Towards Tractable QSAT for Structural Parameters below Treewidth

Friedrich Slivovsky

Quantified Boolean Formulas

$$\forall U_1 \exists E_1 \forall U_2 \exists E_2 \ldots \forall U_n \exists E_n \,.\, \varphi$$

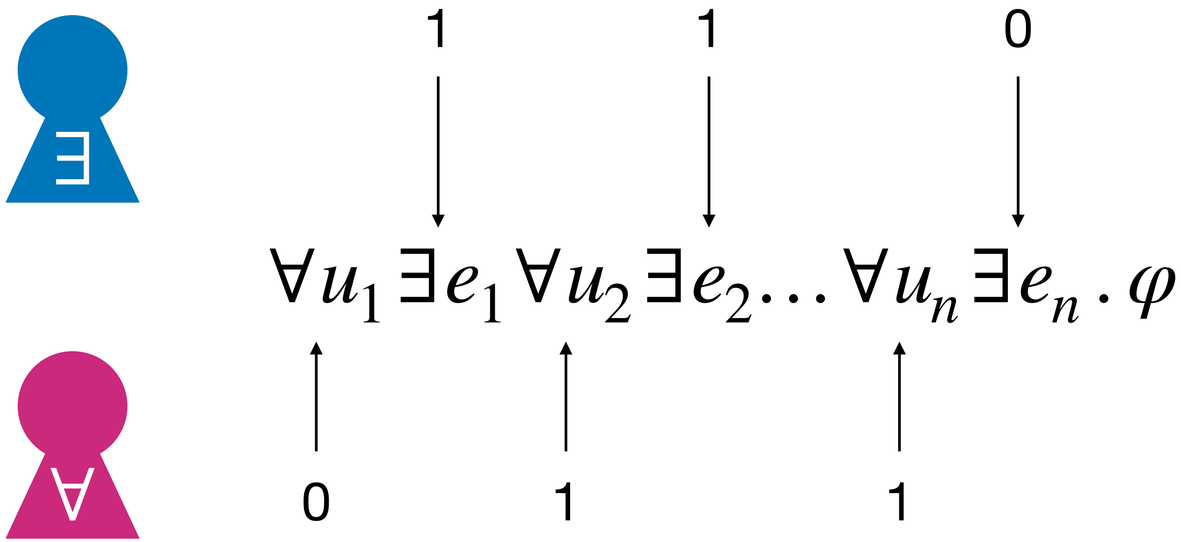
"Matrix" usually CNF

"Prefix"

Succinct encodings of

- Synthesis
- Model Checking
- Planning

QBF Game Semantics



∃ wins play σ ⟺ σ satisfies φ

QBF is **true** ⟺ ∃ has a winning strategy

QBF is **false** ⟺ ∀ has a winning strategy

QSAT

Input: A QBF Φ in prenex CNF normal form.

Question: Decide whether Φ is true.

QSAT is PSPACE-complete

Σ_i^P-complete for Σ_i prefix

$\exists E_1 \forall U_1 \ldots \forall U_{i-1} \exists E_i$

Π_i^P-complete for Π_i prefix

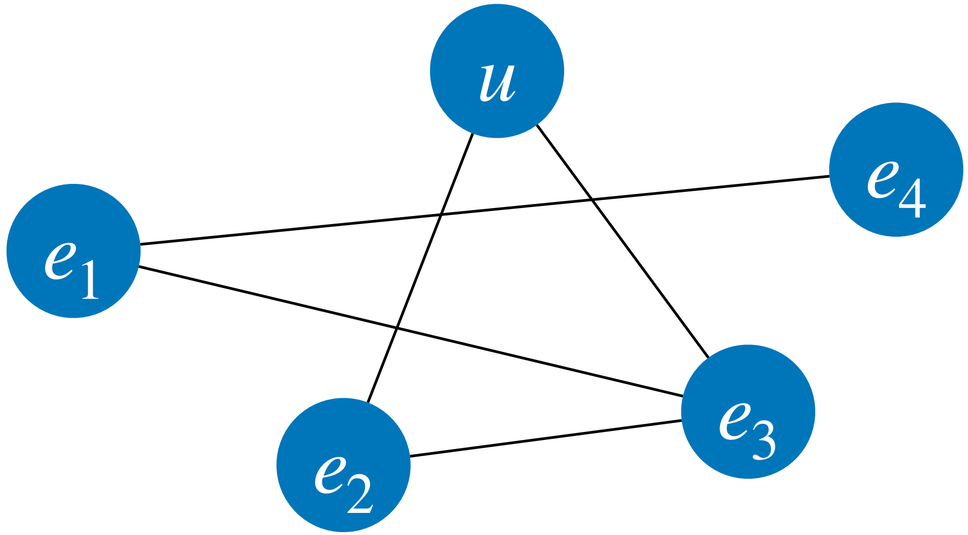
QSAT Parameterized by Treewidth

$$\exists e_1, e_2 \forall u \exists e_3, e_4 \,.\; (e_1 \vee \neg e_3) \wedge (\neg e_1 \vee \neg e_4) \wedge (e_2 \vee u \vee e_3) \wedge (\neg e_2 \vee \neg u \vee e_3)$$

