Interpolation and SAT-based Model Checking

K.L. McMillan
Cadence Berkely Labs

A "How-To" presented by

Scott Cotton

IMPRS Software Model Checking Seminar
Summer, 2005

scotton@mpi-sb.mpg.de

Main results

- Fully symbolic, fully SAT-based method for model checking.
- Can do *unbounded* model checking.
- Does not heavily depend on number of inputs or free variables in systems.
- Works well for *localizable* properties.

How-To Roadmap

- Background and Overview.
- Encoding the transition system for SAT.
- Interpolants.
- SAT-based reachability algorithm.
- Implementation and optimizations.
- Conclusion and discussion.

How-To Roadmap

- Background and overview.
- Encoding the transition system for SAT.
  - Unfoldings.
  - Prefixes and suffixes.
- Interpolants.
- SAT-based reachability algorithm.
- Implementation and optimizations.
- Conclusion and discussion.

How-To Roadmap

- Background and overview.
- Encoding the transition system for SAT.
- Interpolants.
  - Definition.
  - As approximate reachability operator.
  - Finding interpolants.
- SAT-based reachability algorithm.
- Implementation and optimizations.
- Conclusion and discussion.

How-To Roadmap

- Background and overview.
- Encoding the transition system for SAT.
- Interpolants.
- SAT-based reachability algorithm.
  - Top level pseudocode.
  - Termination conditions
- Implementation and optimizations.
- Conclusion and discussion.

How-To Roadmap

- Background and overview.
- Encoding the transition system for SAT.
- Interpolants.
- SAT-based reachability algorithm.
- Implementation and optimizations.
  - Generating resolution proofs.
  - Increasing bounds.
  - Using precise suffixes.
  - Reusing suffixes.
- Conclusion and discussion.

How-To Roadmap

- Background and overview.
- Encoding the transition system for SAT.
- Interpolants.
- SAT-based reachability algorithm.
- Implementation and optimizations.
- Conclusion and discussion.

Symbolic Model Checking

- Reduction of verification properties to properties of finite state systems.
- State space defined by an indexed set of Boolean variables $V = \{v_1, \ldots, v_n\}$.
- Each state s is a bit-vector $(s_1, \ldots, s_n)$.
- A transition system $TS = (I, T, F)$ is represented by Boolean formulas $(\varphi_I, \varphi_T, \varphi_F)$.
- No need to build an explicit representation of the TS.

Symbolic Model Checking with BDDs

- Translation of specifications and transition systems to QBF $(\exists V.\varphi)$.
- Representation of QBF in canonical graphical form.
- Evaluation of fixed point QBF formulas $\mu X.\varphi$ is used
  - To translate CTL formula.
  - To determine reachable set.
- Since QBF formulas are represented in canonical graphical form, fixed points are easy to detect.
- Some formulas will make the size of BDDs explode.
- Space requirements are the bottle neck and are hard to predict.

Basic Symbolic Model Checking with SAT

- Represent bounded runs of a transition system and verification properties in propositional logic.
- Every satisfying assignment is a run, every run is a counterexample.
- No counterexample exists if none is found for a sufficiently large bound.
- More space efficient than BDDs.
- The bound is often large and hard to determine precicesly.
- The procedure does not scale well with the bound.

Symbolic Model Checking with SAT + Interpolation

- Can also prove no counterexample exists with small bounds.
- Based on fixed point detection for approximate reachable set R.
- R is constructed with interpolants.
- R is a forward overapproximation *and also* a backward underapproximation.
- R becomes more precise as the bound increased.
- Bounds increased until fixed point or counterexample found.

