Making IP=PSPACE practical for Automated Reasoning

Javier Esparza

Joint work with Eszter Couillard, Philipp Czerner and Rupak Majumdar

CAV 23

SAT Competition

SAT Competition 2022

Affiliated with the 25th International Conference on Theory and Applications of Satisfiability Testing taking place on the 2nd - 5th of August 2022 in Haifa, Israel.

Disqualification

A SAT solver will be disqualified if the solver produces a wrong answer. Specifically, if a solver reports UNSAT on an instance that was proven to be SAT by some other solver, or SAT and provides a wrong certificate. A solver disqualified from the competition is not eligible to win any award. Disqualified solvers will be marked as such on the competition results page.

Note that there is a dedicated period when the participants can check their results to ensure that no problems are caused by the competition framework.

SAT Competition

SAT Comp

Affiliated with th
Applications of S
of August 2022 i

Disqualific

A SAT so
Specifically, if a solve
by some other solve
from the competition is not eligible to win any award. Disqualified solvers will be
marked as such on the competition results page.

Note that there is a dedicated period when the participants can check their results to ensure that no problems are caused by the competition framework.

A SAT solver will be disqualified if the solver produces a wrong answer. Specifically, if a solver reports UNSAT on an instance that was proven to be SAT by some other solver, or SAT and provides a wrong assignment.

SAT Competition

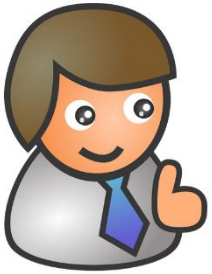
Strategy: if the time limit is approaching and the tool has not found a satisfying assignment, answer UNSAT.

SAT Competition

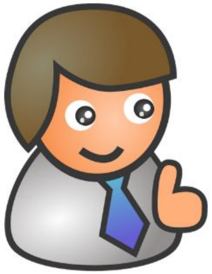
Strategy: if the time limit is approaching and the tool has not found a satisfying assignment, answer UNSAT.

Has been used already!

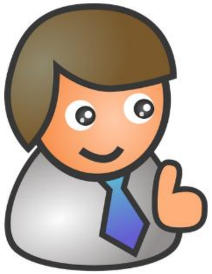
Cloud scenario



Cloud scenario

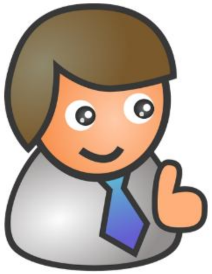


Cloud scenario



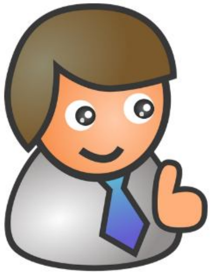
- Please check whether this formula is SAT or UNSAT

Cloud scenario



- Please check whether this formula is SAT or UNSAT
- It'll cost you 10$

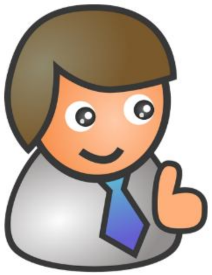
Cloud scenario



- Please check whether this formula is SAT or UNSAT
- OK

- It‘ll cost you 10$

Cloud scenario



- **Please check whether this formula is SAT or UNSAT**
- **OK**

- **It'll cost you 10$**
- (Tries for a while, doesn't find assignment) **It's UNSAT**

Certification

SAT certificate:

Truth assignment that makes the formula true

Checkable on a standard laptop for formulas of GB size

Certification

SAT certificate:

Truth assignment that makes the formula true

Checkable on a standard laptop for formulas of GB size

UNSAT certificate:

Resolution proof, DRAT proof

Certification

SAT certificate:

Truth assignment that makes the formula true

Checkable on a standard laptop for formulas of GB size

UNSAT certificate:

Resolution proof, DRAT proof

- Compulsory in the Main Track of the SAT Competition

Certifying UNSAT

SAT Competition 2022

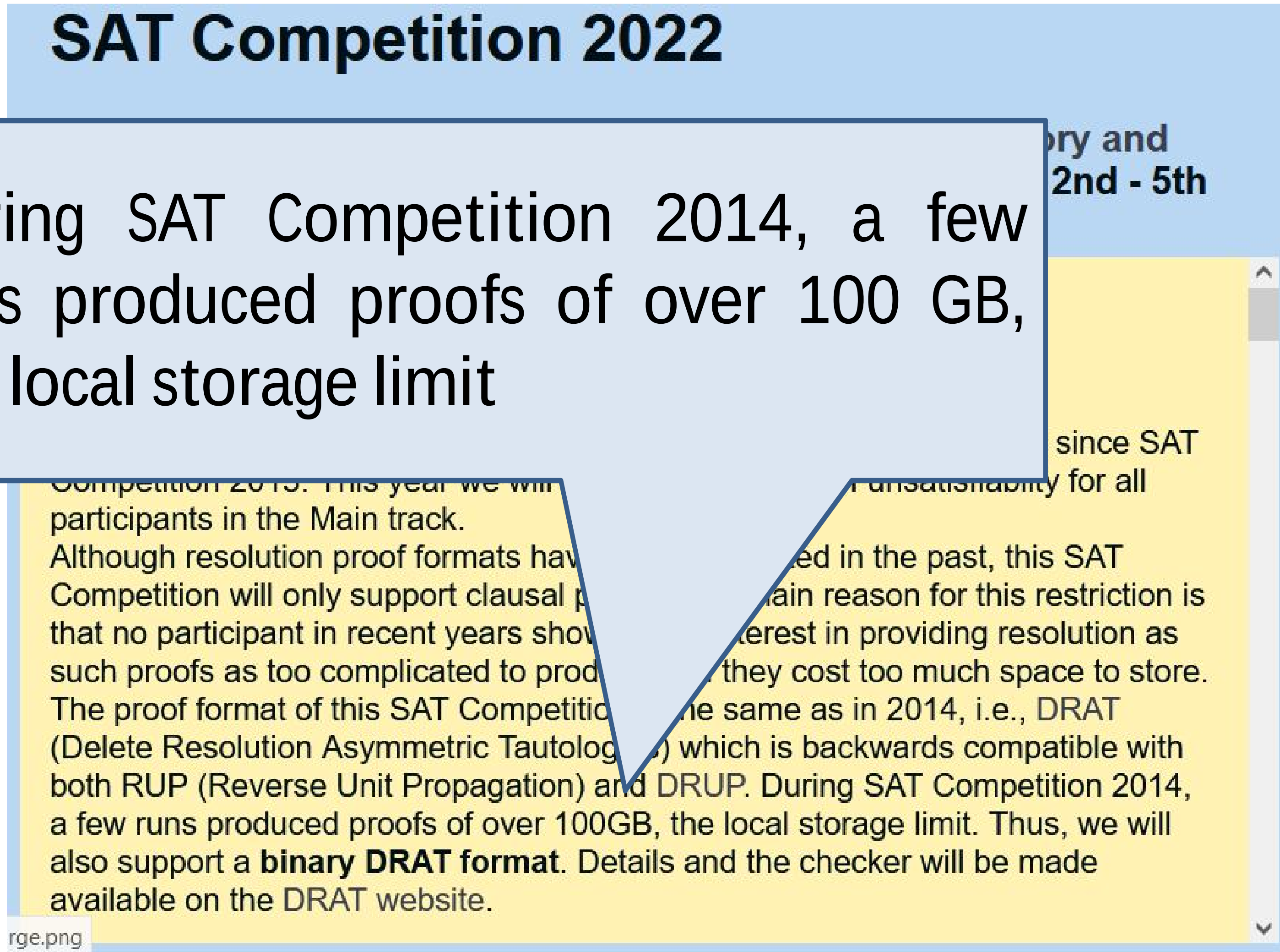
Affiliated with the 25th International Conference on Theory and Applications of Satisfiability Testing taking place on the 2nd - 5th of August 2022 in Haifa, Israel.

Certified UNSAT

Certificates of unsatisfiability have been required for the UNSAT tracks since SAT Competition 2013. This year we will require certificates of unsatisfiablity for all participants in the Main track.
Although resolution proof formats have been supported in the past, this SAT Competition will only support clausal proofs. The main reason for this restriction is that no participant in recent years showed any interest in providing resolution as such proofs as too complicated to produce and they cost too much space to store. The proof format of this SAT Competition is the same as in 2014, i.e., DRAT (Delete Resolution Asymmetric Tautologies) which is backwards compatible with both RUP (Reverse Unit Propagation) and DRUP. During SAT Competition 2014, a few runs produced proofs of over 100GB, the local storage limit. Thus, we will also support a **binary DRAT format**. Details and the checker will be made available on the DRAT website.

rge.png

Certifying UNSAT



During SAT Competition 2014, a few runs produced proofs of over 100 GB, the local storage limit

Certifying UNSAT

Solving and Verifying the boolean Pythagorean Triples problem via Cube-and-Conquer

Marijn J. H. Heule, Oliver Kullmann, and Victor W. Marek

The University of Texas at Austin, Swansea University, and University of Kentucky

Abstract. The *boolean Pythagorean Triples problem* has been a long-standing open problem in Ramsey Theory: Can the set $\mathbb{N} = \{1, 2, \ldots\}$ of natural numbers be divided into two parts, such that no part contains a triple (a, b, c) with $a^2 + b^2 = c^2$? A prize for the solution was offered by Ronald Graham over two decades ago. We solve this problem, proving in fact the impossibility, by using the *Cube-and-Conquer* paradigm, a hybrid SAT method for hard problems, employing both look-ahead and CDCL solvers. An important role is played by dedicated look-ahead heuristics, which indeed allowed to solve the problem on a cluster with 800 cores in about 2 days. Due to the general interest in this mathematical problem, our result requires a formal proof. Exploiting recent progress in unsatisfiability proofs of SAT solvers, we produced and verified a proof in the DRAT format, which is almost 200 terabytes in size. From this we extracted and made available a compressed certificate of 68 gigabytes, that allows anyone to reconstruct the DRAT proof for checking.

Certifying UNSAT

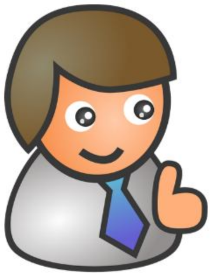
Solving and Verifying the boolean Pythagorean Triples problem via Cube-and-Conquer

Marijn J. H. Heule, Oliver Kullmann, and Victor W. Marek

> … we produced and verified a proof in the DRAT format, which is almost 200 TB in size. From this we extracted […] a compressed certificate of 68 gigabytes …

heuristics, which indeed allowed to solve t … uster with 800 cores in about 2 days. Due to the ge … n this mathemati-cal problem, our result requires a forma … Exploiting recent progress in unsatisfiability proofs of SAT solv…s, we produced and verified a proof in the DRAT format, which is almost 200 terabytes in size. From this we extracted and made available a compressed certificate of 68 gigabytes, that allows anyone to reconstruct the DRAT proof for checking.

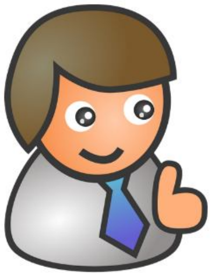
Cloud scenario again



- **Please check whether this formula is SAT or UNSAT**
- **OK**

- **It‘ll cost you 10$**
- (Tries for a while, doesn‘t find assignment) **It‘s UNSAT**

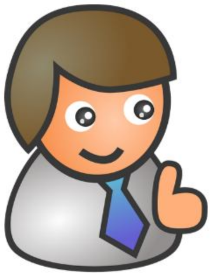
Cloud scenario again



- **Please check whether this formula is SAT or UNSAT**
- **OK**
- **Oh, can I please have a certificate?**

- **It‘ll cost you 10$**
- (Tries for a while, doesn‘t find assignment) **It‘s UNSAT**

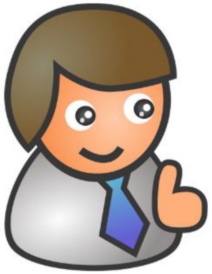
Cloud scenario again



- **Please check whether this formula is SAT or UNSAT**
- **OK**
- **Oh, can I please have a certificate?**

- **It'll cost you 10$**
- (Tries for a while, doesn't find assignment) **It's UNSAT**
- **Sure! I can send you a 10 TB proof for only 10$ more!**

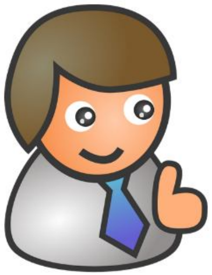
Cloud scenario again



- **Please check whether this formula is SAT or UNSAT**
- **OK**
- **Oh, can I please have a certificate?**
- **But I can‘t check that!**

- **It‘ll cost you 10$**
- (Tries for a while, doesn‘t find assignment) **It‘s UNSAT**
- **Sure! I can send you a 10 TB proof for only 10$ more!**

Cloud scenario again



- **Please check whether this formula is SAT or UNSAT**
- **OK**
- **Oh, can I please have a certificate?**
- **But I can‘t check that!**

- **It‘ll cost you 10$**
- (Tries for a while, doesn‘t find assignment) **It‘s UNSAT**
- **Sure! I can send you a 10 TB proof for only 10$ more!**
- **No problem, for another 20$ we also check it for you!**

Cloud scenario again

How can we help?
Can we produce smaller proofs?

- Pleas… form…
- OK
- Oh, c… certi…
- But I can‘t check that!

- …oesn‘t
- …‘s UNSAT
- …u a 10 TB … more!
- No problem, for another 20$ we also check it for you!

Error in SPIN (CAV 2019)

What's Wrong with On-the-Fly Partial Order Reduction

Stephen F. Siegel(✉)

University of Delaware, Newark, DE, USA
siegel@udel.edu

Abstract. Partial order reduction and on-the-fly model checking are well-known approaches for improving model checking performance. The two optimizations interact in subtle ways, so care must be taken when using them in combination. A standard algorithm combining the two optimizations, published over twenty years ago, has been widely studied and deployed in popular model checking tools. Yet the algorithm is incorrect. Counterexamples were discovered using the Alloy analyzer. A fix for a restricted class of property automata is proposed.