QSAT is **FPT** parameterized by primal TW k + quantifier depth ℓ

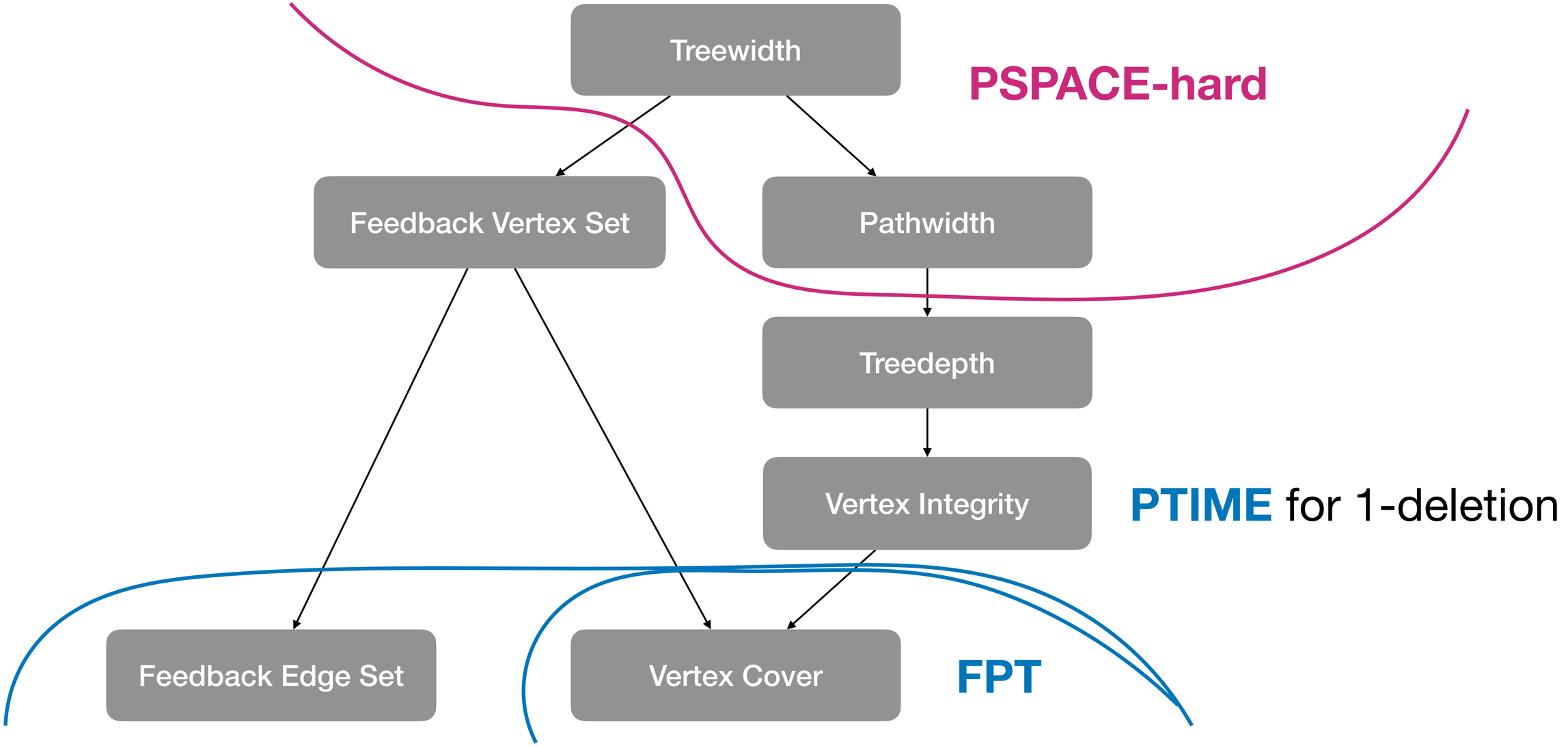
$$\ell\{2^{2^{\cdots^{2^k}}}\} \cdot \mathbf{poly}(n)$$

