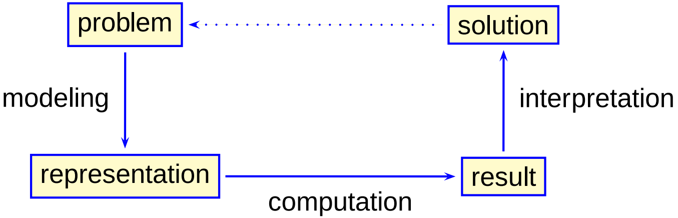
Quantified Boolean Formulas
Part 2

Uwe Egly

Knowledge-Based Systems Group
Institute of Information Systems
Vienna University of Technology

The lazy programmer's approach

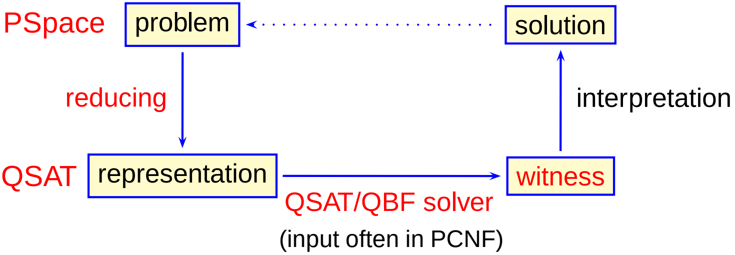
Allows us to implement problem solving programs rapidly



We want to model a problem by compiling it into a suitable representation s.t. the result of the compiled problem can be interpreted as a solution to the original problem.

The lazy programmer's approach

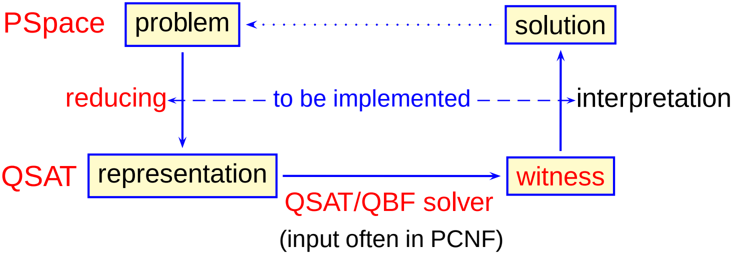
Allows us to implement problem solving programs rapidly



We want to model a problem by compiling it into a suitable representation s.t. the result of the compiled problem can be interpreted as a solution to the original problem.

The lazy programmer's approach

Allows us to implement problem solving programs rapidly



We want to model a problem by compiling it into a suitable representation s.t. the result of the compiled problem can be interpreted as a solution to the original problem.

The topics for today

- Normal form translation again
  - What are the difficulties with prenexing?
  - What to do with the propositional matrix?
- How to implement a QSAT solver?
  - Only few remarks on QDPLL but
  - More on alternative and less explored possibilities
  - ☞ Q-resolution calculus
  - ☞ Gentzen (sequent) systems

Normal forms again

Prenex normal form (PNF), prefix, matrix, PCNF, closed

- Let $\mathsf{Q}_i \in \{\forall, \exists\}$ and $p_i \in \mathcal{P}$. A QBF

  $$\Phi = \mathsf{Q}_1 p_1 \ldots \mathsf{Q}_n p_n \, \psi$$

  is in prenex (normal) form (PNF) if ψ is purely propositional
- $\mathsf{Q}_1 p_1 \mathsf{Q}_2 p_2 \cdots \mathsf{Q}_n p_n$ is the prefix of Φ; ψ is the matrix of Φ.
- Φ is in PCNF if ψ is in CNF
- Φ is closed if the variables in ψ are in $\{p_1, \ldots, p_n\}$

Convention: Each quantifier binds another variable and bound variables do not occur free.

Generating PCNFs

Why are formulas in PCNF necessary?

