Alpine Verification Meeting 2013, FBK, Trento, Italy.

Optimization Techniques For Craig Interpolant Compaction In Unbounded Model Checking

Danilo Vendraminetto

PhD student at Formal Methods Group, Politecnico di Torino, Torino, Italy.

Before starting..

- Talk in part based on a paper we presented at DATE 2013 Conference:

Gianpiero Cabodi, C. Loiacono, D. Vendraminetto.
Optimization techniques for craig interpolant compaction in unbounded model checking.
DATE 2013: 1417-1422

Outline

- Motivations & background
  - *Hardware designs verification*
  - Craig Interpolants in MC
  - ITP size compaction & scalability
- Contributions
  - Redundancy removal and reduction of
    - UNSAT proofs
    - Craig interpolants
  - Heuristic procedure for scalable ITP compaction
- Experimental results & Conclusions

Motivations

- Can ITPs compete with IC3 ?

IC3	ITP
2-level (AND-OR) characteristic functions	Multiple level circuits
Single instance of TR	TR unrollings

- Main limitations of ITP
  - BMC-based model (vs. cube/clause-based reachability)
  - ITPs are highly redundant

Motivations

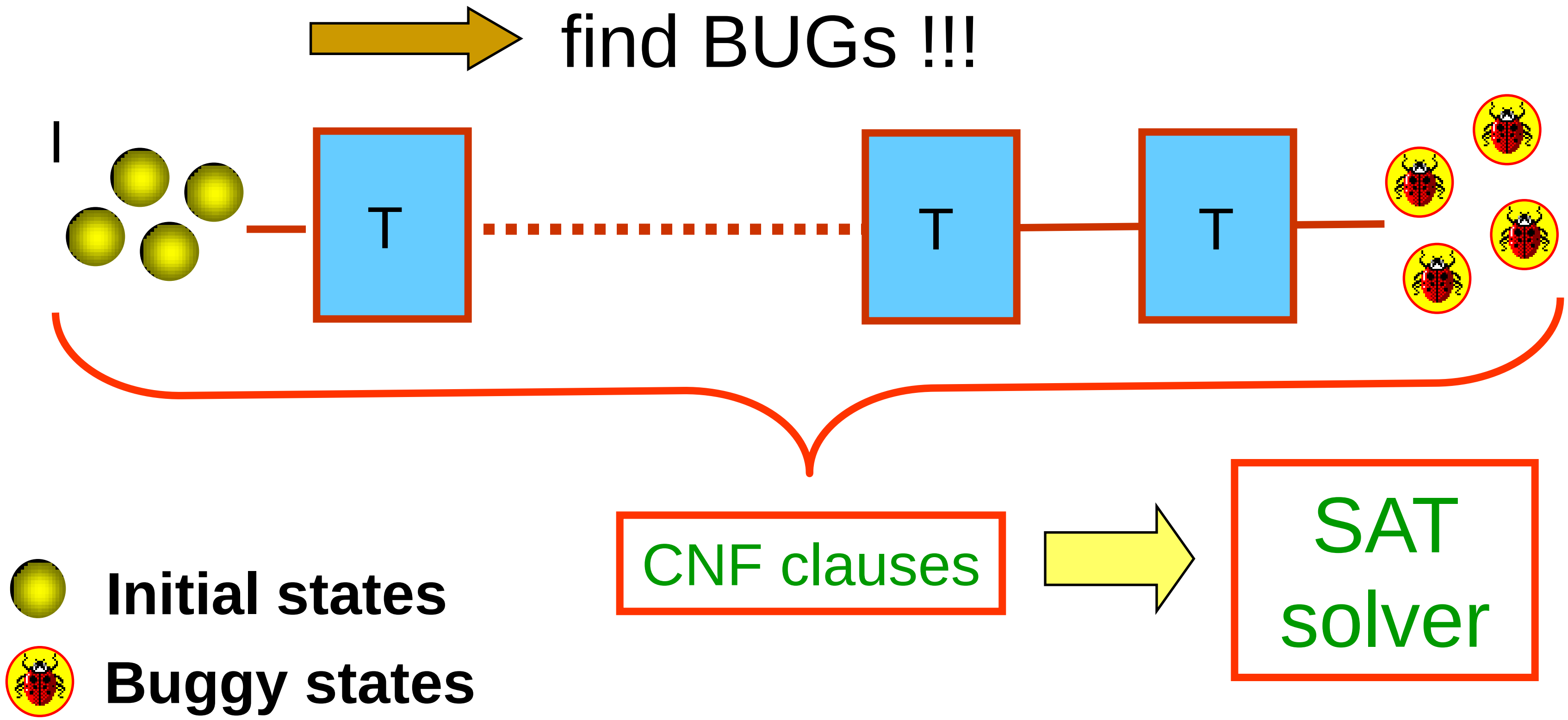
- Can ITPs compete with IC3 ?

IC3	ITP
2-level (AND-OR) characteristic functions	Multiple level circuits
Single instance of TR	TR unrollings

- Main limitations of ITP
  - ~~BMC-based model (vs. cube/clause-based reachability)~~
  - <u>ITPs are highly redundant</u>

Bounded Model Checking

- Trading off completeness for productivity



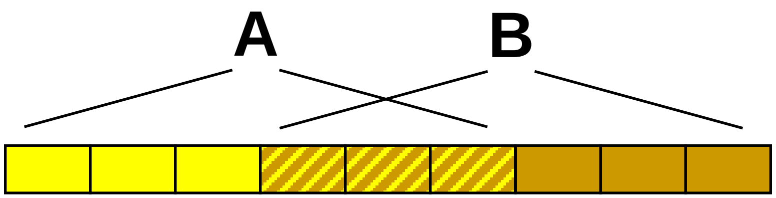
Interpolation [Craig'57]

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

Interpolation [McMillan'03]

- Interpolant as over-approx. image operator
  - Over-approximation
  - Variable quantification
- Works whenever a representation of *backward reachable* space is given
  - A: From $\wedge$T (FWD)
  - B: paths to failure states (BWD)
  - A': over-approx image
- Approx image is called *adequate* w.r.t. B

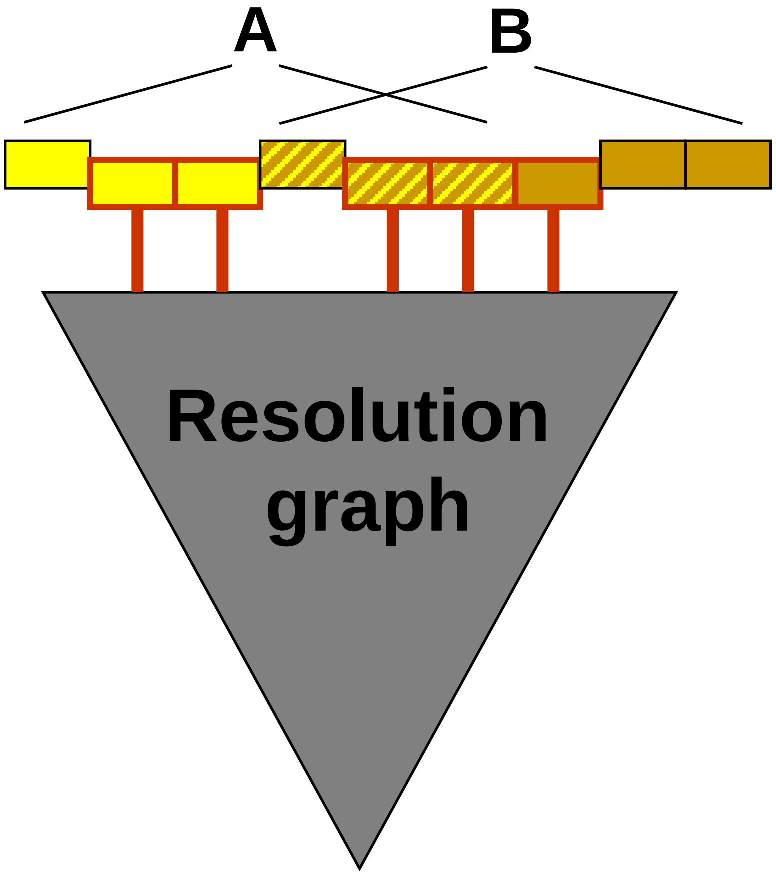
ITP from refutation proof



CNF clauses

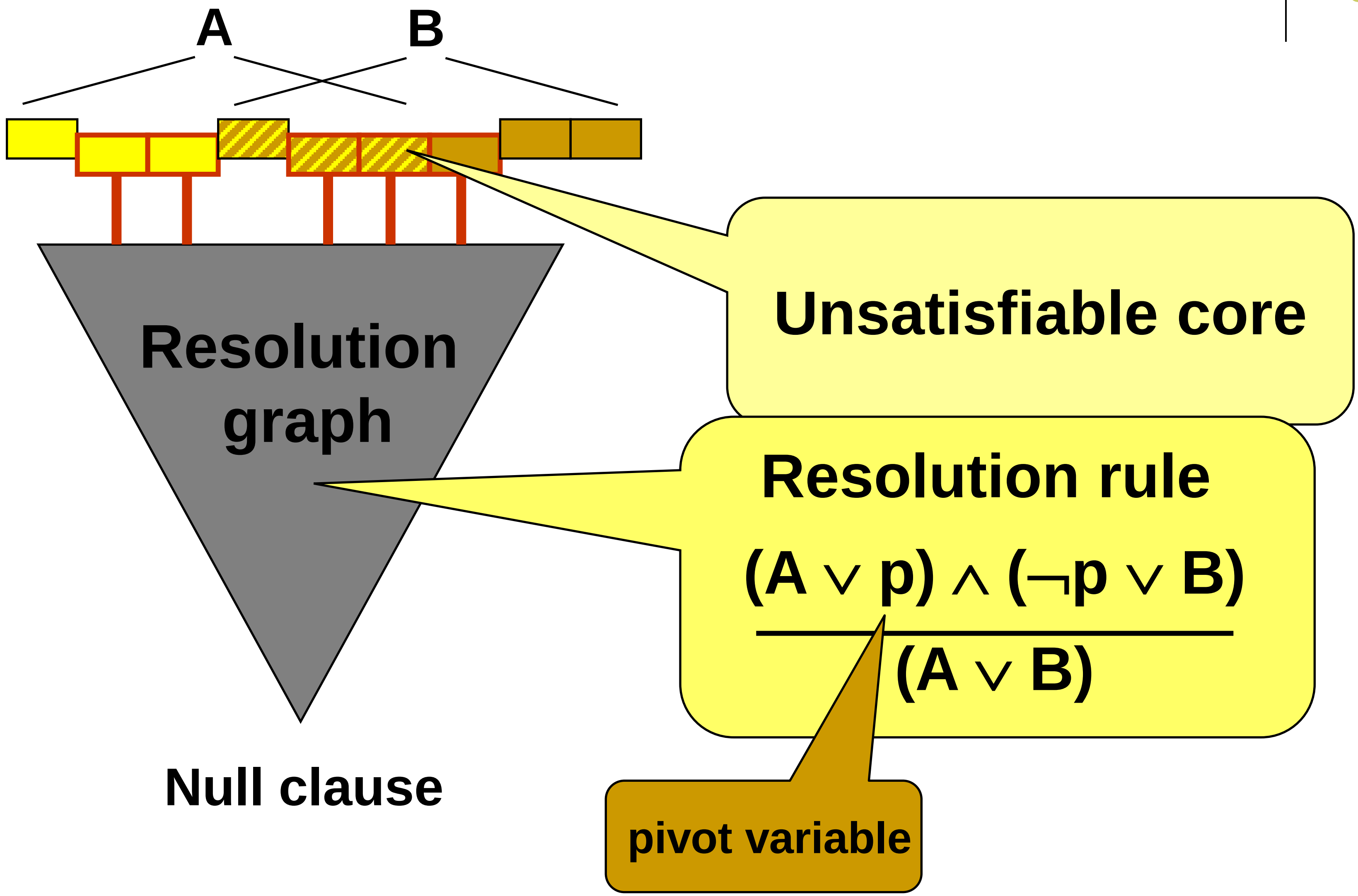
UNSAT problem ($A \wedge B = 0$)

