Craig Interpolants
- Given: A ∧ B = 0, A ⇒ A’, A’ ∧ B = 0
  - A’ refers only to common variables of A,B
- Interpolants from proofs
- Given a resolution refutation of A ∧ B
  - A’ is derived in linear time and space [Pudlak,Krajicek’97]
- Interpolant as over-approximated image operator [McMillan’03]
- Works whenever a representation of \textit{bwd. reachable} space is given
  - A: From \wedge T (FWD)
  - B: paths to failure states (BWD)
  - A’: over-approximated image
- Approximated image is called 	extit{adequate} w.r.t. B

Interpolant Abstraction
- ITP+: take on improved Craig’s interpolation
- Incremental computation of interpolants using alternative techniques
  - Equivalence classes, mutual implications of state variables
  - Cube-based over-approximation, based on the detection of those state variables that are stuck at constant values

Contributions:
- Redundancy removal and reduction of UNSAT proofs and ITPs
  - Heuristic procedure for scalable ITP compaction
- Abstraction and refinement techniques for ITPs
  - Heuristic procedure for abstracting without resorting to resolution proofs

Abstraction by Iterative Refinements
Pseudo code:

1. Find atomic abstraction
2. Select abstraction
3. Until adequate
4. Return \(\text{To}^+ \land \text{ITP}((\text{From} \wedge T \land \text{Cone})^i)\)
   - or return Craig’s ITP

Abstraction classes
- Tightening
  - Equivalence state variables
  - Constant state variables
  - SAT-based enumeration
- Loosening
  - Localization abstraction
  - Ternary abstraction

ITP Proof Compaction
- Proof reduction
  - Recycle-pivots [Bar-inal & al.]
  - Exploiting proof topology (proof node chains)
- Logic synthesis manipulations on the proof
  - Constant propagation
  - BDD-based sweeping (for equivalences)
  - Observability Don’t Care (lightweight)

ITP Circuit Compaction
- Proof into AIG
  - ODC (structural)
  - Logic synthesis
    - rewrite / refactor
    - AIG balance
  - ITE-based decomposition (iff necessary)

Ad-hoc ITP Compaction Pseudo-code:

```
if (max recursions OR [ITP] < th)
  standardLogicSynth (ITP)
else do 
  searchNode(N) //with highest FO
  //compute collecters; equals to ITP ITE(N,ITE(ITP)))
  //size-based heuristic
  if (accept (ITE decomp))
    AigiteDecomp (N)
    AigiteDecomp (ITP)
    ITE = It(Comb(N,ITE,ITP))
  while max try reached
```

Obervability don’t care
- If \(A = 1 \rightarrow f(.)\) and \(g(.)\) can be simplified
- If \(A = 0 \rightarrow \text{out} = 0\), no matters \(f(.)\) or \(g(.)\)

Experimental results
- \begin{itemize}
  - Best Opt Time
  - Std Up Time
  - ITP Compaction
  - ITP Abstraction (partial data)
\end{itemize}