no $o(k)$ under ETH



QSAT is **PSPACE-hard** for QBFs of **bounded pathwidth**

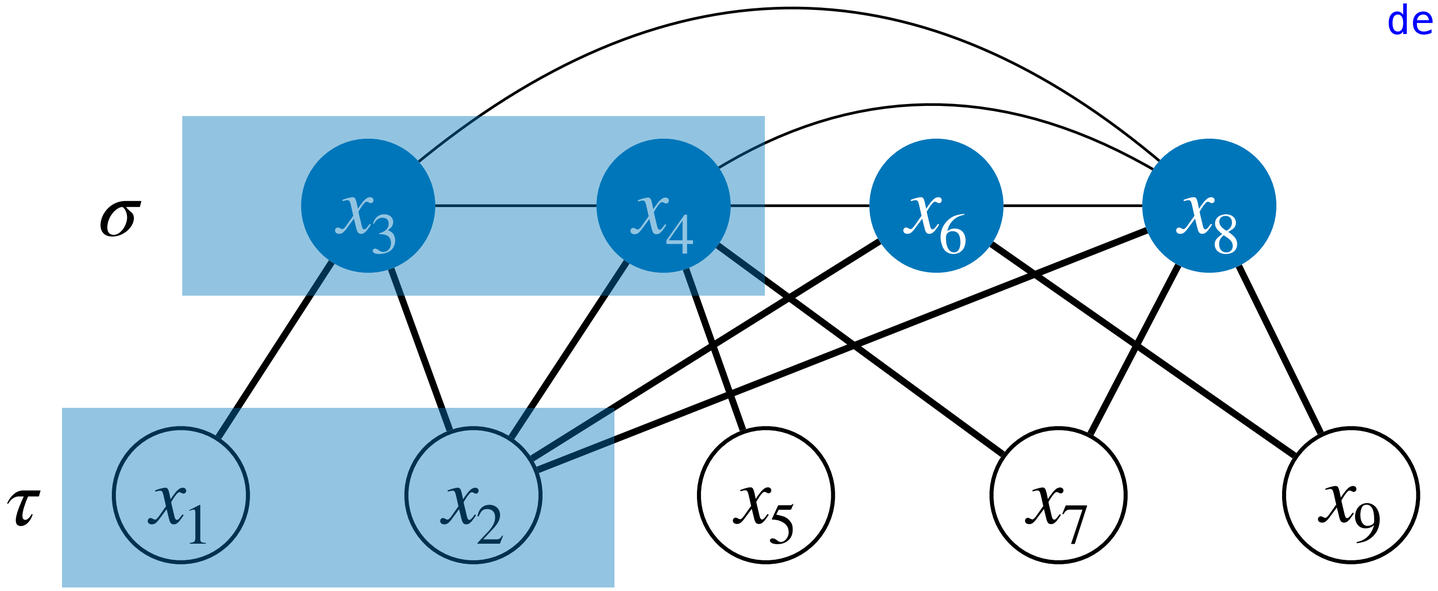
Structural Parameter Zoo



Dropping Alternations from the Parameter

- Quantifier alternations can vastly overestimate “real” dependencies.
- Requires development of new techniques.
- Identify restricted cases that are tractable regardless of quantifiers.

Vertex Cover Number

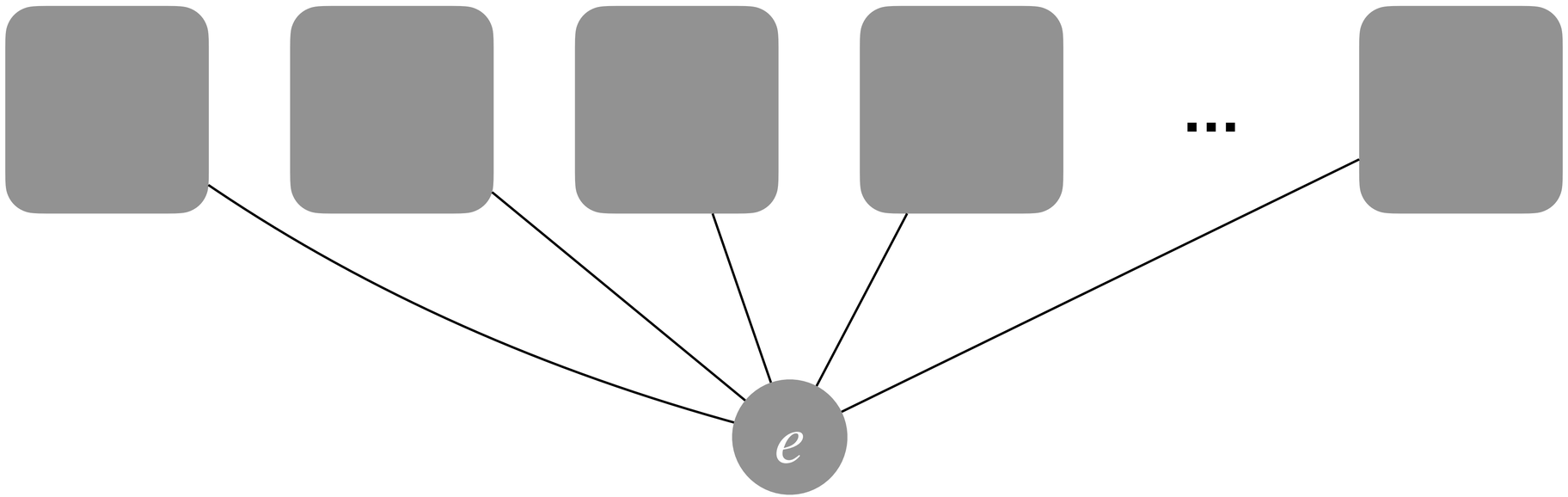



```
def DPLL(QxQ, φ):
    if φ in cache:
      return cache[φ]
    if φ == []:
      result = True
    elif [] in matrix:
      result = False
    else:
      result_true = DPLL(Q, φ[x ← 1])
      result_false = DPLL(Q, φ[x ← 0])
      if Q == ∃:
        result = result_true or result_false
      else:
        result = result_true and result_false
    cache[φ] = result
    return result
```