SAT Encoding: Input and Assumptions

- A symbolic transition system $(\varphi_I, \varphi_T, \varphi_F)$ over variables V and V'.
- Runs of the TS are counterexamples.
- The transition relation T is total.
- Formulas represent the TS:

$$s \models \varphi_I \iff s \in I$$

$$s \models \varphi_F \iff s \in F$$

$$(s, s') \models \varphi_T \iff (s, s') \in T$$

SAT Encoding: Unfolding the TS (1/3)

- Use variables $\mathbf{W} = W_0, W_1, \ldots, W_k$.
- Each W_i is an indexed copy of V.
- Each W_i represents the states reachable in i steps.
- Notation:

$$\begin{aligned} \varphi_I^i &\stackrel{def}{=} \varphi_I[W_i/V] \\ \varphi_F^i &\stackrel{def}{=} \varphi_F[W_i/V] \\ \varphi_T^i &\stackrel{def}{=} \varphi_T[W_i/V, W_{i+1}/V'] \end{aligned}$$

SAT Encoding: Unfolding the TS (2/3)

We encode the runs of length $[j \dots k]$ as follows:

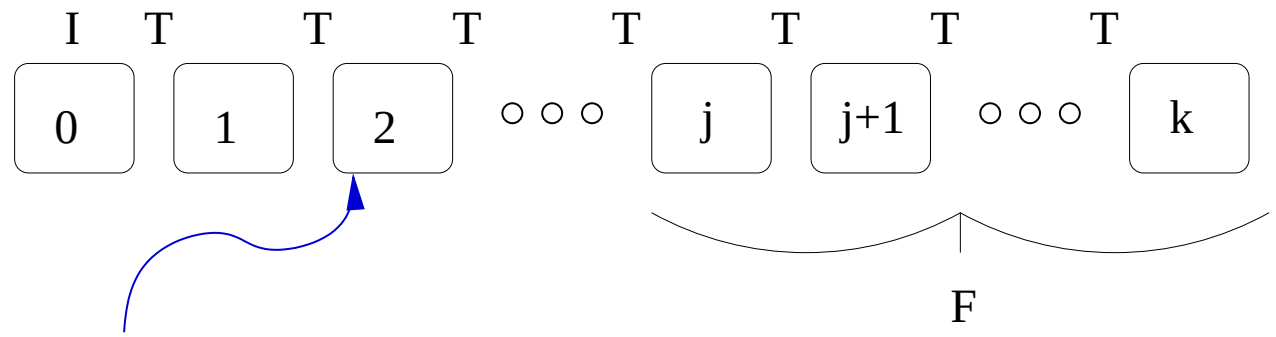
$$\mathrm{BMC}_j^k \stackrel{def}{=} \varphi_I^0 \wedge \Big(\bigwedge_{0 \le i < k} \varphi_T^i \Big) \wedge \Big(\bigvee_{j \le i \le k} \varphi_F^i \Big)$$

- Any satisfying assignment $A : \mathbf{W} \to \{0, 1\}$ encodes a run $s_0, s_1, \dots, s_h$ with $h \in [j \dots k]$.
- If BMC_j^k is not satisfiable, there is no run of length $[j \dots k]$ in the transition system.

Remark: Requires a total transition relation

SAT Encoding: Unfolding the TS (3/3)

$$\mathrm{BMC}_j^k \stackrel{def}{=} \varphi_I^0 \wedge \Big(\bigwedge_{0 \le i < k} \varphi_T^i \Big) \wedge \Big(\bigvee_{j \le i \le k} \varphi_F^i \Big)$$



SAT Encoding: Prefixes and Suffixes (1/2)

- Prefixes are valid paths, starting in some initial state but without considering accepting states:

$$\text{PREF}_h = \varphi_I^{-h} \wedge \Big(\bigwedge_{-h \leq i < 0} \varphi_T^i \Big)$$