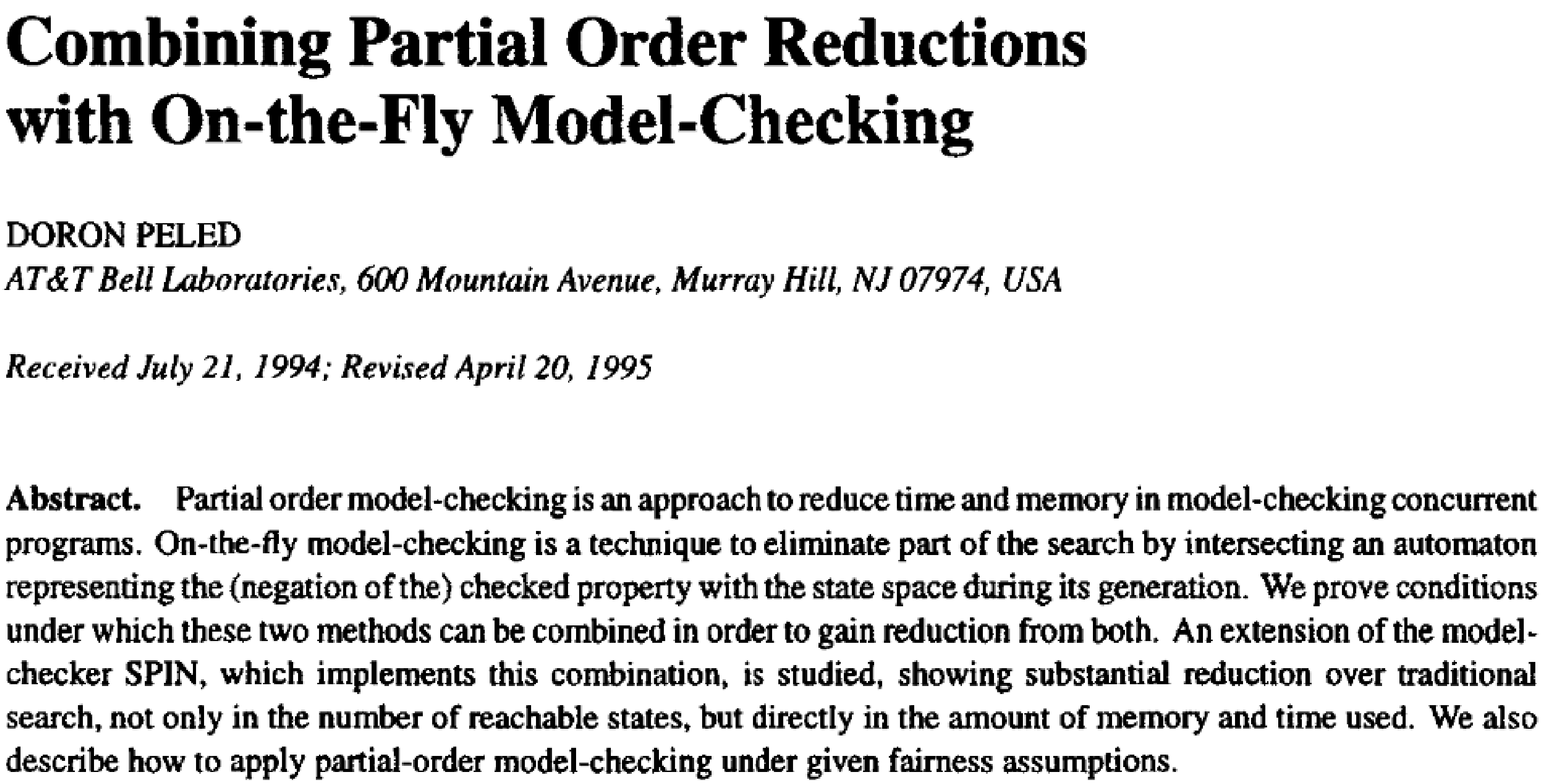
Error in SPIN (CAV 2019)

What's Wrong with On-the-Fly Partial Order Reduction

Partial order reduction and on-the-fly model checking [...] interact in subtle ways [...]. A standard algorithm combining the two optimizations, published over twenty years ago [1995], has been widely studied and deployed in popular model checking tools [SPIN]. Yet the algorithm is incorrect.

using them ... A standard algorithm combining the two optimizations, pu...ed over twenty years ago, has been widely studied and deployed in popular model checking tools. Yet the algorithm is incorrect. Counterexamples were discovered using the Alloy analyzer. A fix for a restricted class of property automata is proposed.

Error in SPIN (CAV 2019)


Combining Partial Order Reductions with On-the-Fly Model-Checking

DORON PELED
AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974, USA

Received July 21, 1994; Revised April 20, 1995

Abstract. Partial order model-checking is an approach to reduce time and memory in model-checking concurrent programs. On-the-fly model-checking is a technique to eliminate part of the search by intersecting an automaton representing the (negation of the) checked property with the state space during its generation. We prove conditions under which these two methods can be combined in order to gain reduction from both. An extension of the model-checker SPIN, which implements this combination, is studied, showing substantial reduction over traditional search, not only in the number of reachable states, but directly in the amount of memory and time used. We also describe how to apply partial-order model-checking under given fairness assumptions.

Error in SPIN (CAV 2019)

Theorem 4.2. *The algorithm* **A2** *will return* true *if the program* P *does not satisfy the property* φ, *and* false *otherwise.*

Error in SPIN (CAV 2019)

Theorem 4.2. *The algorithm* **A2** *will return* true *if the program P does not satisfy the property φ, and* false *otherwise.*

Two page proof containing the line:

It is easy to see that $L(\mathcal{A}') = L(G') \cap L(\mathcal{B})$.

Error in SPIN (CAV 2019)

Theorem 4.2. *The algorithm* **A2** *will return* true *if the program P does not satisfy the property φ, and* false *otherwise.*

Two page proof containing the line:

It is easy to see that $L(\mathcal{A}') = L(G') \cap L(\mathcal{B})$.

(Counterexample to the proof by Brunner, counterexample to the theorem by Siegel)

Certification

Proof size problem is even worse.
Can we produce smaller proofs?

Ask a complexity theorist ...

Ask a complexity theorist ...

PSPACE: class of problems decidable by an algorithm that uses polynomial memory

Ask a complexity theorist ...

- **UNSAT** is coNP-complete
- If there are polynomial certificates for **UNSAT** then **NP=coNP**

- The model-checking problem solved by SPIN is PSPACE-complete
- If there are polynomial certificates for it then **NP=PSPACE**

Ask a complexity theorist ...

But why don‘t you just apply the

IP=PSPACE

theorem?

IP=PSPACE (Lund et al 90, Shamir 92)

- **IP**: class of decision problems with

  interactive proof systems

  (interactive certification systems would be a better name.)

Standard (polynomial) certification

- **Prover** computes a fact and wants to prove to **Verifier** that the fact holds.
- **Prover** sends **Verifier** a certificate; an object that **Verifier** can check in polynomial time in the size of the instance.
- Example : SAT, satisfying assignment, polynomial checker

Interactive proof system

Prover

Verifier

Polynomial time

Interactive proof system

Prover

Verifier

Polynomial time

Interactive proof system

Prover

Verifier

φ

Polynomial time

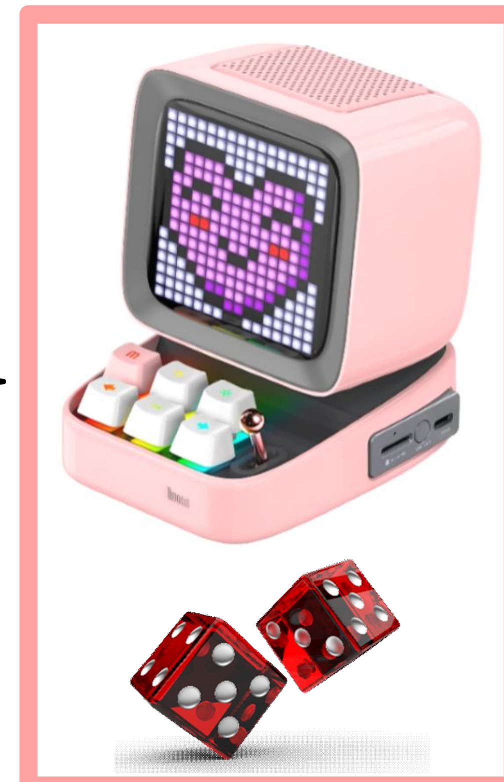
Interactive proof system

Prover

φ

unsat

Verifier



Polynomial time

Interactive proof system

Prover

Challenge 1

Answer 1

Verifier

Polynomial time

Interactive proof system

Prover

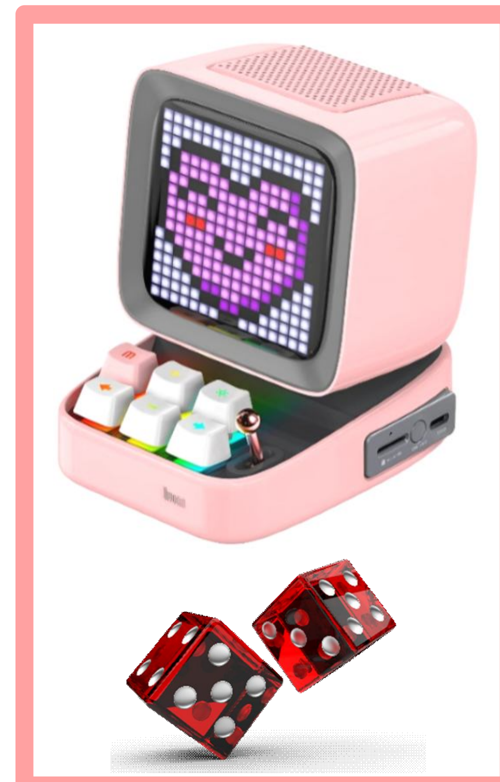
Verifier

Challenge 1

Answer 1

Challenge 2

Answer 2

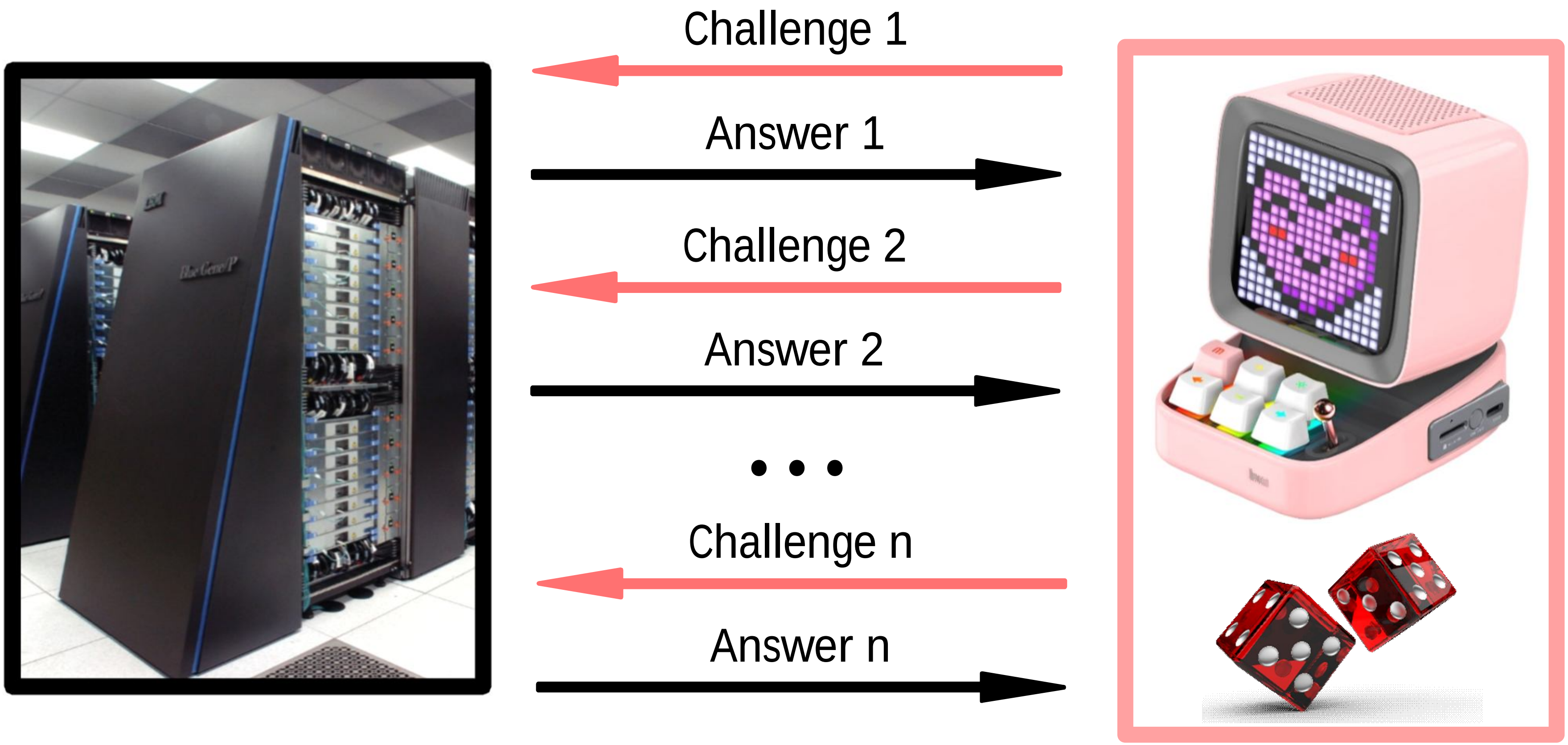


Polynomial time

Interactive proof system

Prover

Verifier



Polynomial time

Interactive proof system

Prover

Verifier

Challenge 1

Verifier can only be fooled by Prover with probability $1/2^{O(n)}$

Polynomial time

IP=PSPACE (Lund et al 90, Shamir 92)

IP = PSPACE

ADI SHAMIR

The Weizmann Institute of Science, Rehovot, Israel

Abstract. In this paper, it is proven that when both randomization and interaction are allowed, the proofs that can be verified in polynomial time are exactly those proofs that can be generated with polynomial space.

