Partial Quantifier Elimination

Eugene Goldberg, Pete Manolios
Northeastern University, USA

HVC-2014, November 18-20,
Haifa, Israel

Outline

- Partial Quantifier Elimination (PQE)
- Solving QE and PQE
- Experimental results

Quantifier Elimination (QE)

Let $F(X,Y)$ be a Boolean CNF formula

QE problem:

Given $\exists X\,[F\,]$, find a CNF formula $F^*(Y)$

such that $F^* \equiv \exists X\,[F\,]$

$F^*(\boldsymbol{y}) = \exists X\,[F(\boldsymbol{y})]$ for every complete assignment $\boldsymbol{y}$ to Y

SATus Quo

- Straightforward QE is hard
- Best model checkers use SAT rather than QE

A different approach based on partial QE:

⬇

Perform reachability analysis light

⬇

A model checker that can break new ground (e.g. finding very deep bugs)

Partial QE (PQE)

Let $F(X,Y)$, $G(X,Y)$ be Boolean CNF formulas

PQE :

Given $\exists X\,[F \wedge G]$, find $F^*(Y)$ s.t.

$$F^* \wedge \exists X\,[G] \equiv \exists X\,[F \wedge G]$$

Replace quantified F with quantifier-free F^*

QE is a degenerate case of PQE where G is empty

Reachability Analysis Light

$T(S,S')$ - transition relation,

s - a state (an assignment to S)

$C_{\mathbf{s}}$ - the longest clause falsified by ***s***

s satisfies $\sim C_{\mathbf{s}}$ and falsifies $C_{\mathbf{s}}$

All $_{\mathbf{s}}$ **:** $R_s \equiv \exists S\, [\sim C_s \wedge T\,]$

The assignments satisfying $R_{\mathbf{s}}$ specify all states reachable from ***s*** in one transition

Only $_{\mathbf{s}}$ **:** $Q_s \wedge \exists S\, [T\,] \equiv \exists S\, [C_s \wedge T\,]$

The assignments falsifying $Q_{\mathbf{s}}$ specify states reachable only from ***s*** in one transition

Reachability Analysis Light
(continued)

- $Only_s \subseteq All_s$
- $Only_s$ can be dramatically smaller than All_s
- It is sufficient to compute $Only_s$ rather than All_s
- $Only_s$ cannot be efficiently computed by a traditional CDCL SAT-solver

Outline

- Partial Quantifier Elimination (PQE)
- Solving QE and PQE
- Experimental results

Our Approach To QE
(FMCAD 12, 13)

Find F^* such that $F^* \equiv \exists X\, [F\,]$

An ***X*-clause** is a clause with a variable of X

1) **Make** X-clauses **redundant** in $\exists X\, [F\,]$ by adding resolvents

Redundancy of X-clause C: $\exists X\, [F\,] \equiv \exists X\, [F \setminus \{C\}\,]$

2) **Use branching** to prove redundancy of X-clauses in subspaces

3) **Use** the **machinery of dependency sequents** to merge results of branches

QE versus SAT
(why one needs dependency sequents)

SAT: Is *F* satisfiable?

QE: Find F^* s.t. $F^* \equiv \exists X\, [F\,]$

Trivial termination condition:

- finding satisfying assignment
- deriving an empty clause

Non-trivial termination condition:

- deriving a "sufficient" number of clauses depending of free variables

No need to reason about subspaces where *F* is satisfiable

One has to reason about subspaces where *F* is satisfiable

Dependency Sequents (D-sequents)