ITP from refutation proof

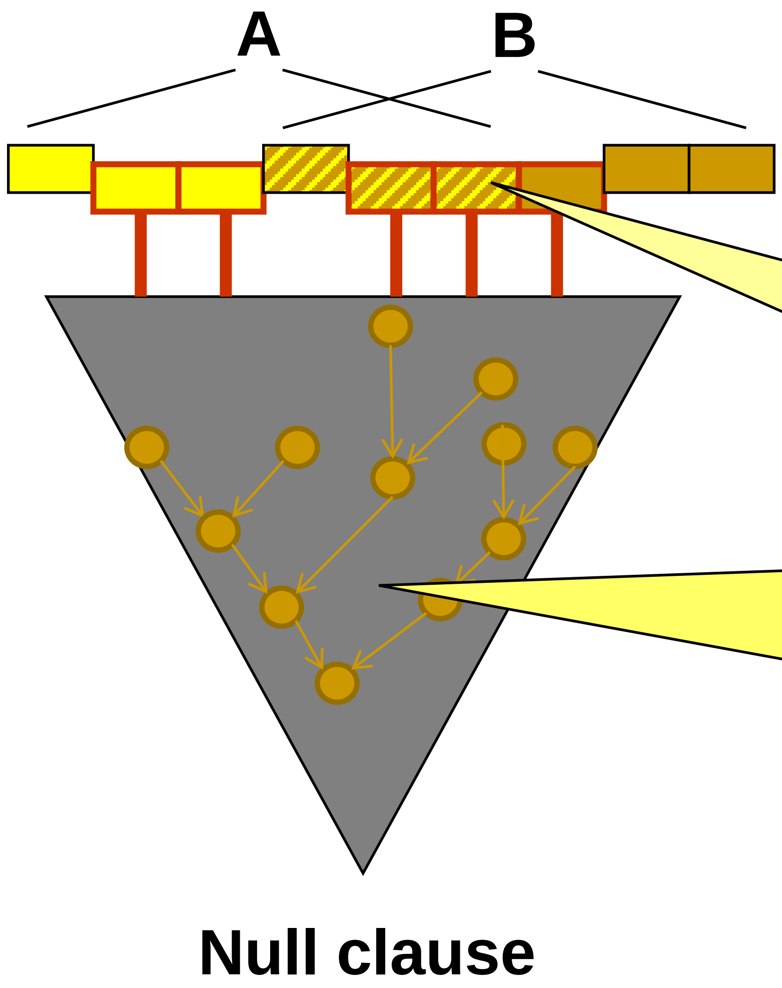


Null clause

ITP from refutation proof



ITP from refutation proof

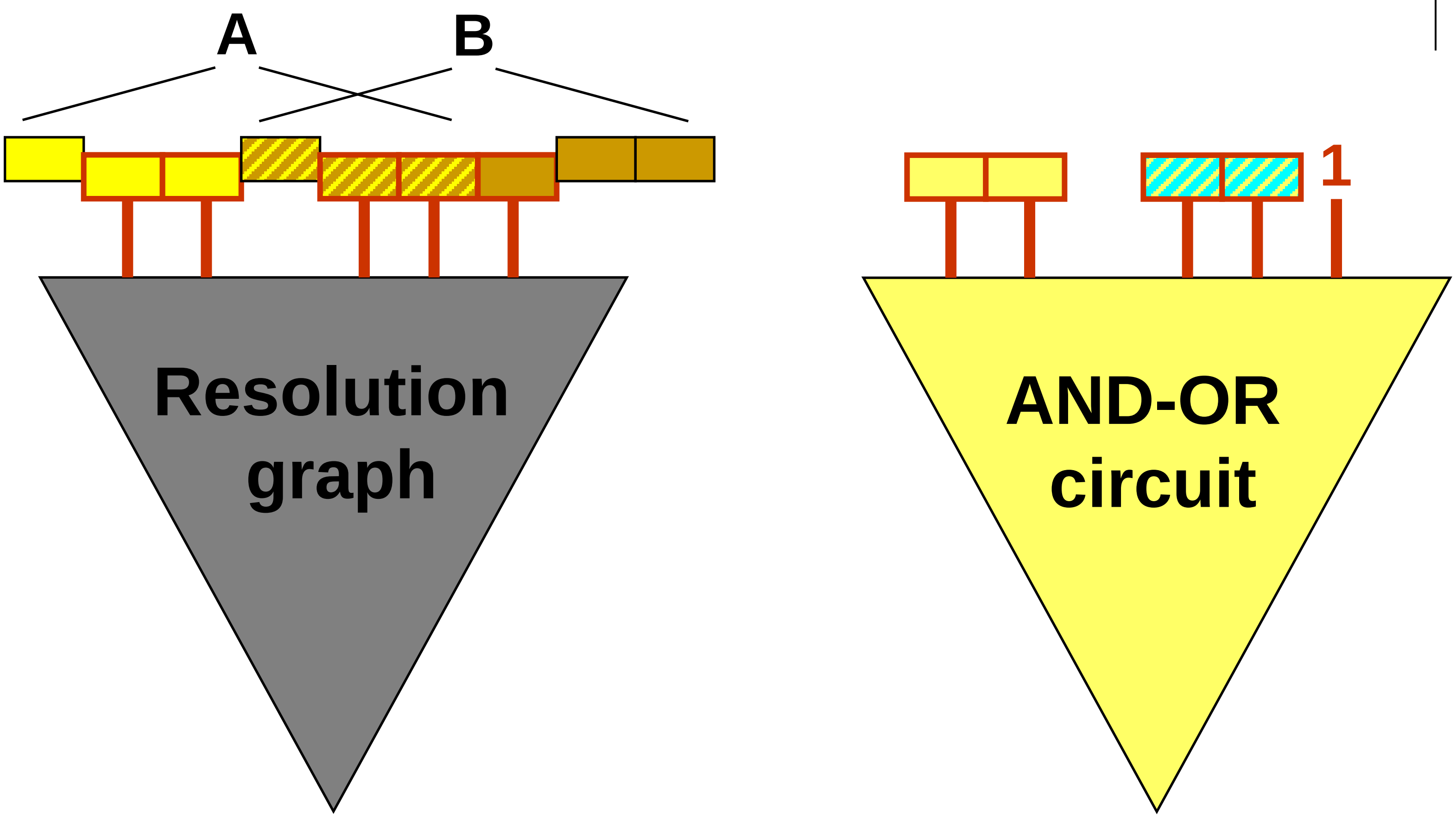


Unsatisfiable core

Resolution rule

$$\frac{(A \vee p) \wedge (\neg p \vee B)}{(A \vee B)}$$

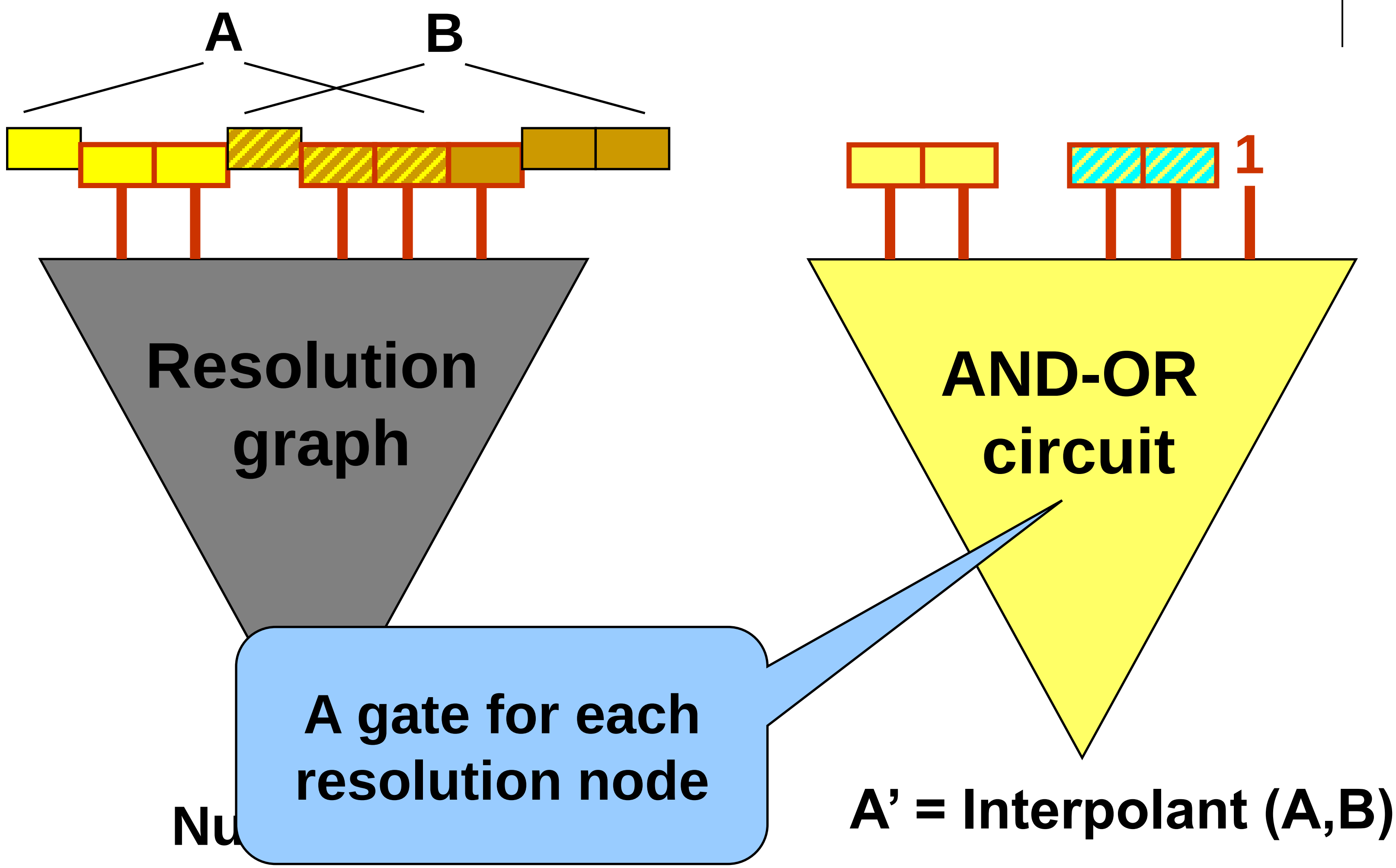
Interpolant from refutation proof



Null clause

A’ = Interpolant (A,B)

