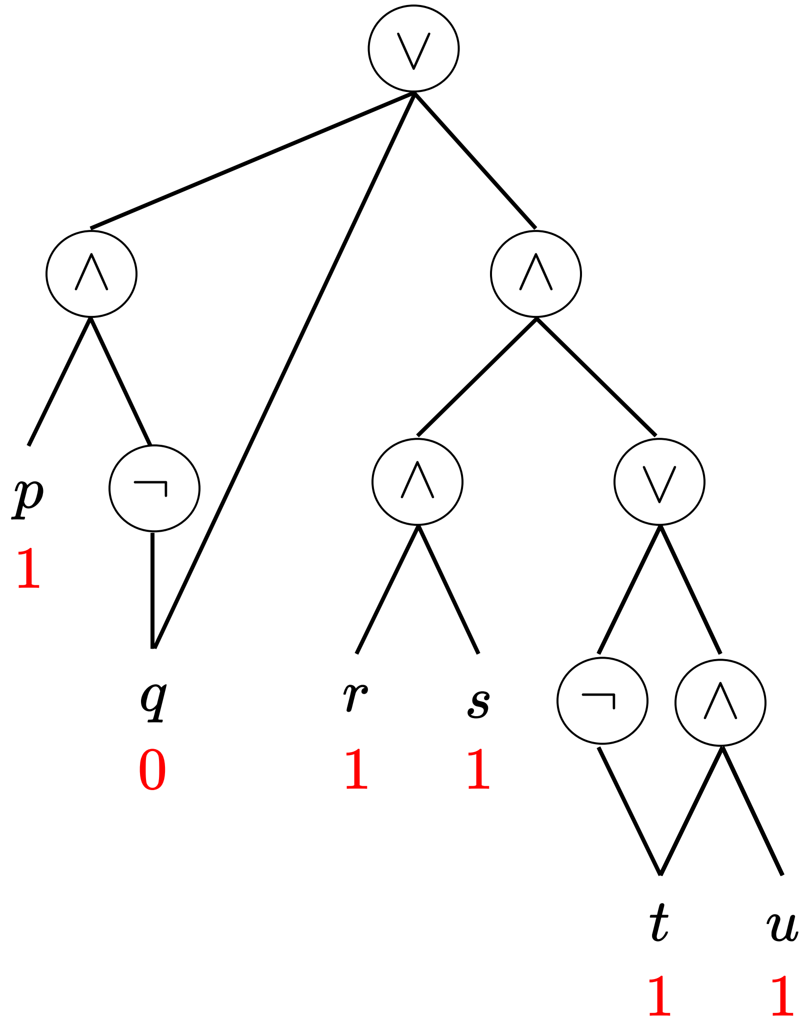


#SDNNF $\leq^p_{\text{par}}$ #TA

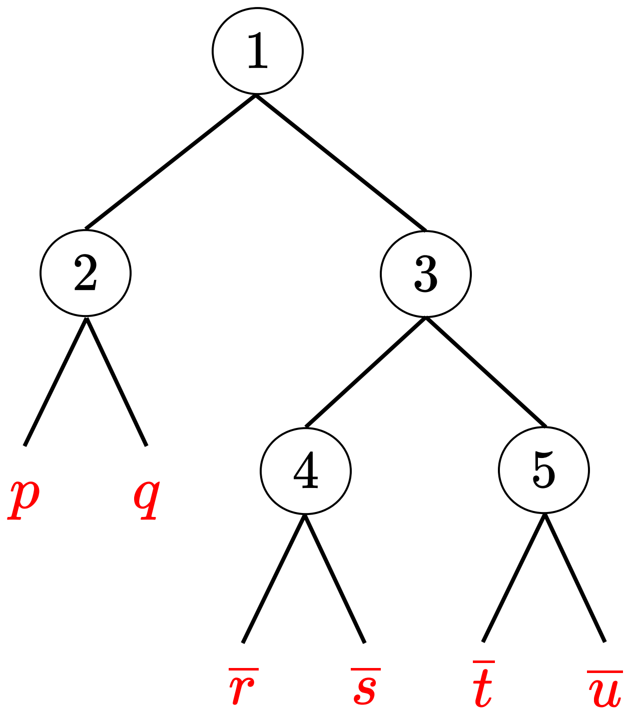


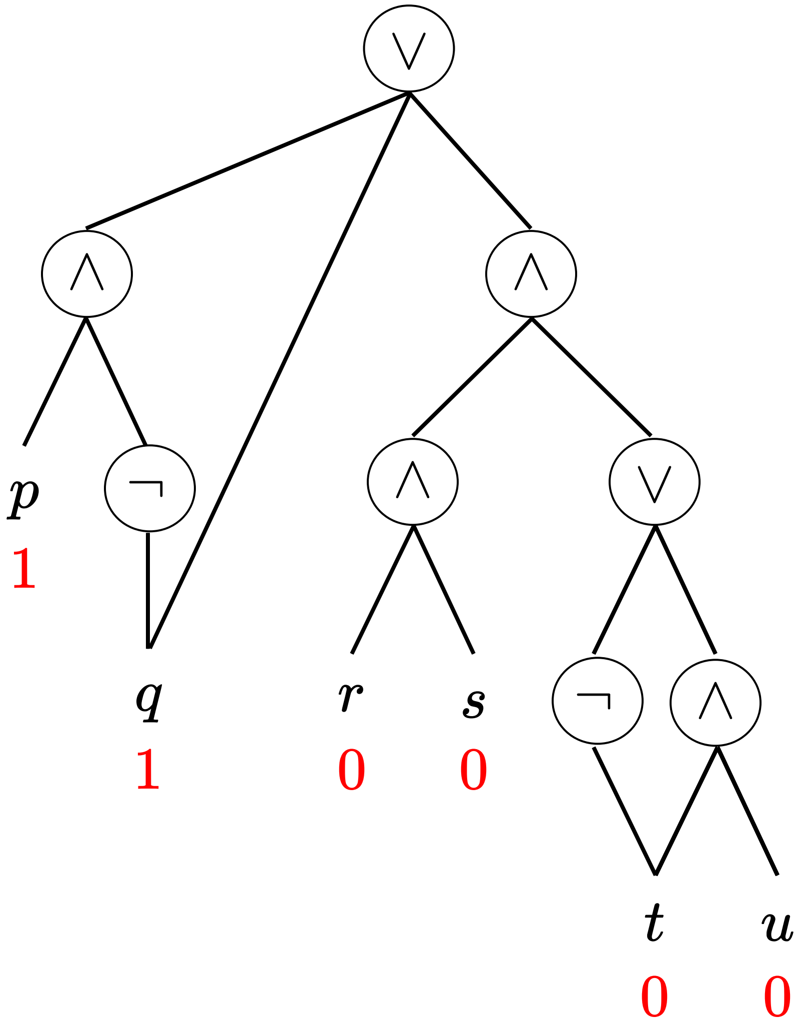
#SDNNF $\leq^{p}_{\text{par}}$ #TA



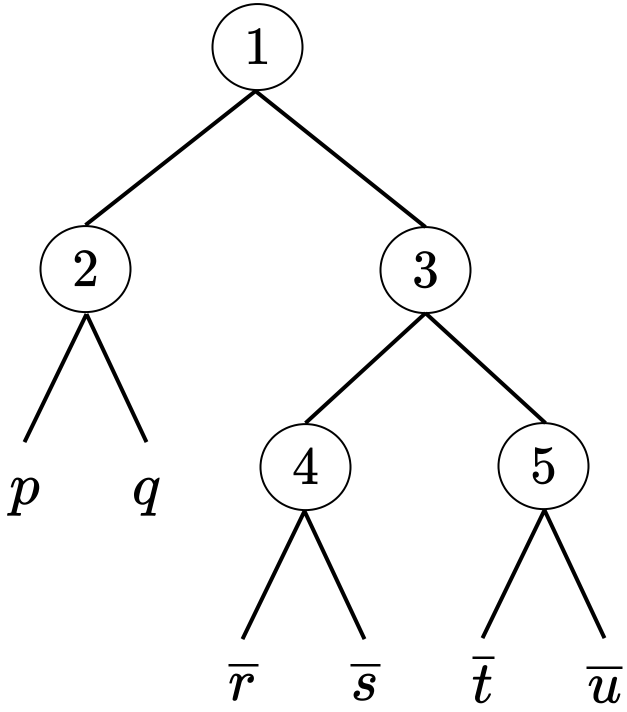


#SDNNF $\leq^{p}_{\text{par}}$ #TA





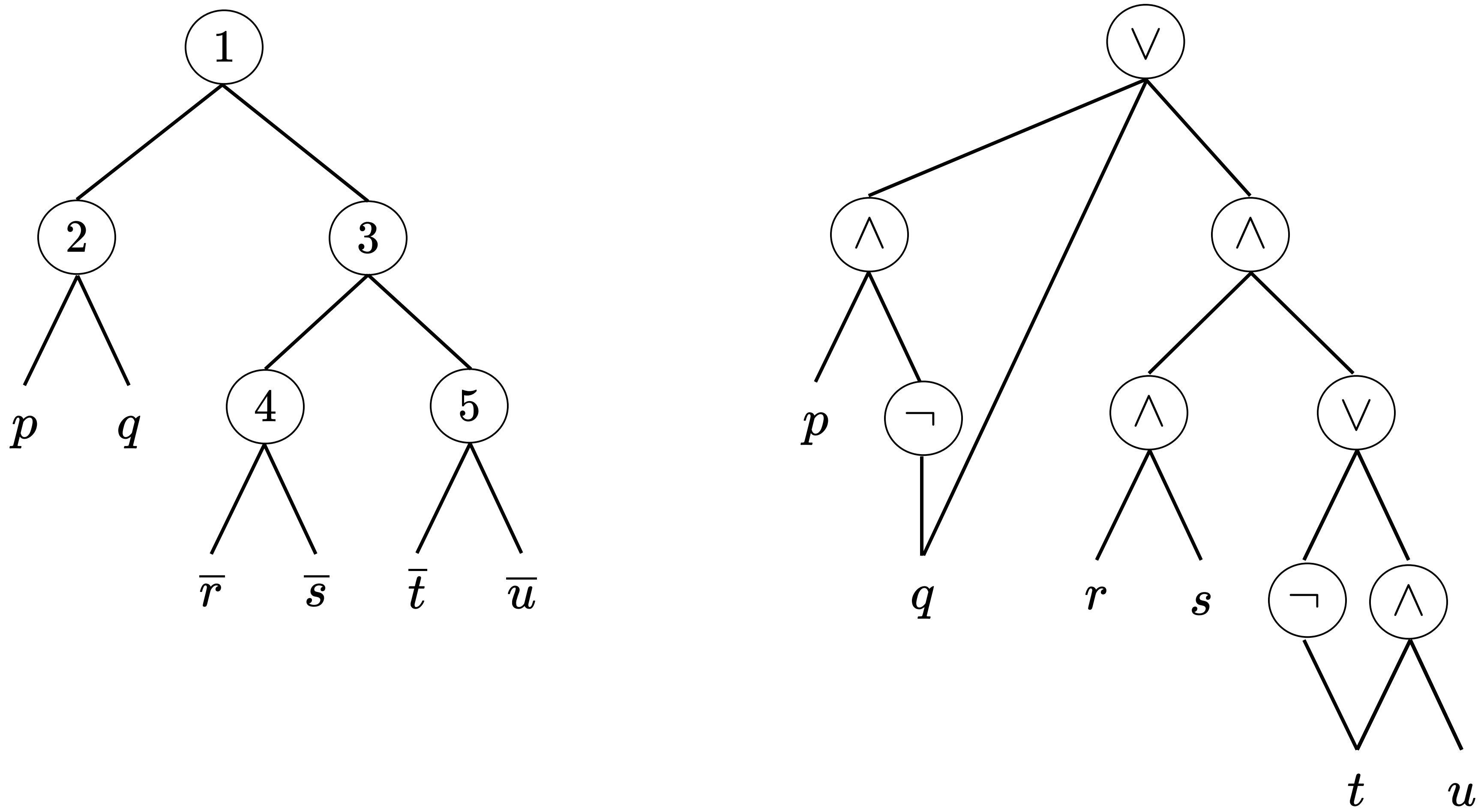
#SDNNF $\leq^p_{\text{par}}$ #TA



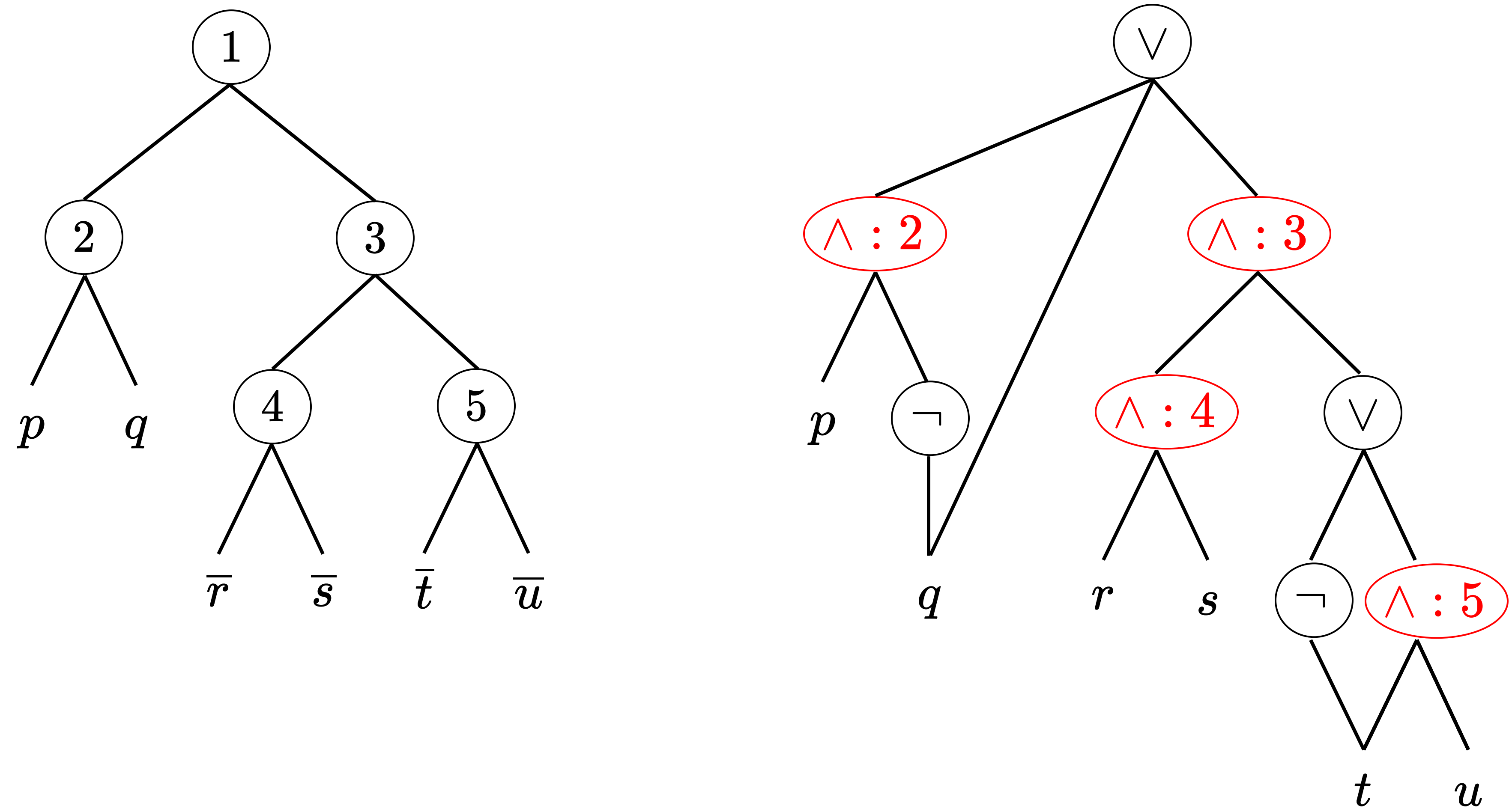
Tree automata: (Q, Σ, Δ, I)