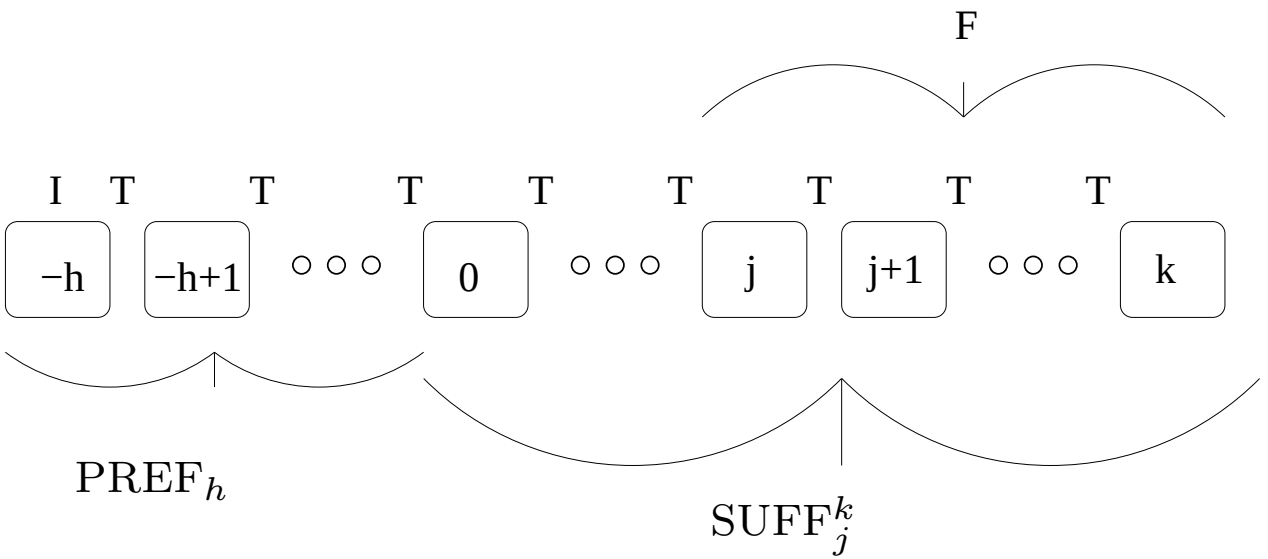
- Suffixes are valid paths, without respect to initial states, but containing an accepting state:

$$\text{SUFF}_j^k = \Big(\bigwedge_{0 \leq i < k} \varphi_T^i \Big) \wedge \Big(\bigvee_{j \leq i \leq k} \varphi_F^i \Big)$$

SAT Encoding: Prefixes and Suffixes (2/2)

$$\mathrm{PREF}_h \stackrel{def}{=} \varphi_I^{-h} \wedge \Big(\bigwedge_{-h \le i < 0} \varphi_T^i \Big)$$

$$\mathrm{SUFF}_j^k \stackrel{def}{=} \Big(\bigwedge_{0 \le i < k} \varphi_T^i \Big) \wedge \Big(\bigvee_{j \le i \le k} \varphi_F^i \Big)$$



SAT Encoding: Conclusion

- Sets of bounded runs are encoded in propositional logic.
- With a total transition relation, we can easily encode runs whose length falls in some range $[j \ldots k]$.
- Prefixes and suffixes: PREF_h, SUFF_j^k.

Interpolants: Definition

Given a pair of formulas (A, B) such that $A \wedge B$ is unsatisfiable, an interpolant for (A, B) is a formula P such that

1. $A \rightarrow P$.
2. $P \wedge B$ is unsatisfiable.
3. P contains only variables common to both A and B.

Example: $A = p \wedge (\neg q \vee \neg r) \quad B = q \wedge r \wedge s \quad P = \neg q \vee \neg r$