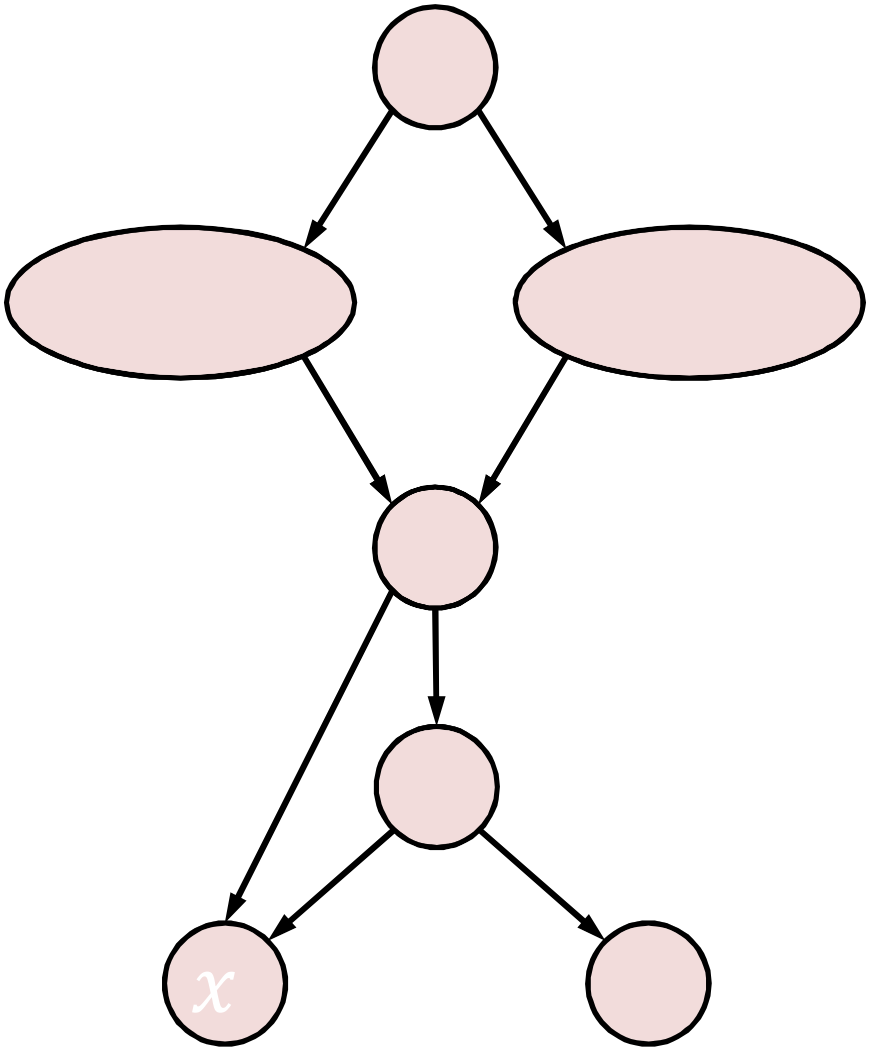
JACM 92

IP=PSPACE (Lund et al 90, Shamir 92)

- To prove PSPACE $\subseteq$ IP (the interesting part), Shamir gives an interactive proof system for QBF (Quantified Boolean Formulas)

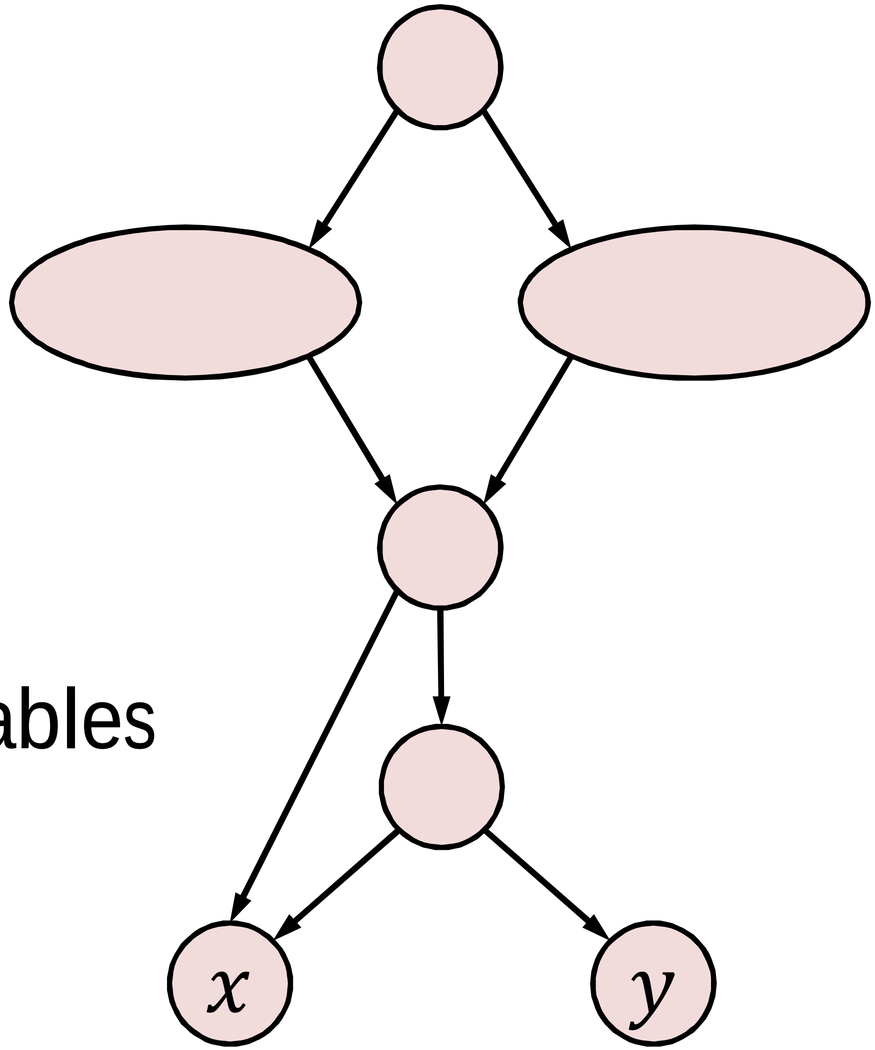
Extended Boolean Circuits: Syntax

Acyclic graph



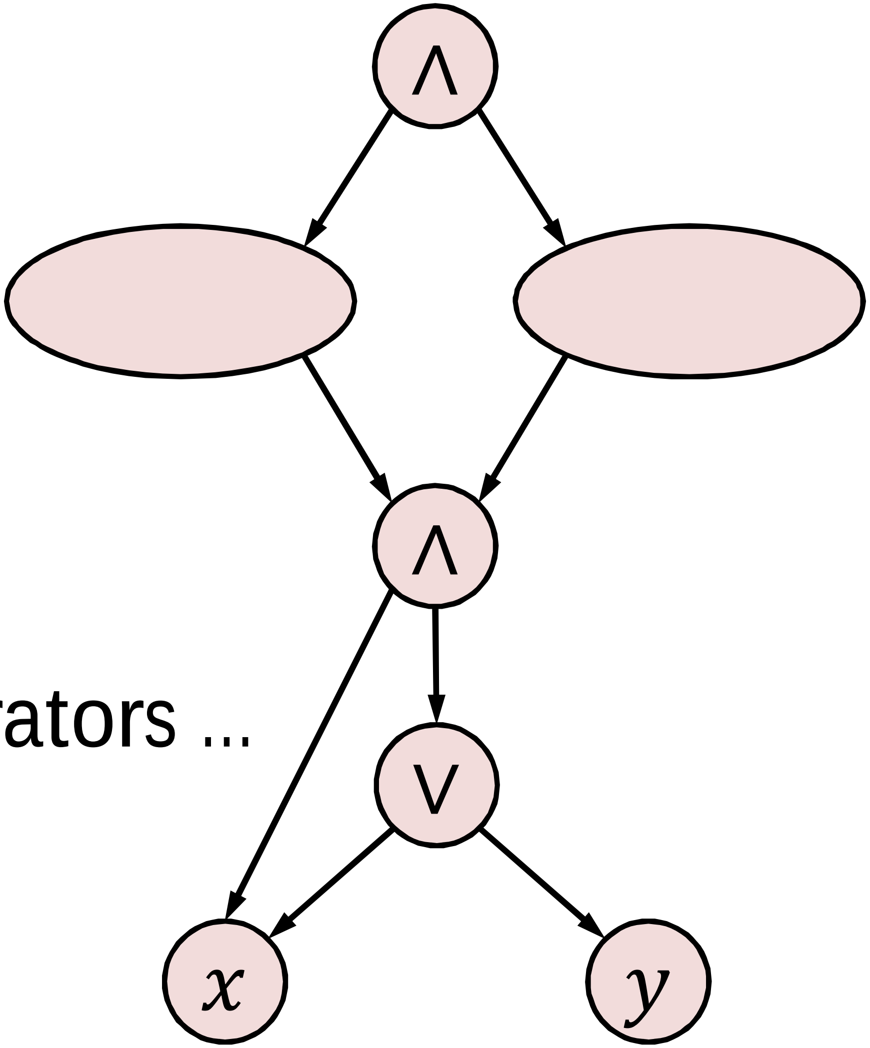
Extended Boolean Circuits: Syntax

Leaves labelled with Boolean variables



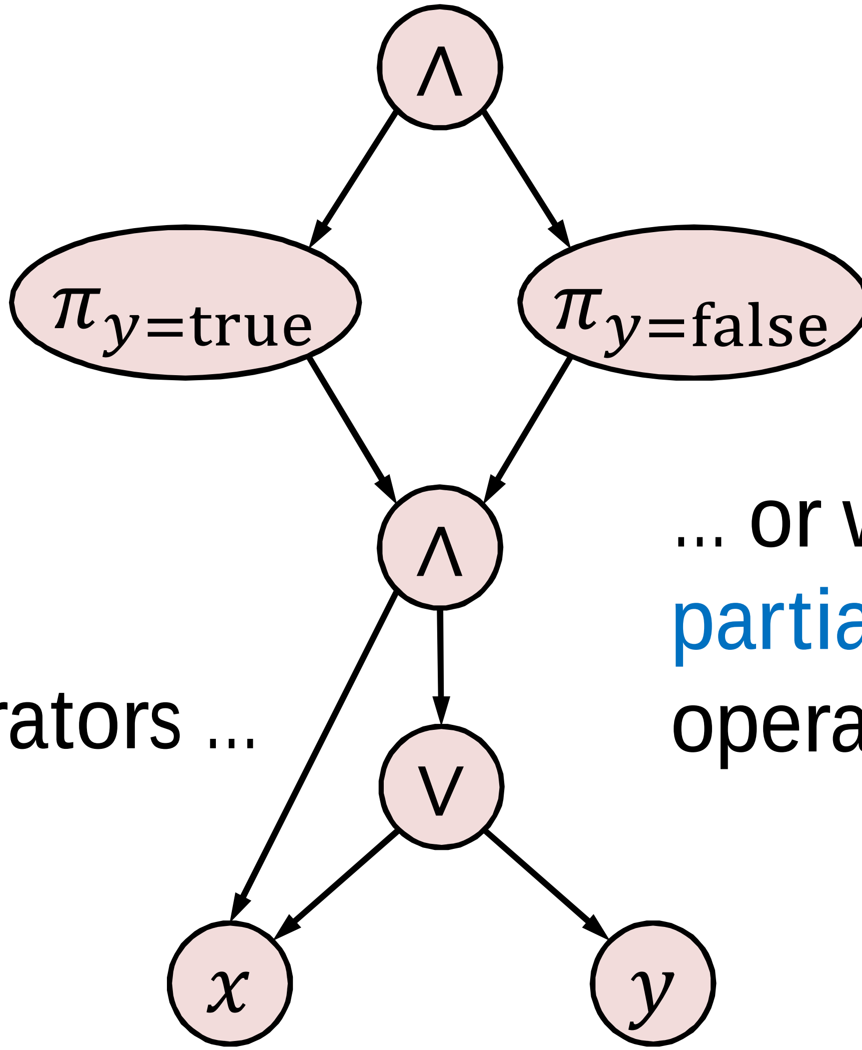
Extended Boolean Circuits: Syntax

Inner nodes labelled with Boolean operators ...



Extended Boolean Circuits: Syntax

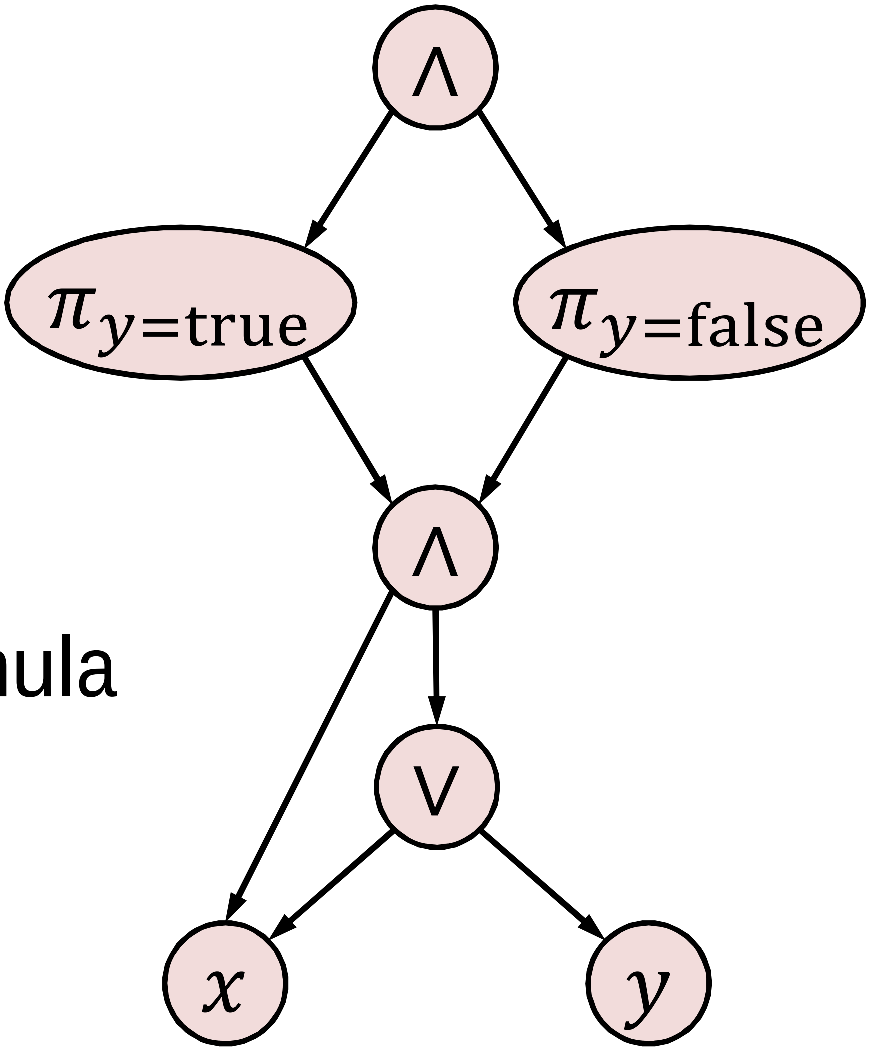
Inner nodes labelled with Boolean operators ...