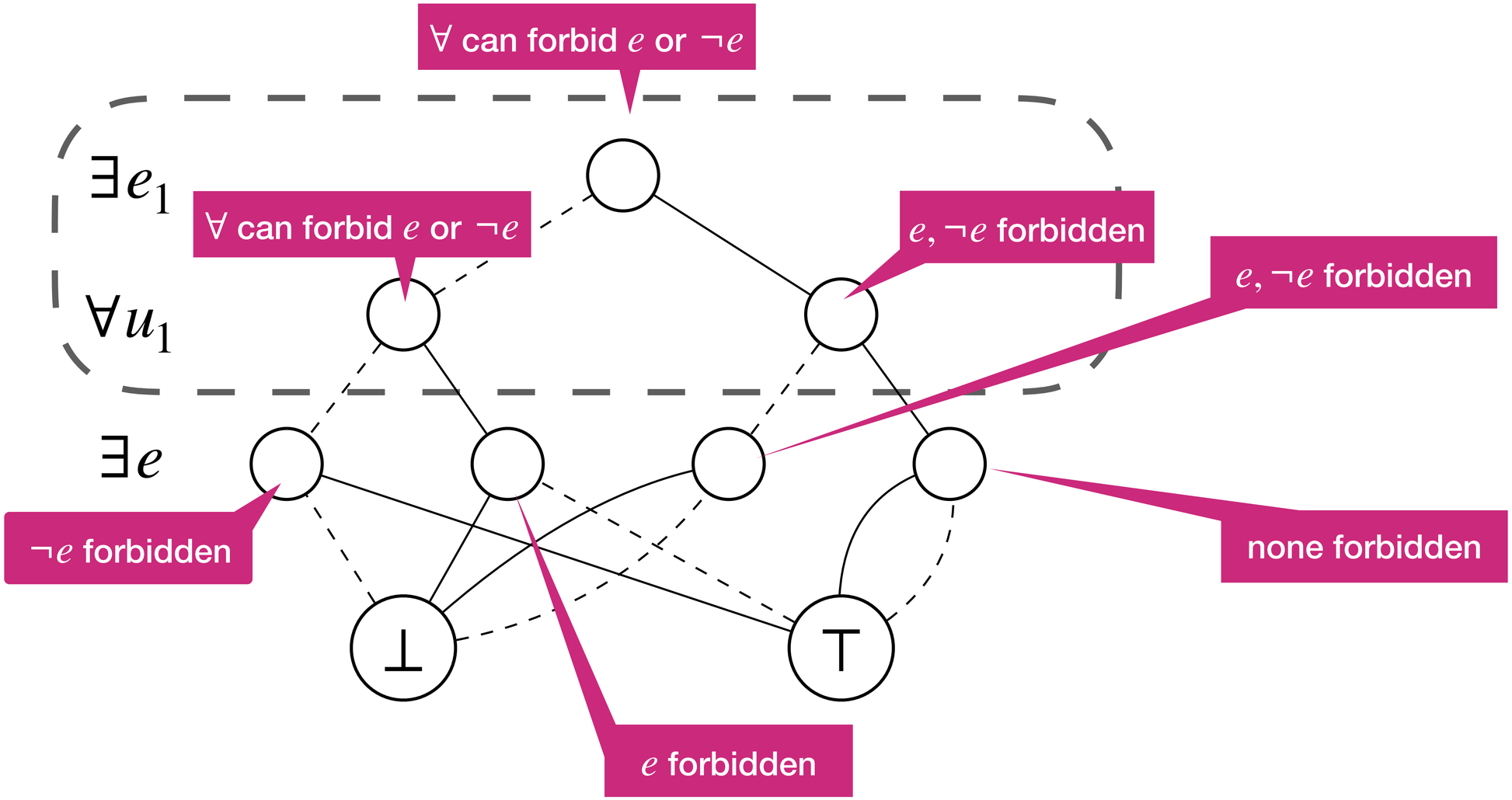
1. Clauses only touched by σ 2^k
2. Clauses touched by τ 2^{3^k}