- $\Sigma = \{1, 2, 3, 4, 5, p, \overline{p}, \ldots, u, \overline{u}\}$

#SDNNF $\leq^p_{\text{par}}$ #TA

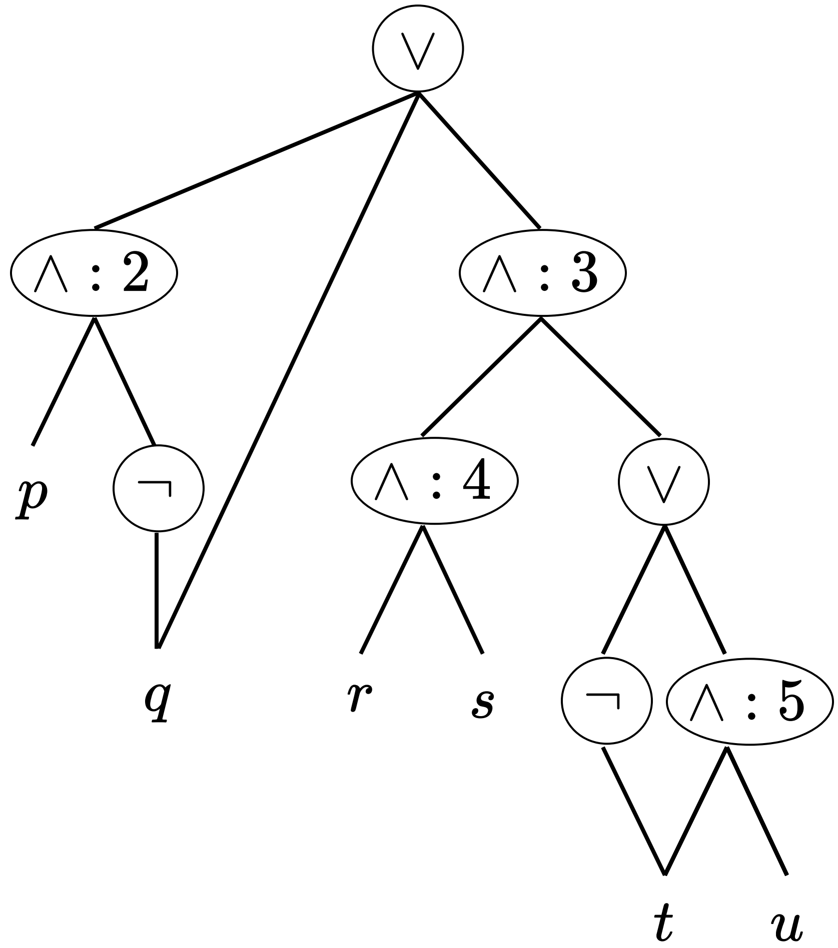


#SDNNF $\leq^p_{\text{par}}$ #TA



#SDNNF $\leq^p_{\text{par}}$ #TA

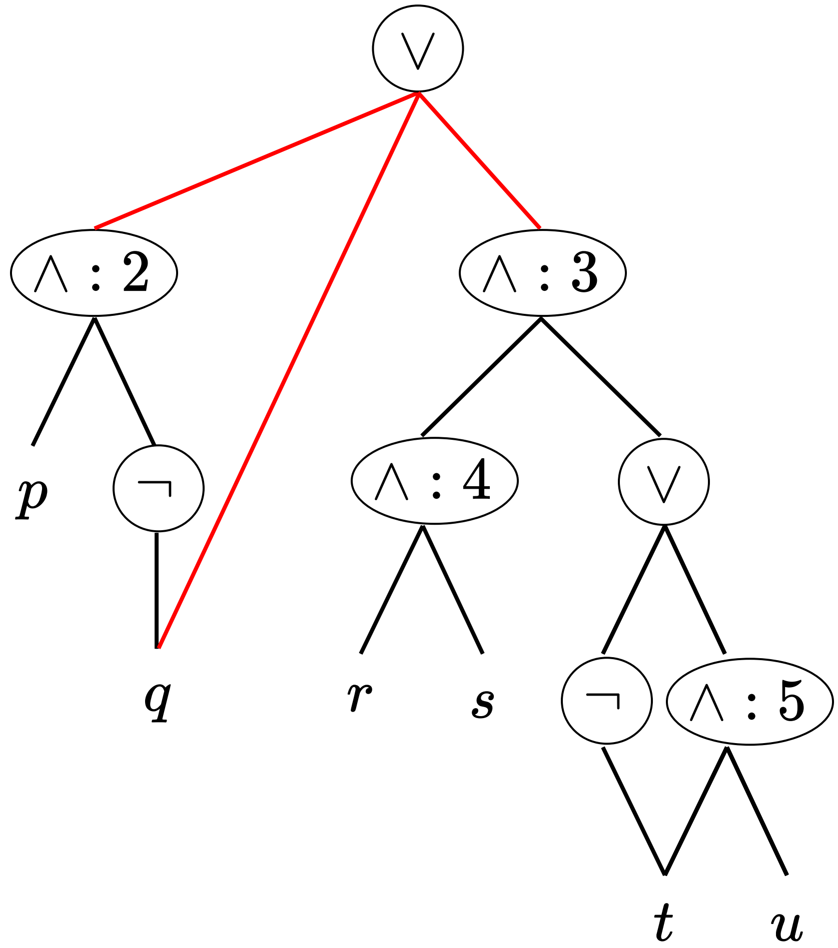
Tree automata: (Q, Σ, Δ, I)



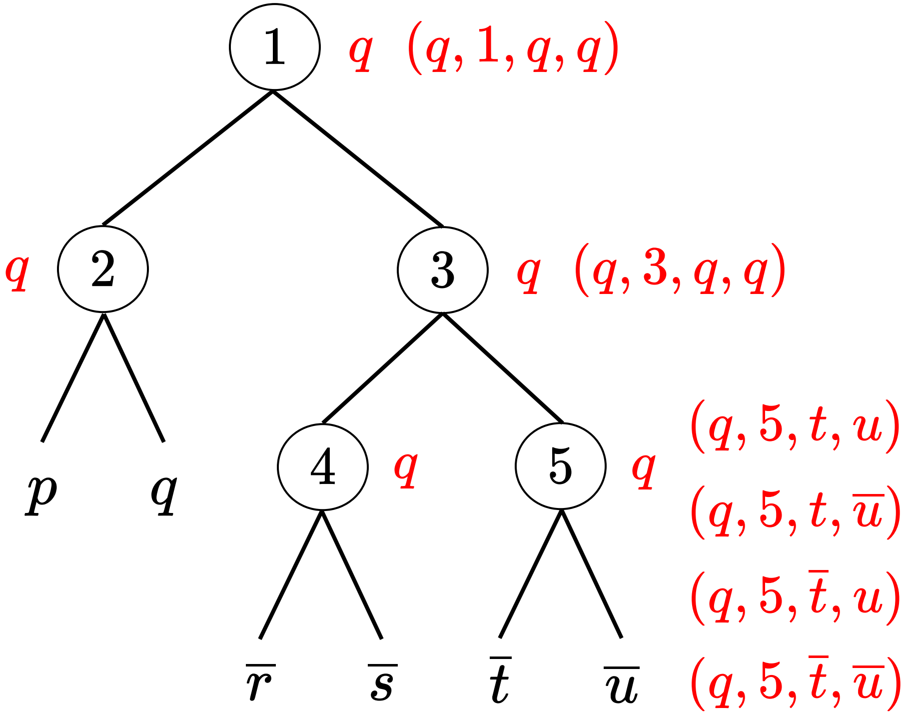
#SDNNF $\leq^p_{\text{par}}$ #TA

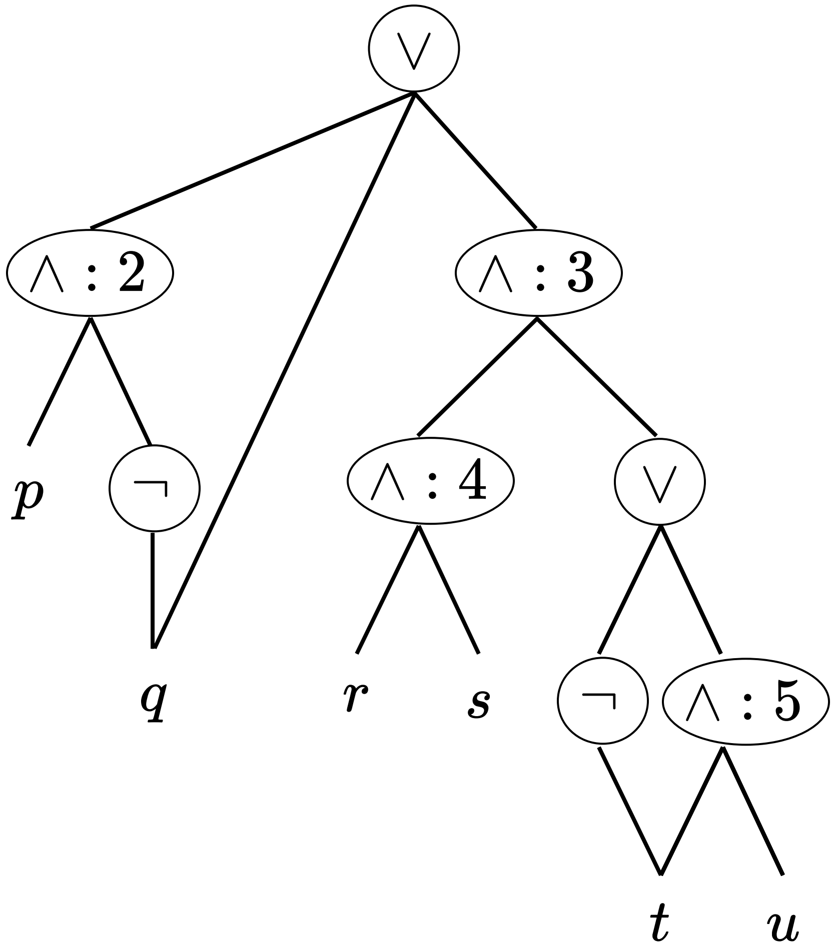
Tree automata: (Q, Σ, Δ, I)

- $I = \{\wedge : 2,\ q,\ \wedge : 3\}$
- $Q = \{\wedge : 2,\ \wedge : 3,\ \wedge : 4,\ \wedge : 5,$
  $p,\ \overline{p}, \ldots, u,\ \overline{u},\ \top\}$