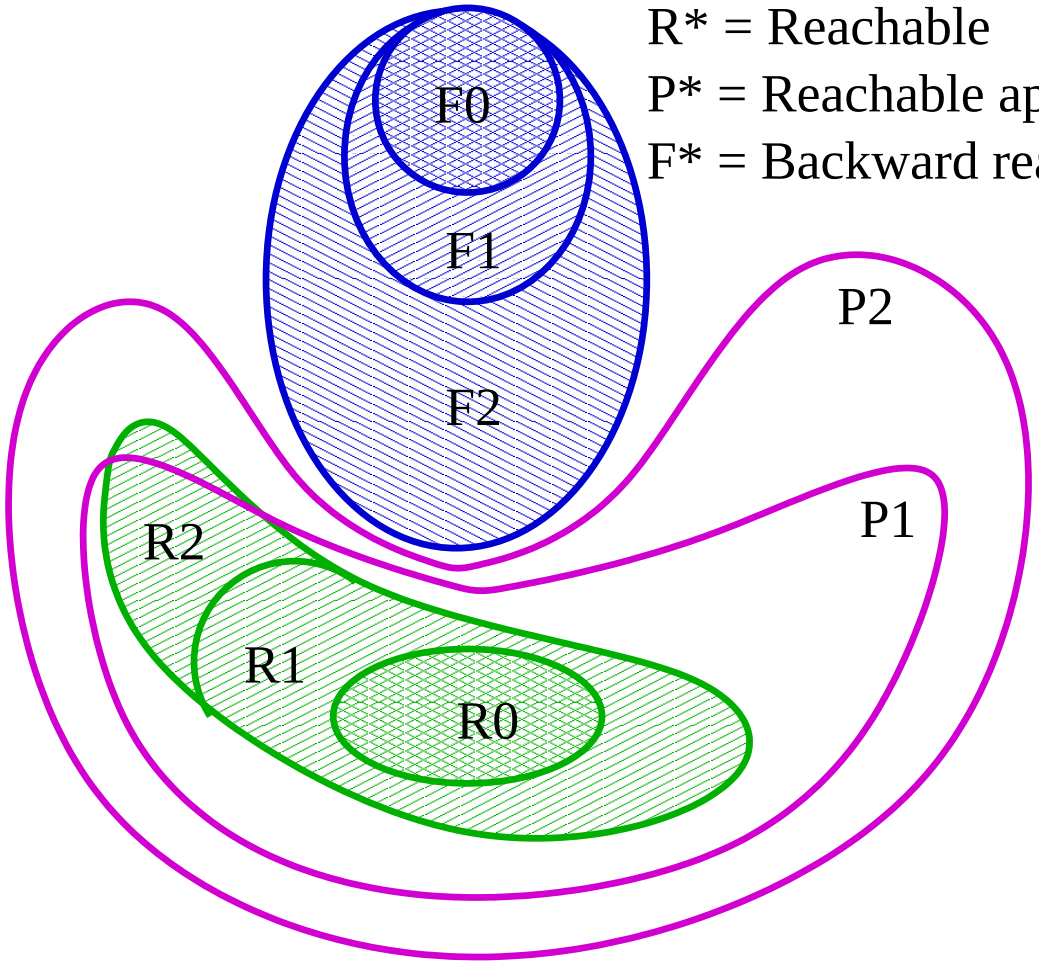
Interpolants and Reachability (1/4)

Main Idea: If $\mathrm{BMC}_0^{k+1}(\varphi_I, \varphi_T, \varphi_F)$ contains no counterexample (unsatisfiable):

1. Split BMC_0^{k+1} into PREF_1 and SUFF_0^k.
2. Find an interpolant P for the pair $(\mathrm{PREF}_1, \mathrm{SUFF}_0^k)$.
3. Reformulate and repeat for $\mathrm{BMC}_0^{k+1}(\varphi_I \vee P, T, F)$

$\mathrm{PREF}_1 \rightarrow P$	P overapproximates 1-reachable states
$P \wedge \mathrm{SUFF}_0^k$ is unsat	P underapproximates states which cannot reach F in $[0 \ldots k]$ steps

Interpolants and Reachability (2/4)



Interpolants and Reachability (3/4)

Proposition: Let $R_{\leq i} = \varphi_I \vee \bigvee_{1 \leq j \leq i} P_j$. If $P_{i+1} \rightarrow R_{\leq i}$, and $\mathrm{BMC}_0^k(R_{\leq j}, \varphi_T, \varphi_F)$ is unsat for $0 \leq j \leq i$ then there is no run for $(\varphi_I, \varphi_T, \varphi_F)$.

Why?