Interpolant from refutation proof

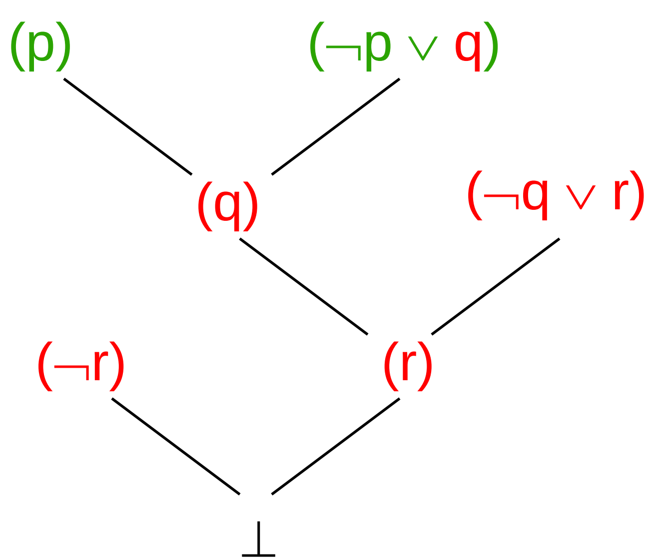


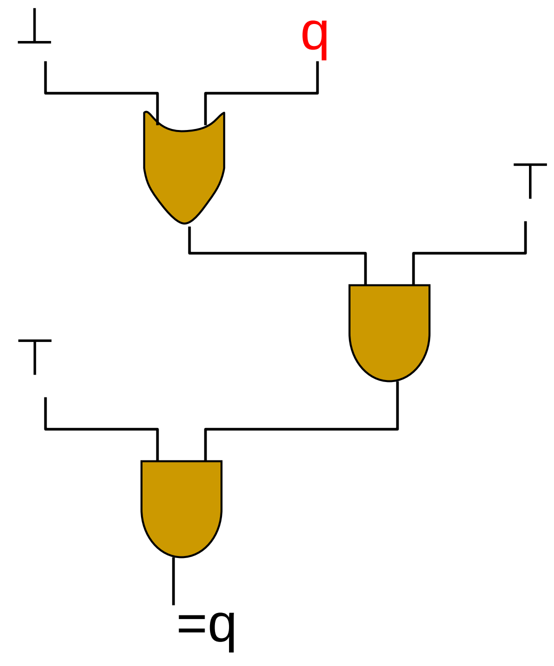
Interpolant rules

- Interpolation is a circuit that follows the *structure* of the proof

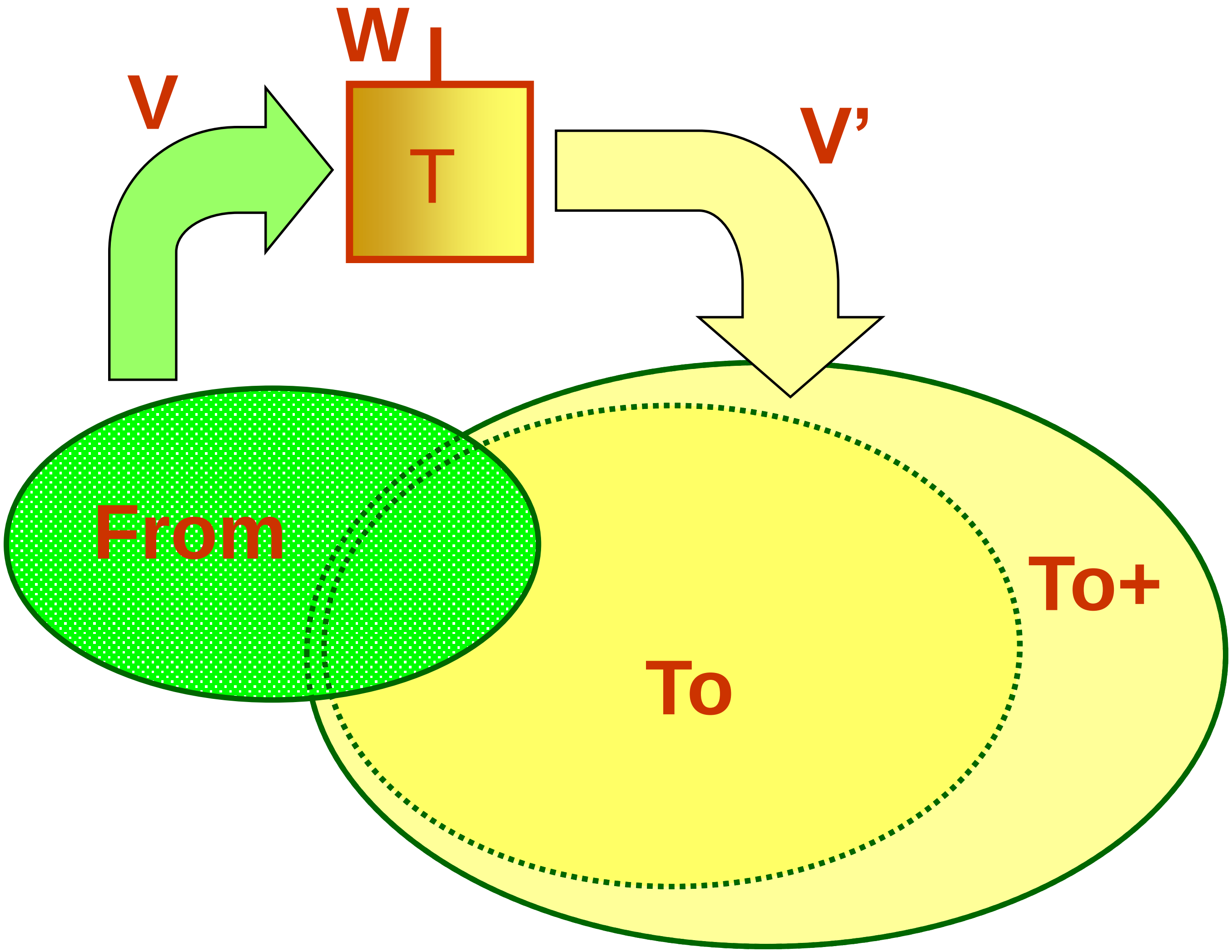
$A = (p)(\neg p \vee q)$

$B = (\neg q \vee r)(\neg r)$



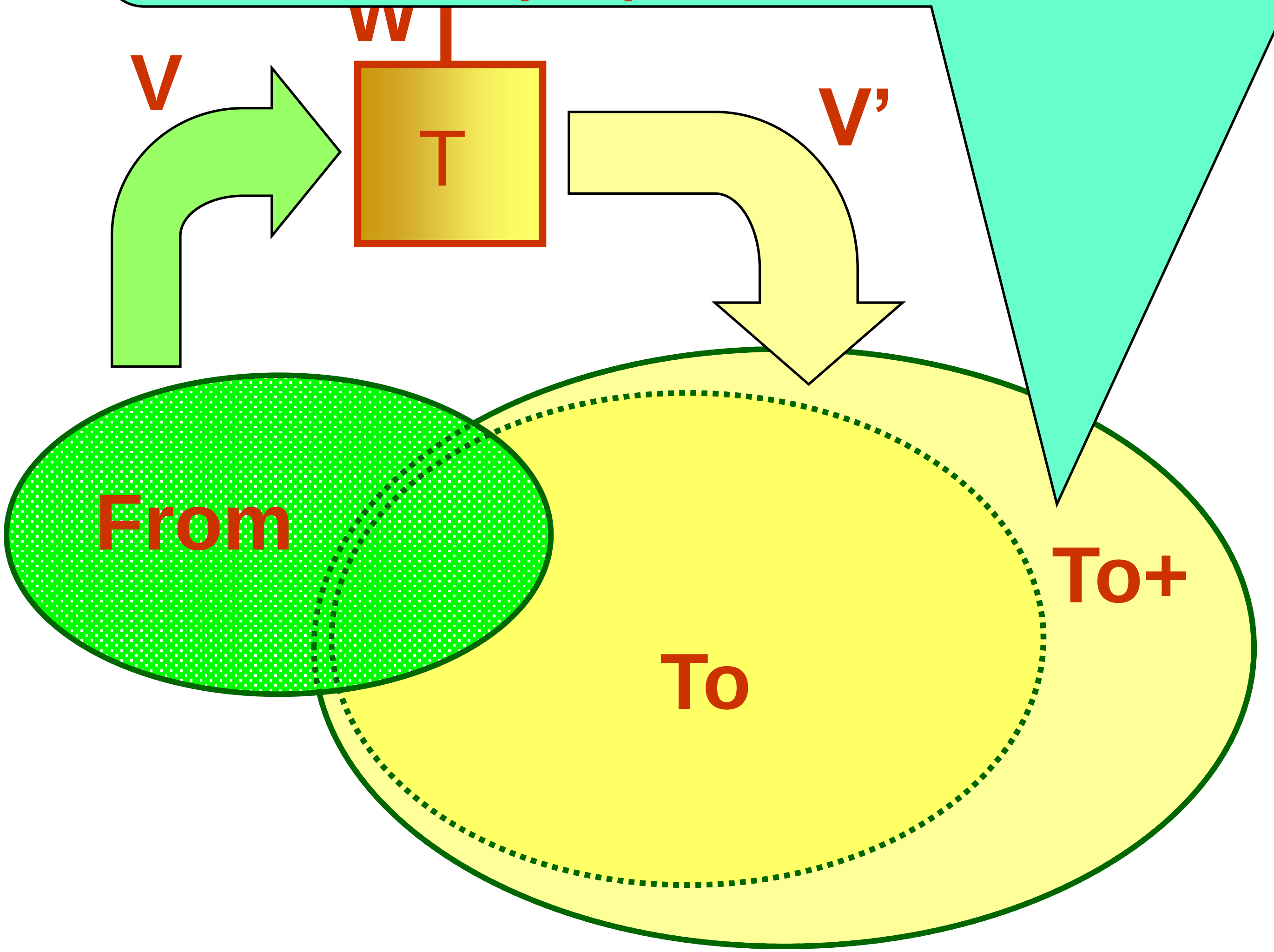


Image+

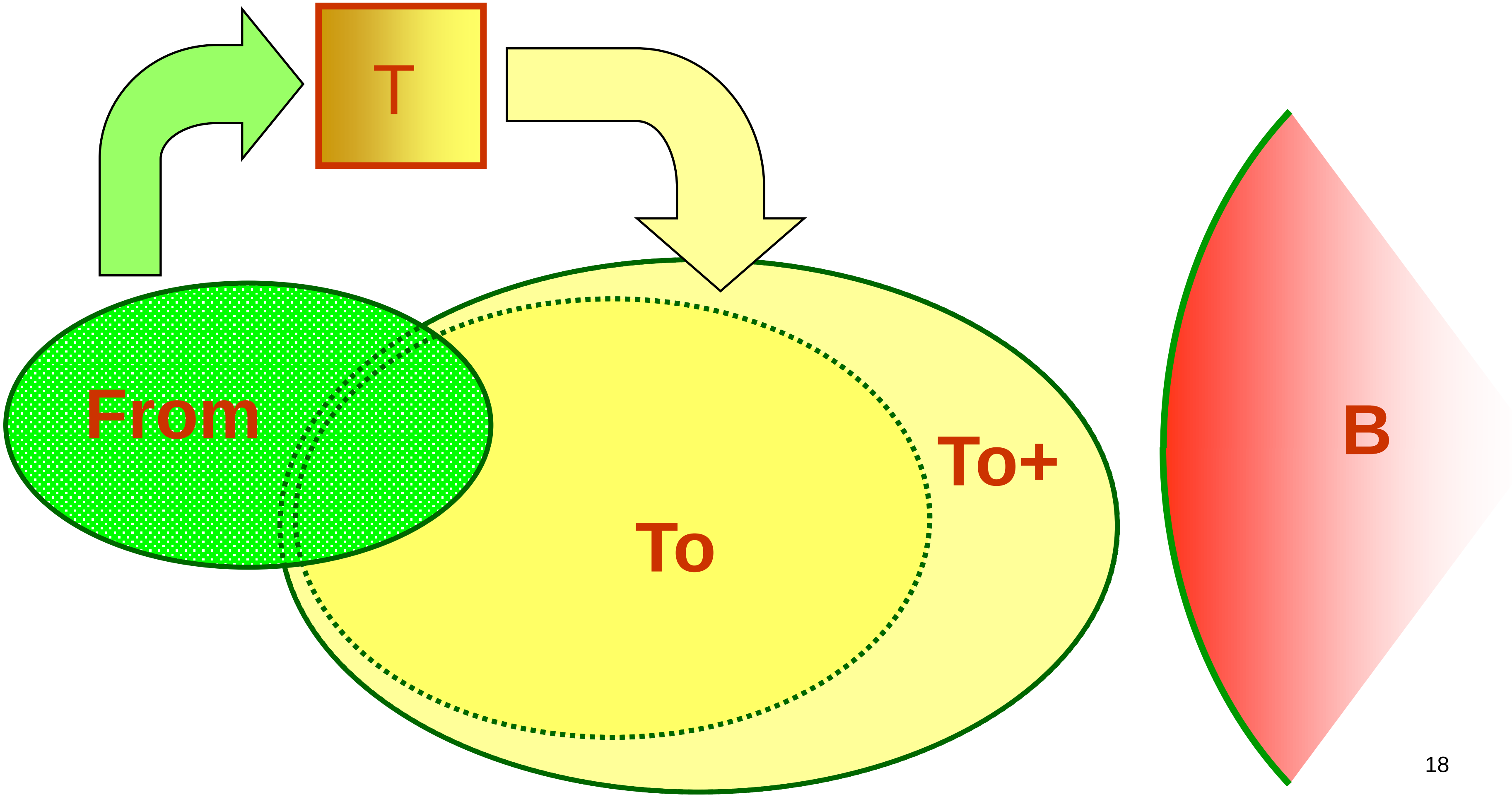


Image

$$To+(V') = IMG+(From,T) = Approx(\exists_{(V,W)} From(v) \wedge T(V,W,V'))$$



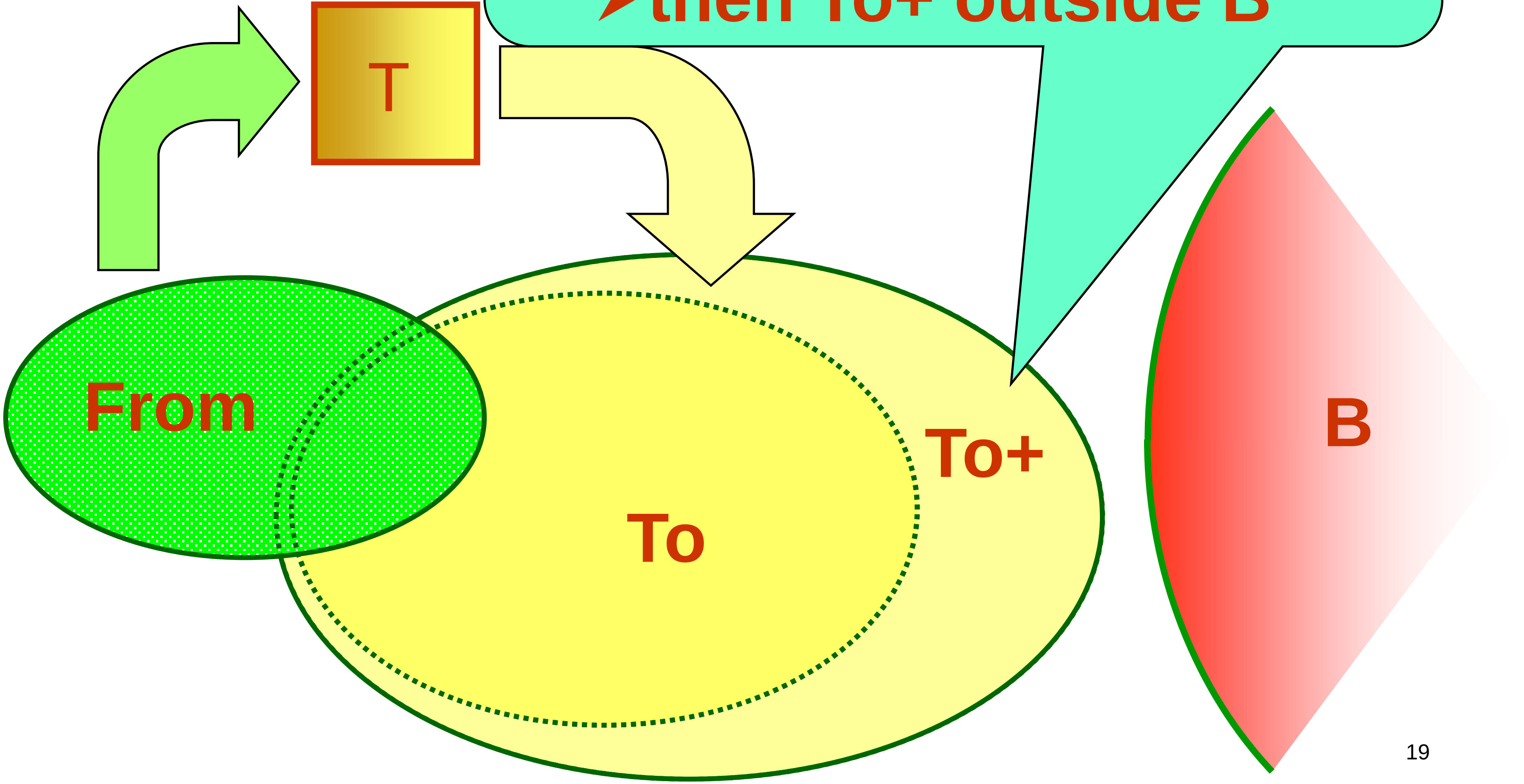