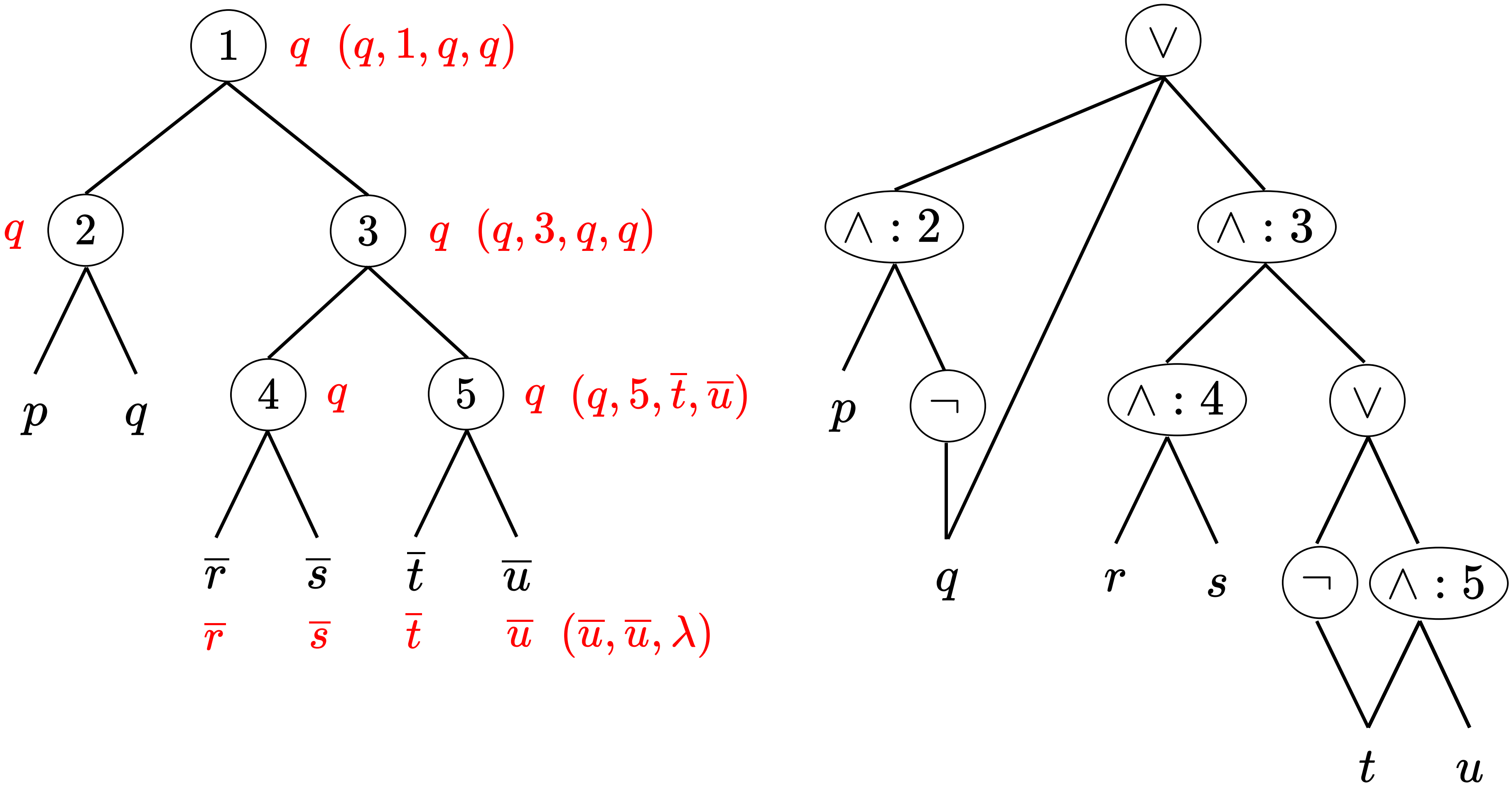


#SDNNF $\leq^{p}_{\mathrm{par}}$ #TA

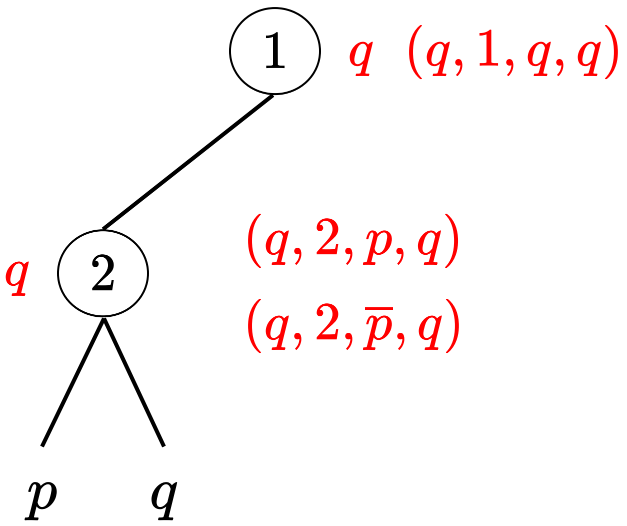


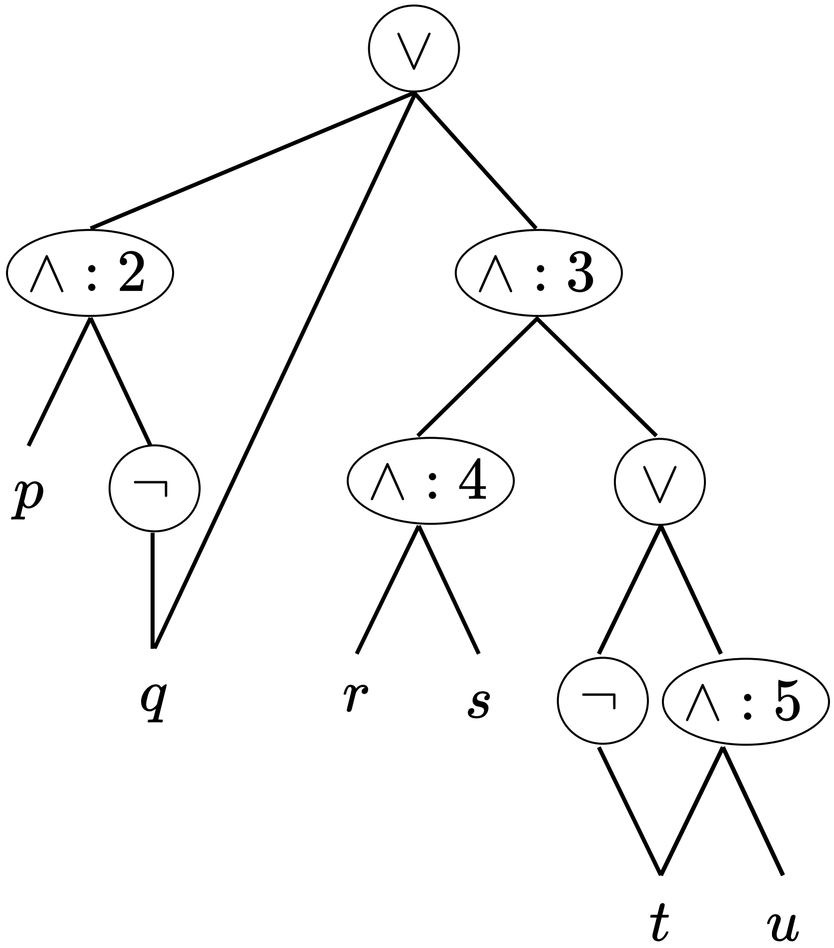


#SDNNF $\leq^p_{\mathrm{par}}$ #TA

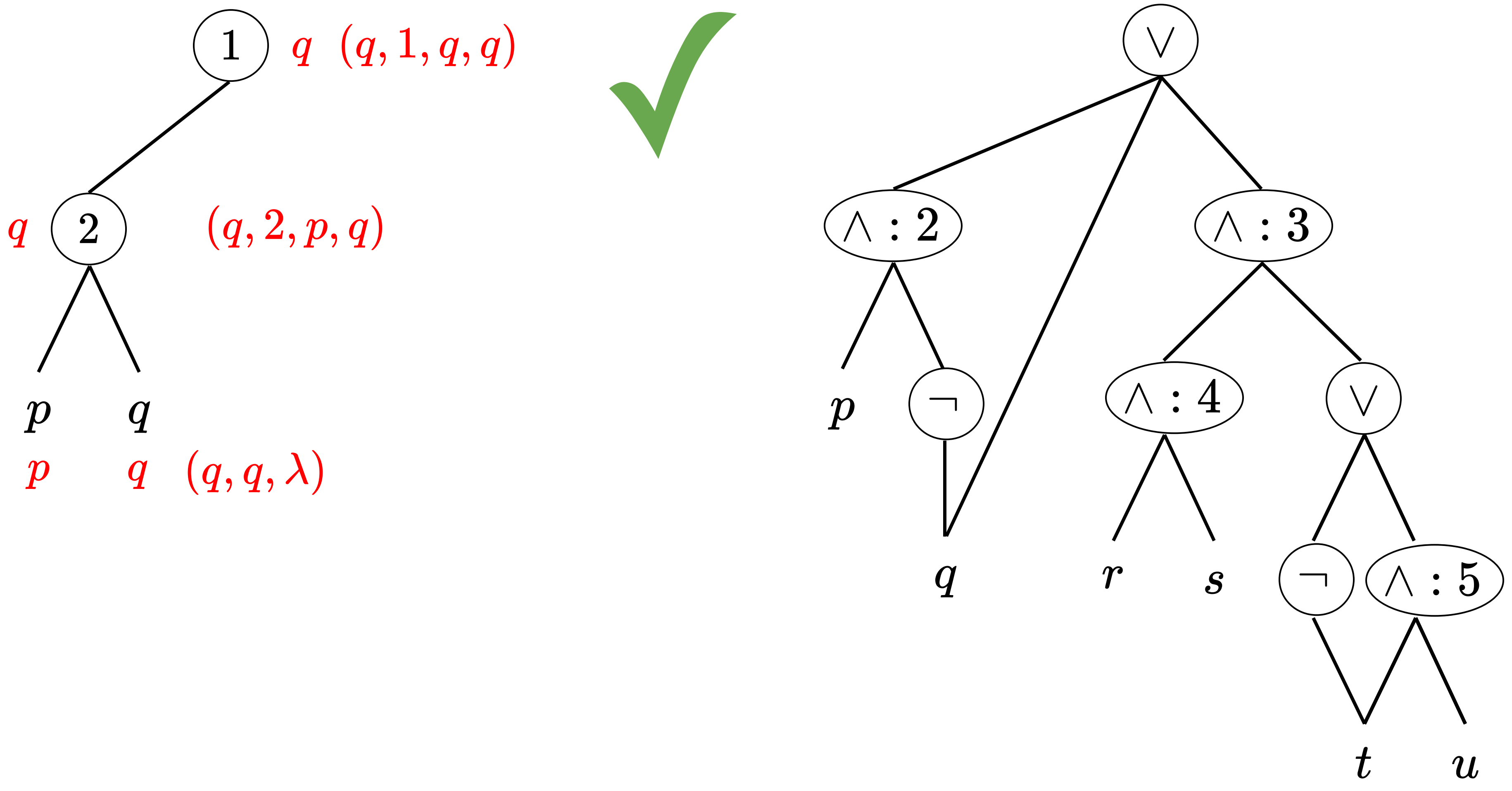


#SDNNF $\leq^{p}_{\text{par}}$ #TA

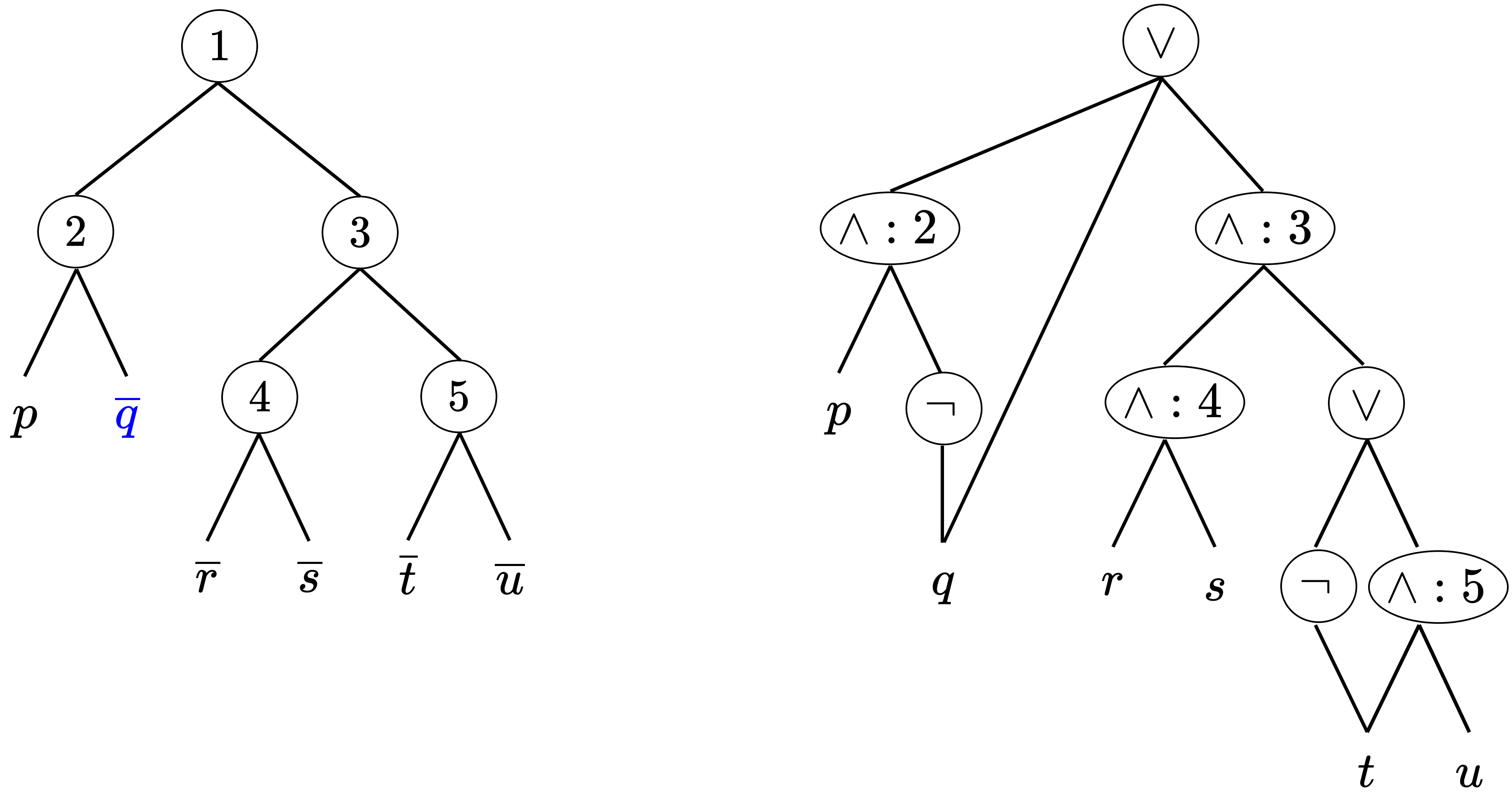