- Most QBF solvers require the input being in PCNF

☞ Translation procedure required

  - Generate a prenex form
  - From the matrix, generate a CNF using Tseitin

☞ All steps here are equivalence-preserving (why?)
(in contrast to, e.g., propositional logic)

Generating a prenex form (cf predicate logic)

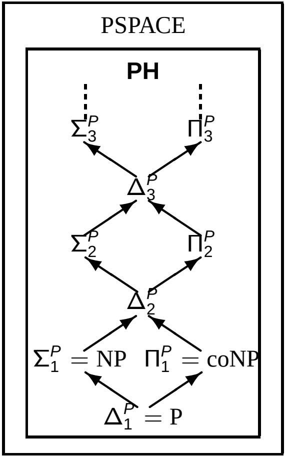
Apply the following rules until a PNF is obtained

R_1	$\mathsf{Q}x\,\Phi \circ \mathsf{Q}y\,\Psi$	$\Rightarrow$	$\mathsf{Q}x\,\mathsf{Q}y\,(\Phi \circ \Psi)$	x not free in Ψ, y not free in Φ
R_2	$(\mathsf{Q}x\,\Phi) \rightarrow \Psi$	$\Rightarrow$	$\mathsf{Q}^{-}x\,(\Phi \rightarrow \Psi)$	x not free in Ψ
R_3	$\Phi \rightarrow (\mathsf{Q}y\,\Psi)$	$\Rightarrow$	$\mathsf{Q}y\,(\Phi \rightarrow \Psi)$	y not free in Φ
R_4	$\forall x\,\Phi \wedge \forall y\,\Psi$	$\Rightarrow$	$\forall x\,(\Phi \wedge \Psi[y/x])$	
R_5	$\exists x\,\Phi \vee \exists y\,\Psi$	$\Rightarrow$	$\exists x\,(\Phi \vee \Psi[y/x])$	

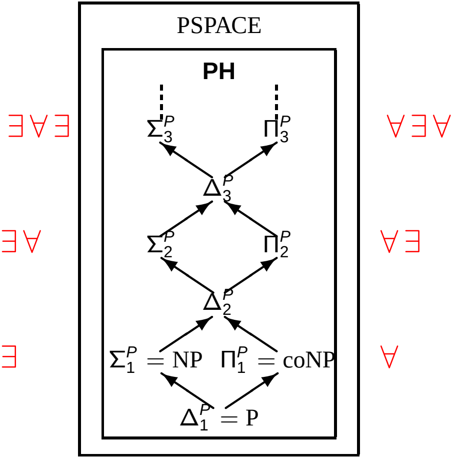
Remarks

- $\mathsf{Q} \in \{\forall, \exists\}$, $(\mathsf{Q}, \mathsf{Q}^{-})$ is $(\forall, \exists)$ or $(\exists, \forall)$ and $\circ \in \{\wedge, \vee\}$
- In general, the PNF of Φ is not unique
  (depends, e.g., on rule choice: R_1 vs R_4 if both are applicable)
- Φ and all of its prenex forms are logically equivalent (why?)

The polynomial hierarchy (PH) again



The polynomial hierarchy (PH) again



➥ Problem complexity determines quantifier prefix of the target QBF (number of alternations, starting quantifier)

How to handle non-prenex QBFs?

Extend the complexity landscape to arbitrary closed QBFs

- Take the maximal number of quantifier alternations along a path in the syntax tree of a QBF into account
- Almost all QBFs can be translated into equivalent QBFs in PNF without increasing the number of quantifier alternations (Which are the problematic QBFs?)
- Translation procedure is fast but non-deterministic
- Can heavily influence the performance of QBF solvers
- Details in E. et al. Comparing Different Prenexing Strategies for Quantified Boolean Formulas. Proc. SAT 2003, pp. 214-228.

Generating CNFs from the matrix

The Tseitin-based algorithm works in three steps

1. Generate a prenex form $\Psi_p : \mathsf{Q}_i X_i \cdots \mathsf{Q}_k X_k \, \psi$ of the input QBF Ψ with minimal number of quantifier alternations. Then the matrix ψ is purely propositional.
2. Use Tseitin's translation to transform ψ into CNF.
3. Place the $\exists$ quantifiers for the newly introduced variables $\ell_1, \cdots \ell_m$ abbreviating $\varphi_1, \ldots, \varphi_m$ "correctly", e.g.,
   - place all the new $\exists$ at the end of the quantifier prefix, or
   - place $\exists \ell_i$ ($1 \leq i \leq m$) after all quantifiers of those variables which occur in φ_i.