$$O^*(2^{3^k+k})$$

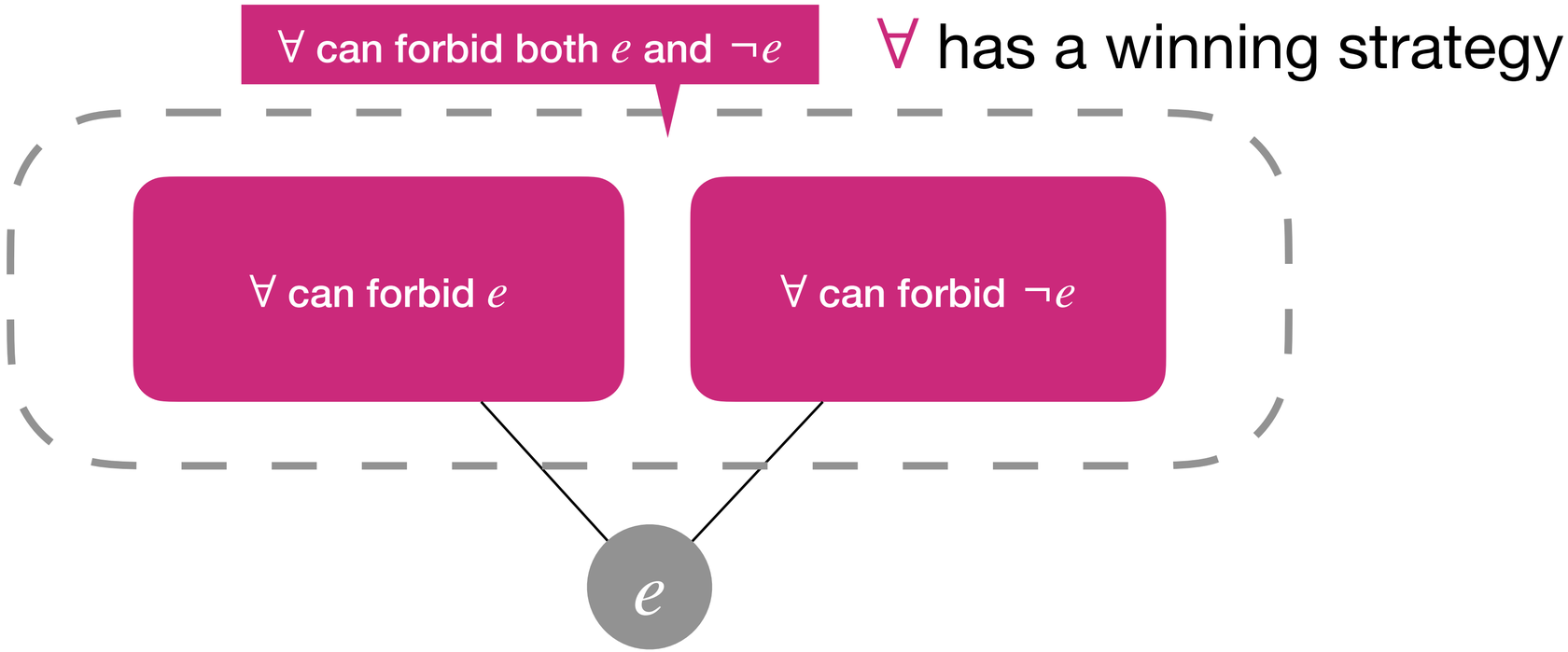
1-Deletion to Small Components



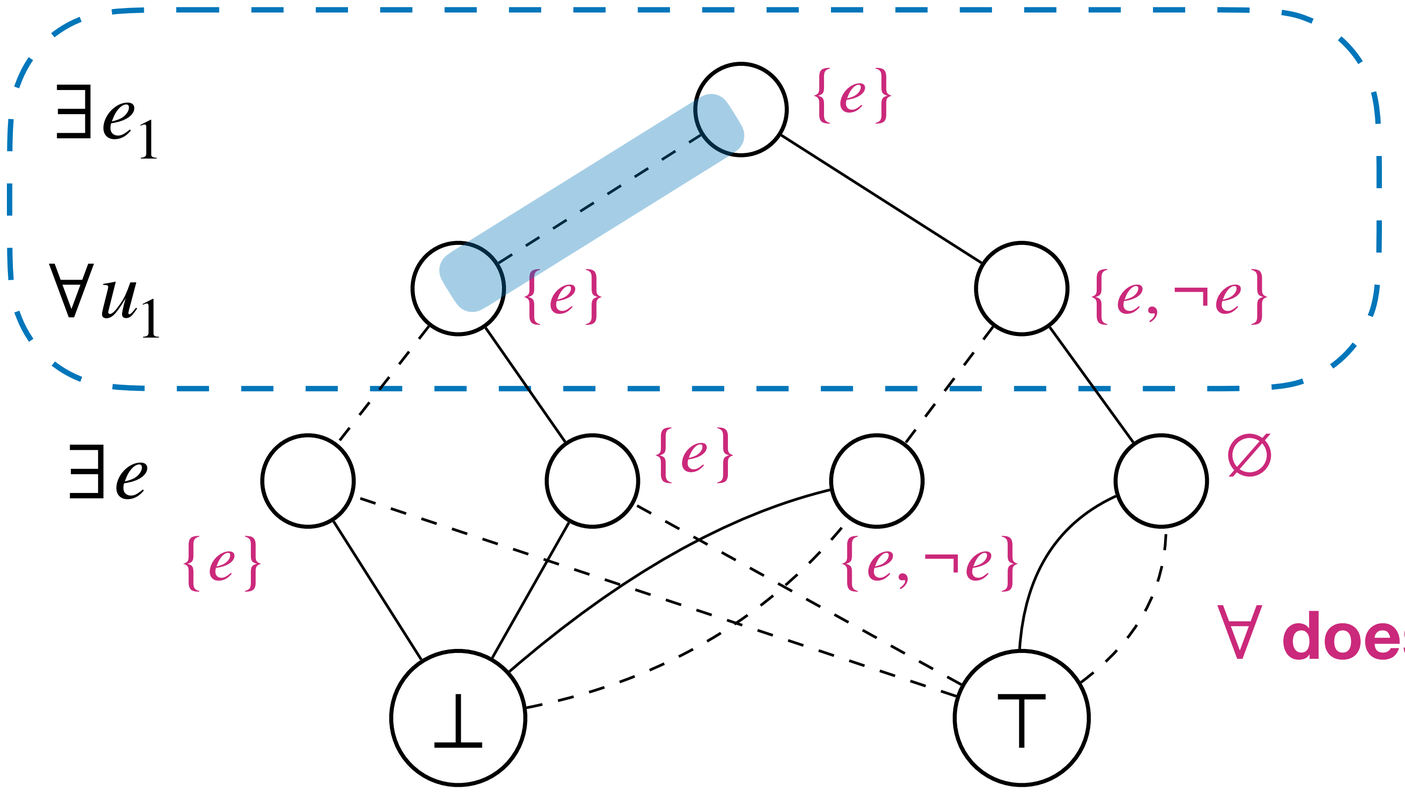
Forbidding Assignments