Adequate Image+



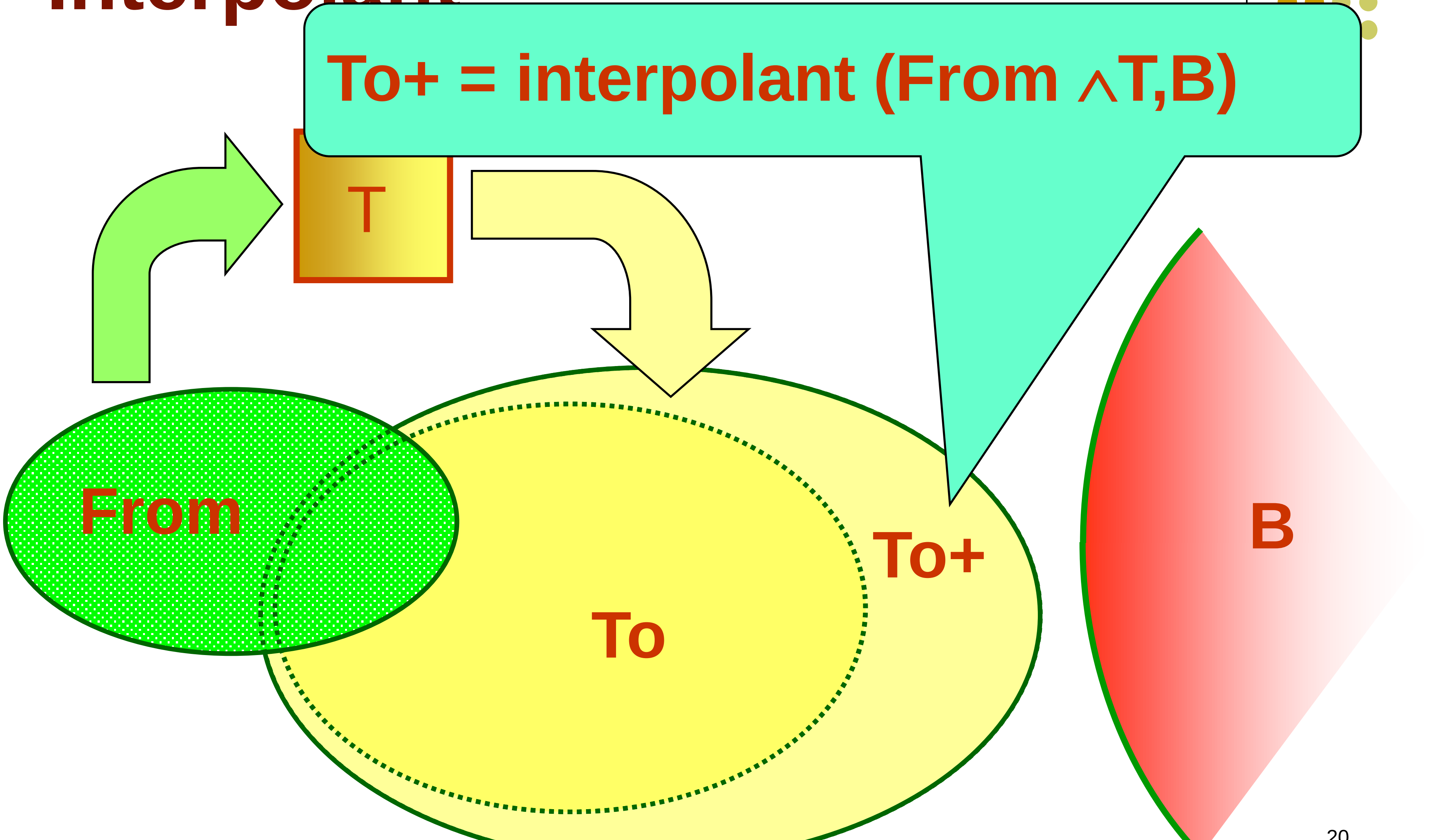
Adequate I

To+ adequate w.r.t. B

- **if To outside B**
- **then To+ outside B**

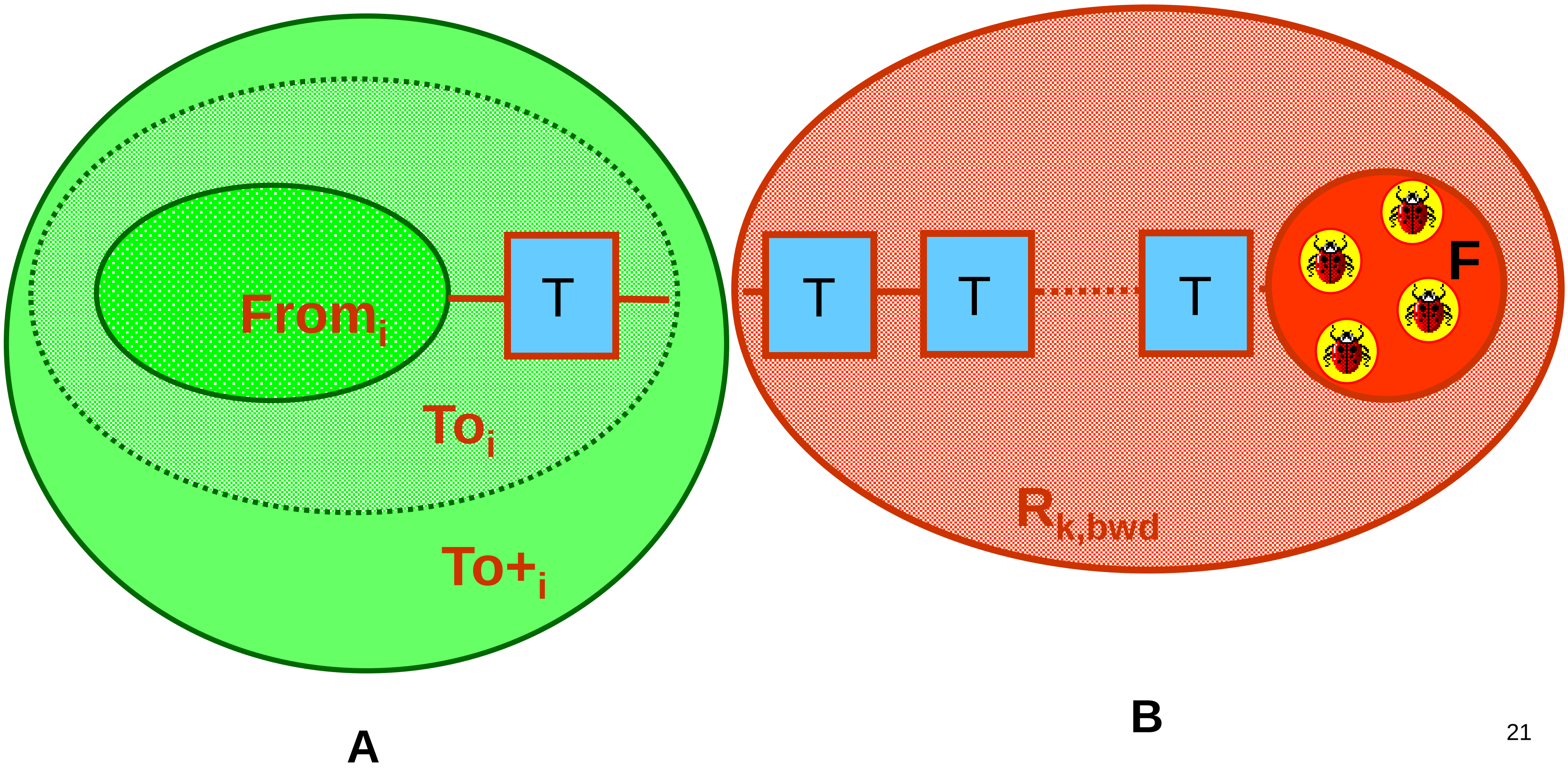


Adequate Image+ by Interpolant



ITP

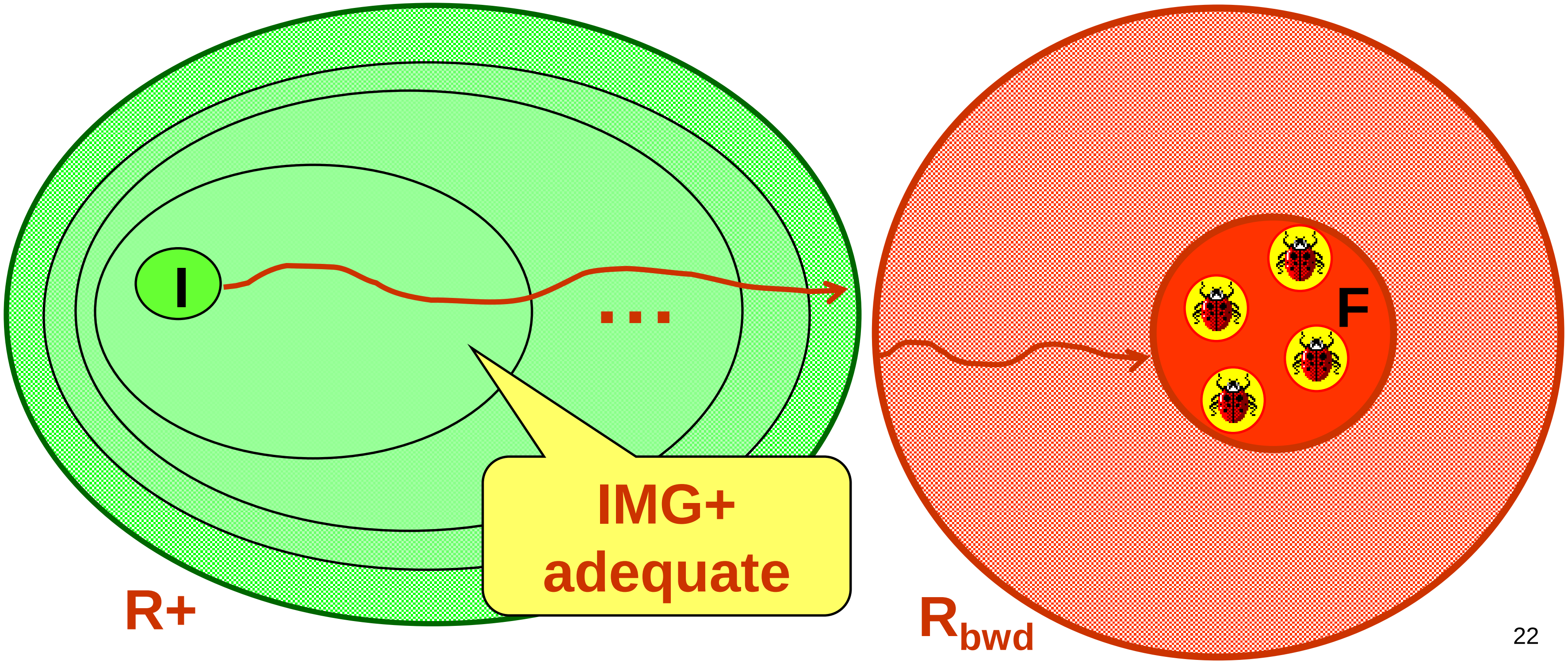
Standard ITP: $to+_i$ computed from appr. $From_i$



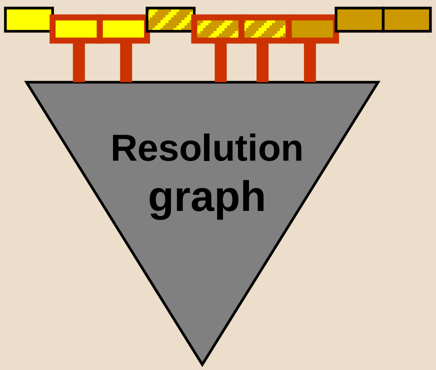
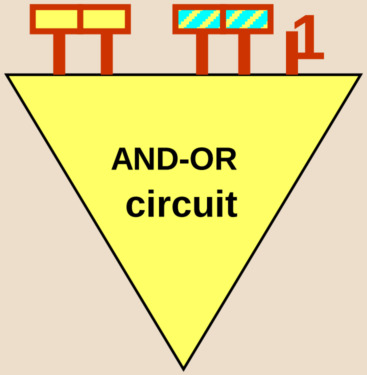
Why use adequate IMG+ ?

⇒ FWD *approximate* reachable states

- computed by adequate IMG+
- do not intersect BWD reachable states