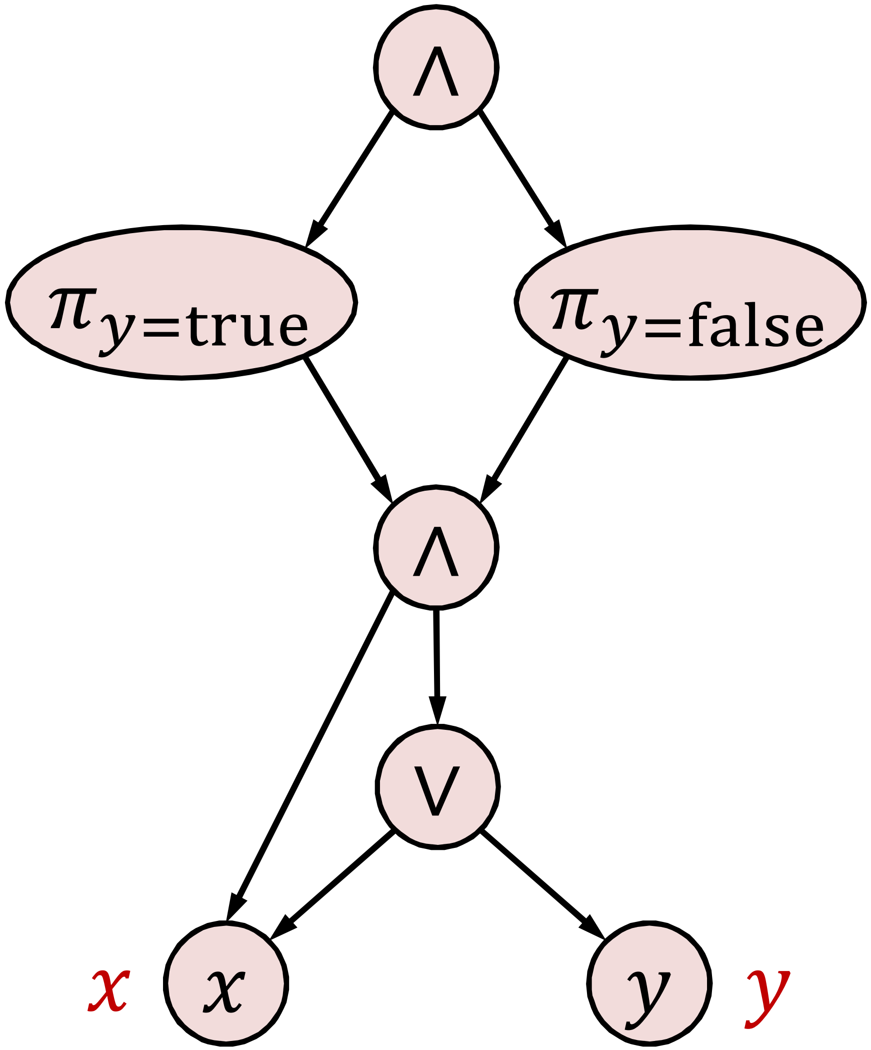
... or with partial evaluation operators.