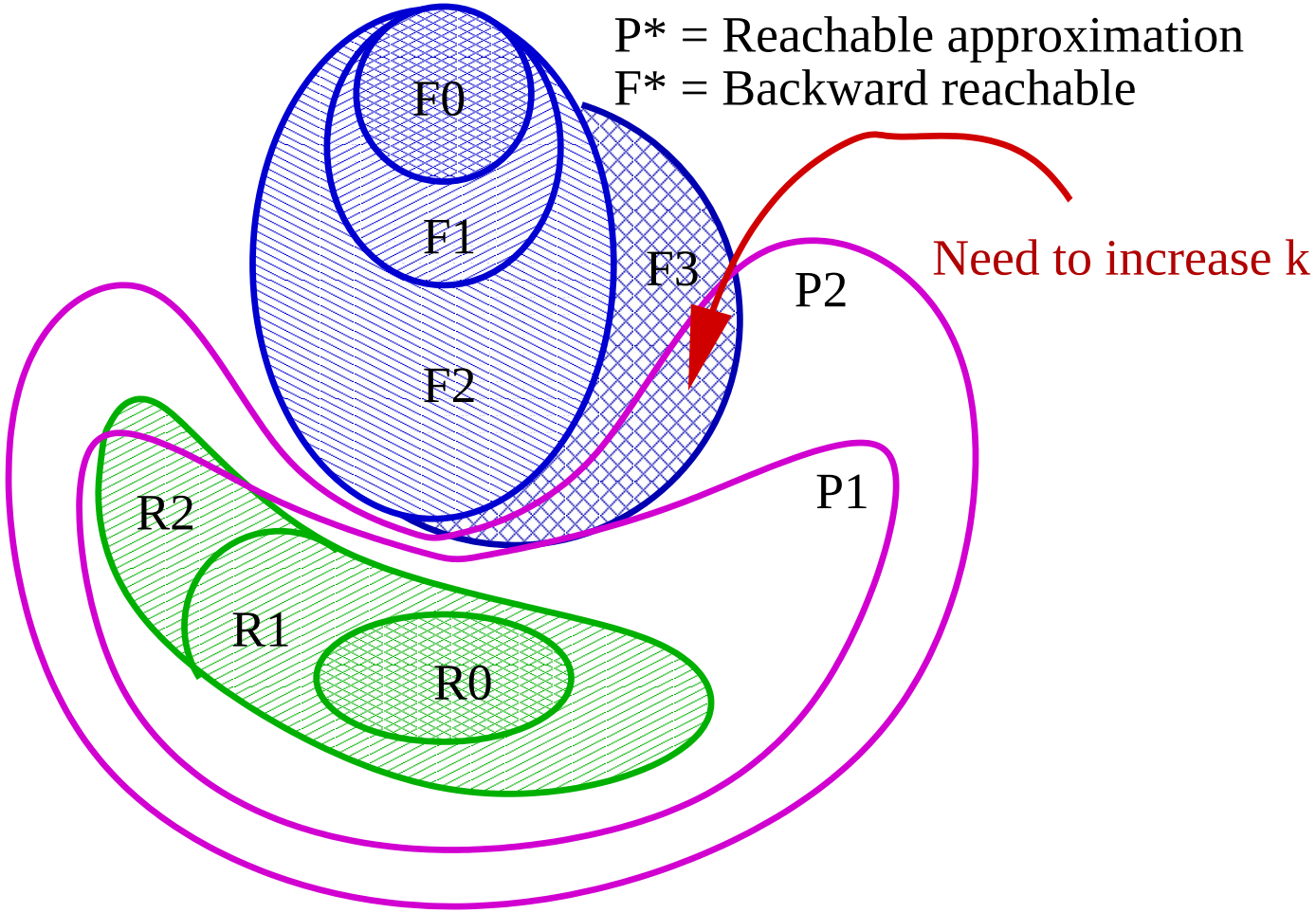
- Each P_i is an overapproximation of the i-reachable set.
- If $P_{i+1} \rightarrow R_{\leq i}$, then $P_{i+1} \subseteq R_{\leq i}$ (formulas represent sets of states).
- $R_{\leq i+1} = R_{\leq i}$ (fixed point reached).
- $\mathrm{BMC}_0^k(R_{\leq i}, \varphi_T, \varphi_F)$ is unsat, so $R_{\leq i} \cap F = \emptyset$.