D-sequents are used to record that a set of X-clauses is redundant in a subspace

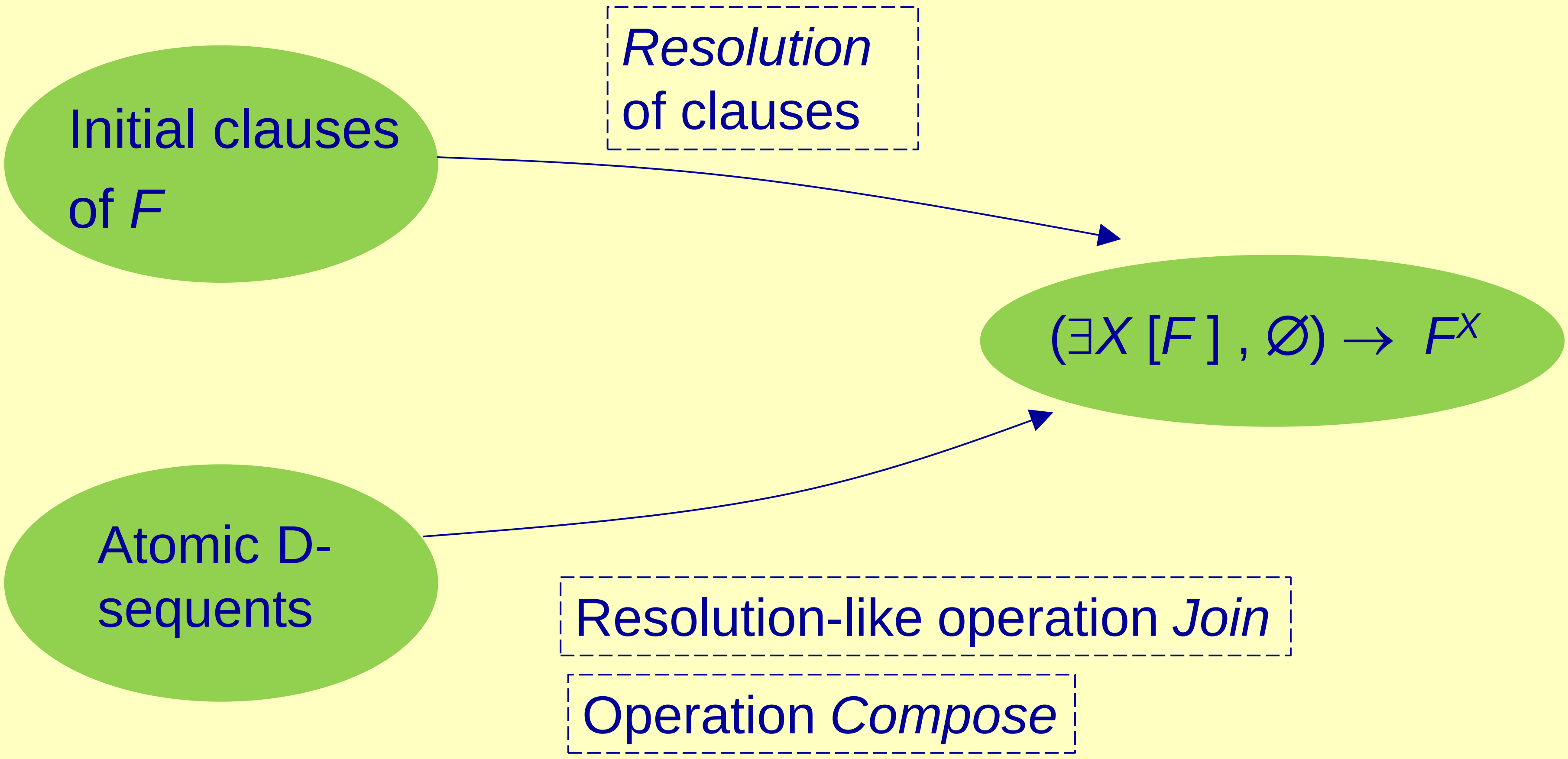
Let $\boldsymbol{q}$ be an assignment to $Vars(F)$.

Let F^X denote the X-clauses of F

A D-sequent: $(\exists X\,[F],\, \boldsymbol{q}) \rightarrow R$, *where* $R \subseteq F^X$

Semantics: R is redundant in $\exists X\,[F]$ in subspace $\boldsymbol{q}$

D-Sequent Calculus



Solving PQE

Given $\exists X\, [F \wedge G\,]$

QE: Derive $(\exists X\, [F \wedge G\,]\,,\, \varnothing) \rightarrow F^X \cup G^X$

PQE: Derive $(\exists X\, [F \wedge G\,]\,,\, \varnothing) \rightarrow F^X$

PQE can be solved similarly to QE by:

- Adding resolvent clauses to *F*
- Proving redundancy of *X*-clauses of *F* and some *X*-clauses of *G* in subspaces
- Merging results of branches using D-sequents

Outline

- Partial Quantifier Elimination (PQE)
- Solving QE and PQE
- Experimental results

PQE versus QE: traditional model checking

We compared two algorithms of backward model checking

MC-PQE: computes pre-image by PQE

MC-QE: computes pre-image by QE (FMCAD-13)

We used HWMCC-10 benchmarks

Time limit: 2,000 s.

Results on Some Concrete Benchmarks

bench-mark	**#latches**	**#gates**	**#iter-ations**	**bug**	**MC-QE (s.)**	**MC-PQE (s.)**
bj08amba3g62	32	9,825	4	no	241	38
kenflashp03	51	3,738	2	no	33	104
pdtvishuffman2	55	831	6	yes	> 2,000	296
pdtvisvsar05	82	2,097	4	no	1,368	7.7
pdtvisvsa16a01	188	6,162	2	no	> 2,000	17
texaspimainp12	239	7,987	4	no	807	580
texasparsesysp1	312	11,860	10	yes	39	25
pj2002	1,175	15,384	3	no	254	47
mentorbm1and	4,344	31,684	2	no	1.4	1.7

Conclusions

- QE is inherently hard $\Rightarrow$ look for QE light
- PQE is a light version of QE
- Experiments show superiority of PQE over QE
- PQE facilitates new methods of model checking
- PQE is enabled by D-sequents

Next step: D-sequent re-using