Extended Boolean Circuits: Semantics

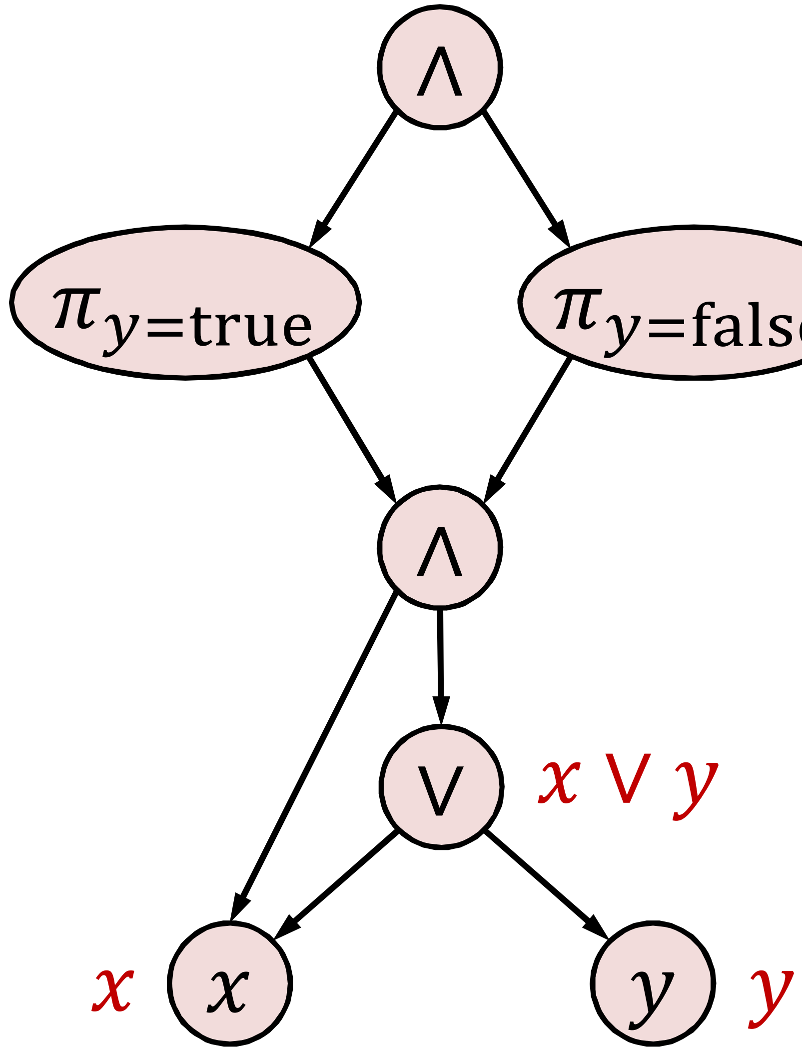
Semantics:
Boolean formula



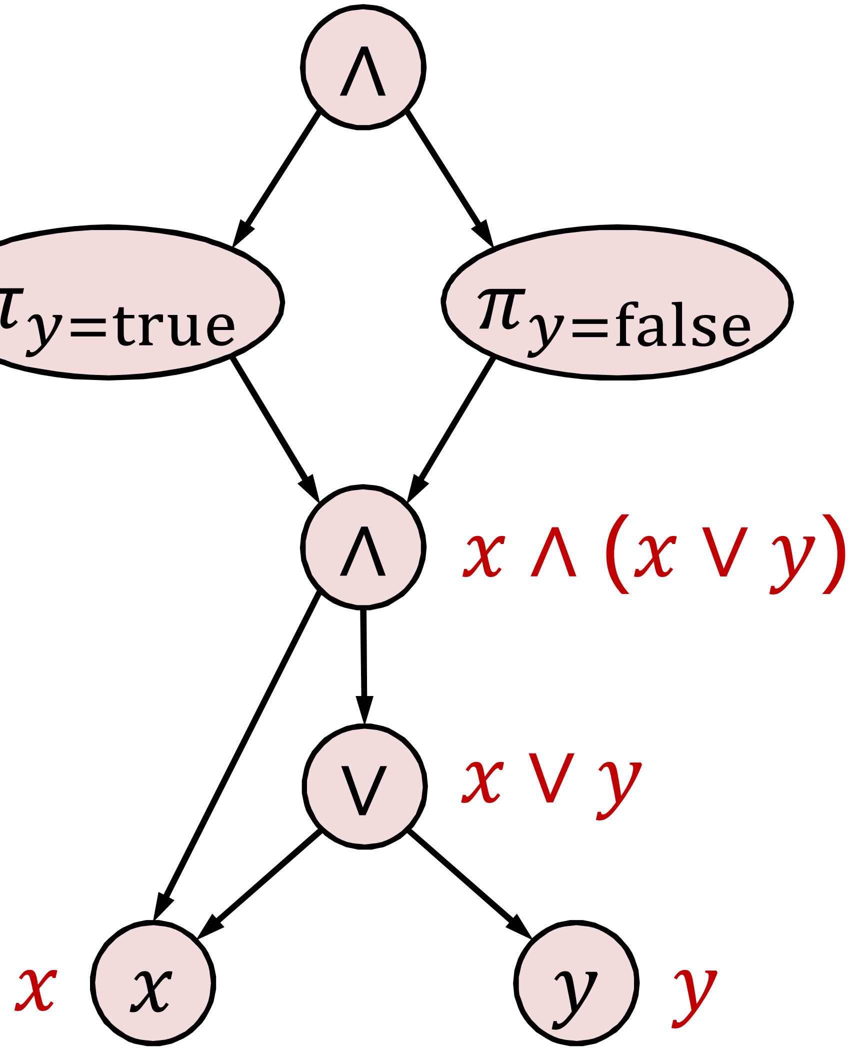
Extended Boolean Circuits: Semantics



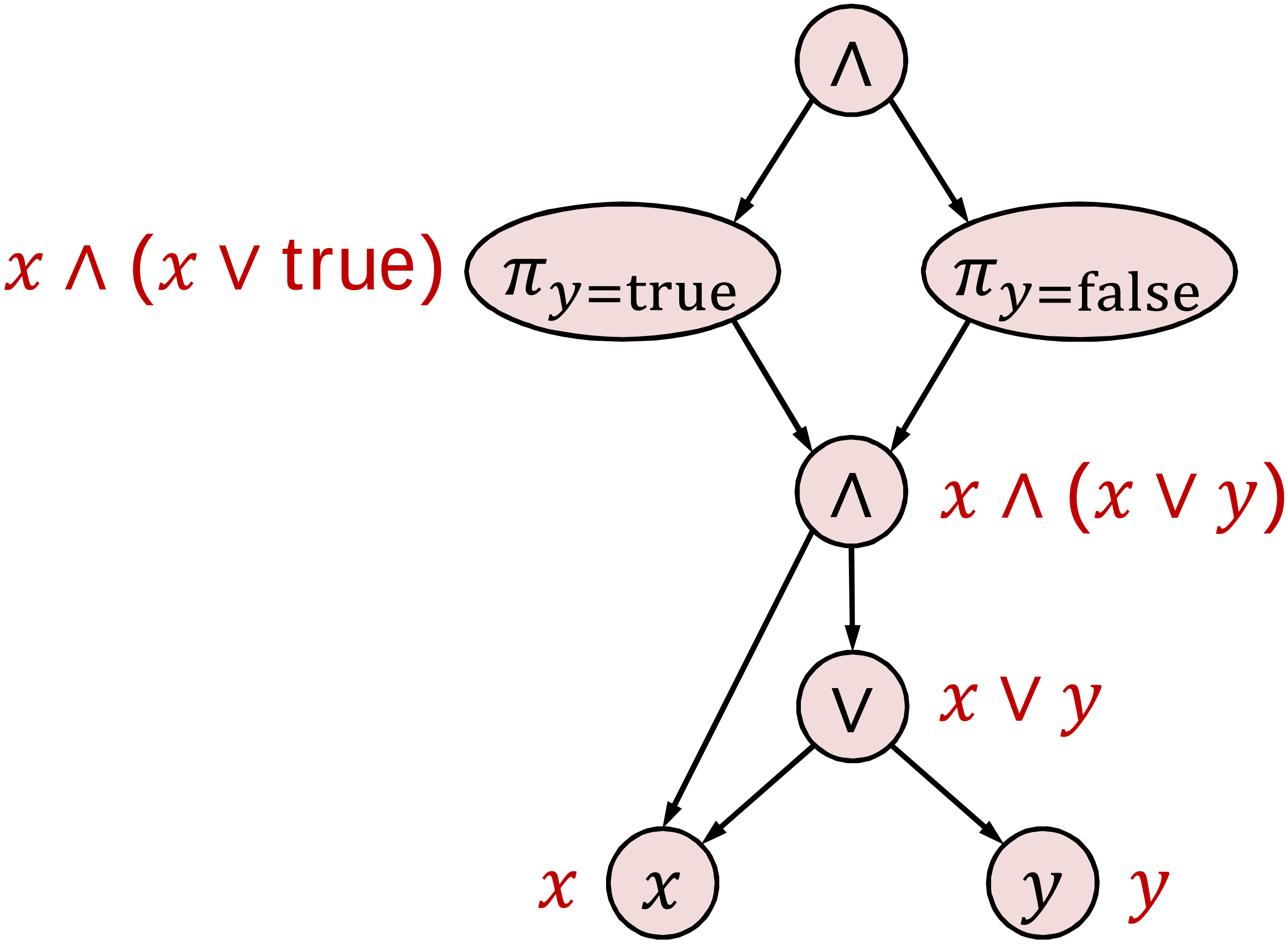
Extended Boolean Circuits: Semantics



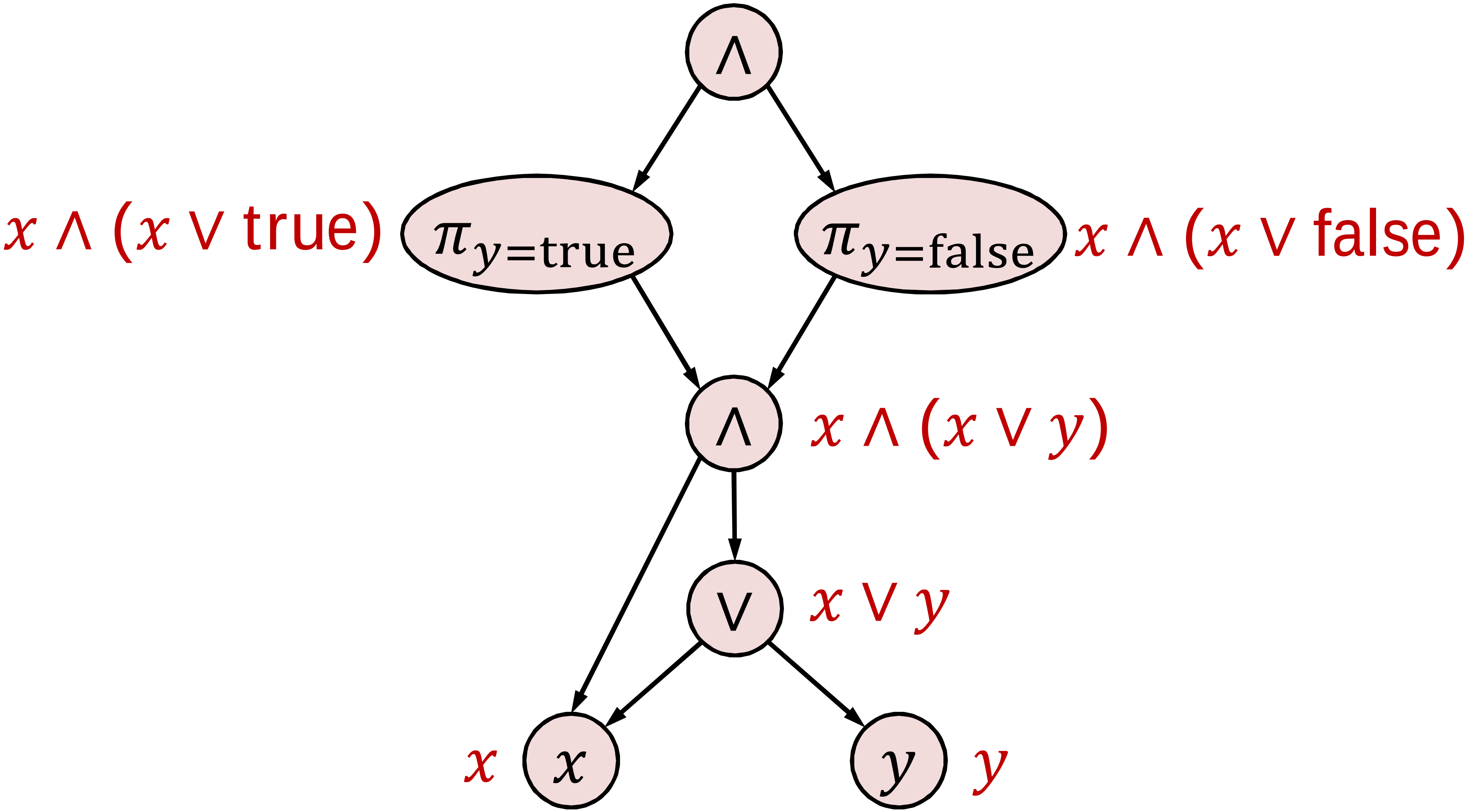
Extended Boolean Circuits: Semantics



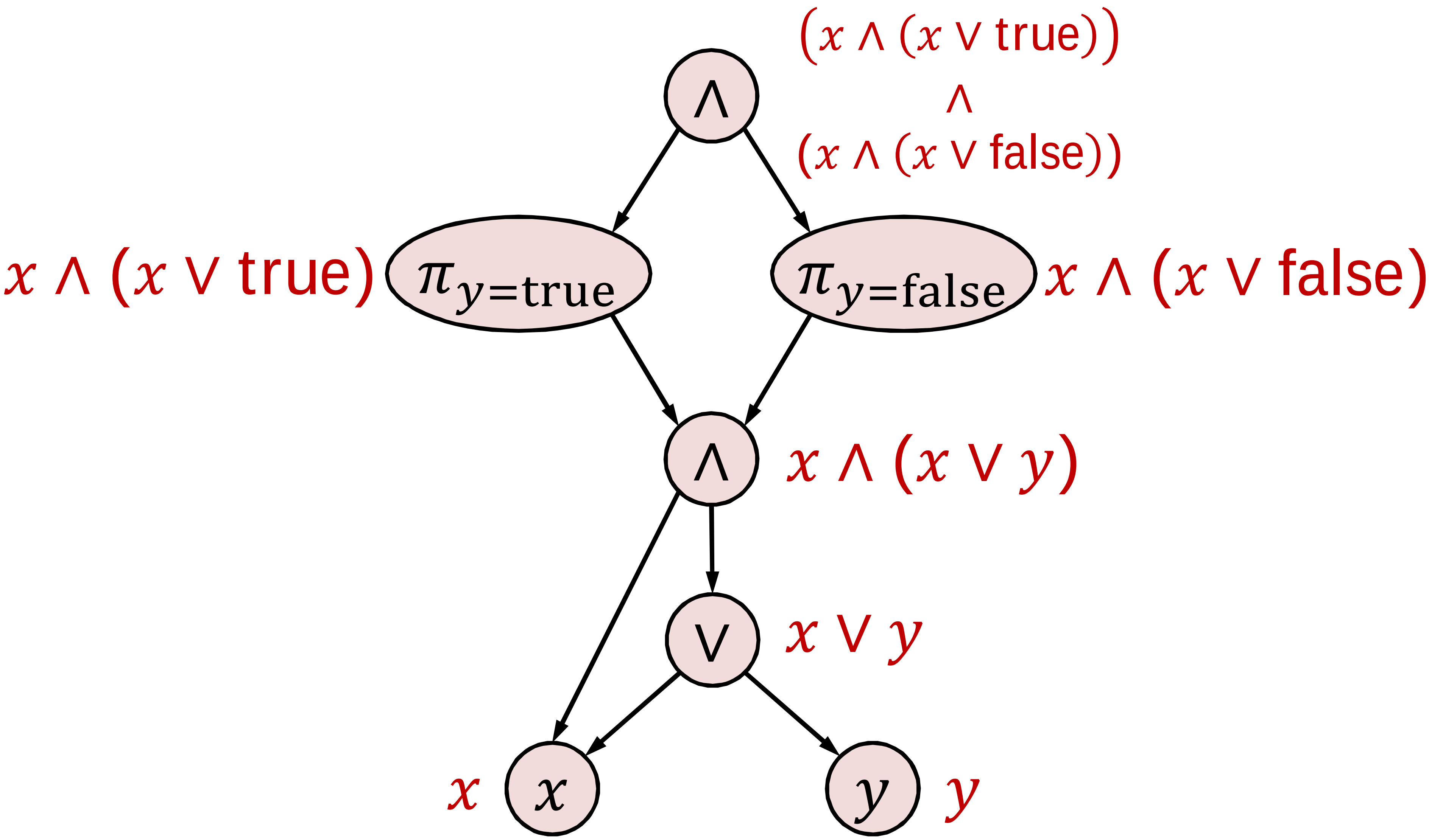
Extended Boolean Circuits: Semantics



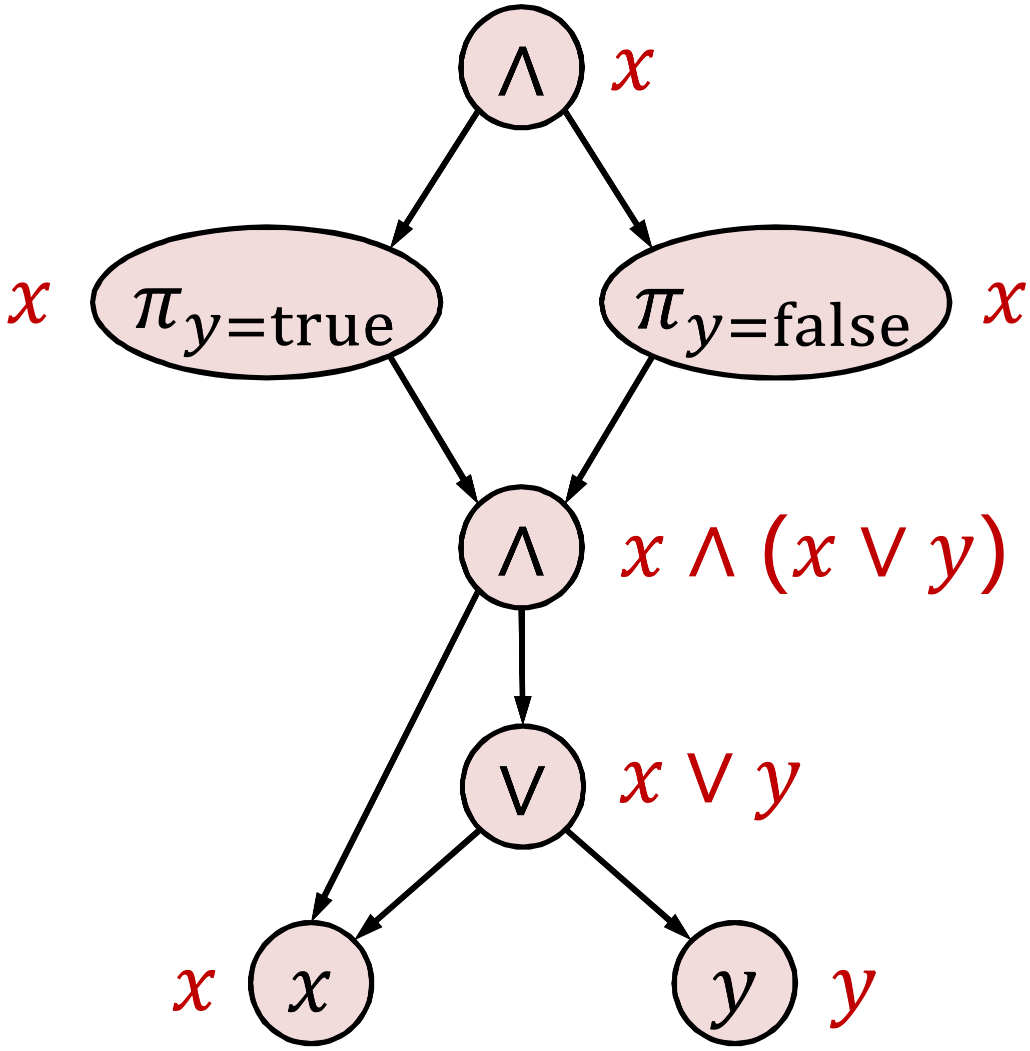
Extended Boolean Circuits: Semantics



Extended Boolean Circuits: Semantics



Extended Boolean Circuits: Semantics



The circuit satisfiability problem

EBC

Input: A (extended boolean) circuit φ

Output: Is (the formula of) φ satisfiable?

An interactive protocol for EBC is also an interactive protocol for QBF

Towards EBC $\in$ IP: Arithmetization

- Fix a finite field $\mathbb{F}$
- Goal: assign to a circuit φ

  a polynomial p_φ

  over $\mathbb{F}$ such that

  φ unsatisfiable iff $p_\varphi = 0$

Towards EBC ∈ IP: Arithmetization

$$p_x : = x$$

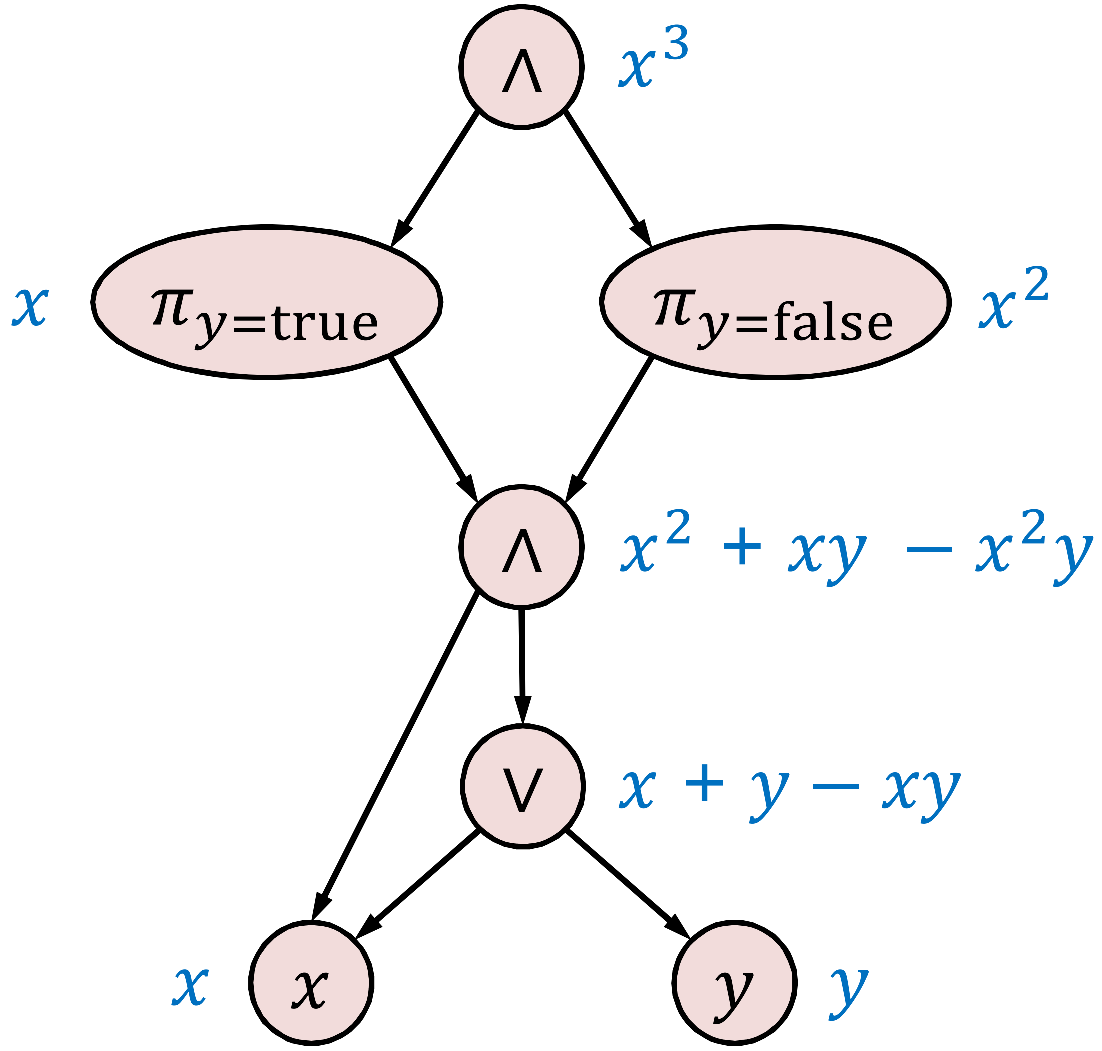
$$p_{\neg\varphi} \coloneqq 1 - p_\varphi$$

$$p_{\varphi_1 \wedge \varphi_2} \coloneqq p_{\varphi_1} \cdot p_{\varphi_2}$$

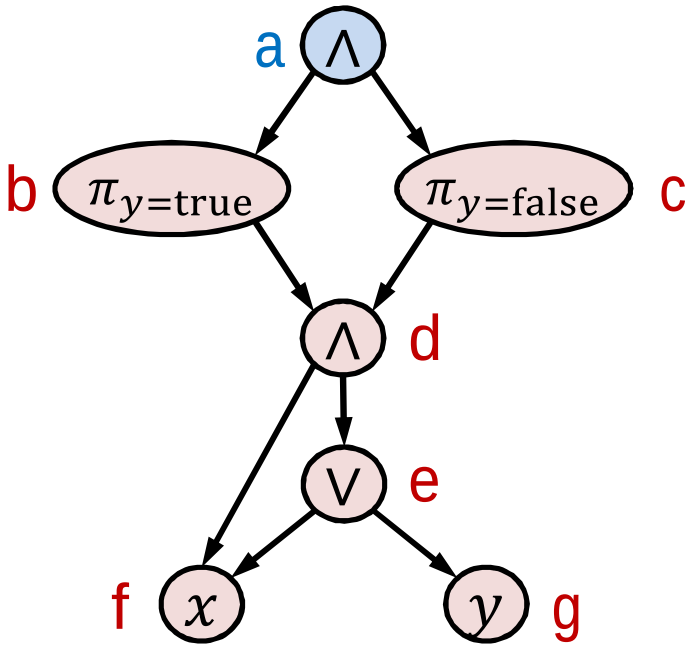
$$p_{\varphi_1 \vee \varphi_2} \coloneqq p_{\varphi_1} + p_{\varphi_2} - p_{\varphi_1} \cdot p_{\varphi_2}$$

$$p_{\Pi_{x \coloneqq b}\varphi} \coloneqq p_\varphi [x \coloneqq b]$$

Towards EBC ∈ IP: Arithmetization



A first (incorrect) IP-system for EBC



Verifier's strategy

To check **Prover**'s claim about the polynomial of a node (e.g a), **Verifier** asks **Prover** to make claims about the polynomials of its children (b, c).

If **Prover**'s claim about the node is dishonest, then at least one of the claims about its children will be dishonest with high probability.

So: if claim about the root is dishonest, then at least one of **Prover**'s claims about the leaves will be dishonest.

Verifier will be able to directly check **Prover**'s claims about leaves

Schwartz-Zippel lemma

Lemma (Schwartz-Zippel):

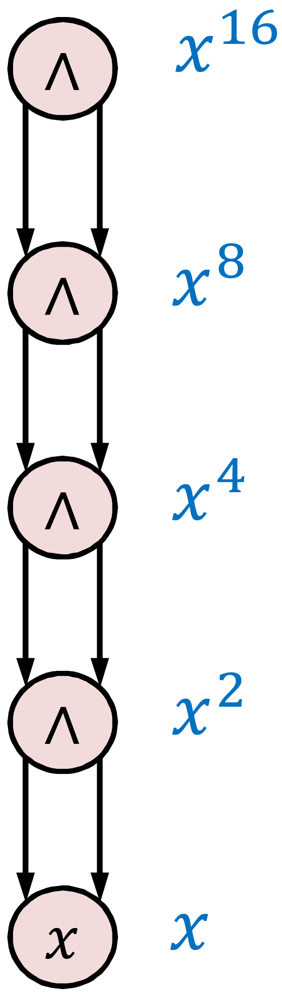
Let $p(x) \neq q(x)$ be polynomials of degree $d \geq 0$ over $\mathbb{F}_p$.