Interpolants and Reachability (4/4)

R* = Reachable

P* = Reachable approximation

F* = Backward reachable



Interpolants and Reachability: Conclusion

- Interpolants can be used to determine that no counterexample exists, provided that
  - BMC_0^k is not satisfiable for an abstraction of $(\varphi_I, \varphi_T, \varphi_F)$.
- BMC_0^k may be satisfiable with a spurious counterexample if k is not sufficiently large.
- Spurious counterexamples come from the underapproximation of the set of states backwards reachable from F.

Finding Interpolants

- Some SAT solvers can produce a resolution proof of unsatisfiability.
- Resolution proofs can be used to efficiently derive interpolants.

Finding Interpolants: Resolution (1/4)

- General form of resolution rule:

$$\frac{\Gamma \vee x, \Delta \vee \neg x}{\Gamma \vee \Delta}$$

- Requires formula to be in CNF.
- $\Gamma \vee \Delta$ is called the resolvent.
- x is called the pivot variable.
- Example: the resolvent of $a \vee \neg b \vee c$ and $b \vee d$ is $a \vee c \vee d$.

Finding Interpolants: Resolution (2/4)

(CNF Reminder):

- A clause is a non tautological disjunction of literals $\bigvee_i l_i$.
- A literal is a variable x or its negation $\neg x$.
- A formula is in CNF if it is a conjunction of clauses.
- Translation to CNF is linear if extra variables are used, can be exponential otherwise.

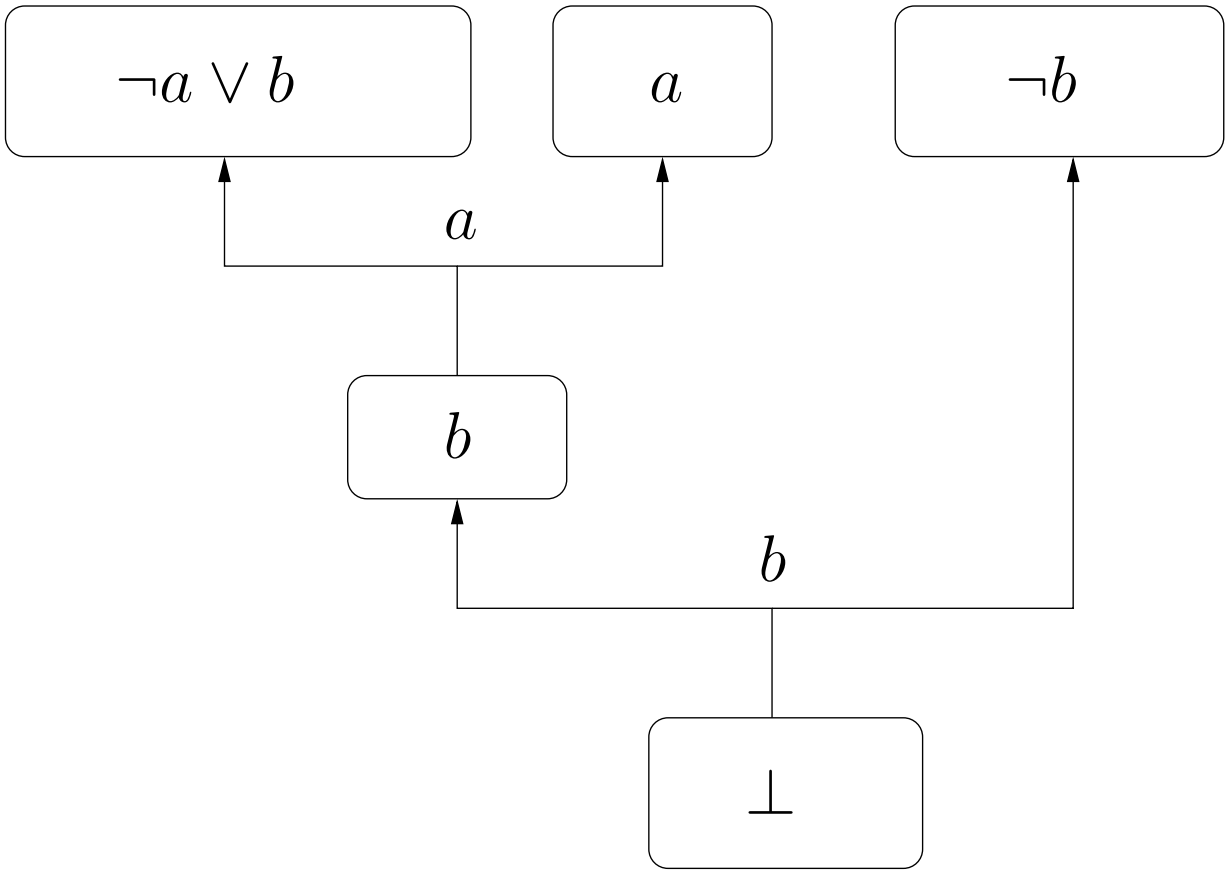
Finding Interpolants: Resolution (3/4)