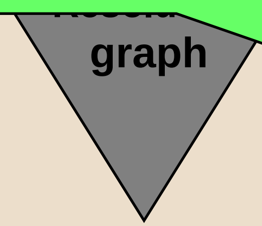
ITP compaction

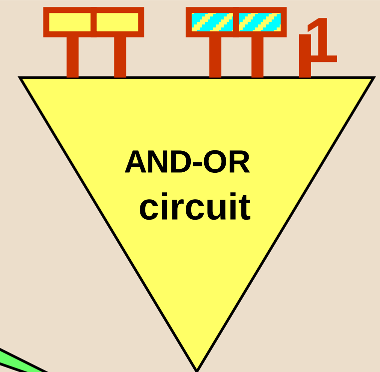
Proof reduction	ITP circuit compaction	
		
Alternative proofs • different resolution schemes	BDD/SAT sweeping	Const propagation
Equivalent proofs • redundancy removal	ODC	Refactor rewrite

ITP compaction

Problem #1:

- **SCALABILITY**

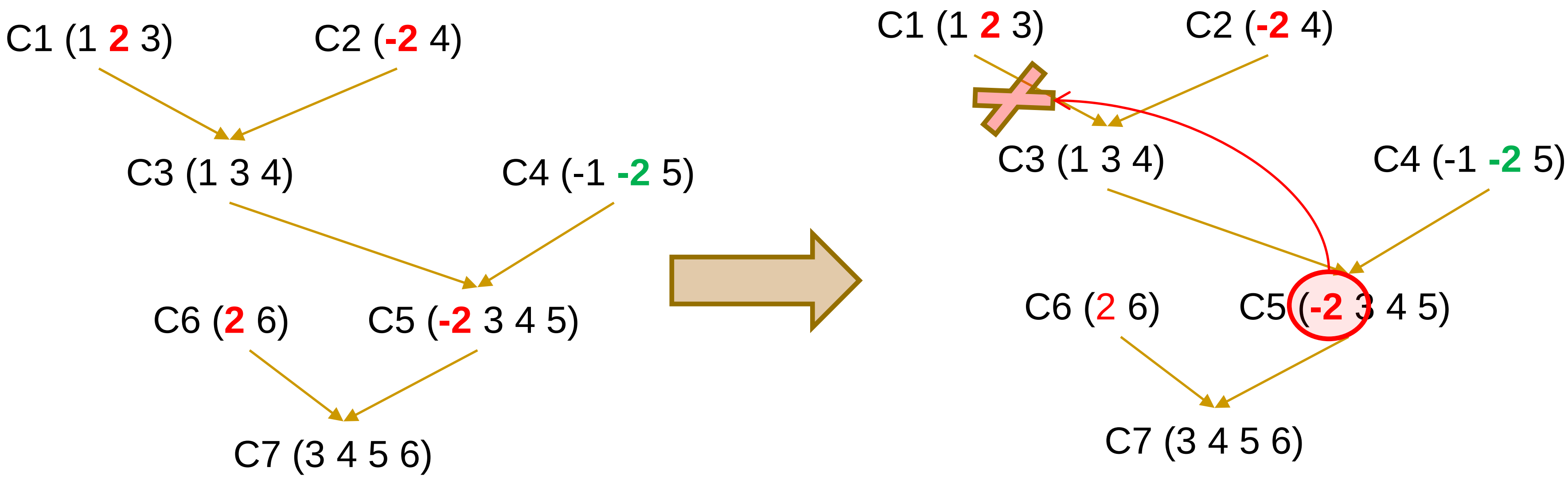




	ITP circuit compaction	
Alternative proofs • different resolution schemes	BDD/SAT sweeping	Const propagation
Equivalent proofs • redundancy removal	ODC	Refactor rewrite