Let r be selected uniformly at random from $\mathbb{F}_p$. Then

$$\Pr[p(r) = q(r)] \leq \frac{d}{p}$$

Probability of error: $\frac{d}{p} \approx 10^{-15}$

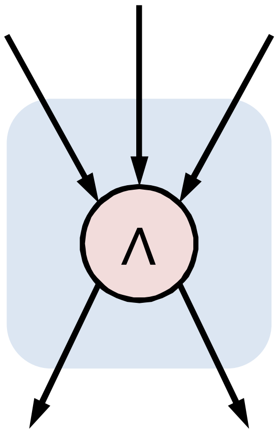
Problem: Exponential degree



Degree of polynomials can grow exponentially in the height of the circuit.

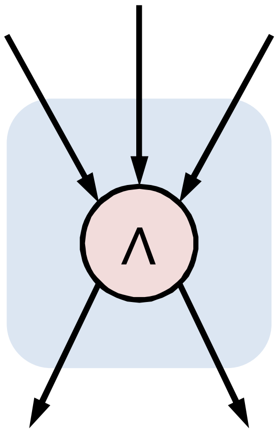
Verifier needs exponential time and Prover can cheat w.h.p.

Degree-reduction trick

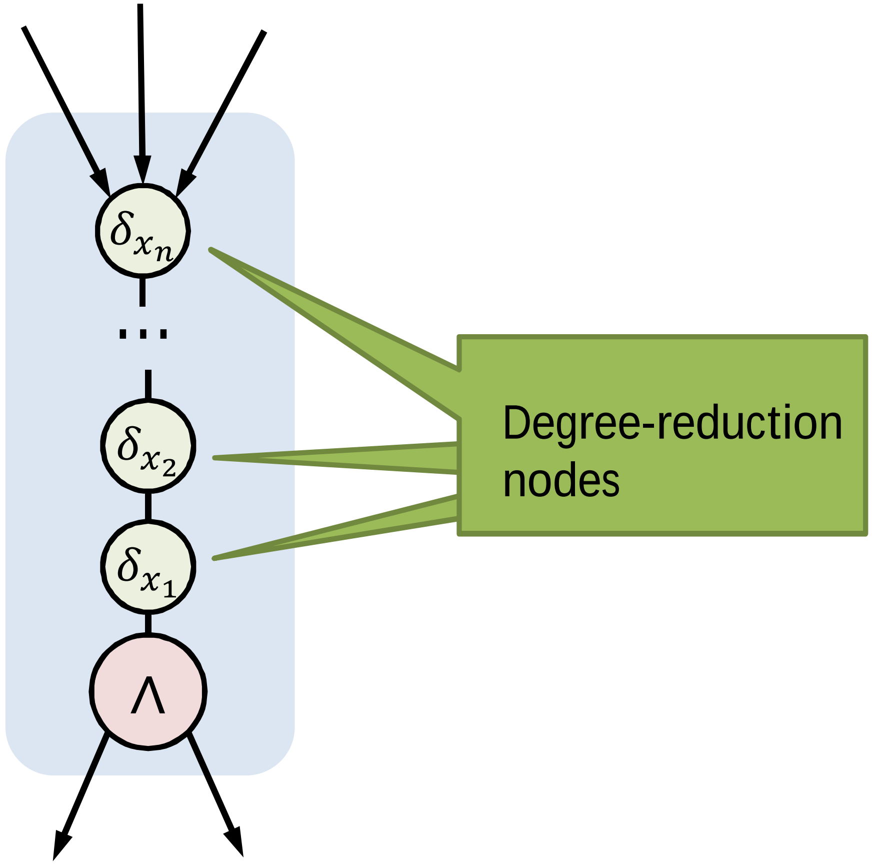


$p(x_1, \ldots, x_n)$

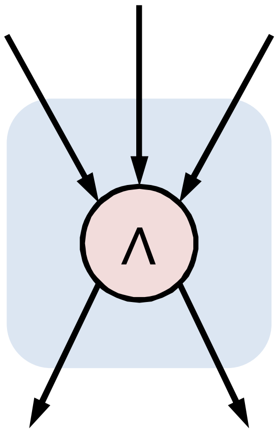
Degree-reduction trick



$p(x_1, \ldots, x_n)$

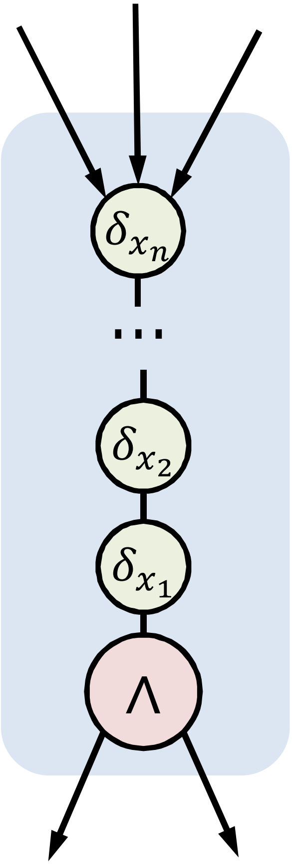


Degree-reduction trick

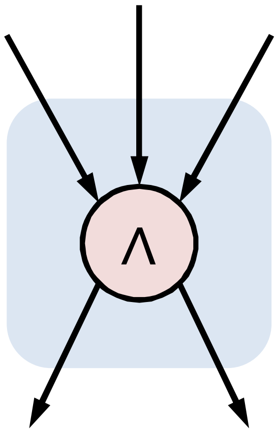


$p(x_1, \ldots, x_n) =$

$x_1^3 + x_1^2 x_2^4 - 3x_2 x_n^5$

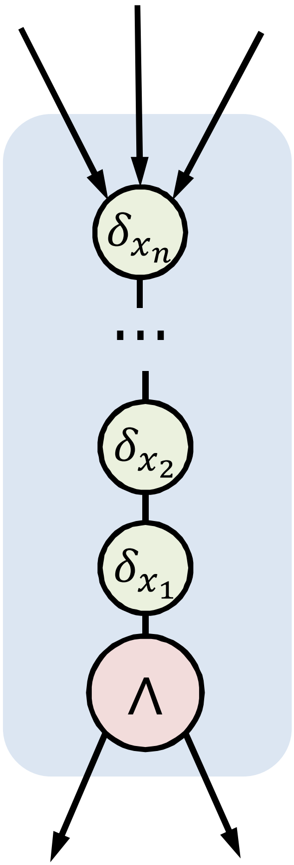


Degree-reduction trick



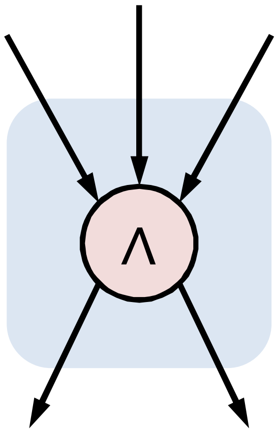
$p(x_1, \ldots, x_n) =$

$x_1^3 + x_1^2 x_2^4 - 3x_2 x_n^5$



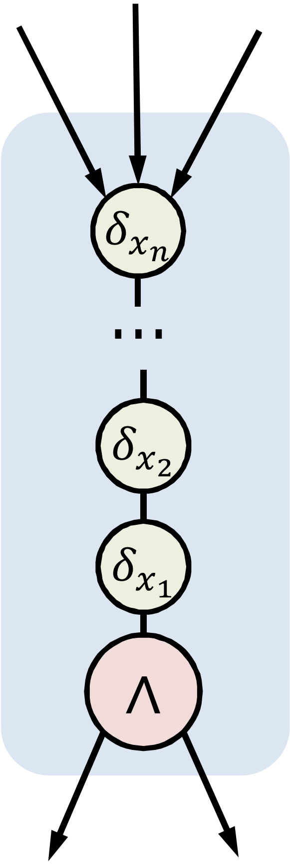
$x_1^3 + x_1^2 x_2^4 - 3x_2 x_n^5$

Degree-reduction trick



$p(x_1, \dots, x_n) =$

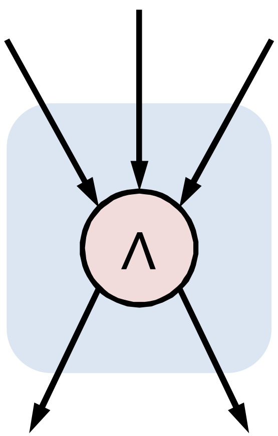
$x_1^3 + x_1^2 x_2^4 - 3x_2 x_n^5$



$x_1 + x_1 x_2^4 - 3x_2 x_n^5$

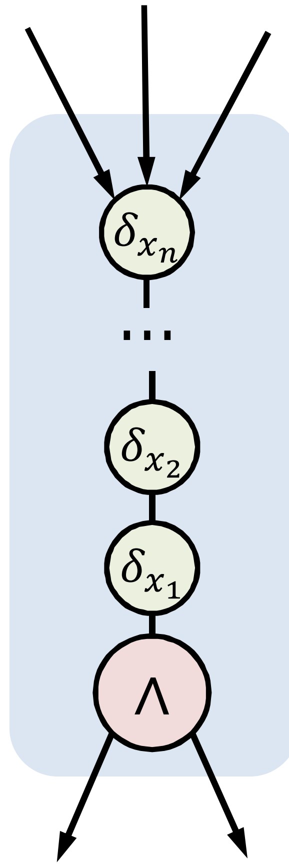
$x_1^3 + x_1^2 x_2^4 - 3x_2 x_n^5$

Degree-reduction trick



$p(x_1, \ldots, x_n) =$

$x_1^3 + x_1^2 x_2^4 - 3x_2 x_n^5$

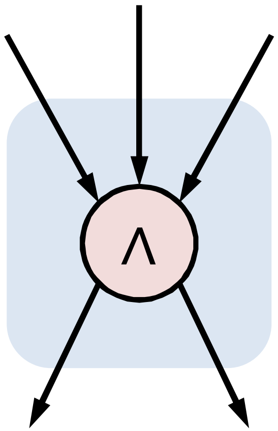


$x_1 + x_1 x_2 - 3x_2 x_n^5$

$x_1 + x_1 x_2^4 - 3x_2 x_n^5$

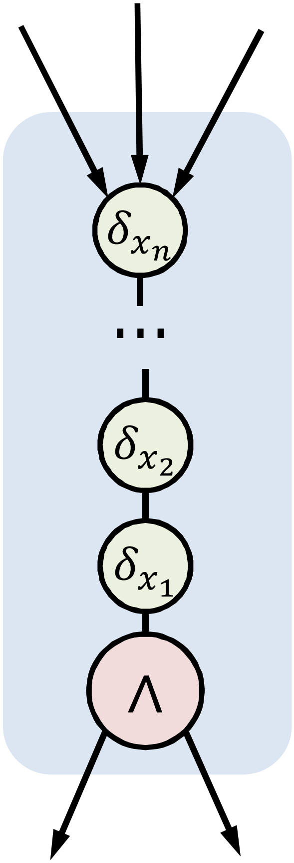
$x_1^3 + x_1^2 x_2^4 - 3x_2 x_n^5$

Degree-reduction trick



$p(x_1, \ldots, x_n) =$

$x_1^3 + x_1^2 x_2^4 - 3x_2 x_n^5$



$x_1 + x_1 x_2 - 3x_2 x_n$

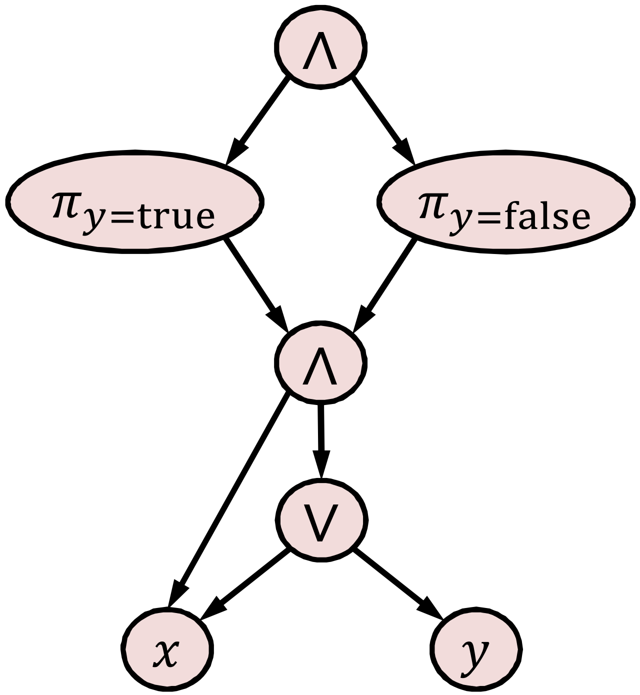
$x_1 + x_1 x_2 - 3x_2 x_n^5$

$x_1 + x_1 x_2^4 - 3x_2 x_n^5$

$x_1^3 + x_1^2 x_2^4 - 3x_2 x_n^5$

Degree-reduction trick

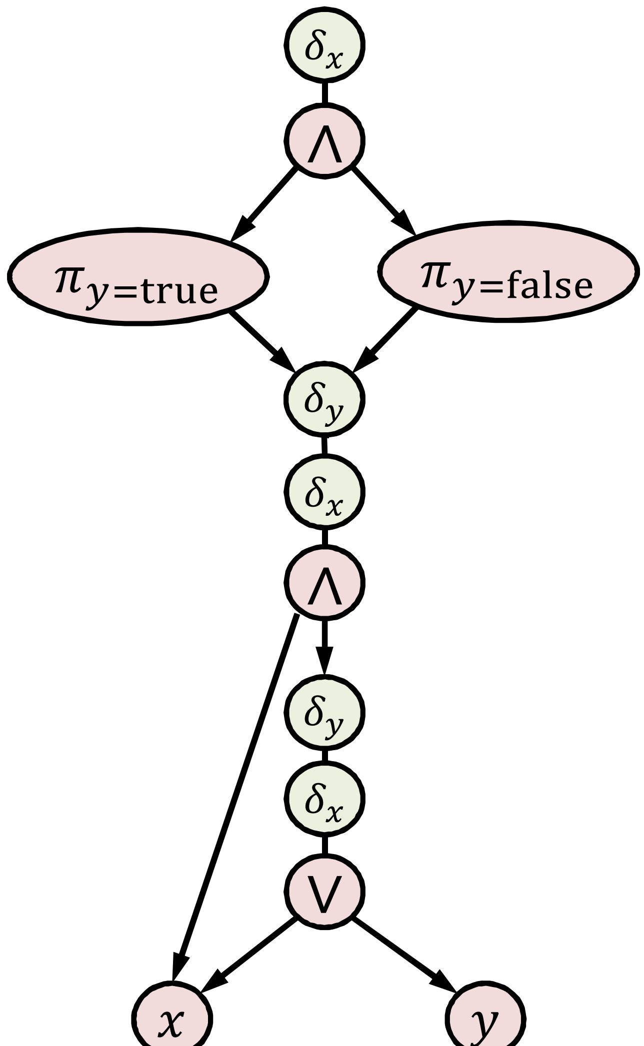
Prover



Verifier

Degree-reduction trick

Prover



Verifier

The big question

Why don‘t we have any probabilistically certified model-checkers or QBF-solvers yet ?