Given an unsatisfiable set of clauses A, a resolution proof Π for A is a directed acyclic graph (V_Π, E_Π) with

- $A \subseteq V_\Pi$
- For every $a \in A$, a is a root ($E_\Pi(a) = \emptyset$).
- $\bot \in V_\Pi$ is the unique leaf ($E_\Pi^{-1}(\bot) = \emptyset$).
- For every $c \in (V_\Pi \setminus A)$,
  - c is the resolvent of two clauses $c_1, c_2 \in V_\Pi$.
  - $E_\Pi(c) = \{c_1, c_2\}$.

Finding Interpolants: Resolution (4/4)

Example Resolution Proof for $A = \{(\neg a \vee b), a, \neg b\}$



Finding Interpolants: Derivation (1/3)

Given a proof Π that $A \cup B$ is unsatisfiable, and an assignment $H : \text{Vars}(B) \to \{0, 1\}$, produce a formula P such that:

- $P[H/\text{Vars}(B)] = \bot \implies A$ is unsatisfiable.
- $P[H/\text{Vars}(B)] = \top \implies B$ is unsatisfiable.

Intuition: P "decides" to refute exactly one of A, B for any input.
Result: P is an interpolant.

Finding Interpolants: Derivation (2/3)

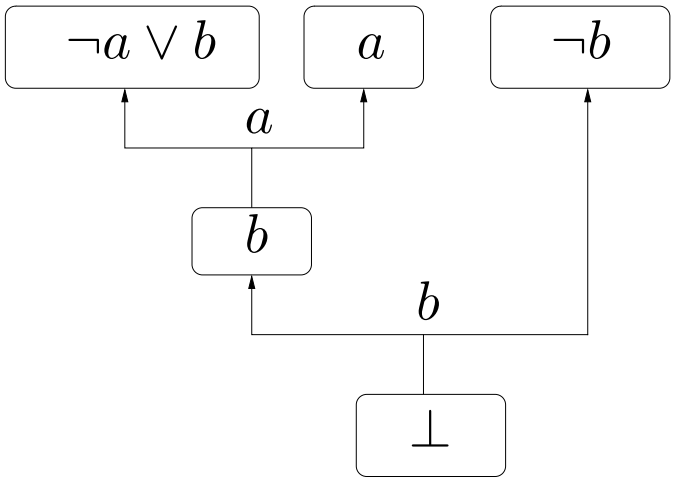
Idea: build P recursively on the structure of the proof by defining a recursive function $\gamma : V_\Pi \to (\text{Vars}(B) \to \{0, 1\})$

- For roots r:
  - If $r \in B$ then $\gamma(r) = \top$
  - If $r \in A$ then
    $\gamma(r) = \bigvee\{l \mid l \text{ is a literal in } r \text{ and } \text{Var}(l) \in \text{Vars}(B)\}$
- For internal nodes c derived from parents
  c_1 and c_2 via variable x:
  - If $x \in \text{Vars}(B)$ then $\gamma(c) = \gamma(c_1) \wedge \gamma(c_2)$.
  - If $x \notin \text{Vars}(B)$ then $\gamma(c) = \gamma(c_1) \vee \gamma(c_2)$.

