Approximate Quantifier Elimination for Propositional Boolean Formulae

Jörg Brauer

Andy King

20.04.2011 @ NFM‘11
Motivation

• Quantifier elimination on Boolean formulae in
  – Unbounded symbolic model checking, predicate abstraction, dependency analysis, transfer function synthesis, information flow analysis, ranking function synthesis, etc.

• Computationally expensive operation
  – Model enumeration using SAT possible
  – Still potentially too expensive
  – Especially when result should be in CNF
Approach

• To compute $\exists x_1, \ldots, x_n : \varphi$ in CNF, you classically eliminate the $x_i$ one after another

• Only final result is free of $x_1, \ldots, x_n$

• We compute $C_i$ such that $\exists x_1, \ldots, x_n : \varphi \models C_i$
  
  – Then $C_i$ over-approximates $\exists x_1, \ldots, x_n : \varphi$

• Refine over-approximation as

  $\exists x_1, \ldots, x_n : \varphi \models C_i \land C_j$

• The $C$ clauses derived from prime implicants
Dual-Rail Encoding for Implicants

- Consider
  \[ \varphi = (\neg x \lor z) \land (y \lor z) \land (\neg x \lor \neg w \lor \neg z) \land (w \lor \neg z) \]
- Goal: eliminate \( z \) from \( \varphi \) such that \( \exists z : \varphi \) in CNF
- Dual-rail encoding
  - Introduce fresh variables
  - Replace positive and negative literals

\[ \tau(\varphi) = \left\{ \begin{align*}
(x^- \lor z) & \land (y^+ \lor z) & \land (x^- \lor w^- \lor \neg z) & \land (w^+ \lor \neg z) & \land \\
(\neg w^+ \lor \neg w^-) & \land (\neg x^+ \lor \neg x^-) & \land (\neg y^+ \lor \neg y^-)
\end{align*} \right. \]
Dual-Rail Encoding for Implicants

• Passing $\tau(\varphi)$ to SAT solver gives a model

$$M = \begin{cases} 
  w^+ &\mapsto 1, & w^- &\mapsto 0, & x^+ &\mapsto 0, & x^- &\mapsto 1, \\
  y^+ &\mapsto 0, & y^- &\mapsto 0, & z &\mapsto 1
\end{cases}$$

• $M$ defines $(w \land \neg x)$, i.e., $(w \land \neg x) \models \exists z : \varphi$
  – Then add blocking clause and proceed

• Observe: $(w \land \neg x)$ under-approximates $\exists z : \varphi$

• So how about applying this to $\neg \varphi$?
Pushing Negations Around

\[ \nu \models \forall z : \neg \varphi \quad \text{iff} \quad \neg \forall z : \neg \varphi \models \neg \nu \]

\[ \text{iff} \quad \exists z : \varphi \models \neg \nu \]

• To find over-approximation \( \neg \nu \) of \( \exists z : \varphi \)
  compute under-approximation of \( \forall z : \neg \varphi \)

• But:
  – Can only derive implicants of \( \exists z : \neg \varphi \)
  – Not implicants of \( \forall z : \neg \varphi \)
Strategy for Over-Approximating Implicants

- Observe that $\forall z : \neg \varphi \models \exists z : \neg \varphi$
  - A model of $\forall z : \neg \varphi$ is also a model of $\exists z : \neg \varphi$
  - But not vice versa

- Algorithm:
  - Negate $\varphi$ to obtain $\tau(\neg \varphi)$
  - Enumerate implicants $C'$ of $\exists z : \neg \varphi$
  - Filter those $C$ such that $C \not\models \forall z : \neg \varphi$
  - Then $\exists z : \varphi \models \neg C'$
Shortest Implicants: Sorting Networks

- Suppose sorter encoded as $\sigma$
- Cardinality constraint $i_1 + i_2 + i_3 = 2$ encoded as $o_1 \land o_2 \land \neg o_3$ in unary encoding
- $\tau(\neg \varphi) \land \sigma \land \land_{i=1}^k o_i \land \land_{i=k+1}^n \neg o_i$ specifies implicants of length $k$
Worked Example

• Take $\tau(\neg \varphi)$
• First, $\nu_1 = (\neg w)$ but $\exists z : \varphi \not\models \neg \nu_1$, so discard
• Then, $\nu_2 = (x)$ and $\exists z : \varphi \models \neg \nu_2$
• No more implicants of length 1
• Now, $\nu_3 = (\neg w \land \neg y)$ and $\exists z : \varphi \models \neg \nu_3$
• No more implicants, thus $\exists z : \varphi = (\neg x) \land (w \lor y)$
Some Experiments

• Written in Java on top of SAT4J
• Benchmark set from CNF encodings of ISCAS-85 hardware circuits
• Observed small CNF representation for quantifier-free formulae
• Runtime suffers from spurious candidates
  – Can be mitigated to some extent using co-factoring
• Traditional SAT-based algorithms rely on model enumeration (giving a DNF stored in BDDs)
  – If too expensive, no result can be computed
  – Our algorithm can still compute over-approximation
So as to not Cause Offense

- McMillan (CAV‘02)
- Lahiri et al. (CAV‘03 & CAV‘06)
- Monniaux (CAV‘10)
- Kettle et al. (TACAS‘06)
- Bryant (IEEE‘87)
- Manquinho et al. (ICTAI‘97)
- Brauer et al. (CAV‘11)
- And many more ...
Conclusion

• Based on dual-rail encoding to derive implicants
• Combined with sorting networks so as to obtain shortest prime implicants
• Start with over-approximation which is then incrementally refined
• Algorithm is thus *anytime*
Thank you very much!