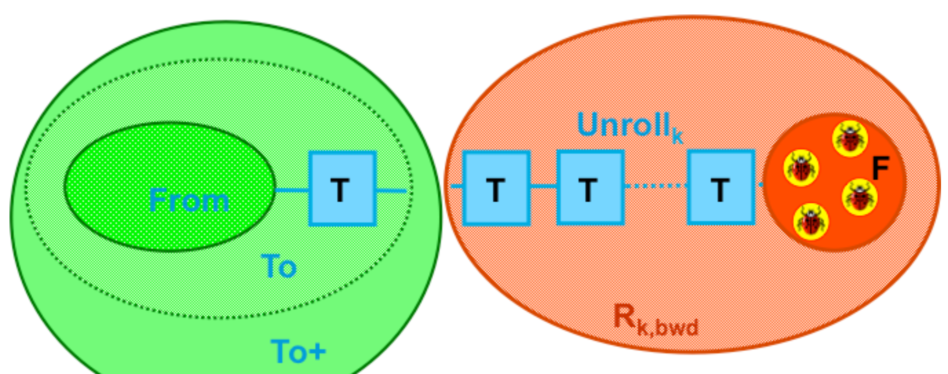
Exploiting Craig Interpolants in Unbounded Model Checking of Hardware Designs

Danilo Vendraminetto

Craig Interpolants

- Given: $A \wedge B = 0$, $A \Rightarrow A'$, $A' \wedge B = 0$
  - A' refers only to common variables of A,B
- A' = *interpolant*(A,B)
- Interpolants from proofs
- Given a resolution refutation of $A \wedge B$
  - A' is derived in linear time and space [Pudlak,Krajicek'97]



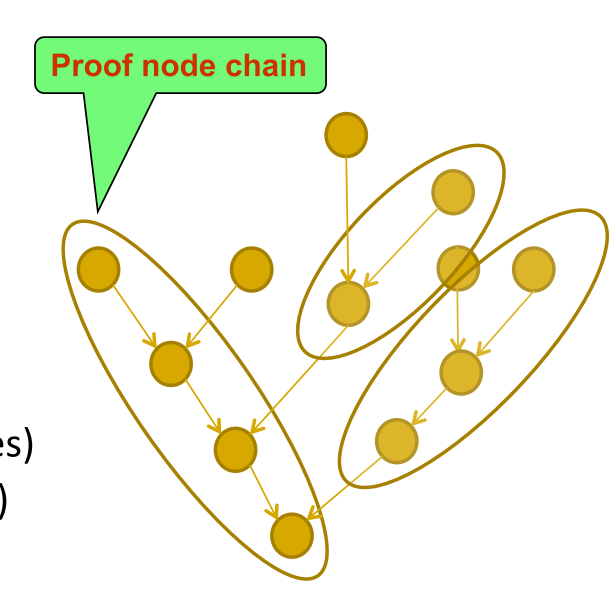
- Interpolant as over-approximated image operator [McMillan'03]
- Works whenever a representation of *bwd. reachable* space is given
  - A: From $\wedge$ T (FWD)
  - B: paths to failure states (BWD)
  - A': over-approximated image
- Approximated image is called *adequate* w.r.t. B

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

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

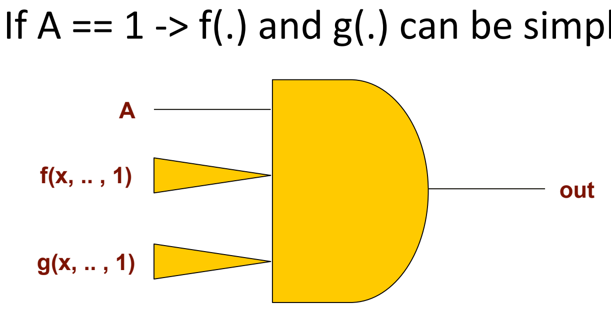
- Proof into AIG
  - ODC (structural)
  - Logic synthesis
    - rewrite / refactor
    - AIG balance
  - ITE-based decomposition (iff necessary)

Ad-hoc ITP Compaction Pseudo-code:

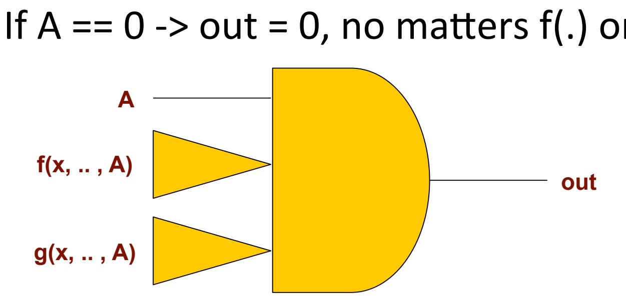
AigIteDecomp (ITP)
if (max recursions OR |ITP| < th)
    standardLogicSynth (ITP)
else do {
    searchNode(N_i) //with highest FO
    //compute cofactors; equals to ITP
    ITE(N_i,ITP_1,ITP_0)
    //size-based heuristic
    if (*accept* (ITE decomp))
        AigIteDecomp (N_i)
        AigIteDecomp (ITP_1)
        AigIteDecomp (ITP_0)
        ITP = iteComb(N_i,ITP_1,ITP_0)
} while max try reached