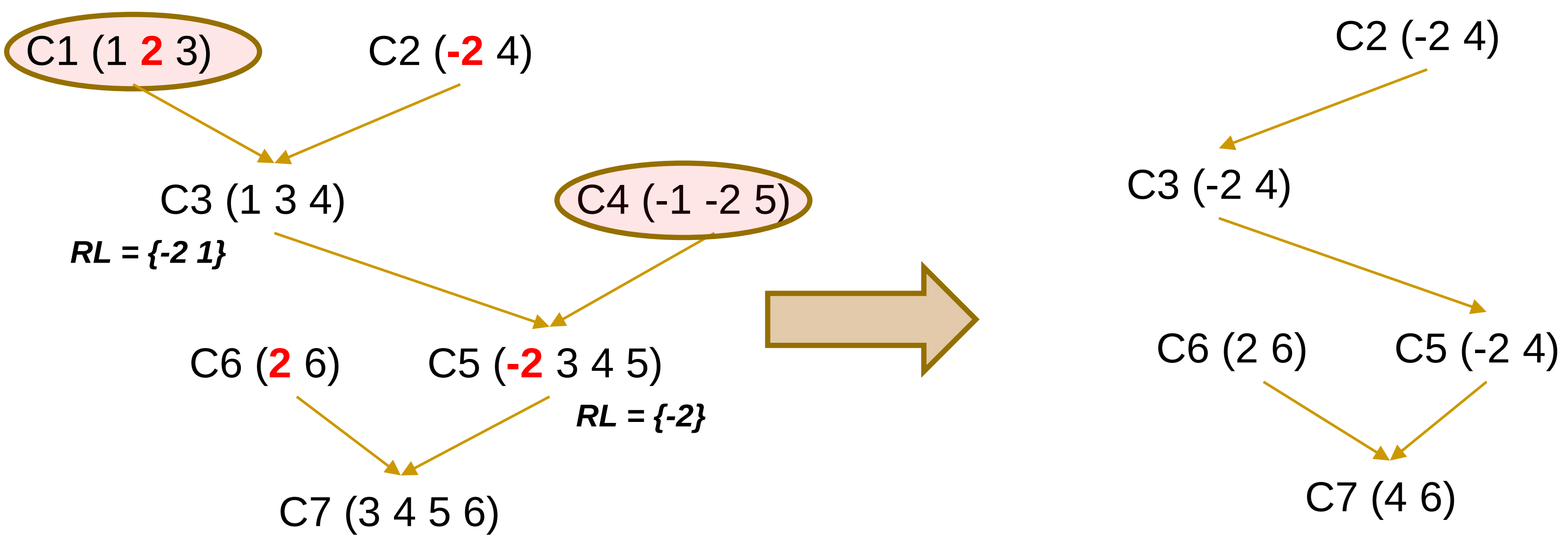
Proof reduction

- Recycle-pivots [Bar-Inal & al. HVC08]



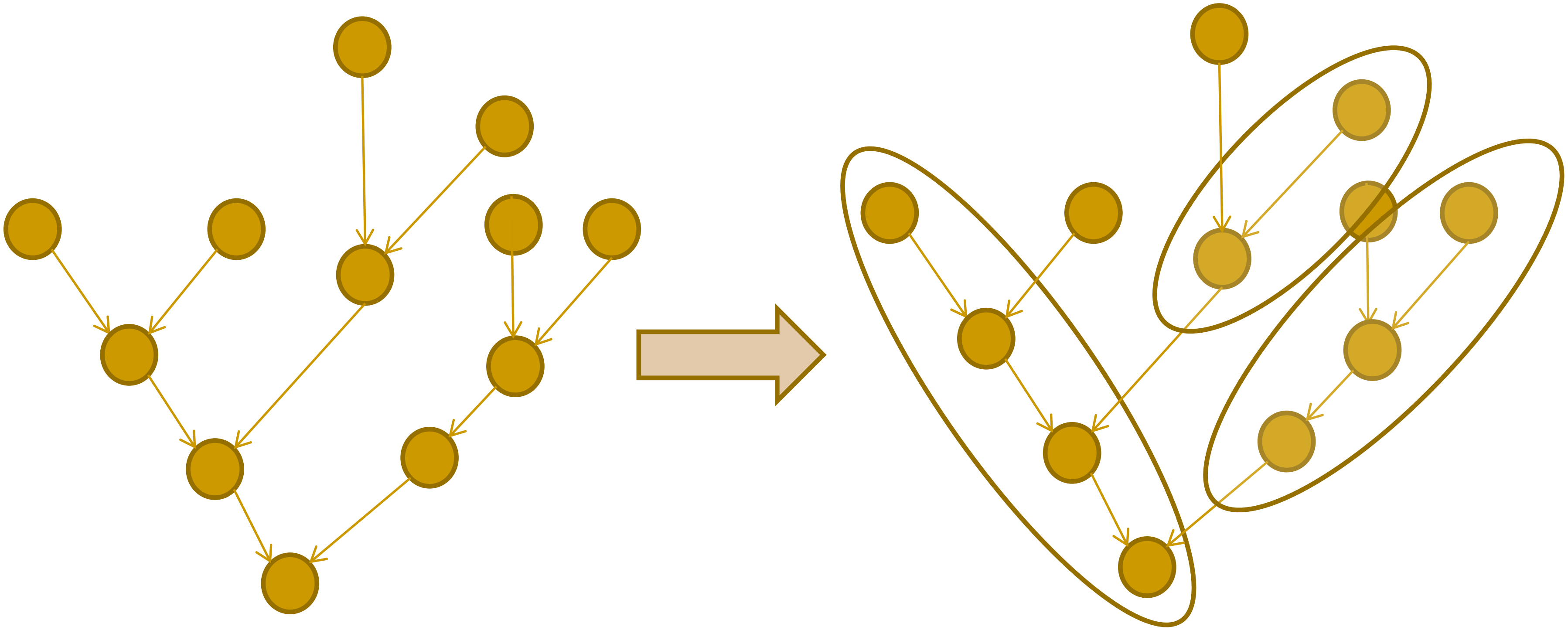
Proof reduction

- Recycle-pivots + restruct proof [Bar-Inal & al. HVC08]

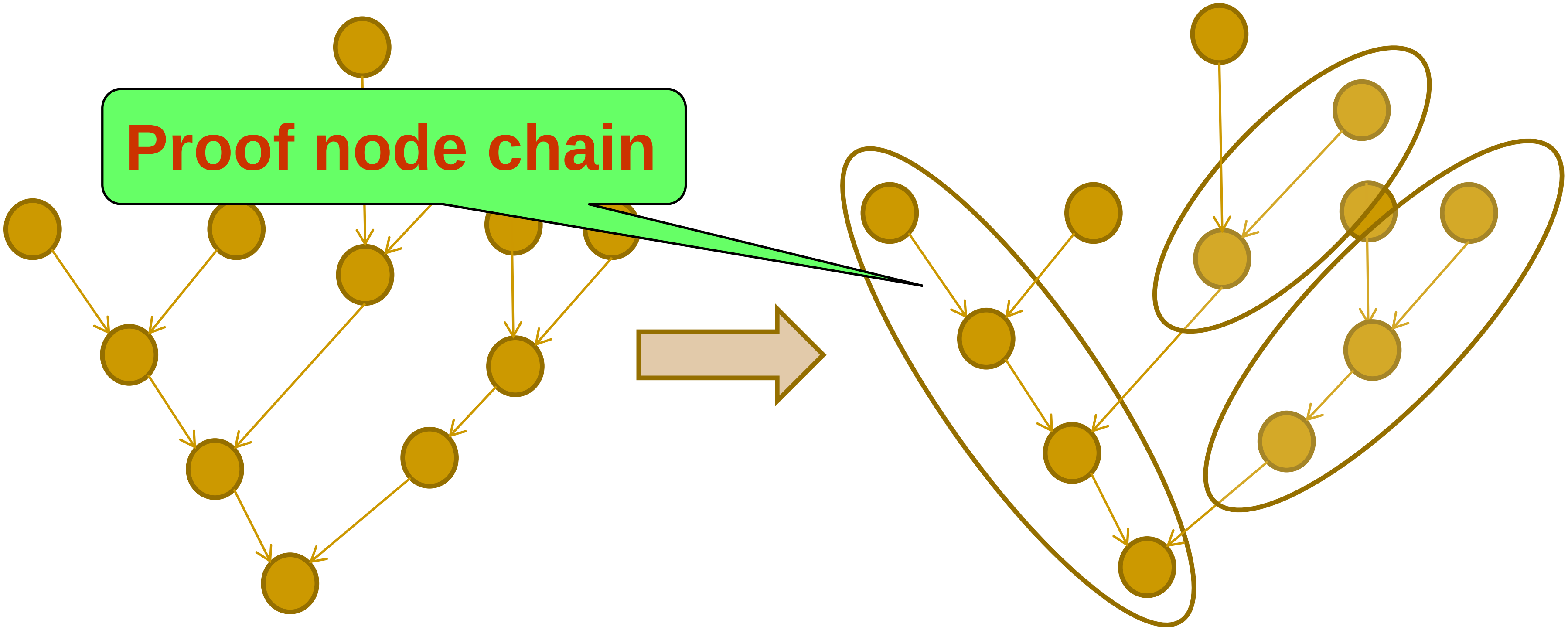


R*L denotes the Removable-Literals*

Our Contribution: exploit proof *topology*



Our Contribution: exploit proof topology



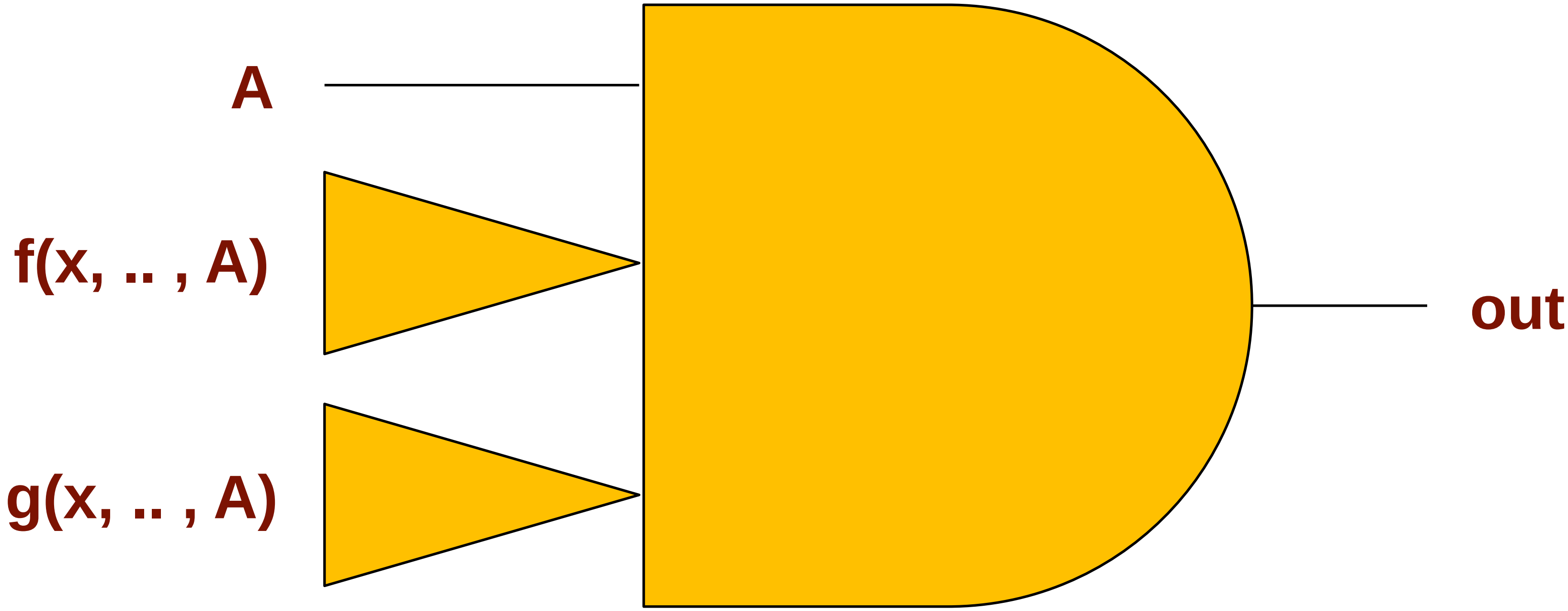
- Simpler data structure for proof reduction algorithms and further techniques