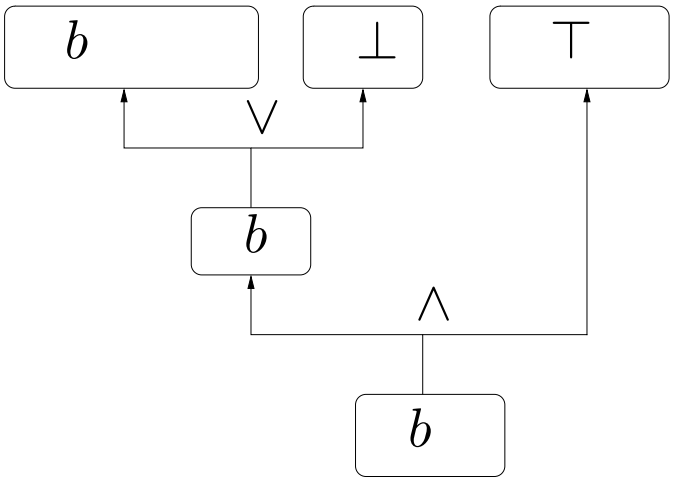
Finding Interpolants: Derivation (3/3)

$$A = \{(\neg a \vee b), a\}, B = \{\neg b\}$$

Refutation of $A \cup B$



Derivation of Interpolant



Interpolants: Conclusion

- Forward overapproximation.
- Backward underapproximation.
- Derivable from resolution proofs.

SAT-based Reachability: Top Level Pseudocode

$k = 0$; $R = \varphi_I$

if $\varphi_I \wedge \varphi_F$ is SAT **return** Reachable.

repeat

  $A = PREF_1(R, \varphi_T, \varphi_F)$

  $B = SUFF_0^k(R, \varphi_T, \varphi_F)$

  if $A \wedge B$ is SAT

    if $R = \varphi_I$ **return** Reachable

    else increase k; $R = \varphi_I$; **continue**

  let P be an interpolant for $A \wedge B$

  if $P \rightarrow R$

    return Not Reachable

  $R = R \vee P$

SAT-based Reachability: Termination

How big might k need to be?

- Let d be the length of the longest *shortest path* leading to a state in F.
- If $k > d$, then the approximation is adequate.
- Often possible that $k \ll d$ suffices.

SAT-based Reachability: Implementation

- SAT solvers work in CNF, but PREF and SUFF must be disjoint sets of clauses, and may not in CNF.
  $\Longrightarrow$ CNFization must be performed separately.
- Interpolants are not in CNF and can be highly redundant.
  $\Longrightarrow$ interpolants must be simplified and translated to CNF.
- Many DPLL SAT solvers can't produce refutations.
  $\Longrightarrow$ record resolutions during learning.
- How to check $P \rightarrow R$?
- What strategy to use to increase k?

SAT-based Reachability: Optimizations

- If the TS loops indefinitely in an accepting state, we can replace SUFF_0^k with SUFF_k^k.
  $\Longrightarrow$ resolution proofs are smaller, SAT solving faster.
- More generally, is there an automatic way of choosing a good j for SUFF_j^k?
- Reusing the clauses representing SUFF_0^k and clauses derived from these clauses across iterations.

Conclusion

Ken McMillan's "Interpolation and SAT-based Model Checking".

- Fully symbolic, fully SAT-based unbounded model checking.
- Approximate reachability, with controllable degree of approximation.
- Effective for localizable properties.
- Can be effective for systems with many inputs.

Questions?

- Encoding the transition system for SAT.
  - Unfolding, prefixes, suffixes.
- Interpolants.
  - Reachability.
  - Derivation.
- SAT-based reachability algorithm.
  - Pseudocode.
  - Termination.
- Implementation and optimizations.

Thank you.