The big question

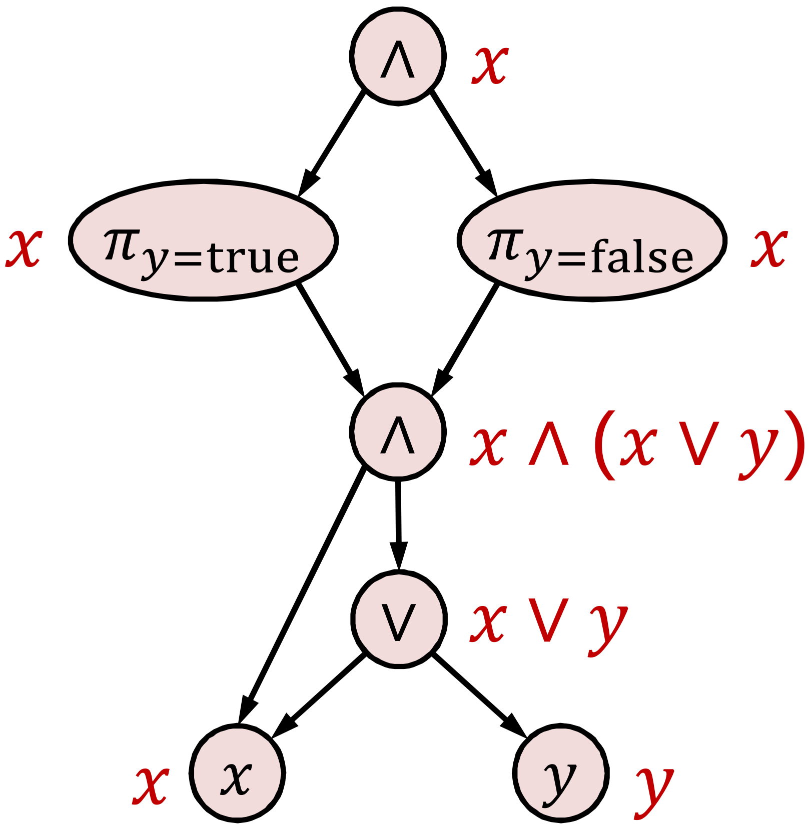
Why don‘t we have any probabilistically certified model-checkers or QBF-solvers yet ?

Seemingly incompatible with our bag of tricks for the „formula explosion“ problem.

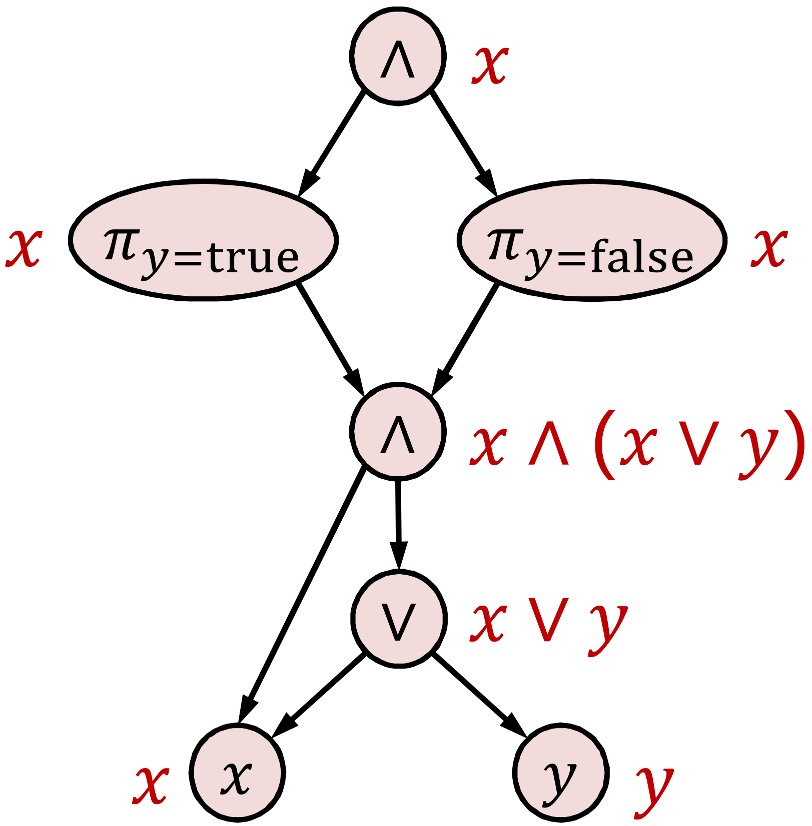
Our result

We add
interactive certification
to a
BDD-solver for EBC
with very small overhead

BDD-solver for EBC



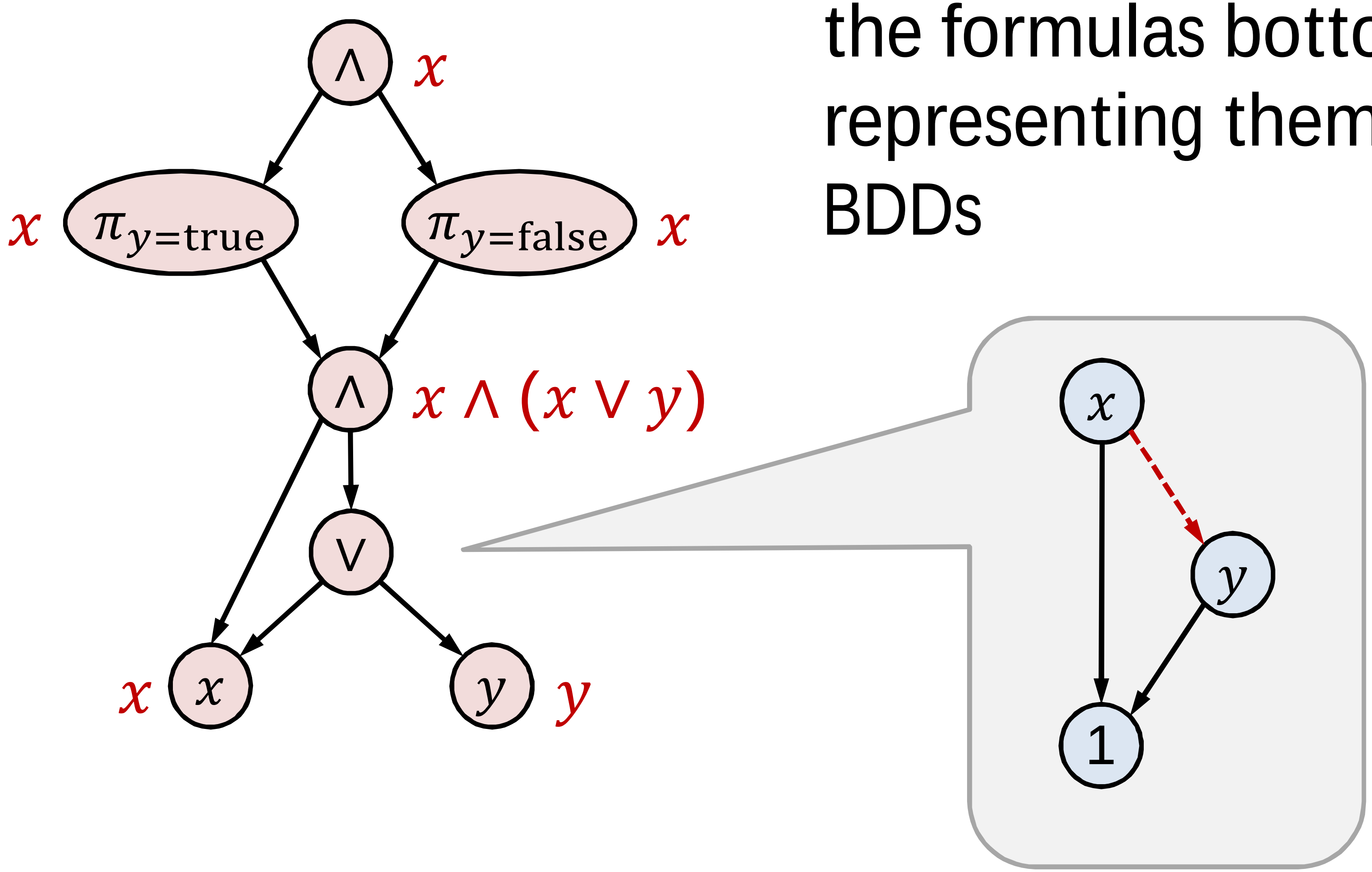
BDD-solver for EBC



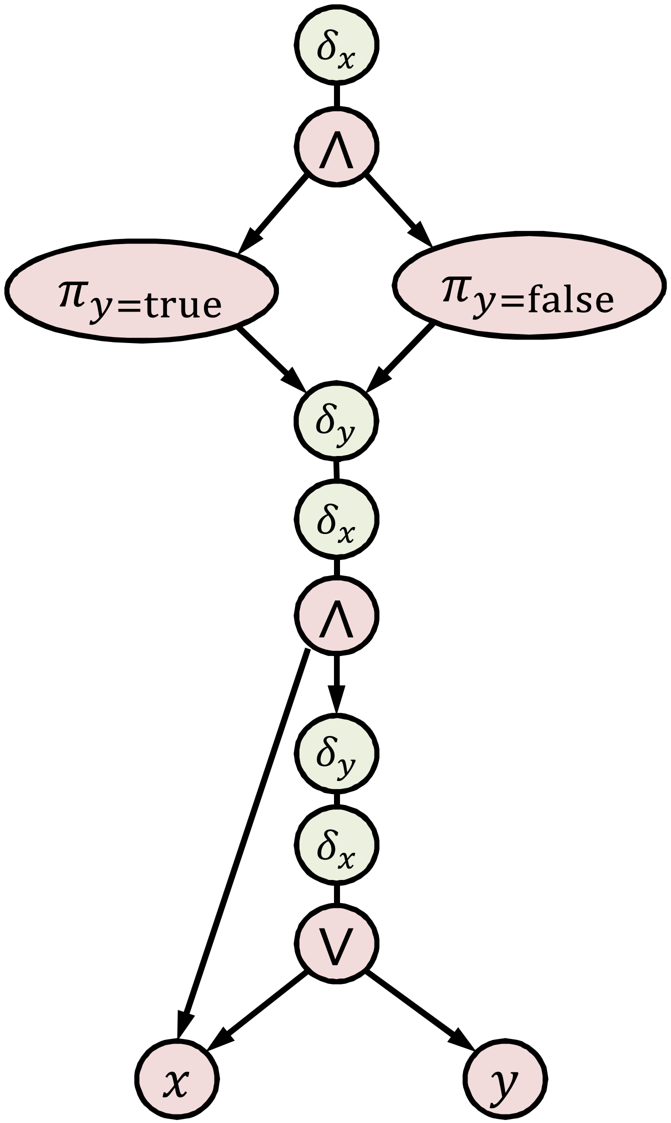
A BDD-solver computes the formulas bottom-up, representing them as BDDs

BDD-solver for EBC

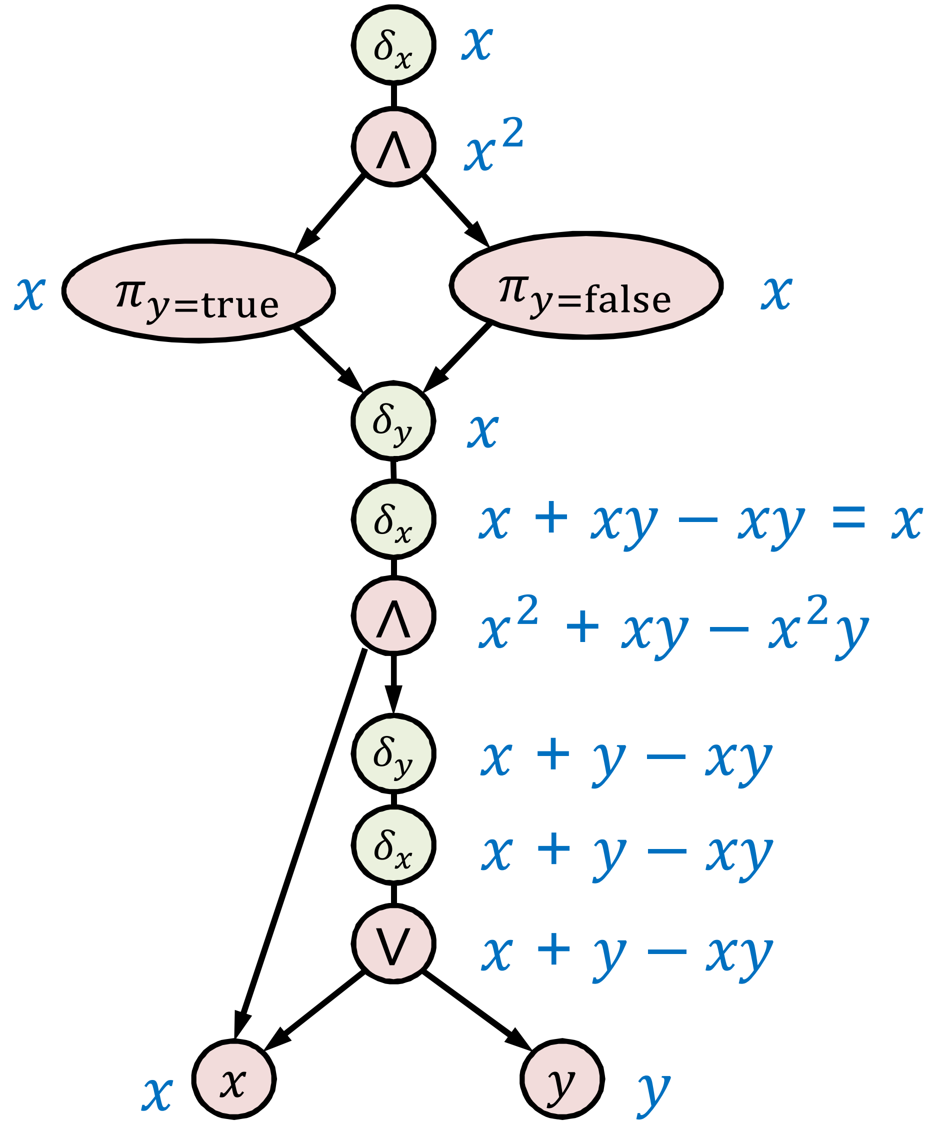
A BDD-solver computes the formulas bottom-up, representing them as BDDs