☞ The use of quantifiers results in an equivalent CNF

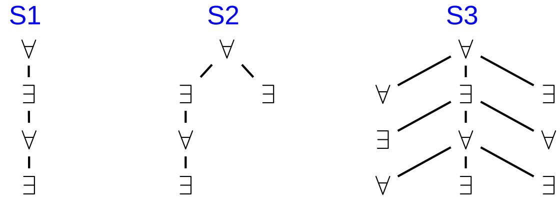
Some problem sets and their quantifier structure

- Test sets for non-normal form solvers not easy to find
  (most test sets are already in PCNF)
- Chosen 3 sets (w increasing complexity of quantifier structure)

S1 Encodings for sat in modal logic K (given by Pan and Vardi)

S2 Encodings for answer set (AS) correspondence checks

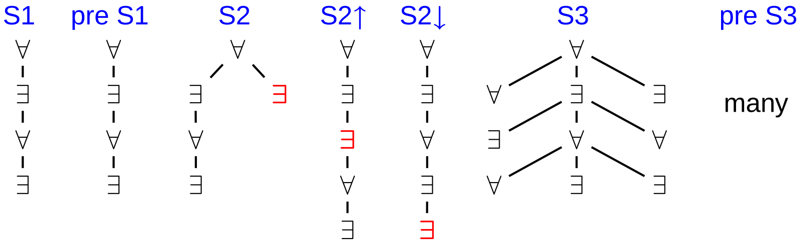
S3 Encodings of reasoning with nested counterfactuals



Prenexing strategies and the quantifier structure

- prenexing: "linearize" quantifier dependencies
  without increasing the number of alternations
- We need two strategies here

$\uparrow$: Place the quantifiers as outermost (high) as possible
$\downarrow$: Place the quantifiers as innermost (low) as possible

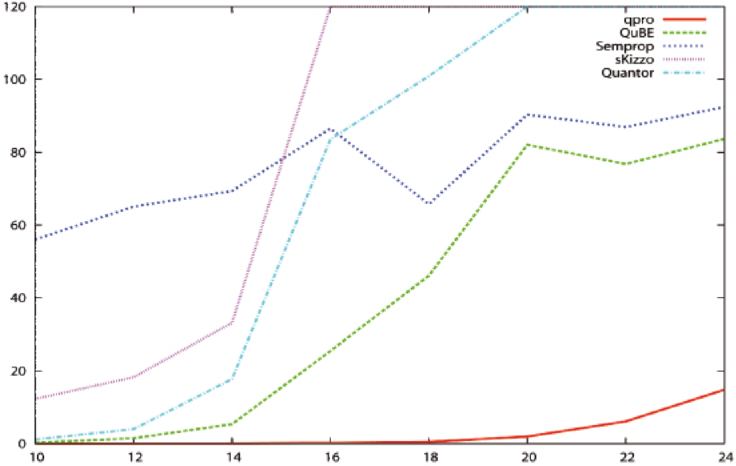


S2: Encodings for AS correspondence checks

- Generation of the instances for the Π_2^P problems (S22) (from [Janhunen & Oikarinen 2004])
  - QBF generated randomly & translated to LP (similar to S24)
  - Take logic program, remove one clause and check equiv (problem has much "propositional" structure!!!)
- Depth of the QBFs: 2, i.e. only 1 quantifier alternation
- Grouped in 8 subsets (QBFs with 10, 12, . . . , 24 variables)
- Number of instances in each subset: 100
- In approx. 50%, the equivalence holds
- Only translation T used
- No specific prenexing strategy (problem is on the 2nd level)

http://www.kr.tuwien.ac.at/research/systems/eq/index.html

Experimental results for the Π_2^P problems



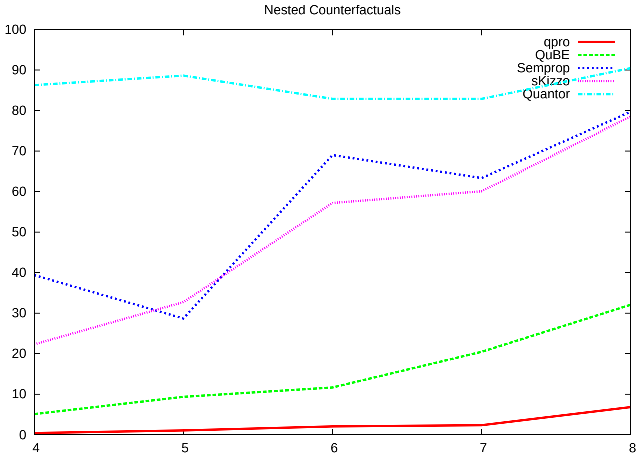
S3: Encodings for nested counterfactuals

- A counterfactual (cf) $p > q$ is a conditional query ...
  ... if we put p to our theory $\mathcal{T}$, can we derive q
- If p, q are cfs, then $p > q$ is called nested cf
- Nested counterfactuals span the polynomial hierarchy
- Nested counterfactuals can be encoded as QBFs.
- Random generation of the problems like before