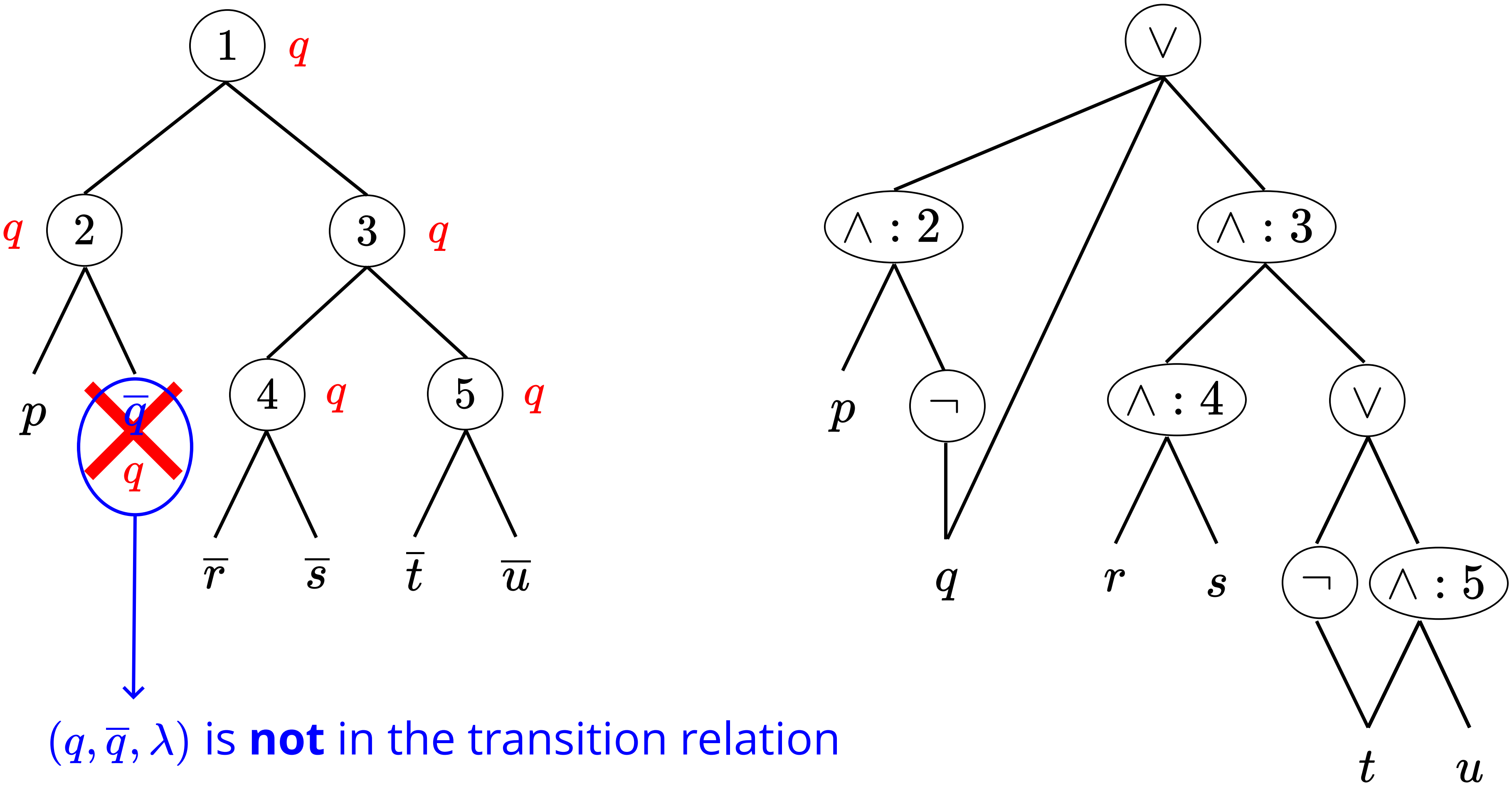
#SDNNF $\leq^p_{\text{par}}$ #TA



#SDNNF $\leq^p_{\text{par}}$ #TA

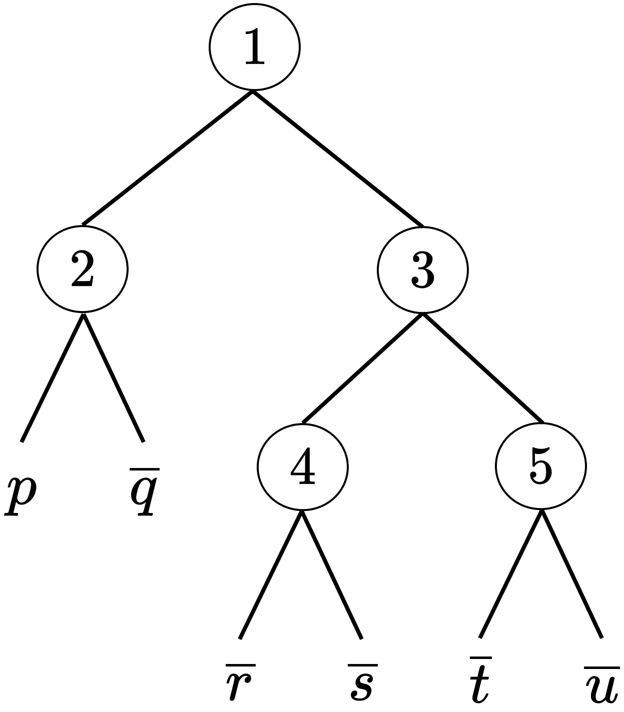


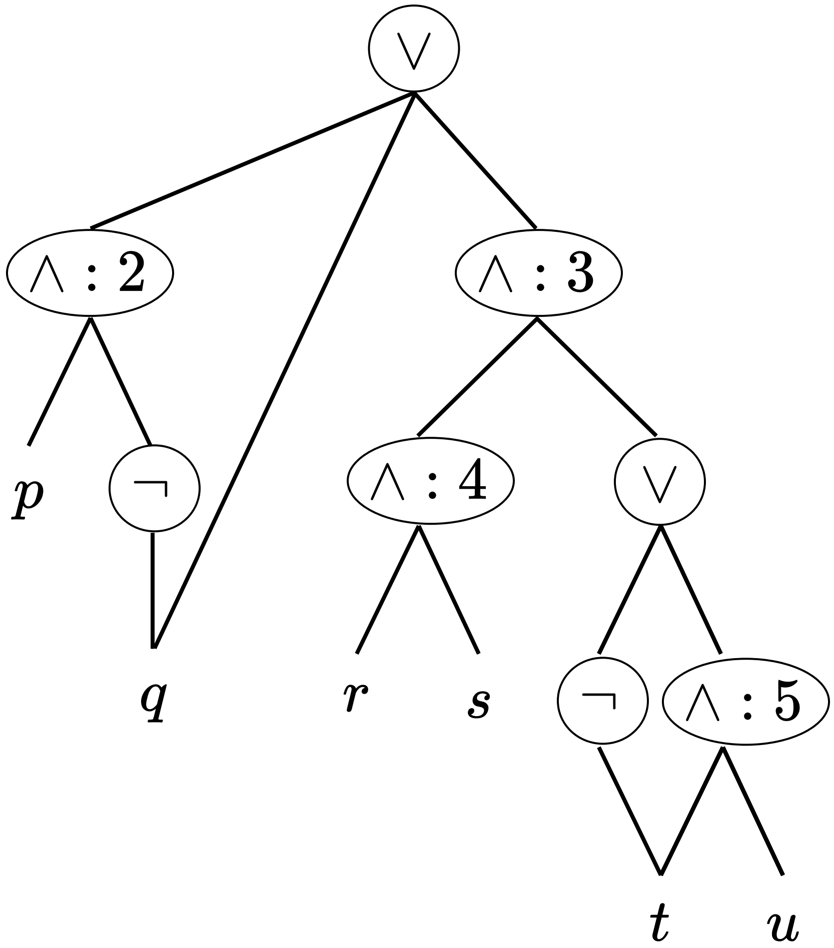
#SDNNF $\leq^{p}_{\text{par}}$ #TA



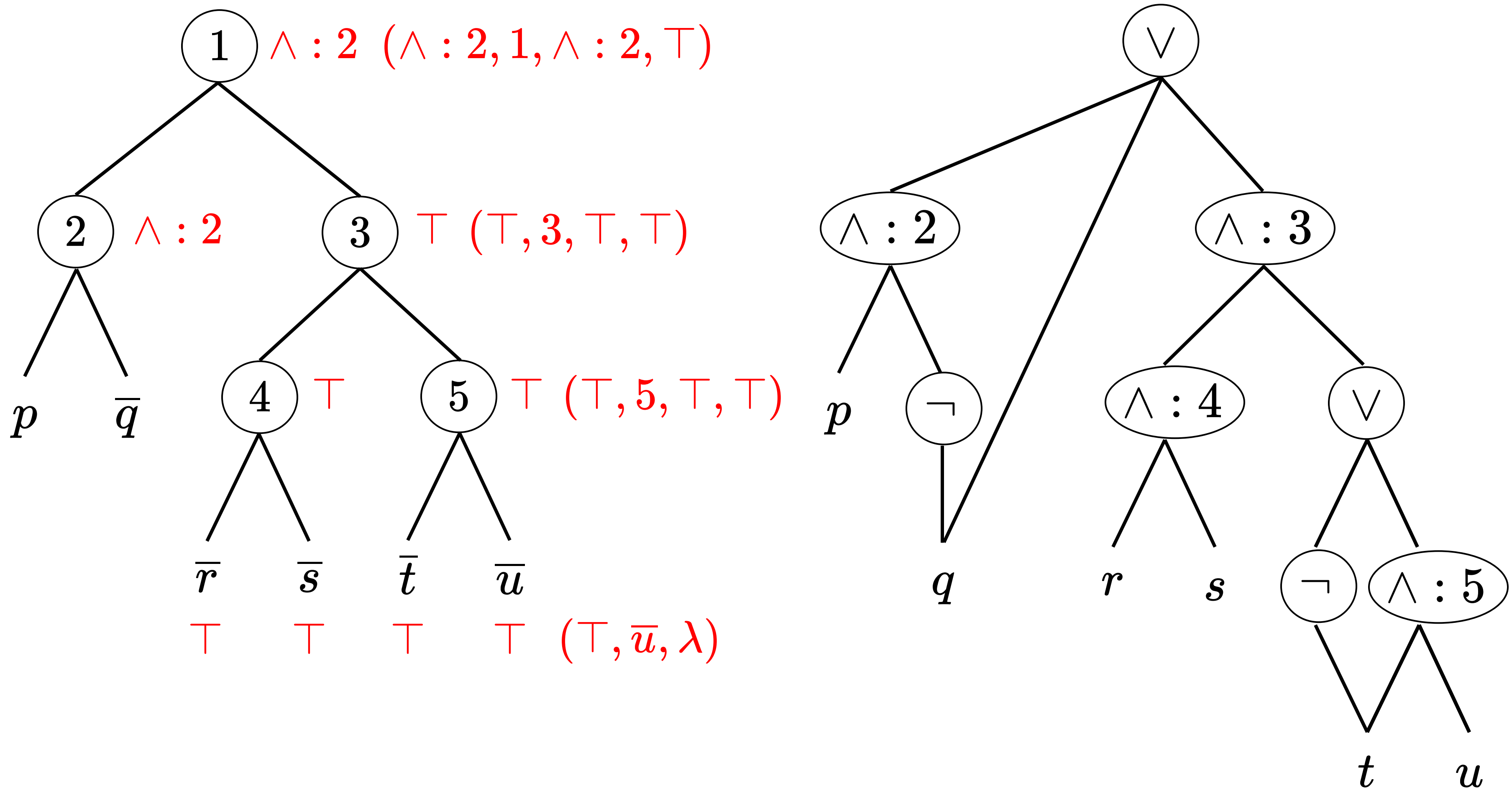
$(q, \overline{q}, \lambda)$ is **not** in the transition relation