BDD-based Prover for EBC

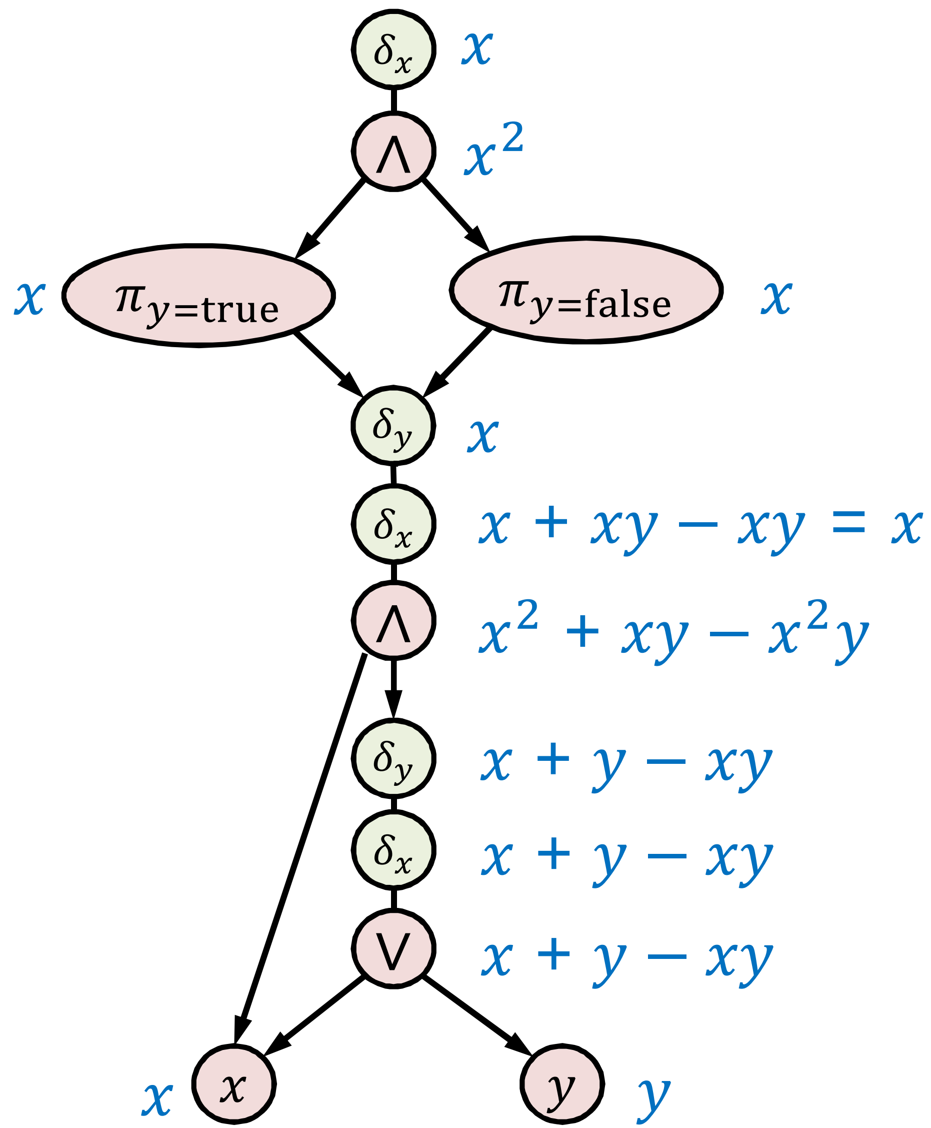


BDD-based Prover for EBC

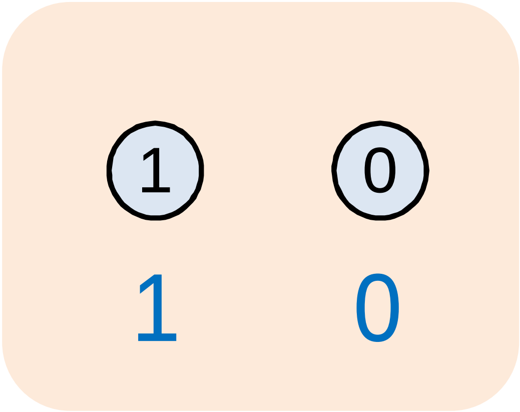


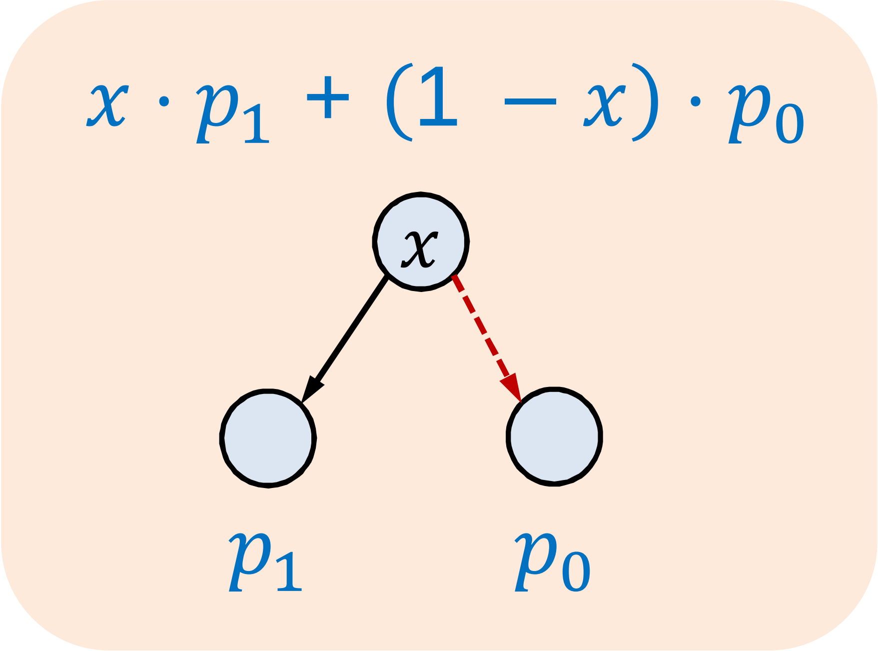
BDD-based Prover for EBC

We represent and evaluate the polynomials using BDDs

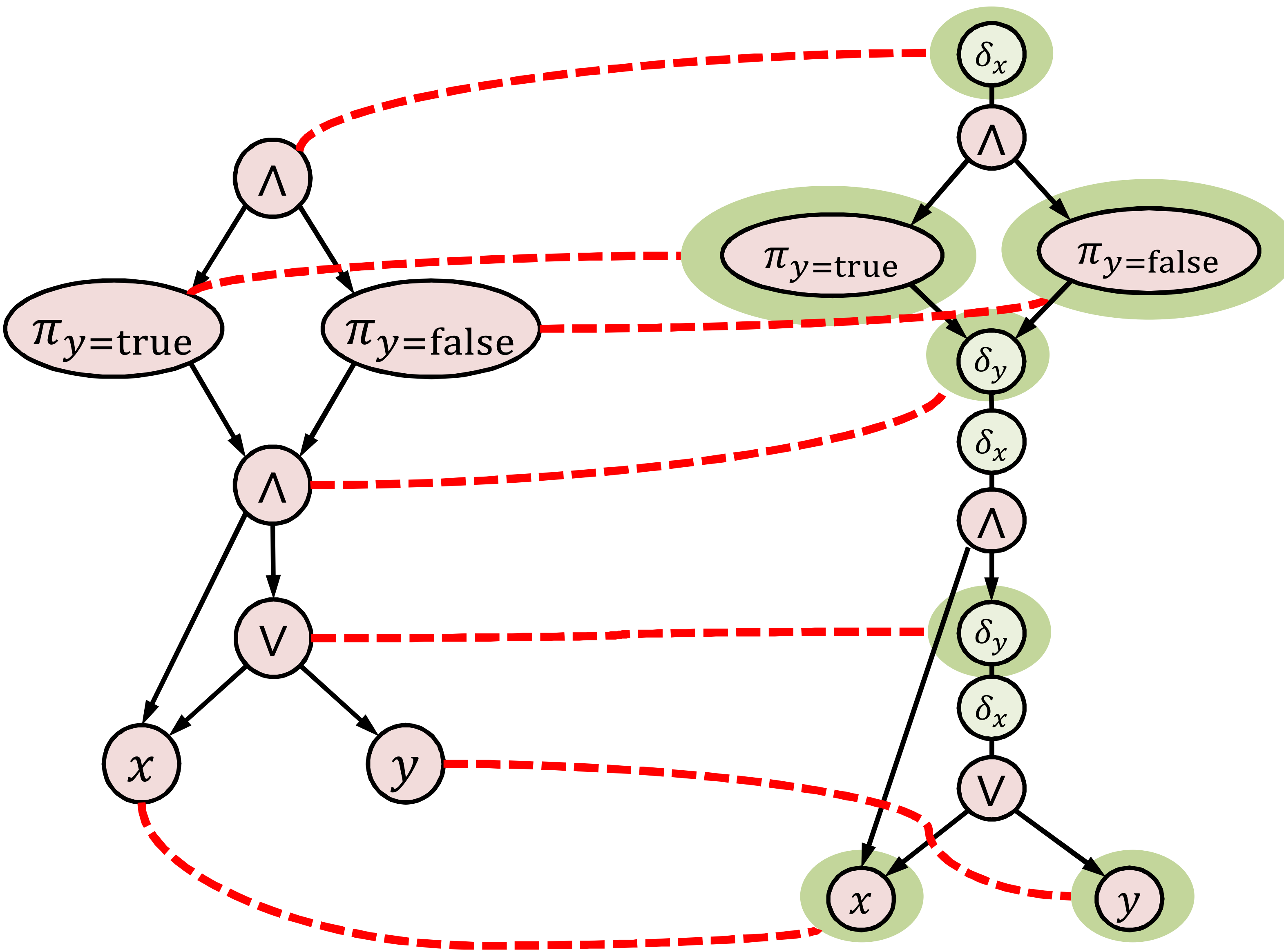


Using BDDs to represent polynomials





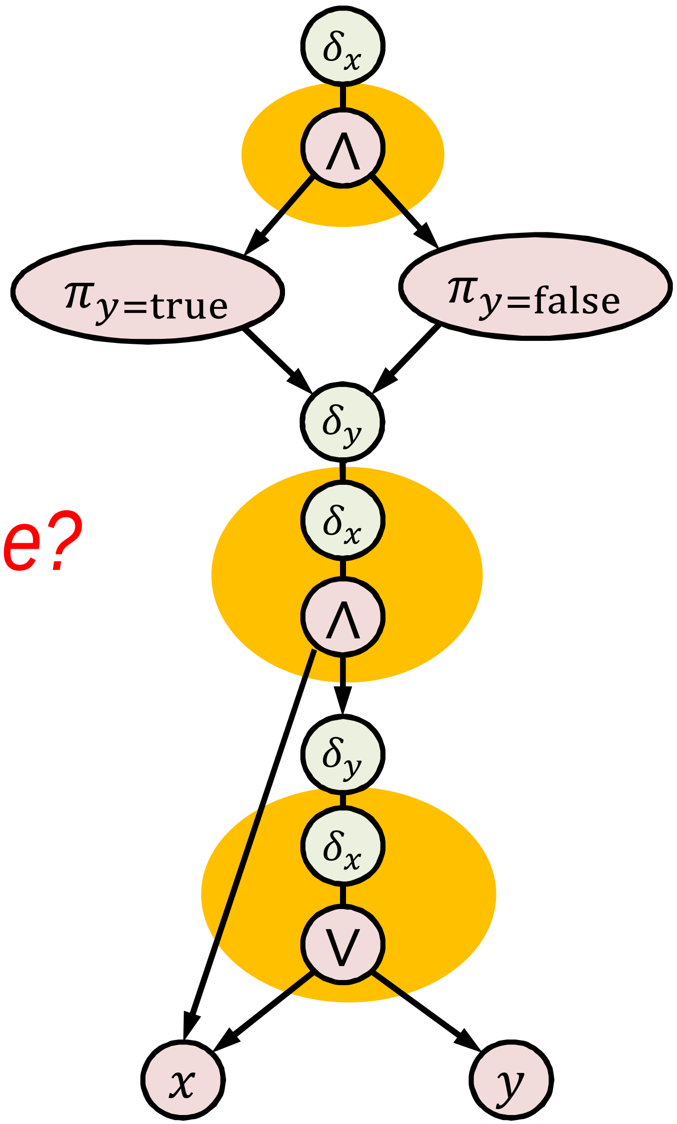
Theorem 1:



∧
$\pi_{y=\text{true}}$
$\pi_{y=\text{false}}$
∧
∨
x
y
δ_x
∧
$\pi_{y=\text{true}}$
$\pi_{y=\text{false}}$
δ_y
δ_x
∧
δ_y
δ_x
∨
x
y

Using BDDs to represent polynomials

What can we do with these?

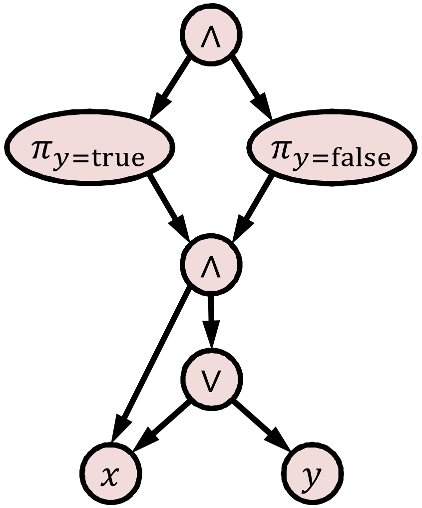


Main result

EBC

Given: An EBC

Decide: Is its binary polynomial 0?



Theorem

If solving an instance of EBC using BDDs takes time $\boldsymbol{t}$,
then solving + IP-certification using eBDDs takes time $\boldsymbol{O(t)}$.

Some QBF experiments

Instance	Var	Quant	Time Prover	Time Eval.	Time Verifier	Bytes exchanged	BDD total size
EQ-N-10	30	3	0.6 s	0.4 s	1 ms	75 KB	6 M
KBKF_QU-N-8	40	17	11 s	7 s	6 ms	176 KB	6 M
KBKF-N-10	40	21	0.5 s	0.3 s	4 ms	187 KB	0.6 M
BEQ-N-10	62	4	14 s	10 s	22 ms	680 KB	10 M
CR-N-10	121	6	6 s	4 s	160 ms	1.2 MB	7 M

Conclusion

- IP=PSPACE is not just a theoretical result
- IP systems are compatible with BDDs
- Which other techniques are they compatible with