Combining Universal Strategies



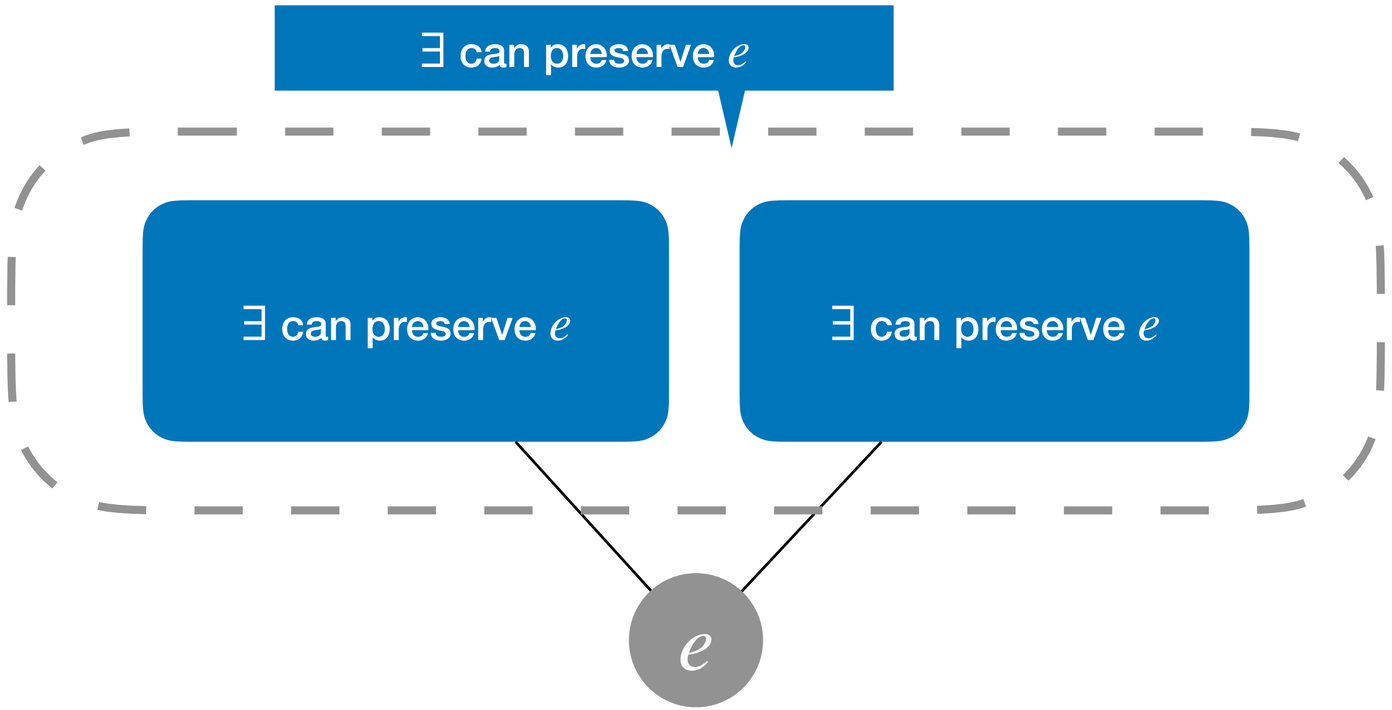
Forbidding vs. Preserving



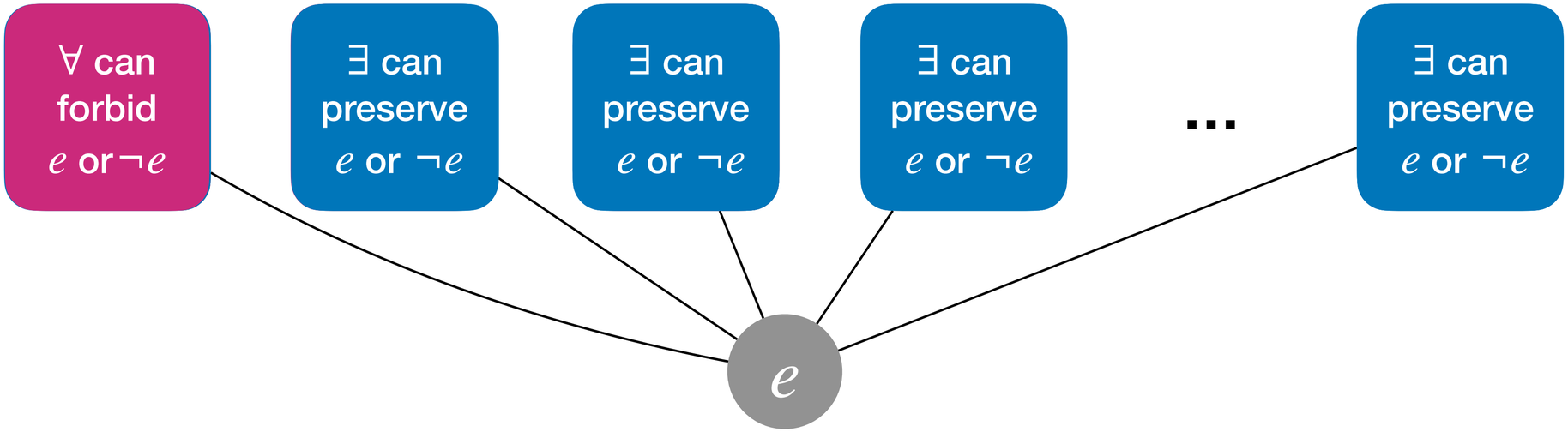
$\exists$ has a strategy preserving $\neg e$