#SDNNF $\leq^p_{\text{par}}$ #TA

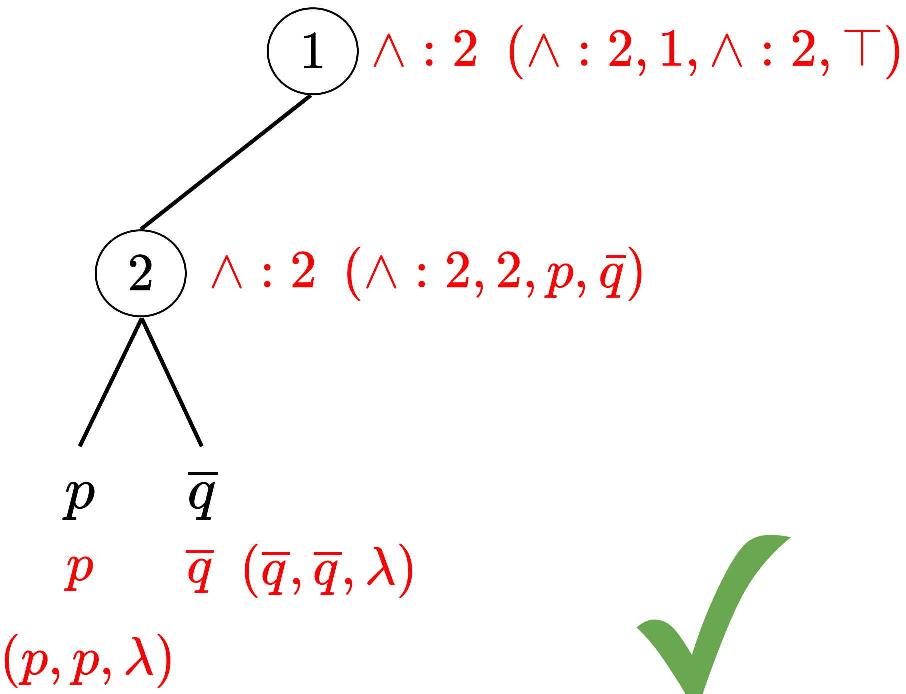


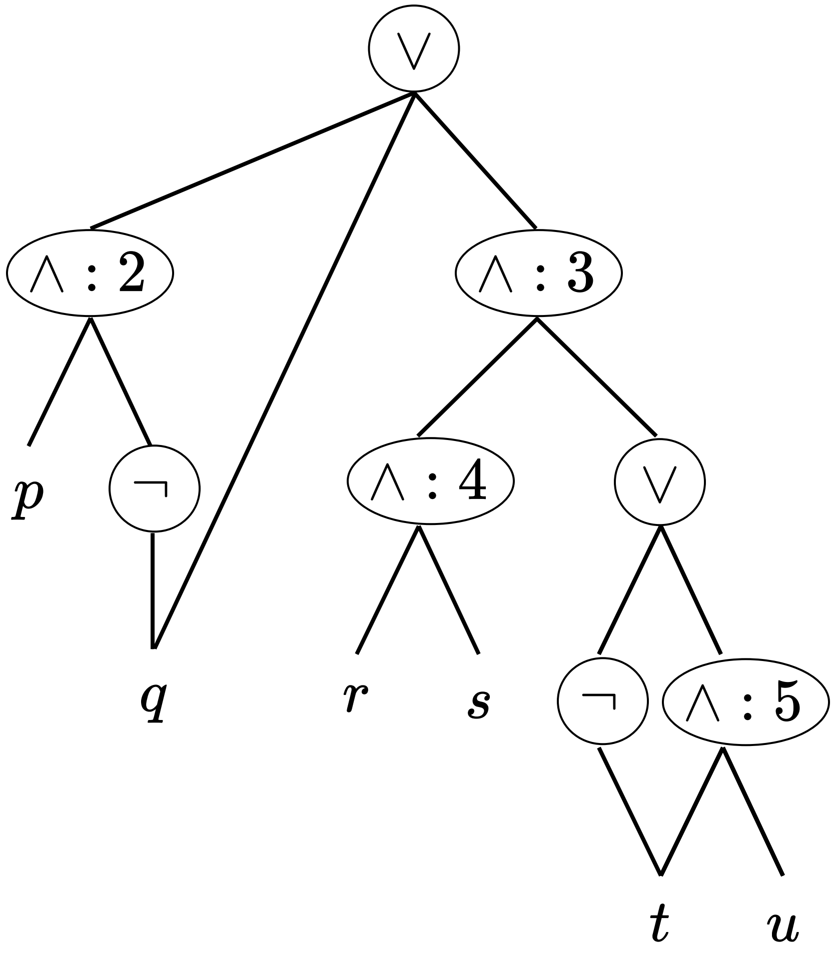


#SDNNF $\leq^p_{\text{par}}$ #TA

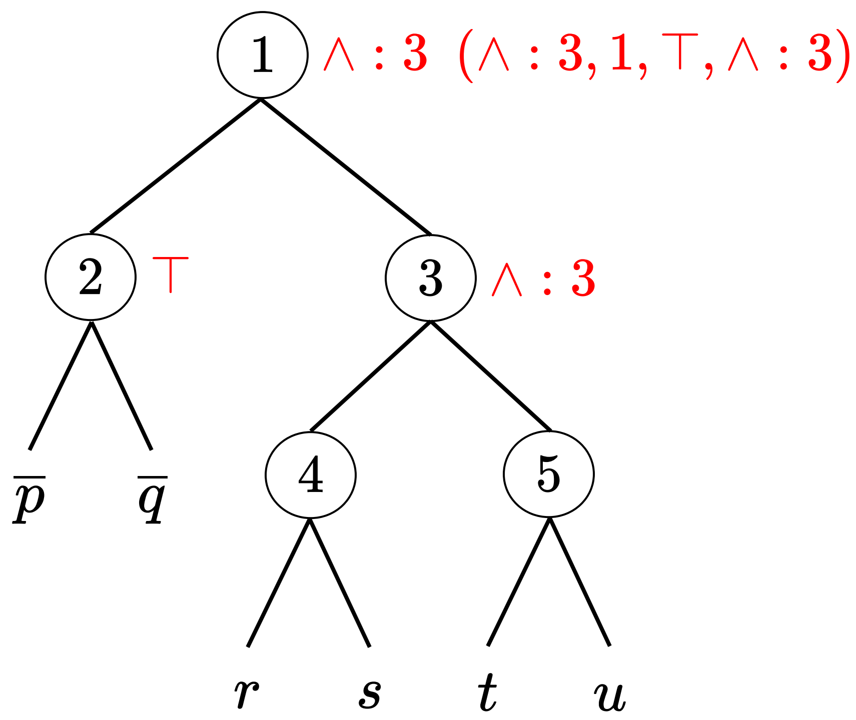


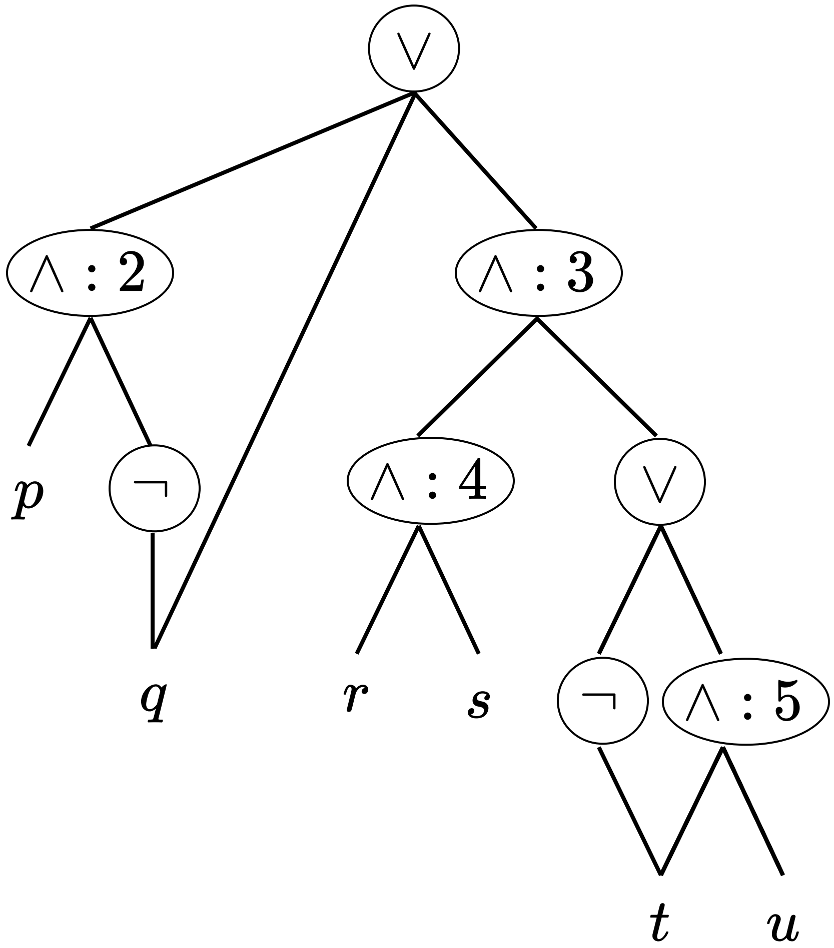
#SDNNF $\leq^p_{\text{par}}$ #TA



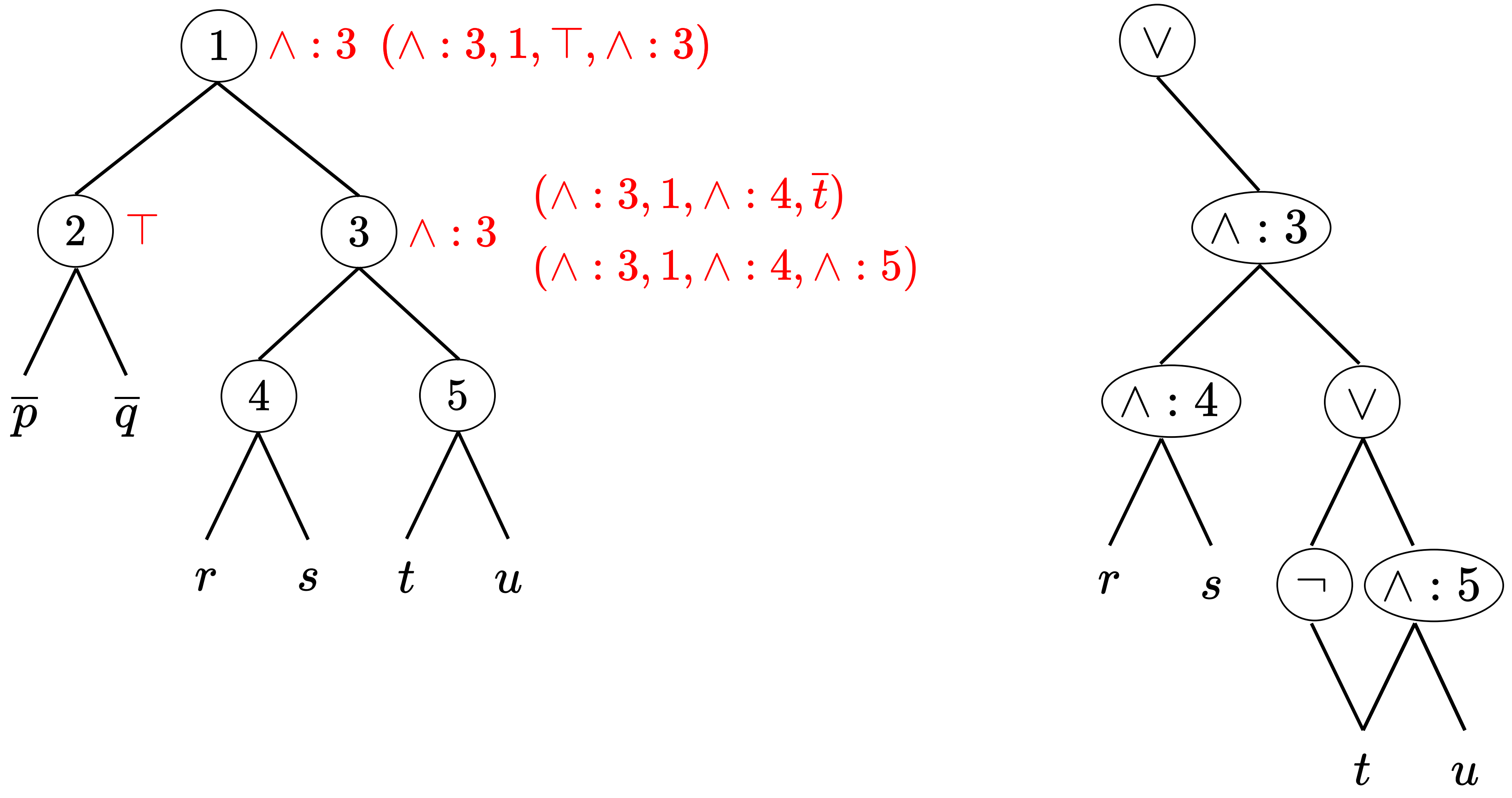


#SDNNF $\leq^p_{\text{par}}$ #TA

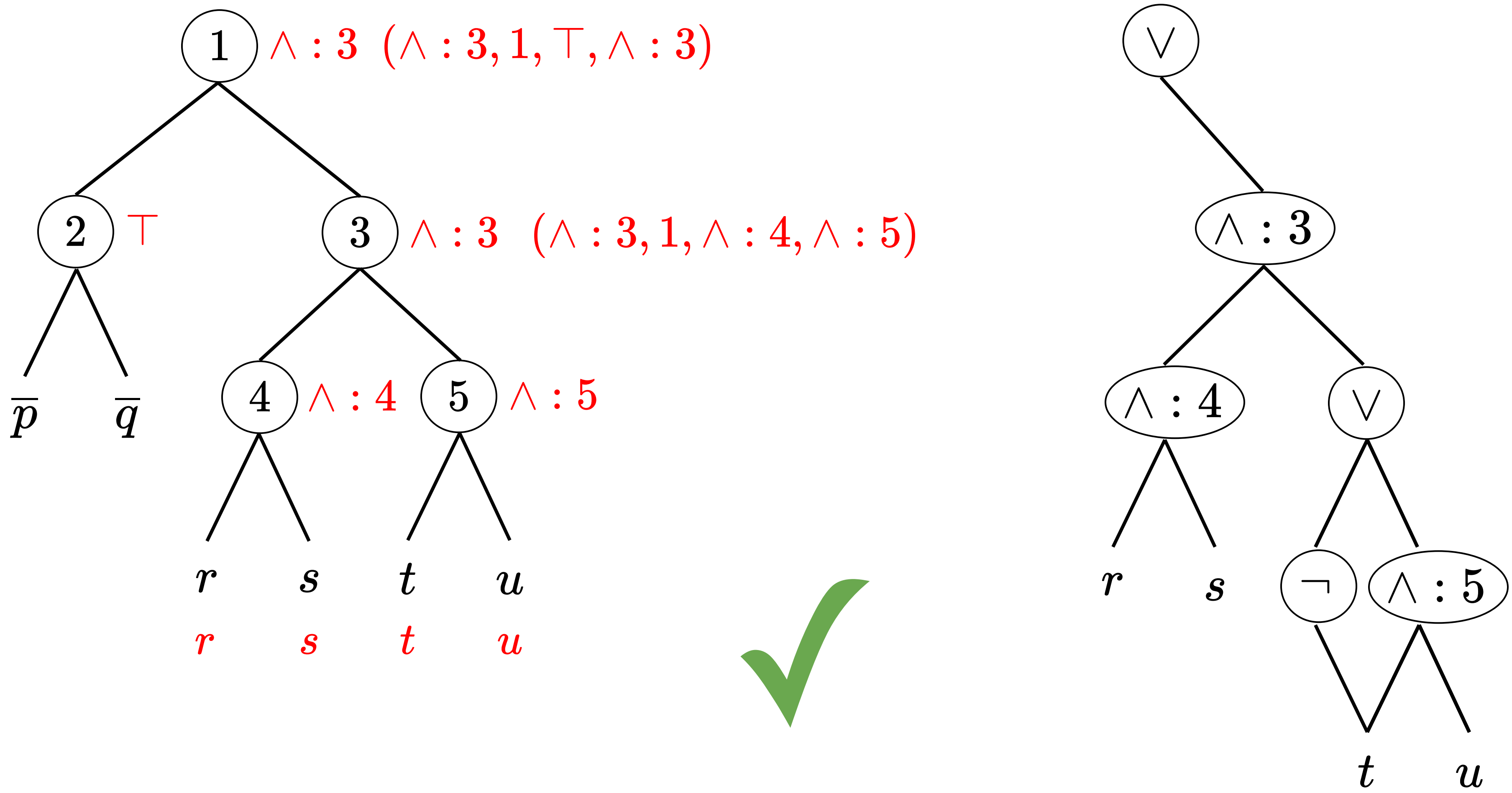




#SDNNF $\leq^p_{\text{par}}$ #TA



#SDNNF $\leq^{p}_{\text{par}}$ #TA



Before the open problems …

A corollary of the existence of an FPRAS for #NFA

Counting complexity classes

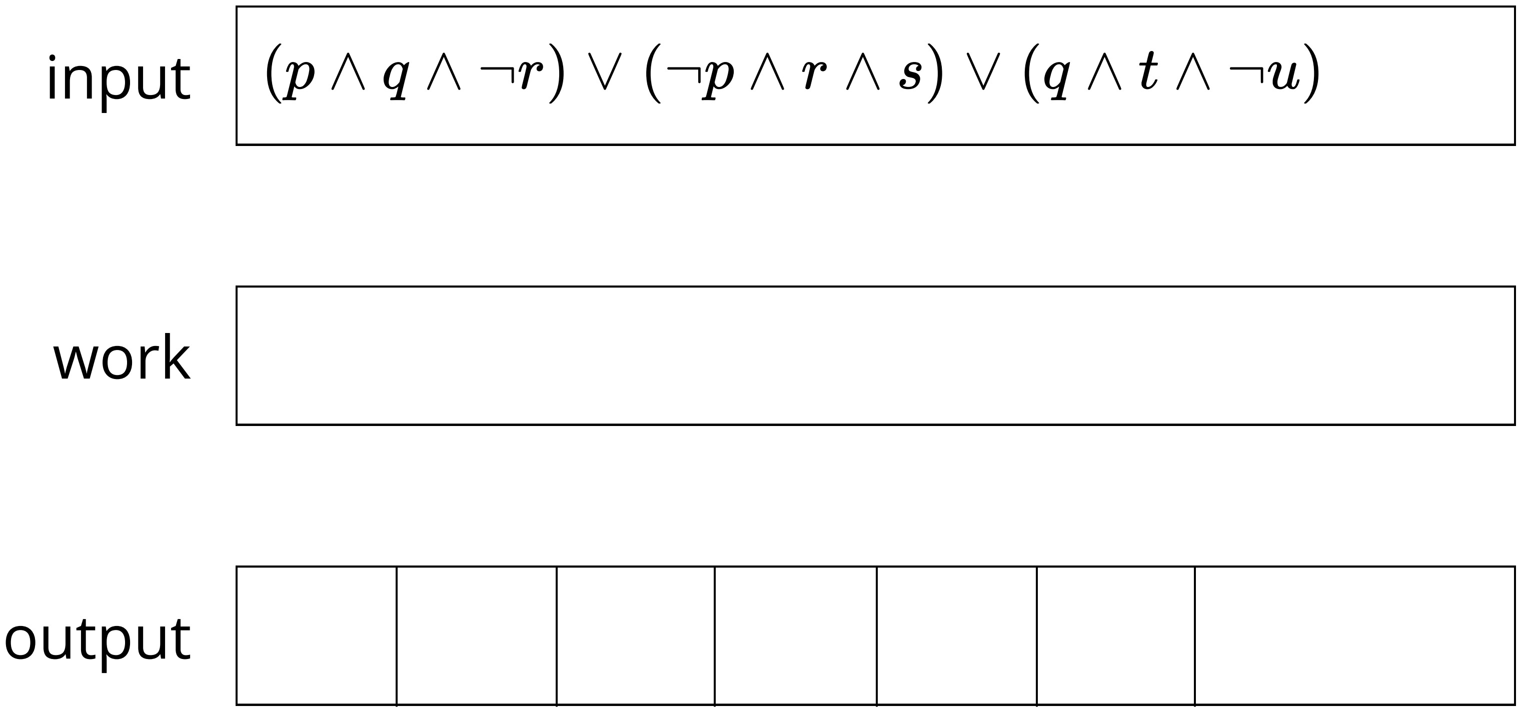
- **#P**: Count the number of witnesses for a problem in NP
- **SpanP**: Count the number of distinct outputs of an NP-transducer
  - Example: given as input a graph G, count the number of subgraphs G' of G such that G' is 3-colorable

Counting complexity classes

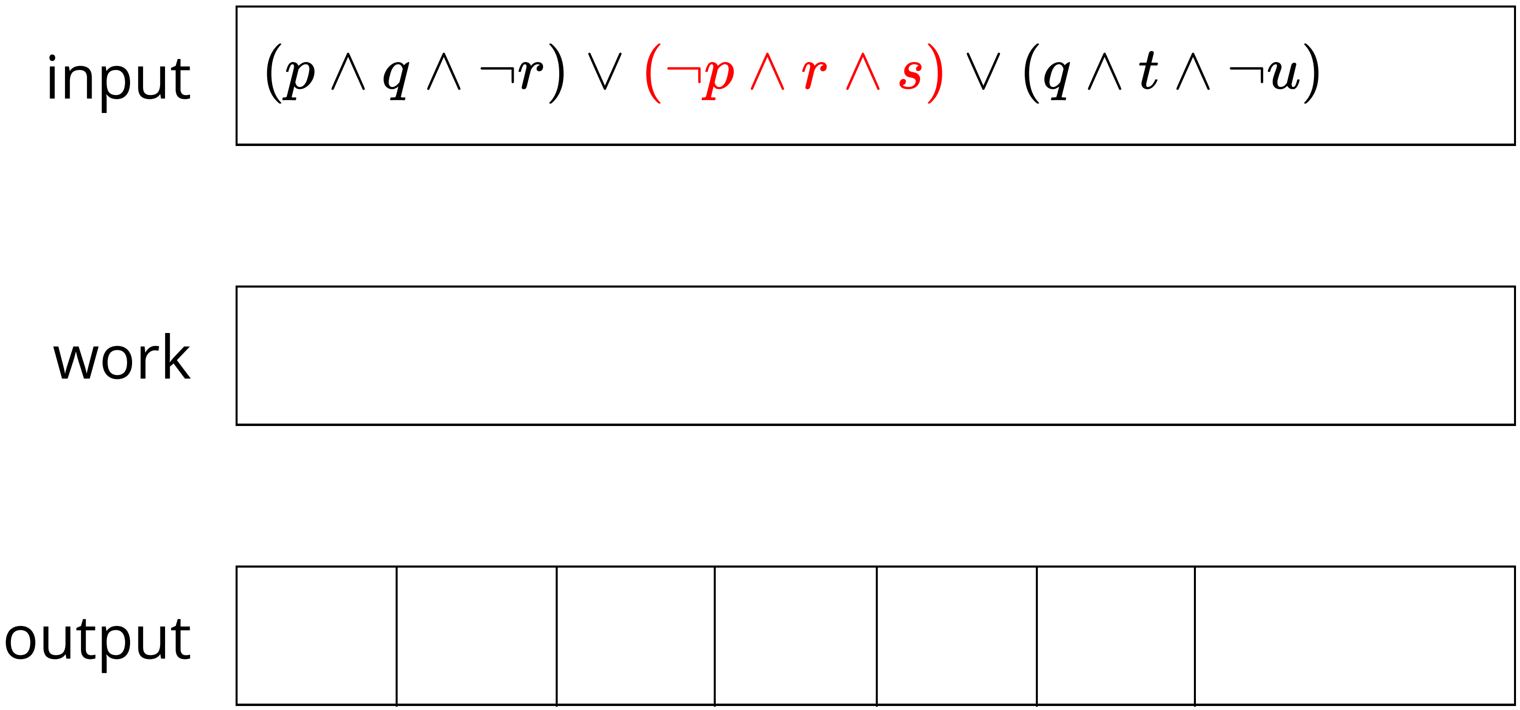
- **SpanL**: Count the number of distinct outputs of an NL-transducer
  - SpanL is contained in #P, and it is a hard class: if every function in SpanL can be computed in polynomial time, then P = NP

- #NFA is SpanL-complete under parsimonious reductions
  - Every function in SpanL admits an FPRAS

