Quantifier Elimination by Dependency Sequents

Eugene Goldberg, Pete Manolios Northeastern University, USA

FMCAD-2012, October 22-25, Cambridge, UK

- Introduction
- Dependency sequents
- Algorithm description
- Experimental results
- Conclusions

Quantifier Elimination (QE)

Let F be a Boolean CNF formula and $X \subseteq Vars(F)$.

QE problem:

Given $\exists X[F]$, find a quantifier free CNF formula *G* such that $G \equiv \exists X[F]$

 $G \equiv \exists X [F]$ means that $G_s = \exists X [F_s]$ for every complete assignment **s** to $Vars(F) \land X$

QE is important in reachability analysis and model checking

Existing QE Methods

BDD based methods suffer from memory explosion

Two kinds of SAT based QE algorithms:

- enumeration of satisfying assignments: McMillan 2002, Ganai,Gupta, Ashar 2004, Jin, Somenzi 2005, Brauer, King, Kriener 2011
- variable elimination:

Davis, Putnam 1960, Jiang 2009, Goldberg, Manolios 2010

SAT based QE methods have poor scalability

Two Ideas of Our Approach

1) Add resolvent-clauses to *F* until variables of *X* become redundant in $\exists X [H]$ where $H \supseteq F$

Redundancy means that $\exists X [H] \equiv \exists X [H \setminus H^X]$ where $H^X = \{$ the clauses of H with a variable of $X \}$.

2) Use branching to prove variable redundancy in subspaces and merge results of different branches

Beyond Resolution

Termination condition: stop when X is redundant in $\exists X[F]$.

Let $Y = Vars(F) \setminus X$ and **s** be a complete assignment to Y.

F_s is unsatisfiable: there is a resolvent *C*, s.t. C(s) = 0. *X* is redundant in $\exists X [(F \land C)_s]$. Resolution works.

 F_s is satisfiable, there is no resolvent C s.t. C(s) = 0. Yet X is redundant in $\exists X [F_s]$. Beyond resolution!

- Introduction
- Dependency sequents
- Algorithm description
- Experimental results
- Conclusions

Dependency Sequents (D-sequents)

Let **s** be a partial assignment to Vars(F)A D-sequent for $\exists X [F]$ has the form $(\exists X [F], \mathbf{s}) \rightarrow Z$, where $Z \subseteq X$

Semantics: Z is redundant in $\exists X [F_s]$ We will call **s** the conditional part of the D-sequent

Properties:

a) If *F* implies *G*, $(\exists X[F \land G], \mathbf{s}) \rightarrow Z$ also holds b) If $\mathbf{s} \subseteq \mathbf{q}$, $(\exists X[F], \mathbf{q}) \rightarrow Z$ also holds

Atomic D-sequents

- 1. Let $v \in X$ be monotone in F in subspace s (pure literal). Then v is redundant in $\exists X[F_s]$ and so D-sequent ($\exists X[F], s$) $\rightarrow \{v\}$ holds.
- 2. Let a clause *C* of *F* be falsified in subspace *s* Every $v \in X \setminus Vars(s)$ is redundant in $\exists X[F_s]$ So D-sequent $(\exists X[F], s) \rightarrow \{v\}$ holds

Often, $(\exists X[F], \mathbf{s}) \rightarrow \{v\}$ can be replaced with $(\exists X[F], \mathbf{s}^*) \rightarrow \{v\}$ where $\mathbf{s}^* \subset \mathbf{s}$

Joining D-sequents



Importantly, $F_{s'}$ and/or $F_{s''}$ may be satisfiable. Here we go beyond resolution that reasons only over subspaces where *F* is unsatisfiable.

- Introduction
- Dependency sequents
- Algorithm description
- Experimental results
- Conclusions

Derivation of D-sequents (DDS)

Ω is the current set of D-sequents $(\exists X [F], \mathbf{s}_v) → \{v\}$, where v ∈ X

DDS($\exists X[F], \mathbf{s}, \Omega$) /* Returns $\exists X[F], \Omega, C$ */

1. $(\Omega, C) := atomic_D_seqs(\exists X [F], s, \Omega)$

2. if (vars of X are assign. or redund.) return $(\exists X [F], \Omega, C)$

3. $v = pick_var(Vars(F) \setminus (Assigned(s) \cup Redundant(\Omega)))$

4. $(\exists X[F], \Omega, C_0) := DDS (\exists X[F], \mathbf{s} \cup (v=0), \Omega)$

5. $(\exists X[F],\Omega,C_1) := DDS (\exists X[F], \mathbf{s} \cup (v=1), \Omega \setminus \Omega_{(v=0)})$

6. Return (*Merge_Branches*($\exists X [F], \mathbf{s}, v, \Omega_{(v=0)}, \Omega, C_0, C_1$))

Merging Branches

 $Merge_Branches(\exists X [F], \mathbf{s}, v, \Omega_{(v=0)}, \Omega, C_0, C_1)$

- 1. if $((C_0 \neq nil)$ and $(C_1 \neq nil))$
- 2. { $C := resolve(C_0, C_1, v)$
- 3. $F := F \wedge C$
- 4. $\Omega := update_D_seqs(s, C, \Omega)$
- 5. return($\exists X [F], \Omega, C$) }
- 6. $\Omega := join_D_seqs(v, \Omega_{(v=0)}, \Omega_{(v=1)})$
- 7. if $(v \in X)$
- 8. $\Omega := \Omega \cup \{ atomic_D_seq(\exists X [F], s, v) \}$
- 9. return(∃*X* [*F*],Ω,*nil*)

A Few Remarks About DDS

- To simplify implementation of DDS
 a) it first branches on variables of Vars(F) \ X
 b) D-sequents are not re-used
- DDS first branches on vars of unit clauses (BCP)
- If *F* is unsatisfiable, DDS behaves as a conflict driven SAT-solver. (But this is a "degenerate" case.)
- The novelty of DDS comes into play in subspaces where *F* is satisfiable.

- Introduction
- Dependency sequents
- Algorithm description
- Experimental results
- Conclusions

Model Checking Experiments

Model	EnumSA	QE-GBL	DDS
checking	solved	solved	solved
mode	(%)	(%)	(%)
forward	425 (56%)	561 (74%)	664 (87%)
backward	97 (12%)	522 (68%)	563 (74%)

- EnumSA: enumeration of satisf. assignments (CAV-11)
- QE-GBL : quantifies variables away globally (HVC-10)
- 758 benchmarks from HWMCC'10
- The algorithms performed one step of forward/backward model checking
- Timeout limit is 1 minute

Forward Model Checking



Backward Model Checking



Conclusions

- We introduced a QE algorithm based on D-sequents
- D-sequents can be used in many other applications
- We experimented with a very simple implementation The results of experiments are very encouraging
- Some points not covered in this talk are addressed in the paper (e.g. the compositionality of DDS)