$\forall$ **does not** have a strategy forbidding $\neg e$

Combining Existential Strategies

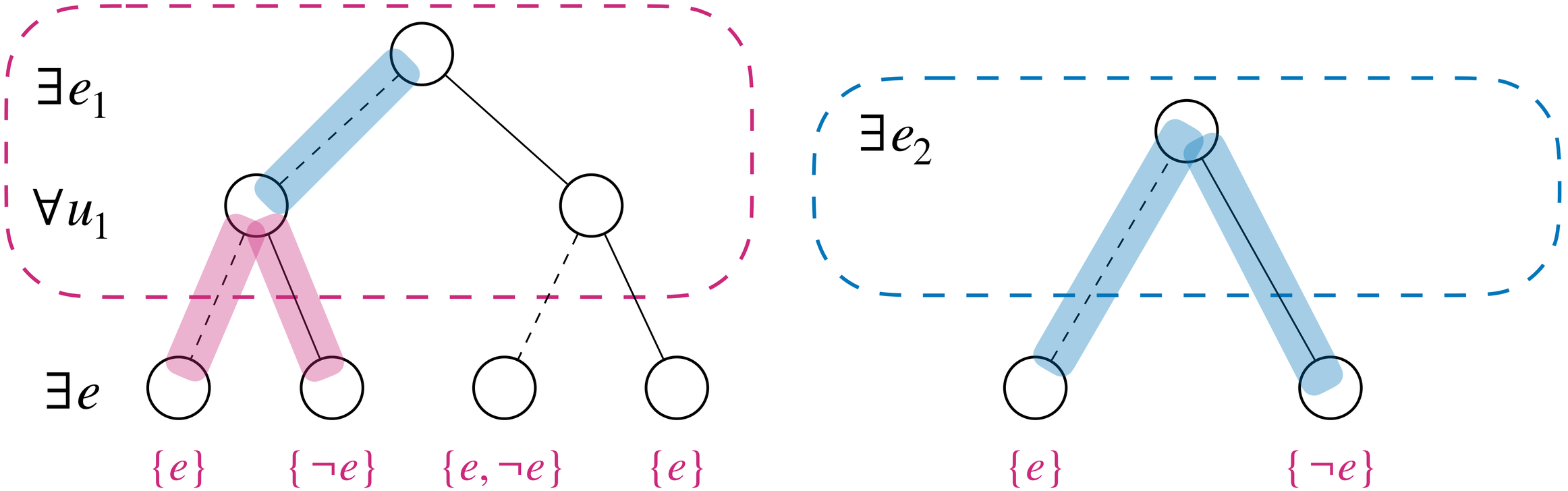


Case Distinction



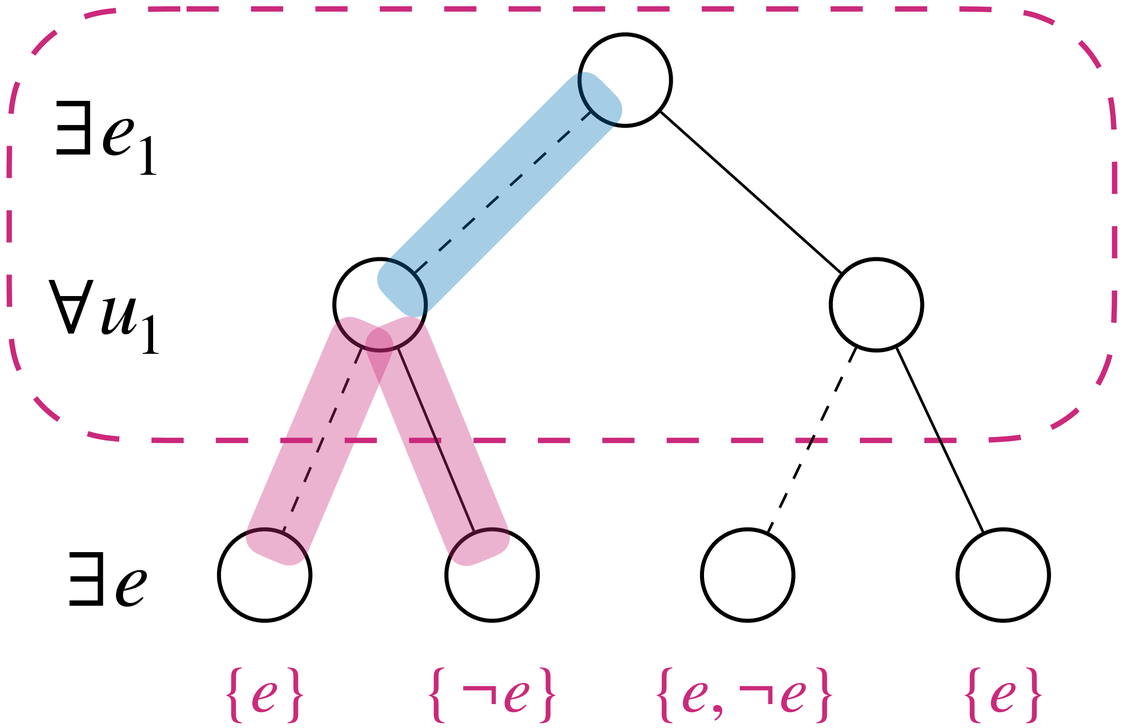
1. component where ∀ can forbid both — **∀ wins**
2. distinct components where ∀ can forbid e, $\neg e$ — **∀ wins**
3. ∃ can preserve a move in all components — **∃ wins**
4. component where ∀ can forbid either, ∃ can preserve either in the remaining components — **?**

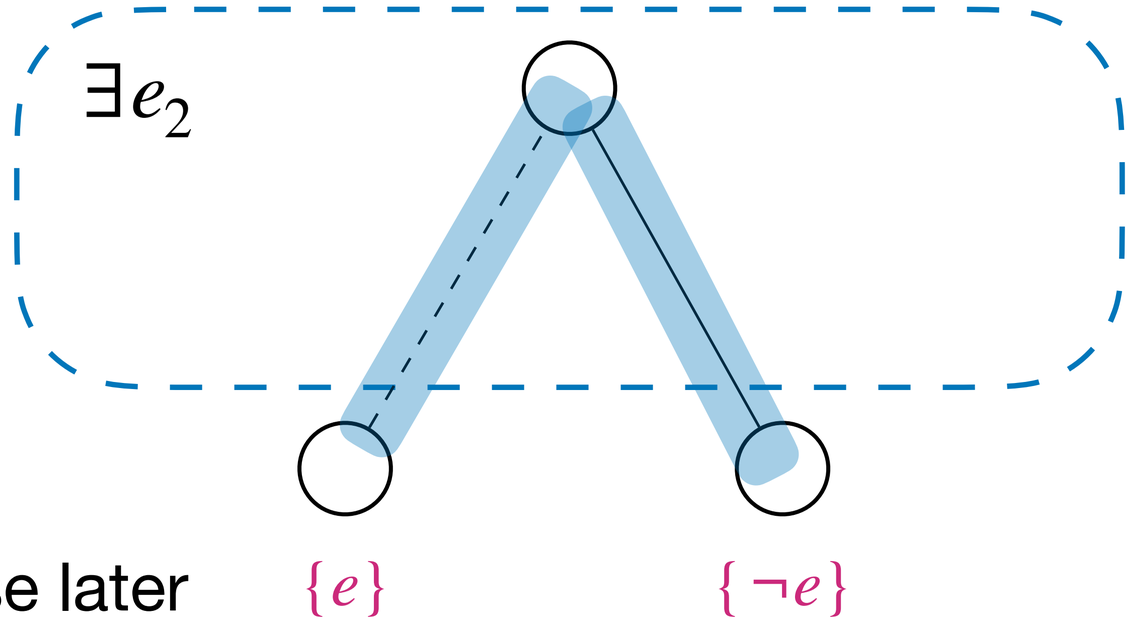
The Remaining Case(s)