ITP Circuit Compaction

- Logic synthesis manipulations on the proof
  - Constant propagation
  - BDD-based sweeping (for equivalences)
  - Observability Don't Care (lightweight)
- Proof into AIG
  - ODC (lightweight)
  - Logic synthesis
    - rewrite / refactor, using *ABC* tool
    - AIG balance
  - ITE-based decomposition (iff necessary)

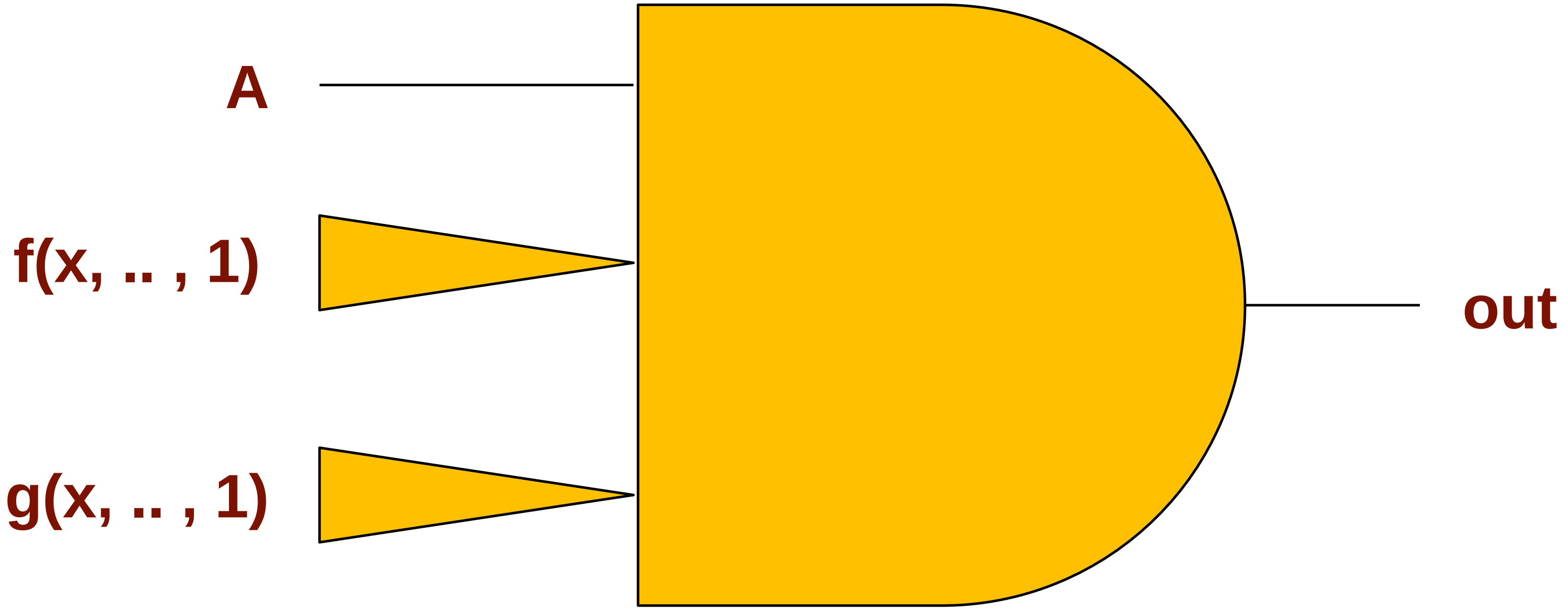
Observability don't care

- If A == 0 $\rightarrow$ out = 0 ; no matters f(.) or g(.)
  - don't-care set

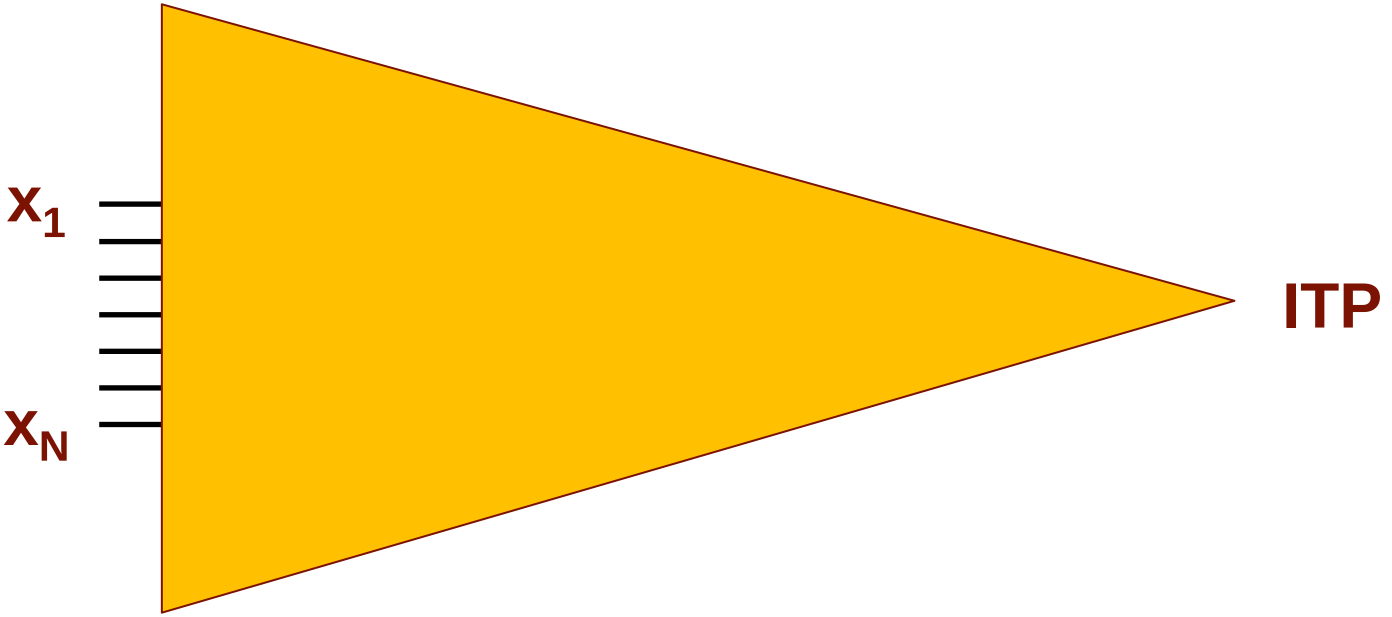


Observability don't care

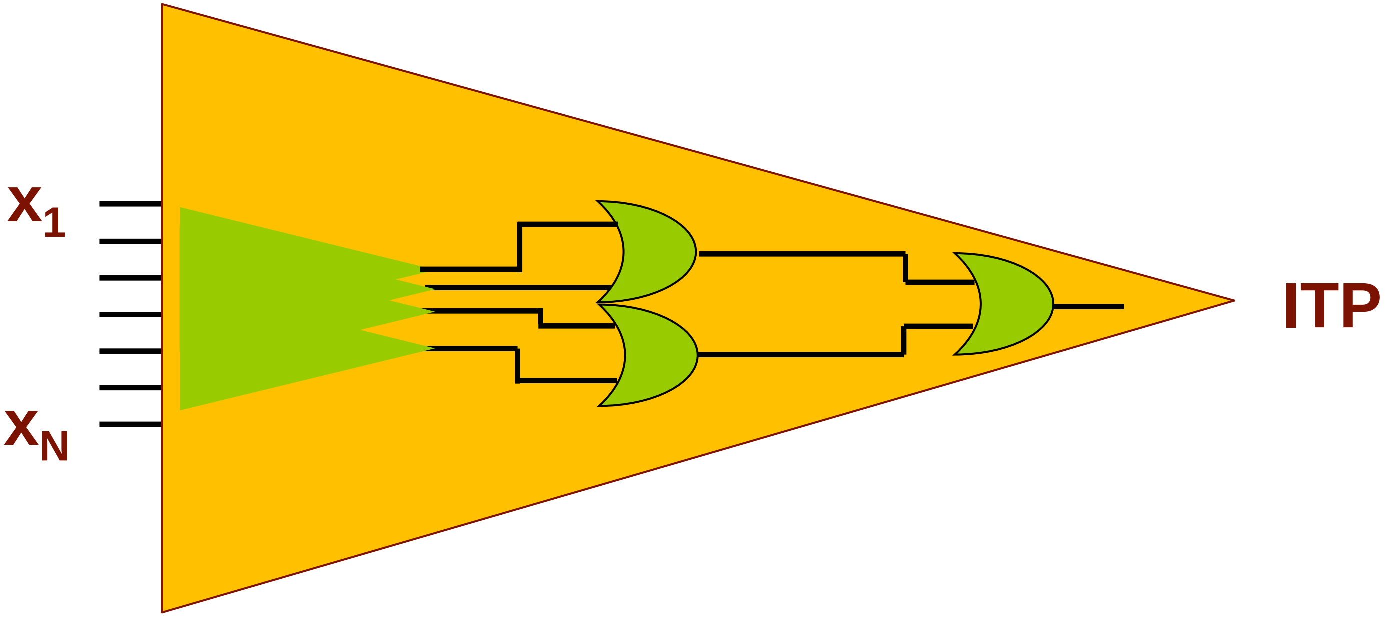
- If A == 1 $\rightarrow$ f(.) and g(.) can be simplified
  - care set