Observability don't care

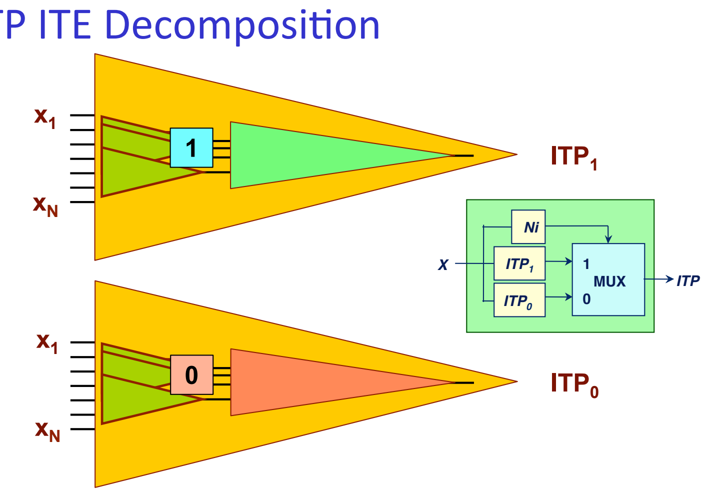
- If A == 1 -> f(.) and g(.) can be simplified



- If A == 0 -> out = 0, no matters f(.) or g(.)

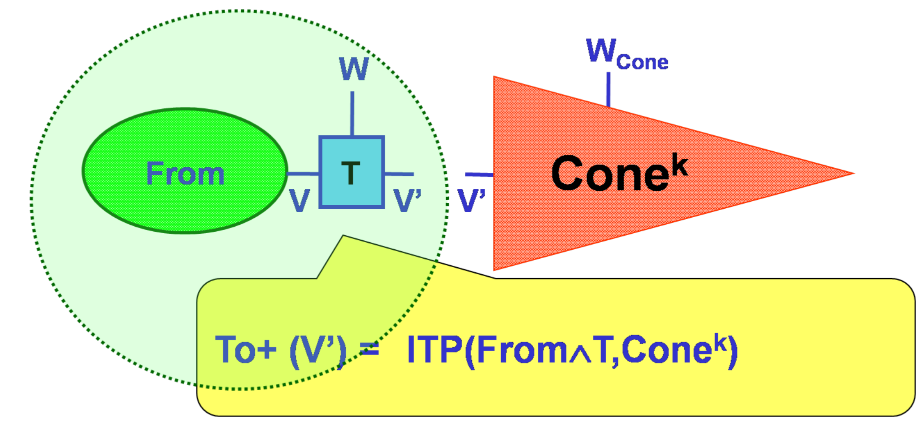


ITP ITE Decomposition



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



Abstraction by Iterative Refinements

Pseudo code:

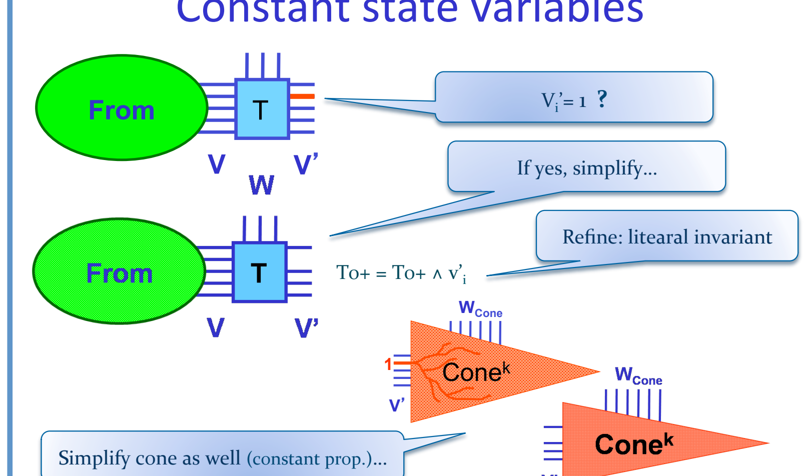
IMG+$_{Adq}$ (From, T, Cone^k)
  To+ = Full_state_space
  Foreach Class ∈ Abstraction _classes
    Select abstraction
    To+$_{Class}$ = IMG+ using abstraction
    To+ = To+ $\wedge$ To+$_{Class}$
    if UNSAT(To+ $\wedge$ Cone^k) return To+
  return (To+ $\wedge$ ITP(From $\wedge$T, Cone^k))

Loop through candidates
Find atomic abstraction
...until adequate
...or return Craig's ITP

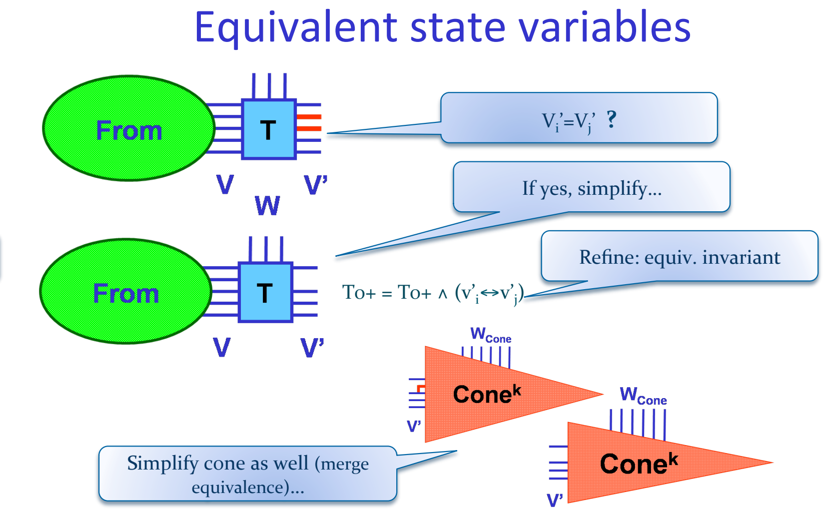
Abstraction classes

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

Constant state variables



Equivalent state variables



Experimental results