Statistics for problems from S3

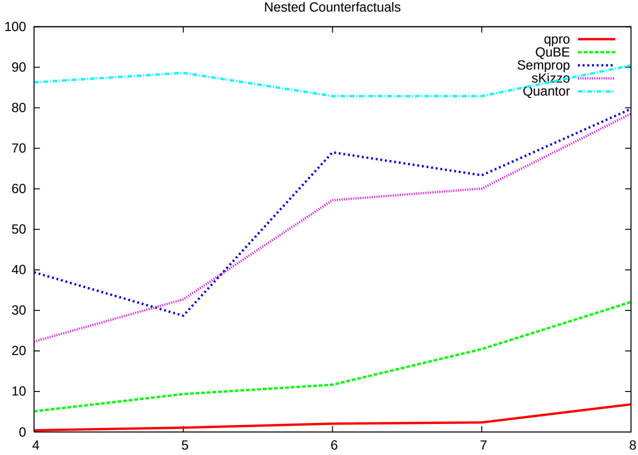
- Grouped in 5 sets (with depth of QBFs from 4 to 8)
- Approximately 60% are valid
- Number of instances per set: 50
- Number of variable: 183, 245, 309, 375, 443
- Number of vars (after PCNF trans): 464, 600, 786, 934, 1132
- Many prenexing strategies tried but ...
- ... we show best strategy for each solver in the graphic

Experimental results for nested counterfactuals



Nowadays, many solvers can solve ncf problems
(by analyzing “dependencies between variables”)

Experimental results for nested counterfactuals



Nowadays, many solvers can solve ncf problems
(by analyzing “dependencies between variables”)

Outline

Why do we need a resolution calculus for QBFs?

- We need a QSAT solver in our rapid implementation approach. Why not Q-resolution?
- Although you will usually not see it, but in nearly every QDPLL solver, there is Q-resolution inside
- Some QDPLL solvers deliver Q-resolution “refutations” as certificates for unsatisfiability.
- From these proofs, one can generate witness functions (as mentioned earlier)

A resolution calculus for QBFs

Definition (Propositional resolution rule)

Let $C \vee x$ and $D \vee \neg x$ be two clauses, where C, D are disjunctions of literals. With the propositional resolution rule

$$\frac{C \vee x \quad D \vee \neg x}{C \vee D}\ PRes$$

we derive the resolvent $C \vee D$ from the indicated parent clauses.

Definition (Propositional factor)

Let $C_1 \vee \ell \vee C_2 \vee \ell \vee C_3$ be a clause with possibly empty subclauses C_1, C_2, C_3 and let ℓ be a literal. With the propositional factor rule

$$\frac{C_1 \vee \ell \vee C_2 \vee \ell \vee C_3}{C_1 \vee \ell \vee C_2 \vee C_3}\ PFac$$

the factor $C_1 \vee \ell \vee C_2 \vee C_3$ of the indicated clause can be derived.

A resolution calculus for QBFs (cont'd)

Definition (Quantification level)

Let Q be a sequence of quantifiers. Associate to each alternation its level as follows. The left-most quantifier block gets level 1, and each alternation increments the level.

Example: $\underbrace{\forall x_1 \forall x_2}_{\text{level 1}} \underbrace{\exists y_1 \exists y_2 \exists y_3}_{\text{level 2}} \underbrace{\forall x_3}_{\text{level 3}} \underbrace{\exists y_4}_{\text{level 4}} \varphi$

Definition ($\forall$ reduction)

Let $C \vee \ell$ be a non-tautological clause, ℓ a universal literal and no other literal in C has higher level. Then, with $\forall$ reduction

$$\frac{C \vee \ell}{C} \ \forall R$$

we can derive C from $C \vee \ell$.

A resolution calculus for QBFs (cont'd)

Definition (Q-resolution calculus)

The Q-resolution calculus consists of the propositional resolution and factor rule and the $\forall$ reduction rule. Resolution operations are only allowed over existential literals and tautological resolvents are deleted immediately.

Theorem (Kleine Büning, Karpinski, Flögel, Inf. Comput., 1995)

A PCNF is false iff there is a derivation of the empty clause $\Box$ in the Q-resolution calculus.

Example

On blackboard

A resolution calculus for QBFs (cont'd)

Is the following rule allowed/sound?

Definition (Possible resolution rule over $\forall$ variables)

Let $C \vee x$ and $D \vee \neg x$ be two clauses, where C, D are disjunctions of literals and x is a universal variable. With the $\forall$ resolution rule

$$\frac{C \vee x \quad D \vee \neg x}{C \vee D} \; \forall Res$$

we derive the resolvent $C \vee D$ from the indicated parent clauses.