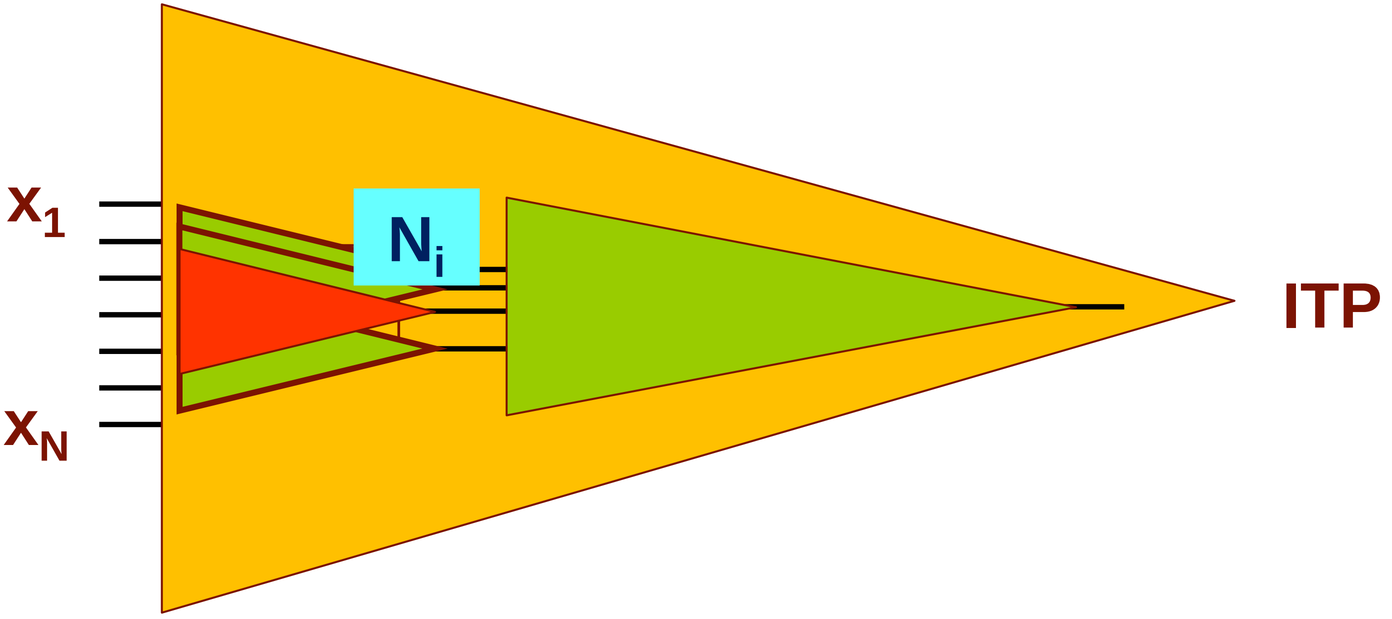
ITP ITE decomposition



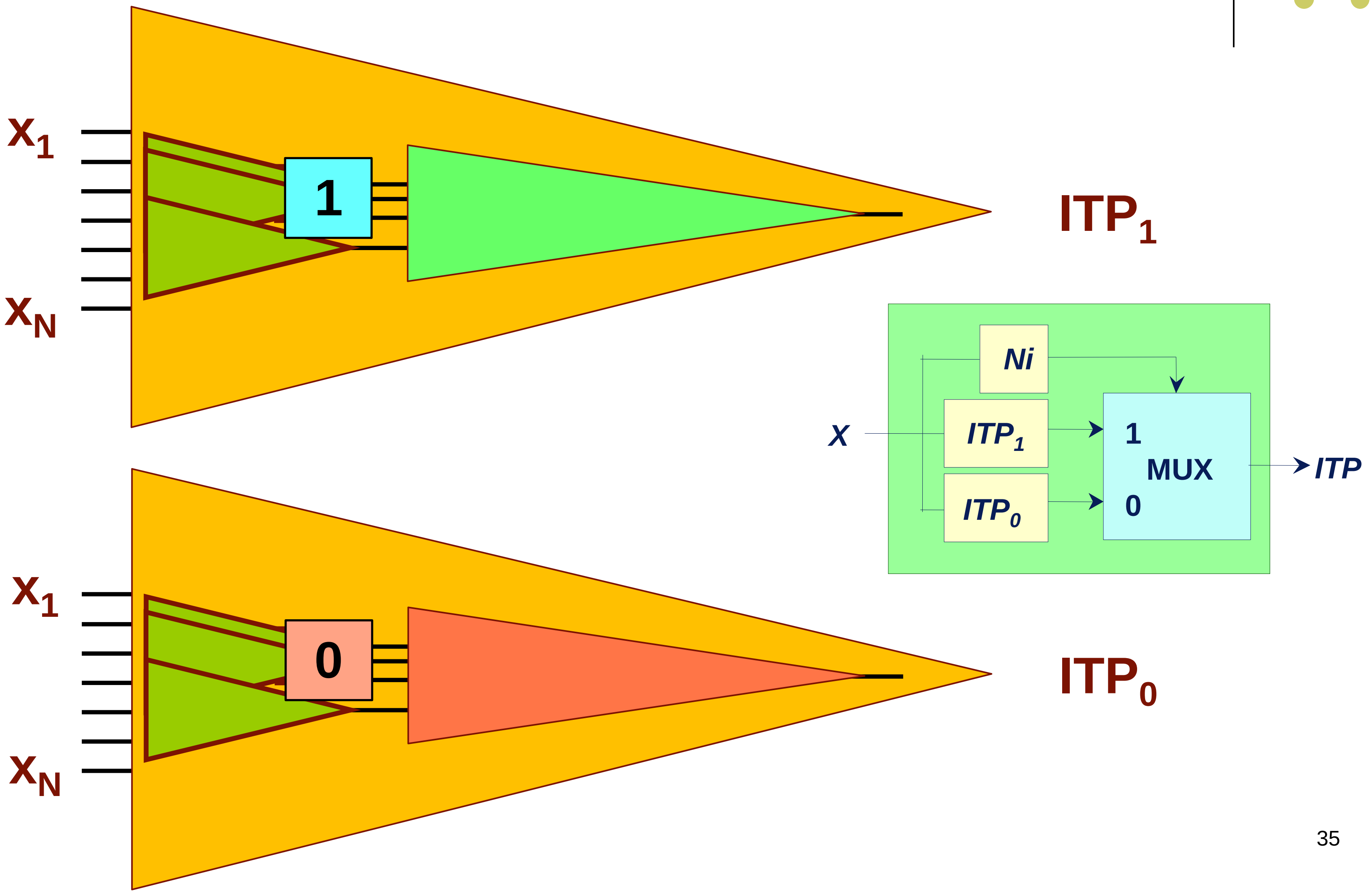
ITP ITE decomposition



ITP ITE decomposition



ITP ITE decomposition



Ad-Hoc ITP compaction

AigIteDecomp (ITP)

if (max recursions || |ITP| < th)

standardLogicSynth (ITP)

do

search node N_i with highest FO

ITE(N_i,ITP_1,ITP_0) //compute cofactors; equals to ITP

if (*accept* (**ITE** decomp)) //size-based heuristic

AigIteDecomp (N_i)

AigIteDecomp (ITP_1)

AigIteDecomp (ITP_0)

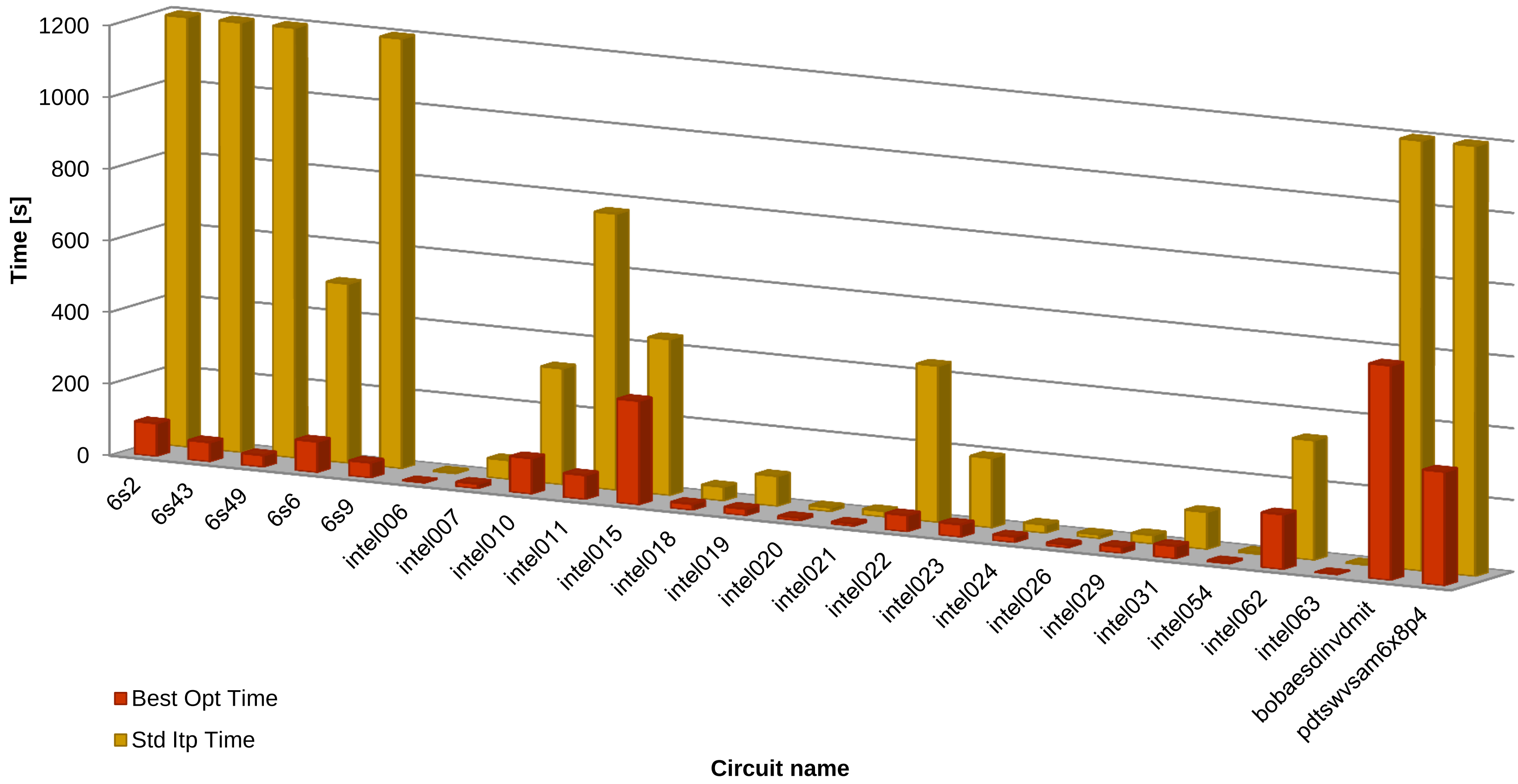
ITP = ITE(N_i,ITP_1,ITP_0)

while max try reached

Experimental results

- Framework: *PdTrav*
  - State-of-the-art academic Model Checker
  - HWMCC ’07 to ‘12
    - Ranked 1st at 2010 Model Checking Competition – UNSAT category
- ITP compaction => better MC runs
- Experience on IBM & Intel benchmarks

Experimental results



Conclusions

- ITP-based MC heavily relies on scalability, i.e. ability to compact ITPs
- We developed effective techniques to compact ITPs.
  - *Scalable techniques, applied incrementally*
- Best suited as a second engine
  - Hard-to-prove properties (hard for IC3)
  - Explosion of standard interpolation
  - Can afford extra time (for memory)

Thank you!