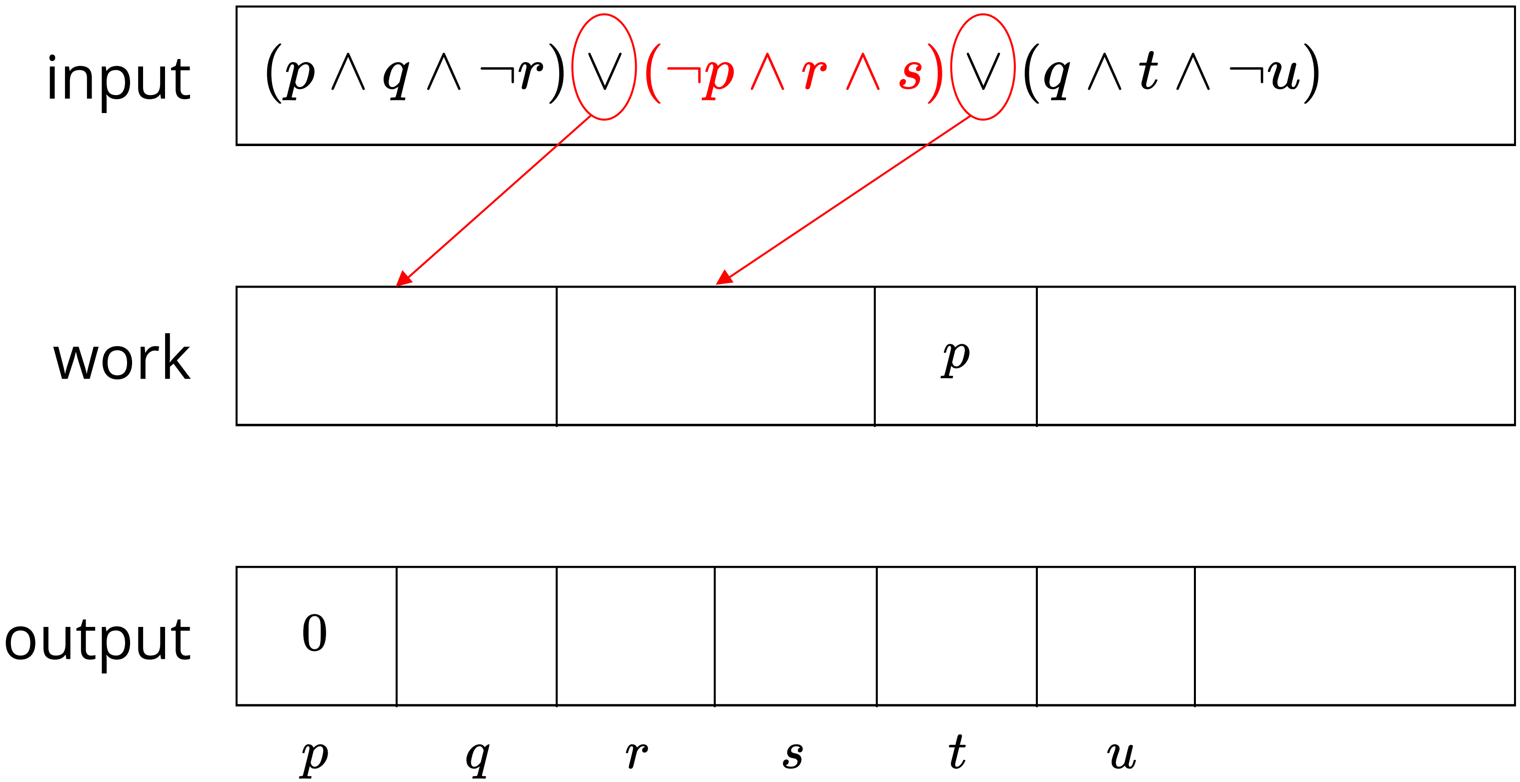
#DNF is in SpanL



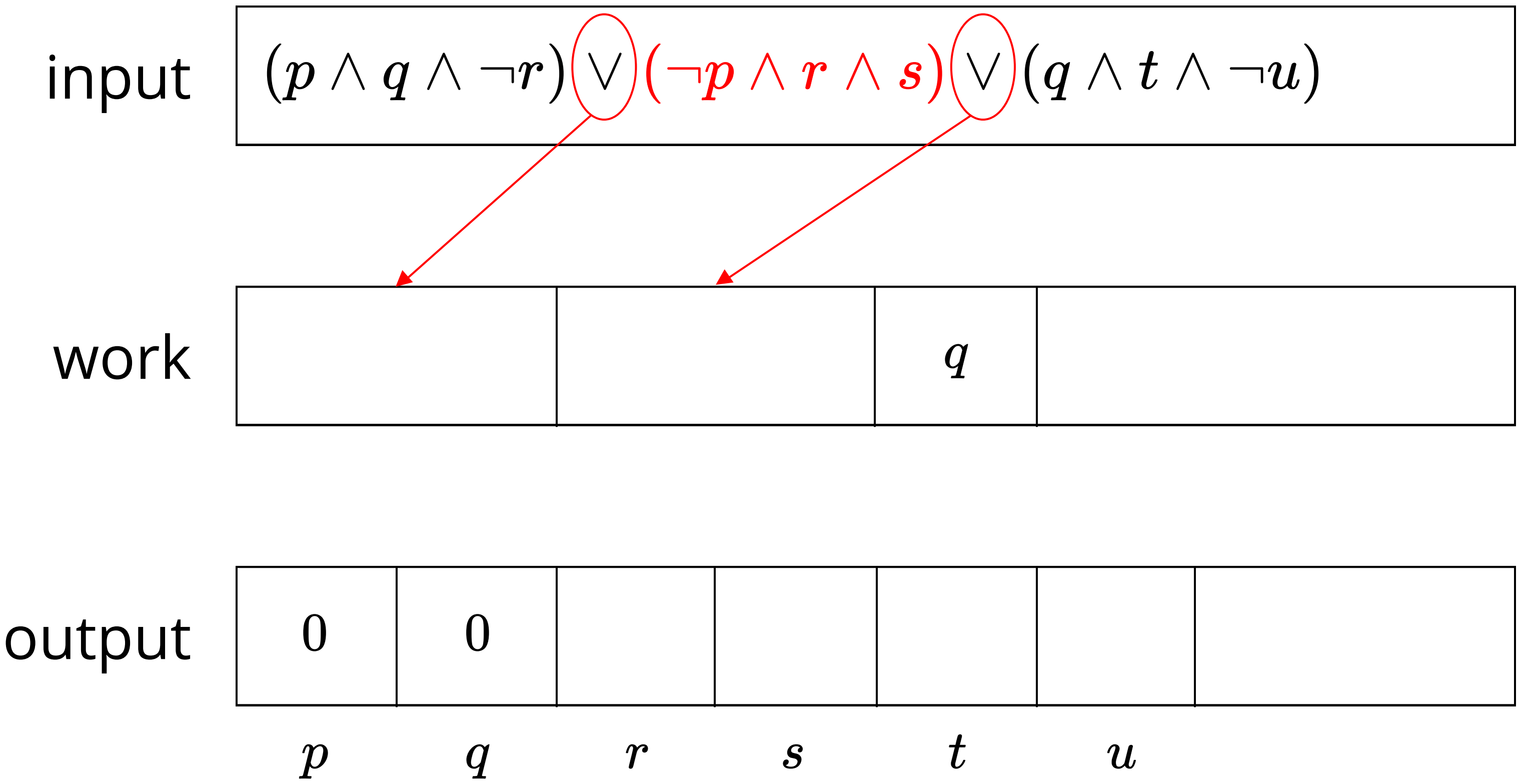
#DNF is in SpanL



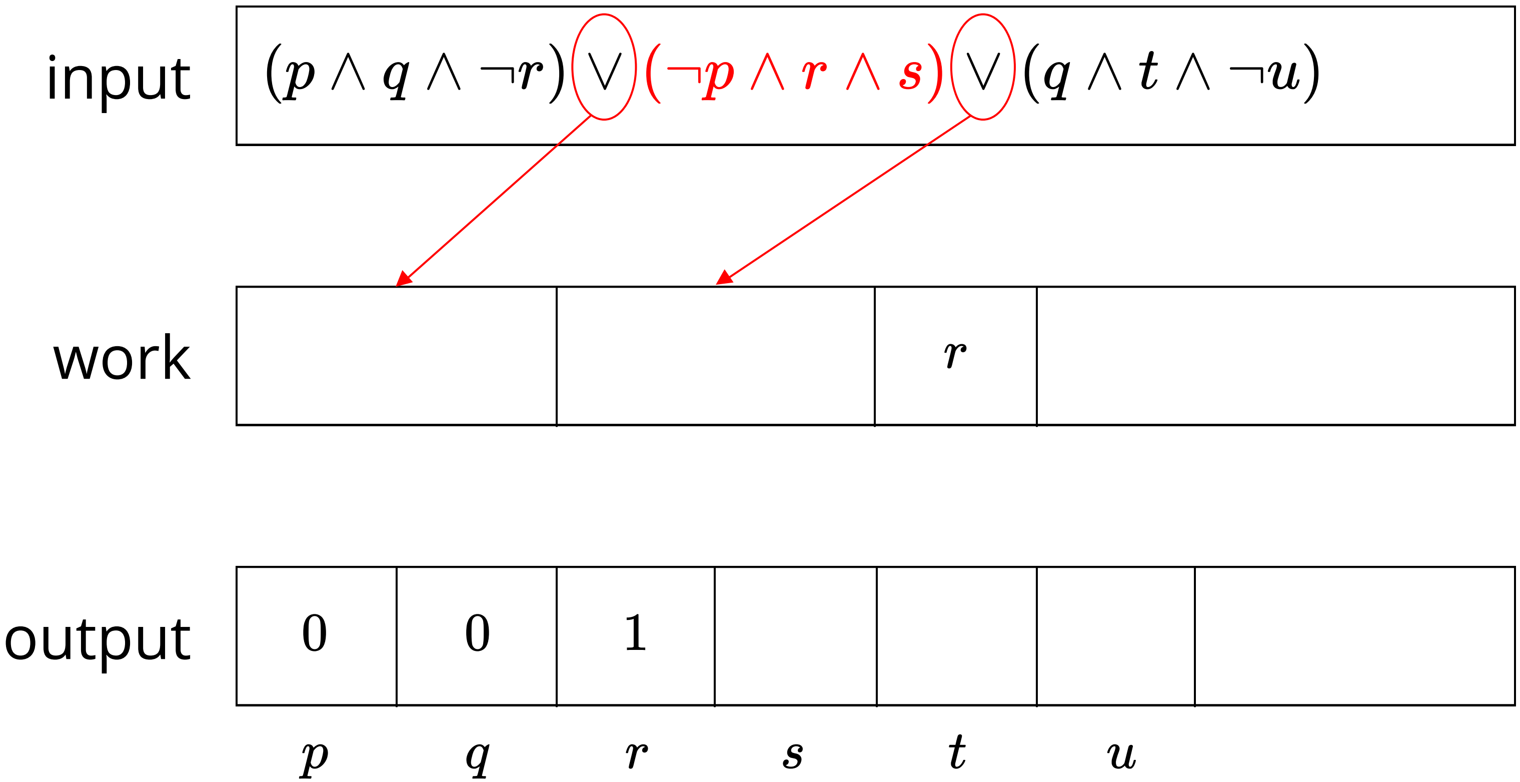
#DNF is in SpanL



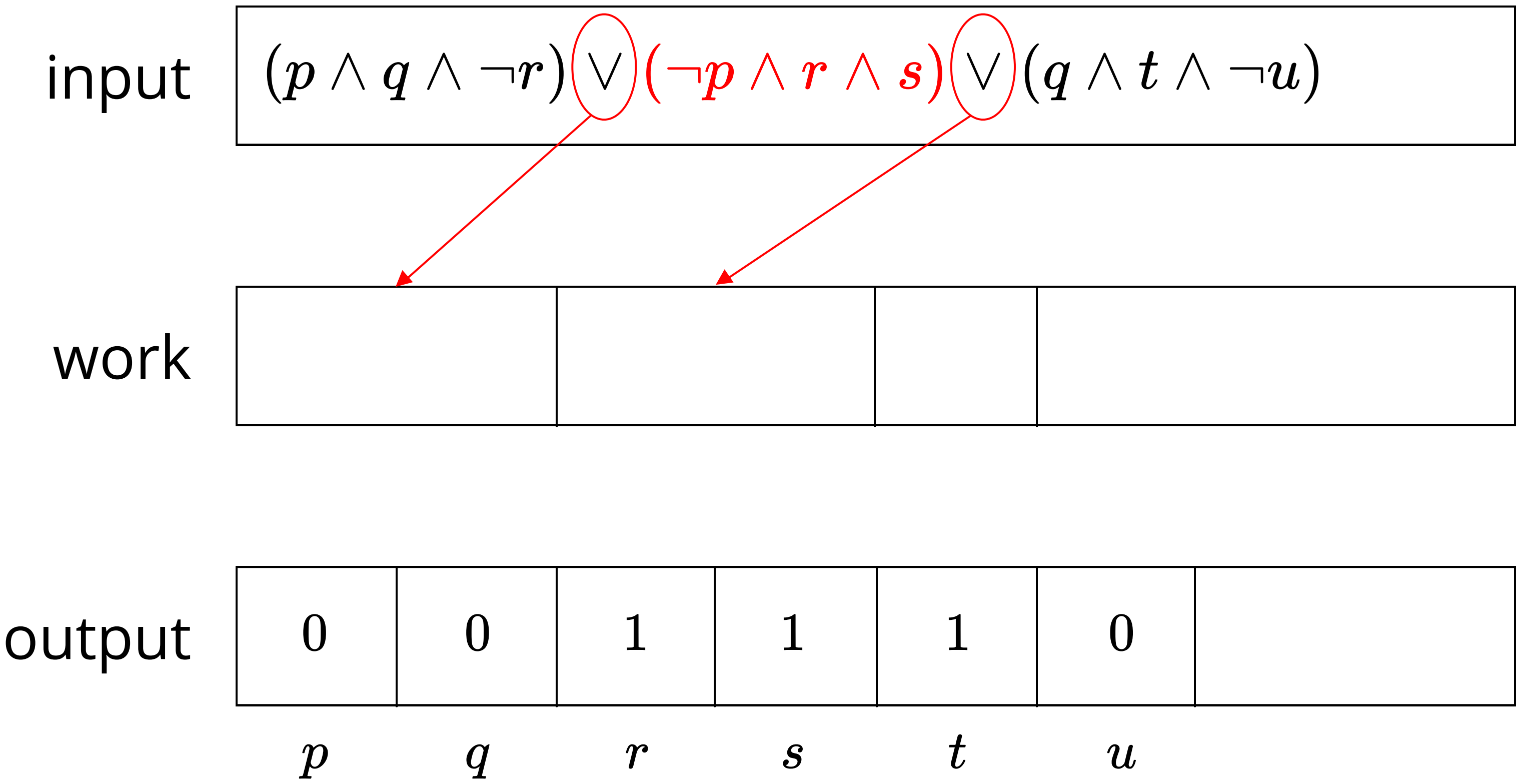
#DNF is in SpanL



#DNF is in SpanL

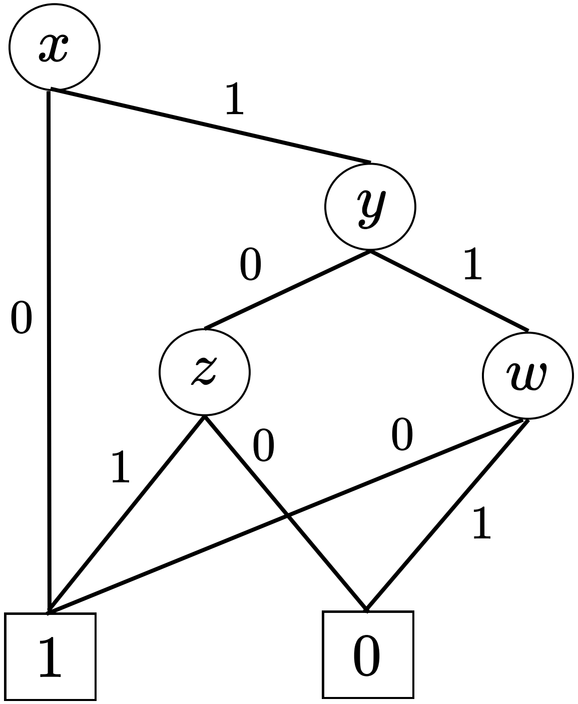


#DNF is in SpanL



#nOBBD is in SpanL

OBDD



$x < y < w < z$

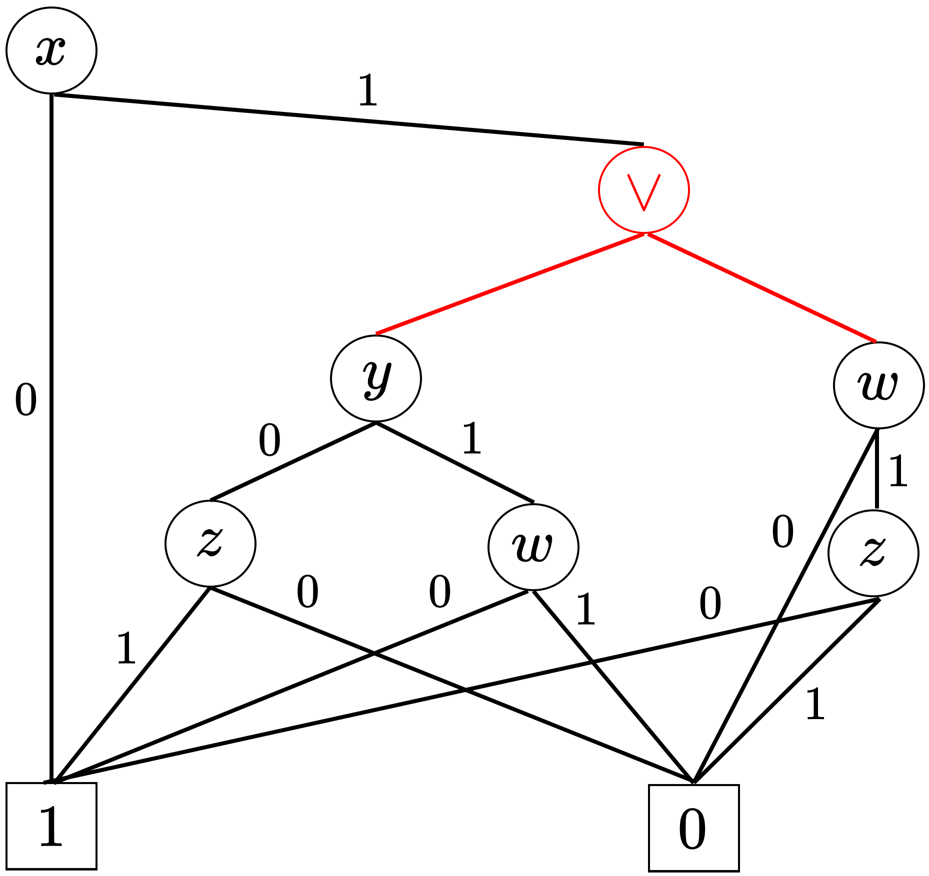
#nOBBD is in SpanL

nOBDD

#nOBBD is in SpanL

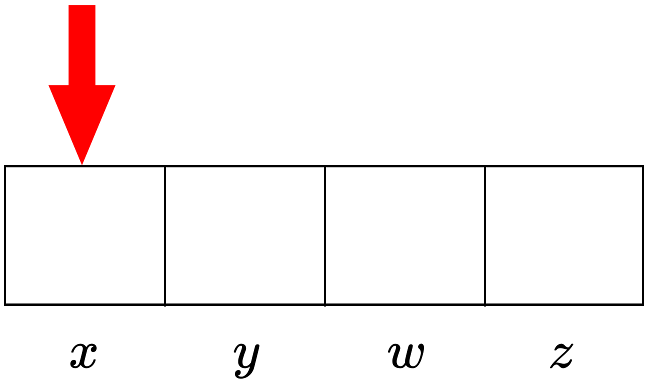
nOBDD

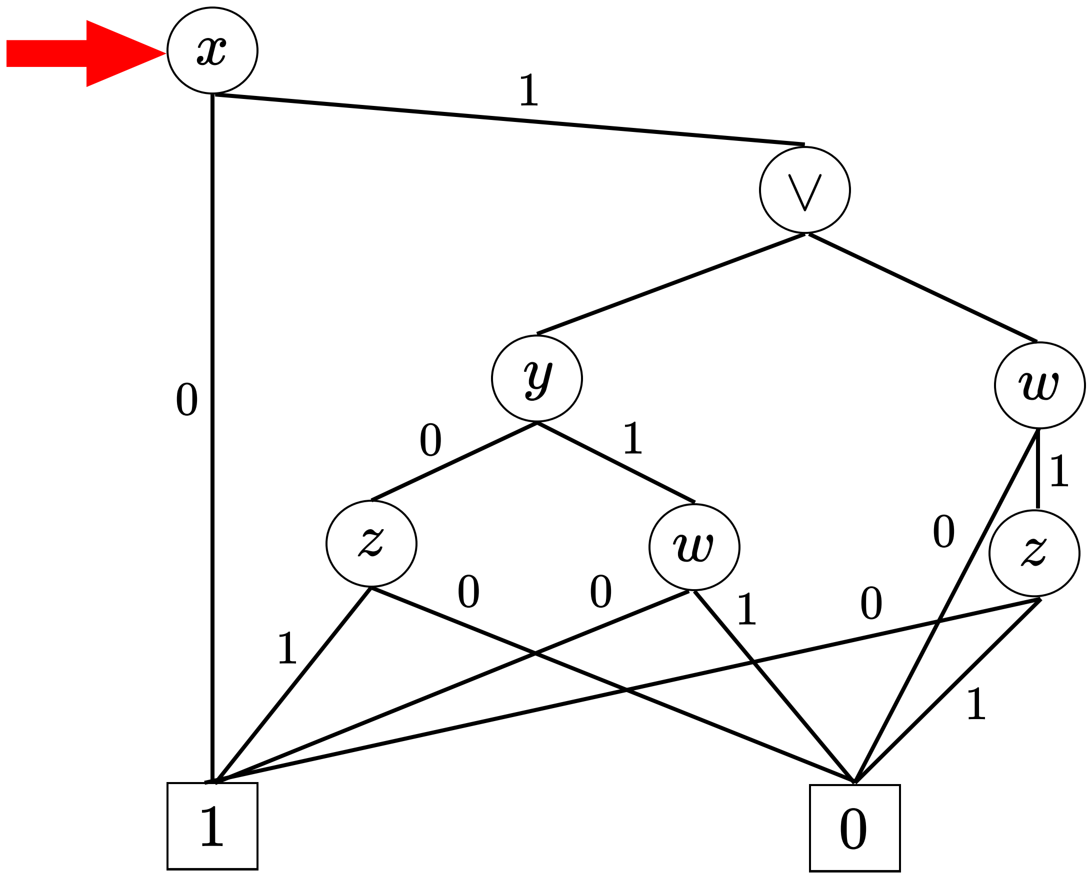
#nOBDD: count the number of satisfying assignments of an nOBDD