Outline

Why sequent systems?

- Give a nice way for non-normal form theorem proving
  (not only for QBFs; also for propositional/FO/non-classical logic)
- Vast amount of proof theoretical knowledge about them
- Easy to implement (as we will see $\hookrightarrow$ qpro)

Definition (Sequent)

A sequent S is an ordered pair of the form $\Gamma \vdash \Delta$, where Γ (antecedent) and Δ (succedent) are finite multisets of formulas. We write "$\vdash \Delta$" or "$\Gamma \vdash$" whenever Γ or Δ is the empty sequence, respectively.

The propositional rules of a sequent calculus for QBFs

$$\frac{\Gamma \vdash \Delta}{\Phi, \Gamma \vdash \Delta}\ wl \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, \Phi}\ wr$$

$$\frac{\Gamma_1, \Phi, \Phi, \Gamma_2 \vdash \Delta}{\Gamma_1, \Phi, \Gamma_2 \vdash \Delta}\ cl \qquad \frac{\Gamma \vdash \Delta_1, \Phi, \Phi, \Delta_2}{\Gamma \vdash \Delta_1, \Phi, \Delta_2}\ cr$$

$$\frac{\Gamma \vdash \Delta, \Phi}{\neg\Phi, \Gamma \vdash \Delta}\ \neg l \qquad \frac{\Phi, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg\Phi}\ \neg r$$

$$\frac{\Phi, \Psi, \Gamma \vdash \Delta}{\Phi \wedge \Psi, \Gamma \vdash \Delta}\ \wedge l \qquad \frac{\Gamma \vdash \Delta, \Phi \qquad \Gamma \vdash \Delta, \Psi}{\Gamma \vdash \Delta, \Phi \wedge \Psi}\ \wedge r$$

$$\frac{\Phi, \Gamma \vdash \Delta \qquad \Psi, \Gamma \vdash \Delta}{\Phi \vee \Psi, \Gamma \vdash \Delta}\ \vee l \qquad \frac{\Gamma \vdash \Delta, \Phi, \Psi}{\Gamma \vdash \Delta, \Phi \vee \Psi}\ \vee r$$

$$\frac{\Gamma \vdash \Delta, \Phi \qquad \Psi, \Gamma \vdash \Delta}{\Phi \rightarrow \Psi, \Gamma \vdash \Delta}\ \rightarrow l \qquad \frac{\Phi, \Gamma \vdash \Delta, \Psi}{\Gamma \vdash \Delta, \Phi \rightarrow \Psi}\ \rightarrow r$$

The axioms and quantifier rules for the calculus

The axioms: $\Phi \vdash \Phi$ Ax $\qquad \bot \vdash \ \bot l \qquad \vdash \top \ \top r$

Some possible quantifier rules:

$$\frac{\Gamma \vdash \Delta, \Phi\{p/q\}}{\Gamma \vdash \Delta, \forall p\,\Phi}\ \forall r_e \qquad \frac{\Phi\{p/q\}, \Gamma \vdash \Delta}{\exists p\,\Phi, \Gamma \vdash \Delta}\ \exists l_e$$

$$\frac{\Phi\{p/\Psi\}, \Gamma \vdash \Delta}{\forall p\,\Phi, \Gamma \vdash \Delta}\ \forall l_f \qquad \frac{\Gamma \vdash \Delta, \Phi\{p/\Psi\}}{\Gamma \vdash \Delta, \exists p\,\Phi}\ \exists r_f$$

$$\frac{\Phi\{p/\top\}, \Phi\{p/\bot\}, \Gamma \vdash \Delta}{\forall p\,\Phi, \Gamma \vdash \Delta}\ \forall l_s \qquad \frac{\Gamma \vdash \Delta, \Phi\{p/\top\}, \Phi\{p/\bot\}}{\Gamma \vdash \Delta, \exists p\,\Phi}\ \exists r_s$$

$$\frac{\Gamma \vdash \Delta, \Phi\{p/\top\} \wedge \Phi\{p/\bot\}}{\Gamma \vdash \Delta, \forall p\,\Phi}\ \forall r_s \qquad \frac{\Phi\{p/\top\} \vee \Phi\{p/\bot\}, \Gamma \vdash \Delta}{\exists p\,\Phi, \Gamma \vdash \Delta}\ \exists l_s$$

q does not occur as a free variable in the conclusion of $\forall r_e$ / $\exists l_e$