$\forall$ **wins** if they get to choose later

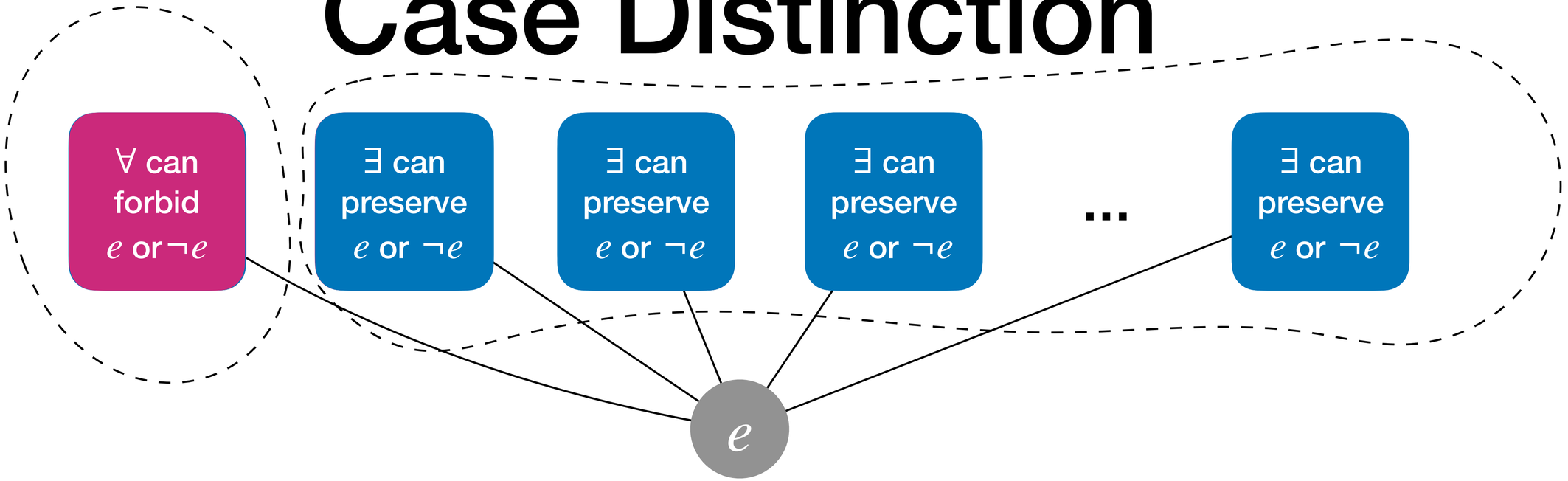
The Remaining Case(s)





$\exists$ **preserves a move** if they get to choose later

Case Distinction



compute the **rightmost variable** x where ∀ must choose

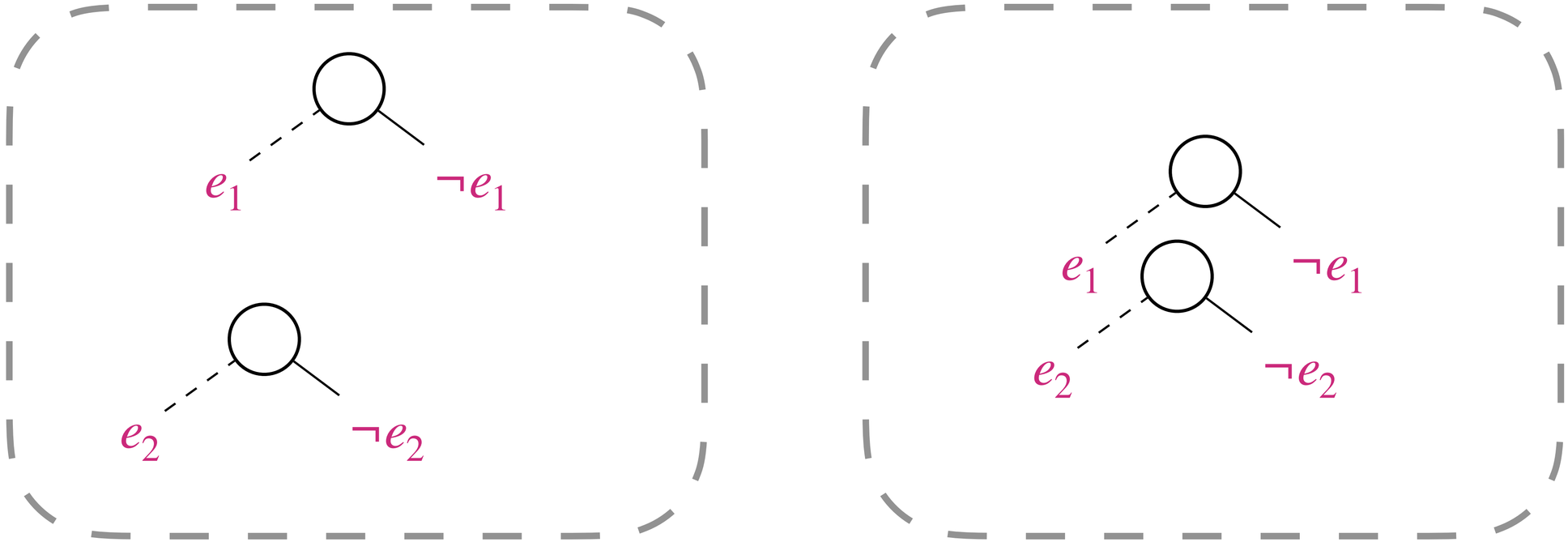
compute the **rightmost variables** Y where ∃ must choose

let $y \in Y$ be the leftmost variable where ∃ must choose

∀ wins $\Leftrightarrow$ $y < x$

Closing Comments

this simple argument does not work for **two deletion variables** e_1, e_2



must understand the interaction of **variable-disjoint** strategies