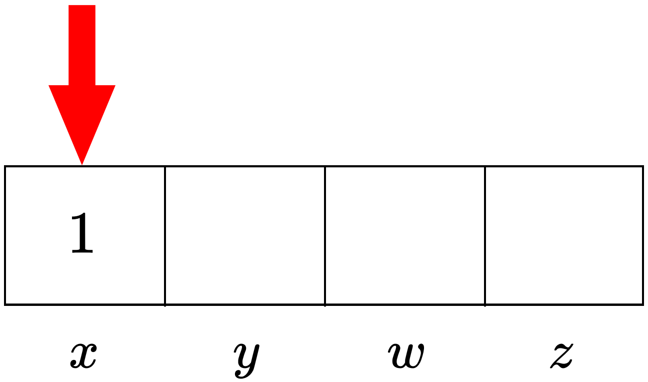
$x < y < w < z$

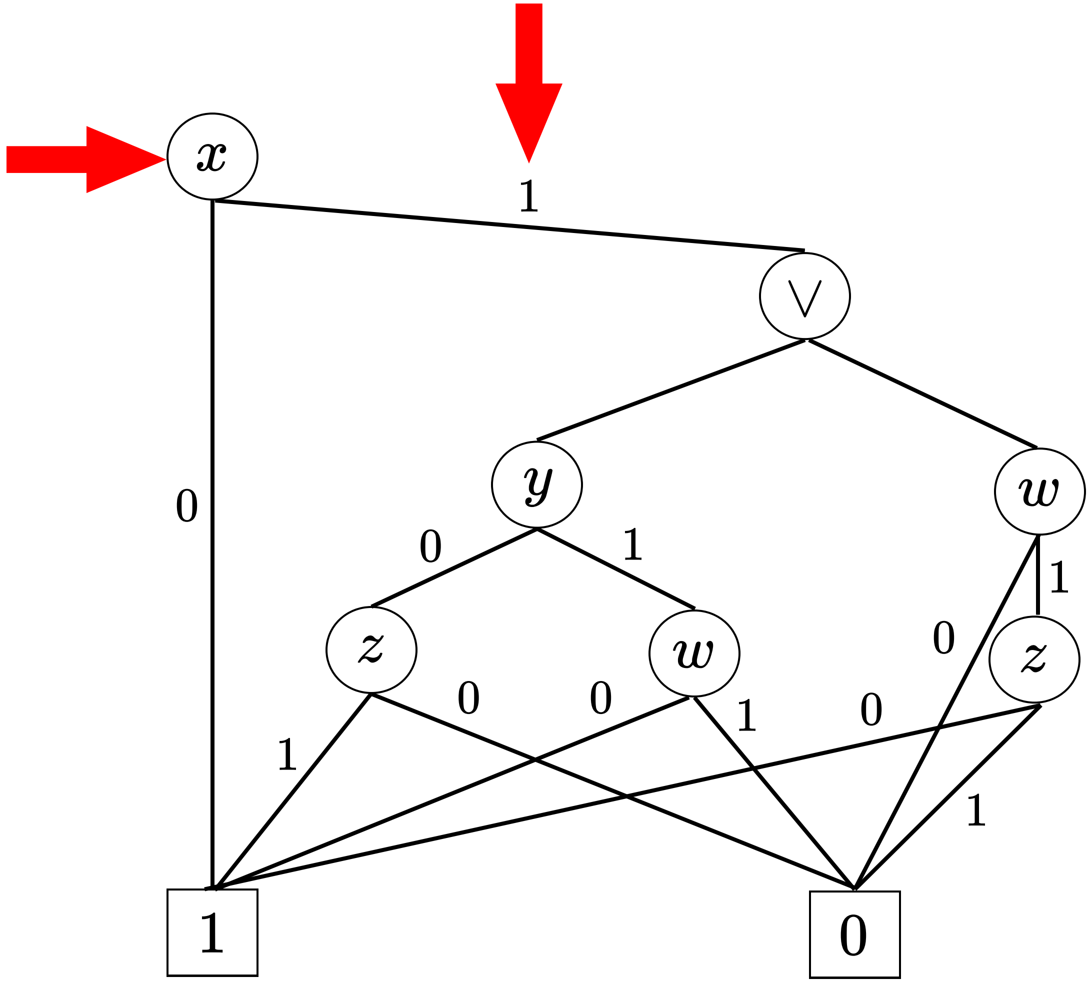
#nOBBD is in SpanL



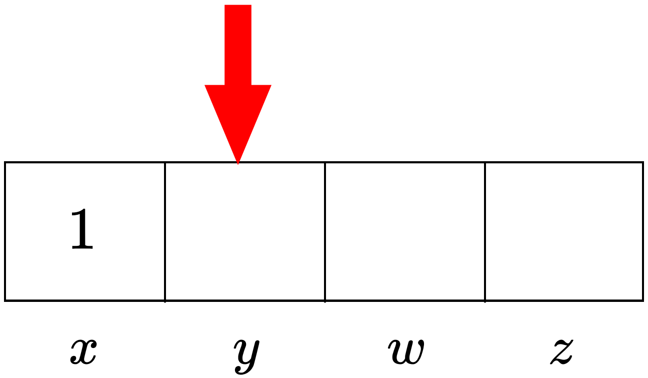


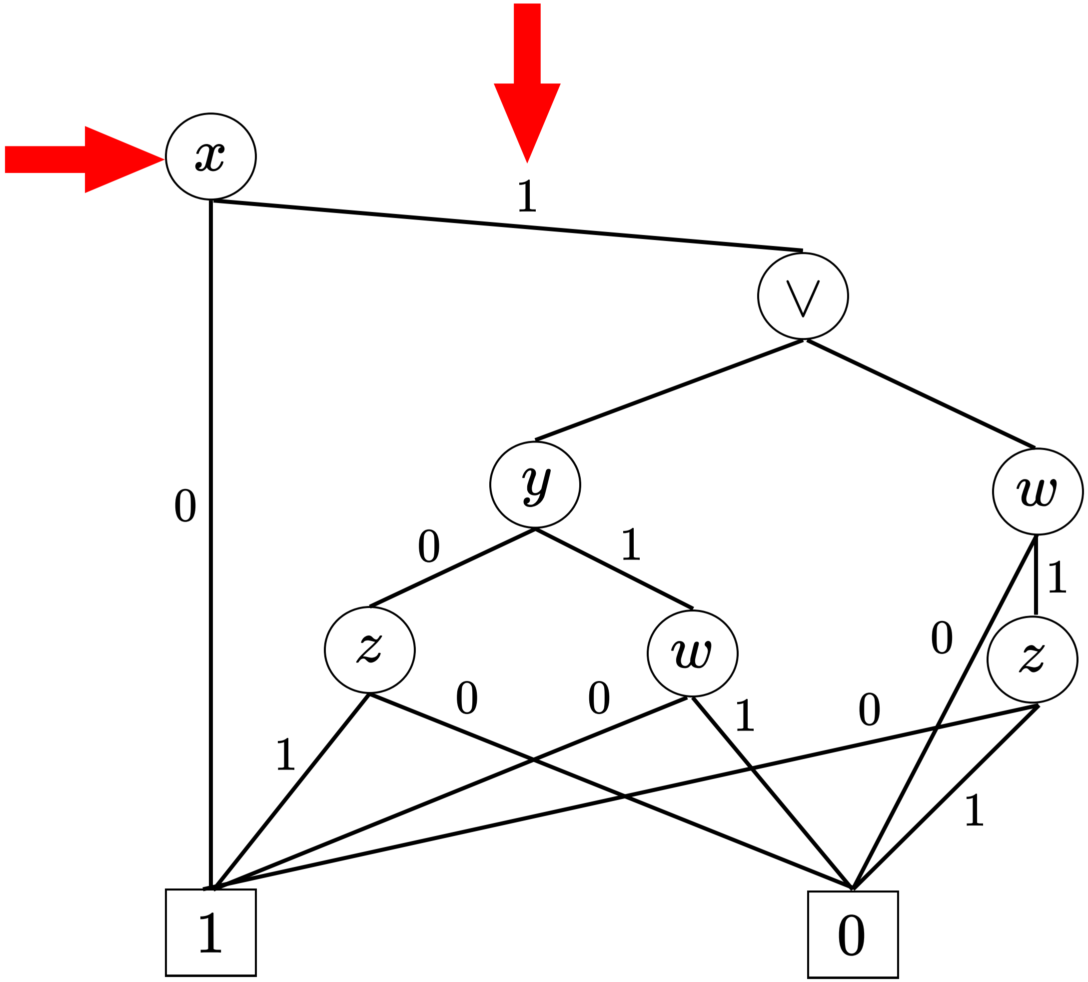
#nOBBD is in SpanL



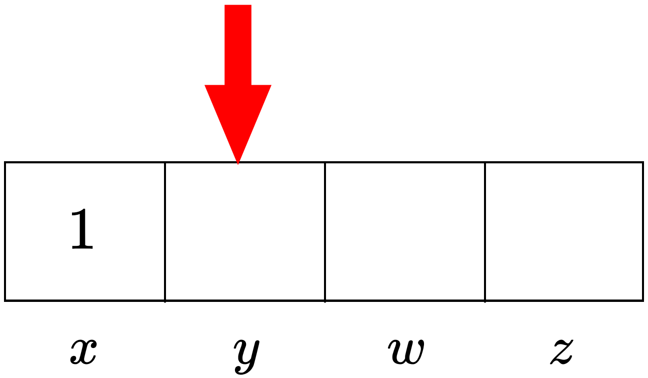


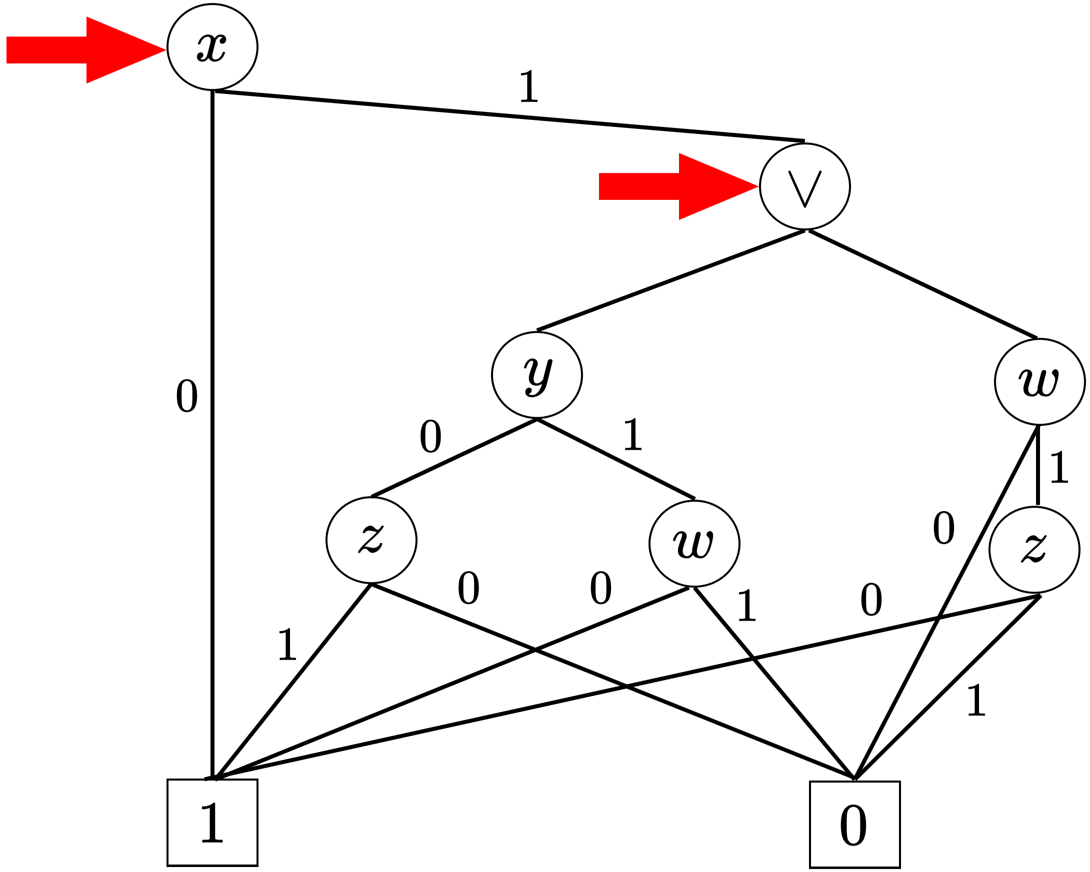
#nOBBD is in SpanL



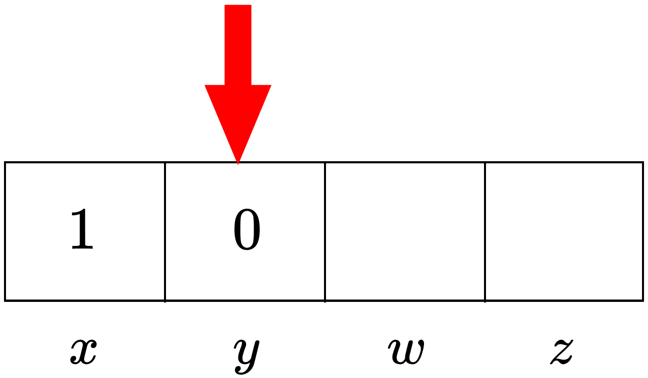


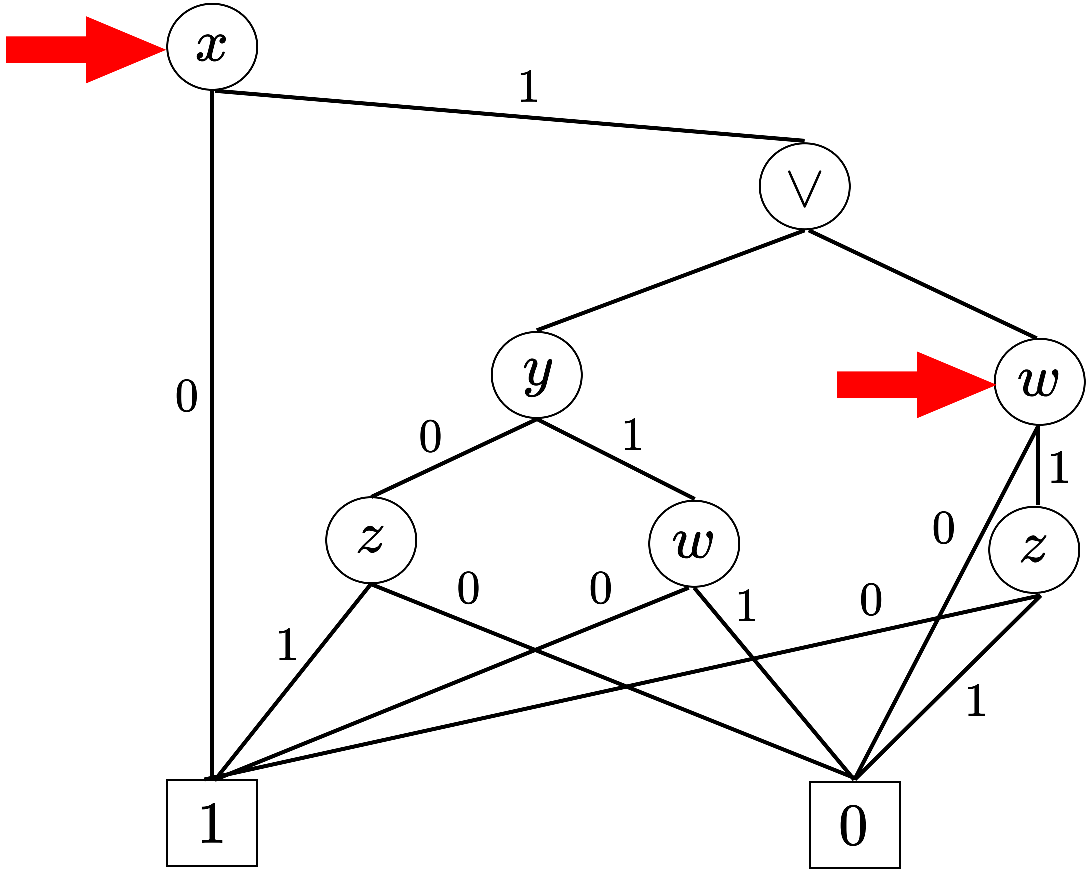
#nOBBD is in SpanL



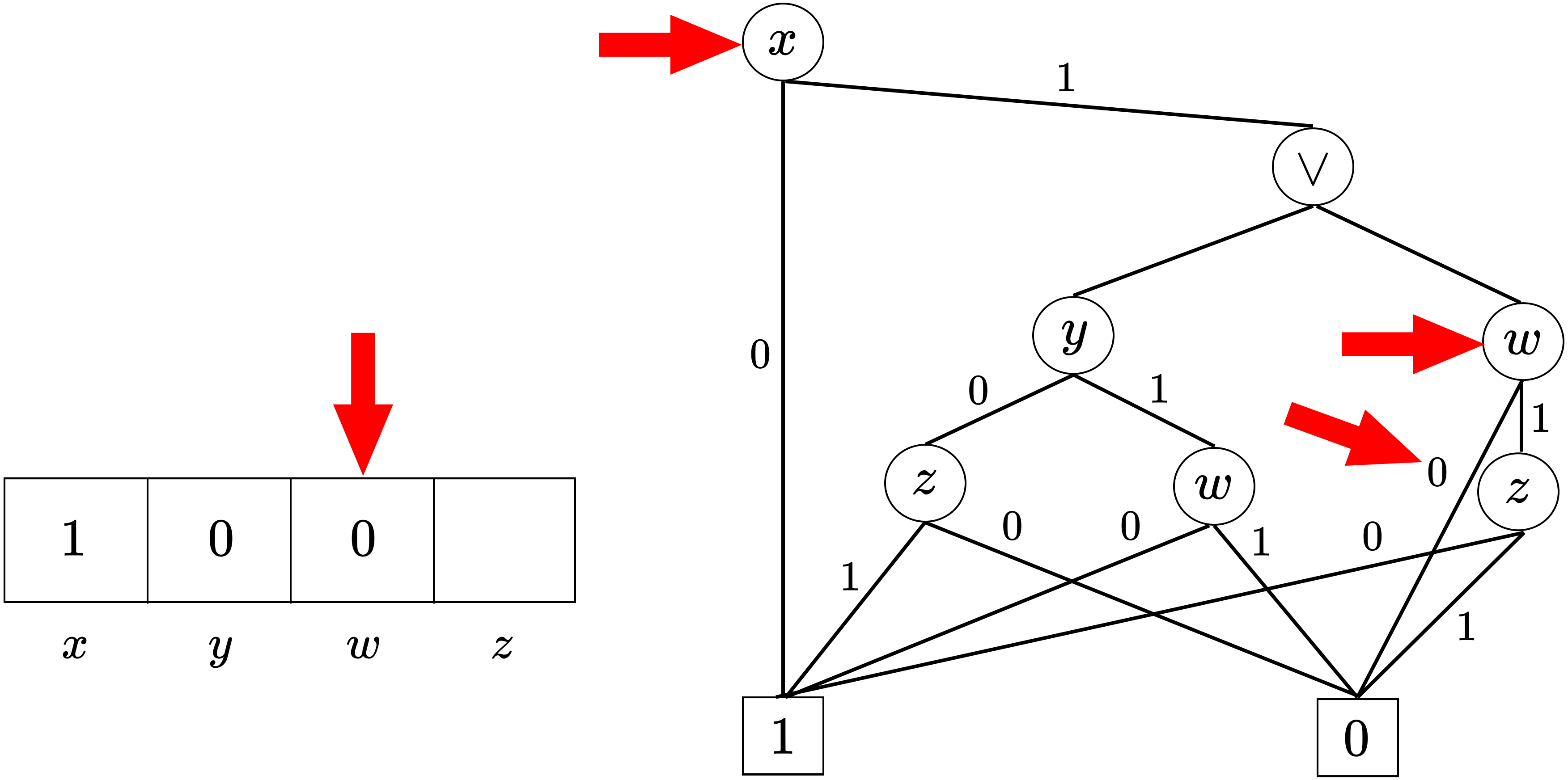


#nOBBD is in SpanL

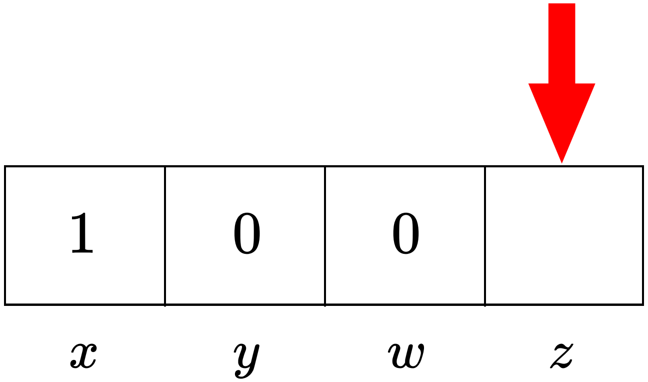


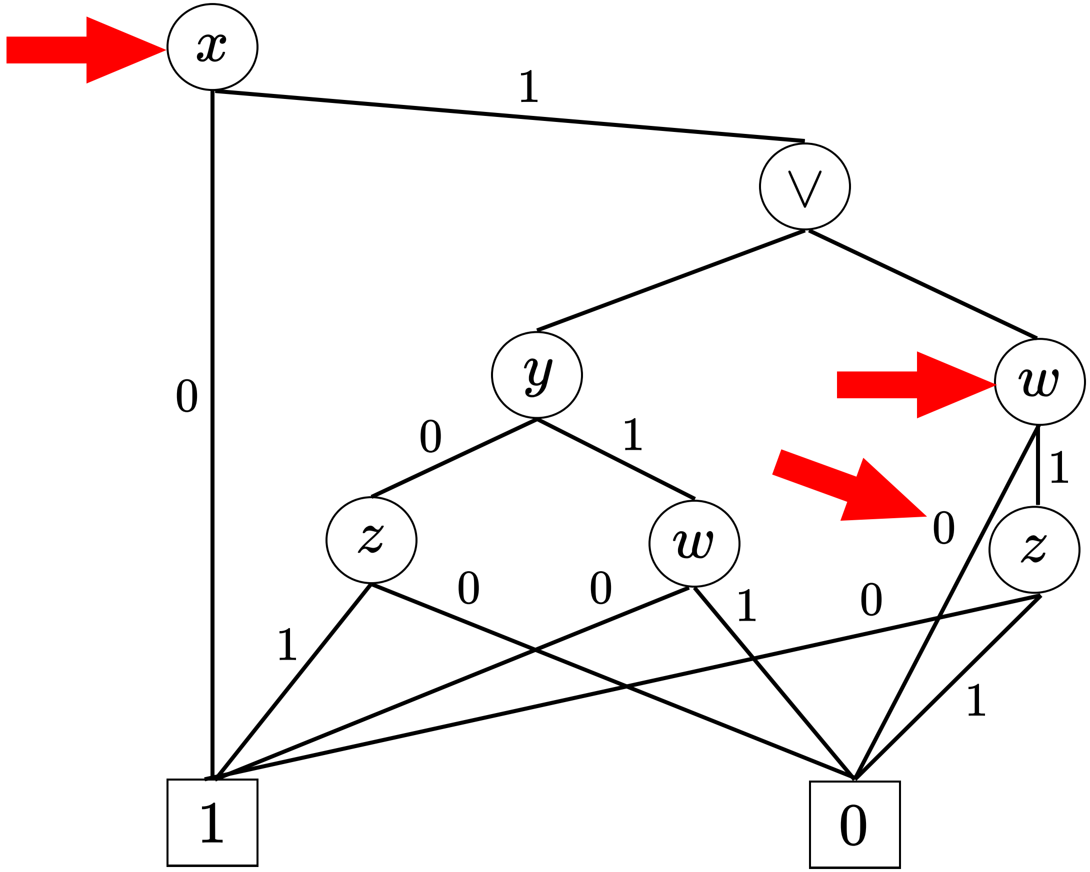


#nOBBD is in SpanL

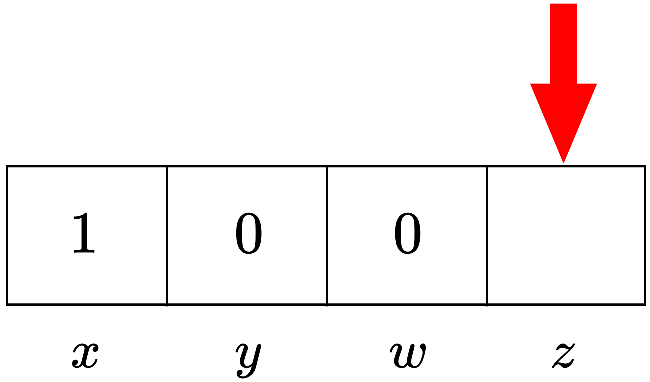


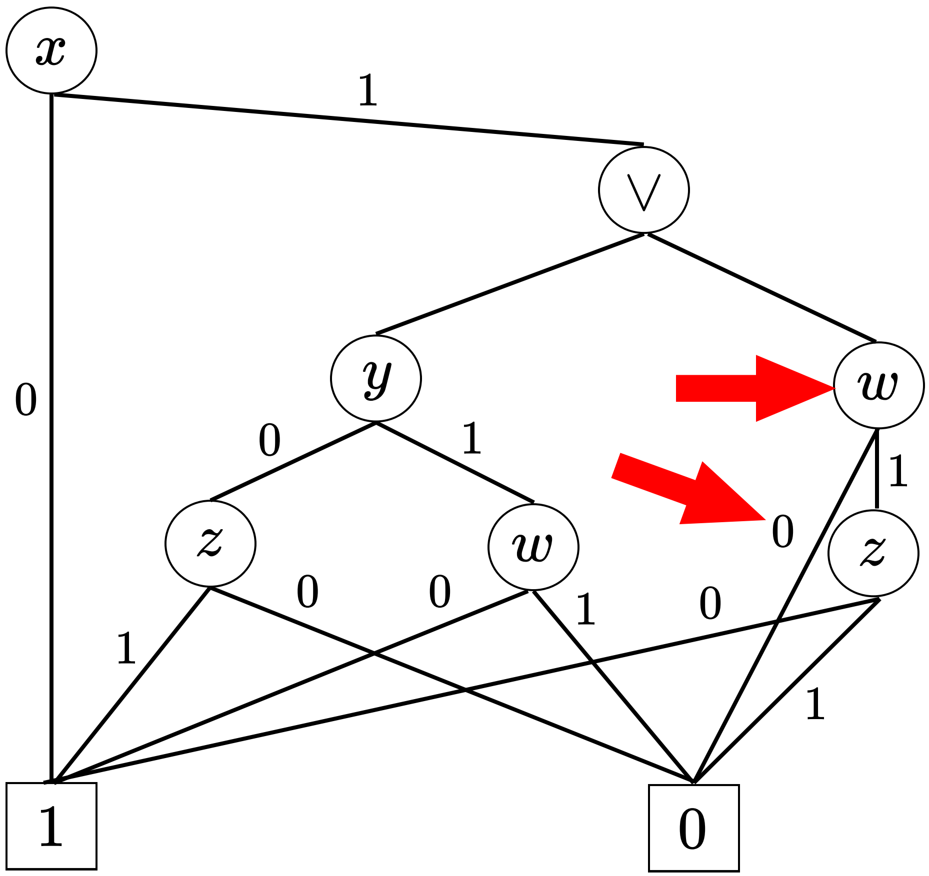
#nOBBD is in SpanL





#nOBBD is in SpanL





Other interesting problems are in SpanL

- Two-terminal network reliability problem on directed acyclic graphs

SpanL gives an alternative approach to prove the existence of an FPRAS for a specific problem

Open problems

- Is #SDNNF in SpanL?
- Can these approaches based on automata be made practical?
- Is #TA complete for a *natural* (and interesting) counting complexity class?
- Does #DNNF admit an FPRAS?
- Does #CFG admit an FPRAS?

Thanks!

Bibliography

[ACJR21a] M. Arenas, L. A. Croquevielle, R. Jayaram, C. Riveros. *#NFA Admits an FPRAS: Efficient Enumeration, Counting, and Uniform Generation for Logspace Classes*. J. ACM 68(6): 48:1-48:40, 2021

[ACJR21b] M. Arenas, L. A. Croquevielle, R. Jayaram, C. Riveros. *When is approximate counting for conjunctive queries tractable?* STOC 2021: 1015-1027

[JVV86] M. Jerrum, L. G. Valiant, V. V. Vazirani. Random Generation of Combinatorial Structures from a Uniform Distribution. Theor. Comput. Sci. 43:169-188, 1986