The basic algorithm

- Based on DPLL (successful in SAT-/QBF-solving in (P)CNF)
- Relatively simple extension for nonprenex QBFs in NNF (implementation follows the semantics using *s* quantifier rules)

```
BOOLEAN split(QBF φ in NNF) {

switch (simplify (φ)):  /* simplify works inside φ */
   case ⊤:  return True;
   case ⊥:  return False;

   case (φ1 ∨ φ2):  return (split(φ1) || split(φ2));
   case (φ1 ∧ φ2):  return (split(φ1) && split(φ2));
   case (QXψ):  select x ∈ X;
      if Q = ∃ return (split(∃Xψ[x/⊥]) || split(∃Xψ[x/⊤]));
      if Q = ∀ return (split(∀Xψ[x/⊥]) && split(∀Xψ[x/⊤]));
}
```

Simplifying formulas

simplify(ϕ): returns ϕ' simplified wrt some equivalences:

(a) $\neg\top \Rightarrow \bot$; $\quad \neg\bot \Rightarrow \top$;

(b) $\top \wedge \phi \Rightarrow \phi$; $\quad \bot \wedge \phi \Rightarrow \bot$; $\quad \top \vee \phi \Rightarrow \top$; $\quad \bot \vee \phi \Rightarrow \phi$;

(c) $(Qx\,\phi) \Rightarrow \phi$, if Q $\in \{\forall, \exists\}$, and x does not occur in ϕ;

(d) $\forall x(\phi \wedge \psi) \Rightarrow (\forall x \phi) \wedge (\forall x \psi)$;

(e) $\forall x(\phi \vee \psi) \Rightarrow (\forall x \phi) \vee \psi$, whenever x does not occur in ψ;

(f) $\exists x(\phi \vee \psi) \Rightarrow (\exists x \phi) \vee (\exists x \psi)$;

(g) $\exists x(\phi \wedge \psi) \Rightarrow (\exists x \phi) \wedge \psi$, whenever x does not occur in ψ.

Rewritings (d)–(g) are known as miniscoping

Additional mechanisms

- Basic procedure clearly not sufficient for competitive solver
- Desirable extension: generalization of pruning techniques
  - Unit literal elimination
  - Pure literal elimination
  - Dependency-directed backtracking
    (works for true and false subproblems)
  - Learning

➥ `split` looks like an implementation of a sequent calculus

➥ Extensions of `split` formalized as a sequent calculus (for NNF)

The Logical Rules of the Sequent Calculus $GQBF_1$

$$\frac{\vdash \phi}{\vdash \phi \vee \psi}\ (\vee') \qquad \frac{\vdash \psi}{\vdash \phi \vee \psi}\ (\vee'') \qquad \frac{\vdash \phi \qquad \vdash \psi}{\vdash \phi \wedge \psi}\ (\wedge)$$

$$\frac{\vdash \phi[x/\bot]}{\vdash \exists x \phi}\ (\exists') \qquad \frac{\vdash \phi[x/\top]}{\vdash \exists x \phi}\ (\exists'') \qquad \frac{\vdash \phi[x/\bot] \quad \vdash \phi[x/\top]}{\vdash \forall x \phi}\ (\forall)$$

- The logical rules (l-rules) simulate the basic algorithm
- Sequents consist of one formula only
- Axioms of $GQBF_1$ are $\vdash \top$ and $\vdash \neg\bot$

➥ Use proofs in $GQBF_1$ to prove properties of techniques like DDB more elegantly

➥ Are such proofs in $GQBF_1$ useful for applications?

Simplification Rules

- GQBF_1 extended by simplification rules like

$$\frac{\vdash \phi(\top)}{\vdash \phi(\top \vee \psi)}\ \text{(S3a)} \qquad \frac{\vdash \phi(\psi)}{\vdash \phi(\bot \vee \psi)}\ \text{(S3b)}$$

$$\frac{\vdash \phi(\psi)}{\vdash \phi(\mathsf{Q}x\,\psi)}\ \text{(S4) no occurrences of } x \text{ in } \psi$$

- Very important that they work inside formulas
(circumventing the usual decomposition strategy)
- Allow for short proofs for (simple) formulas like

$$\exists x_n \forall y_n \cdots \exists x_1 \forall y_1\, (x_n \vee y_n \vee \cdots \vee x_1 \vee y_1)$$

- ... which have only long proofs in GQBF_1