SINGLE CLAUSE ASSUMPTION WITHOUT ACTIVATION LITERALS TO SPEED-UP IC3

FMCAD'21

Nils Froleyks, Armin Biere

2021-10-20

You will see

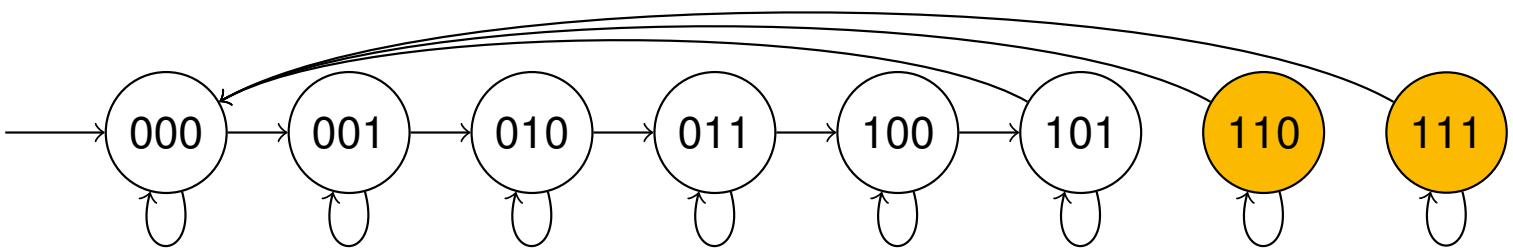
- Extension to the assumption based interface
- Allowing the addition of a single temporary clause
- Optimizing incremental SAT solvers for IC3

$$
\begin{aligned}
&F_0 \\
&F_1 = F_0 \wedge C_1 \wedge C_2 \\
&F_2 = F_1 \wedge C_3 \\
&\vdots
\end{aligned}
$$

$$
F_i \wedge \underbrace{\ell \wedge \overline{\ell'}}_{\text{Assumptions}} \wedge \underbrace{C}_{\text{Constraint}}
$$

Introduction

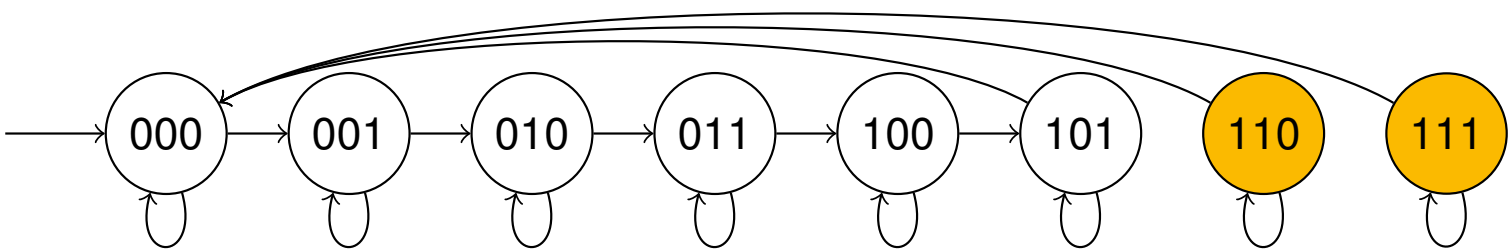
- 3-bit ($b_2 b_1 b_0$) counter
- Property: Counter does not exceed 5



Introduction

- 3-bit ($b_2 b_1 b_0$) counter
- Property: Counter does not exceed 5
- $T = \quad (\overline{b_2} \wedge \overline{b_1} \wedge \overline{b_0}) \Rightarrow ((\overline{b_2'} \wedge \overline{b_1'} \wedge \overline{b_0'}) \vee (\overline{b_2'} \wedge \overline{b_1'} \wedge \overline{b_0'}))$
  $\wedge (\overline{b_2} \wedge \overline{b_1} \wedge b_0) \Rightarrow ((\overline{b_2'} \wedge \overline{b_1'} \wedge b_0') \vee (\overline{b_2'} \wedge b_1' \wedge \overline{b_0'}))$
  $\wedge \ldots$
- Can state 6 be reached?

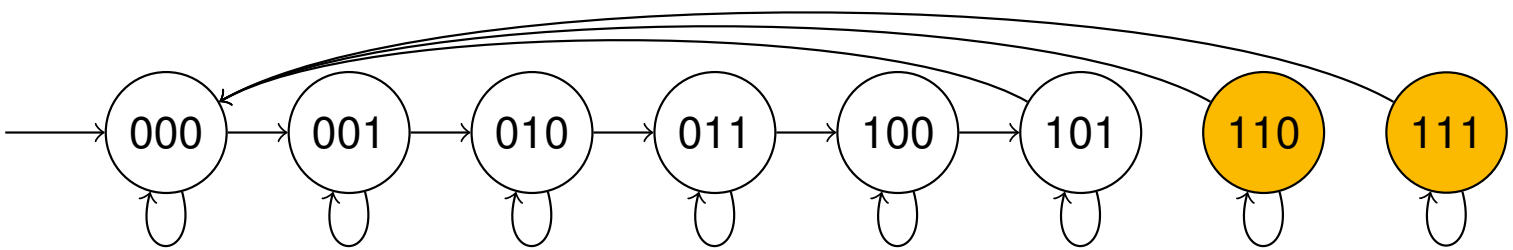
$$T \wedge b_2' \wedge b_1' \wedge \overline{b_0'} \quad \text{SAT}$$



Introduction

- 3-bit ($b_2 b_1 b_0$) counter
- Property: Counter does not exceed 5
- $T = \quad (\overline{b_2} \wedge \overline{b_1} \wedge \overline{b_0}) \Rightarrow ((\overline{b'_2} \wedge \overline{b'_1} \wedge \overline{b'_0}) \vee (\overline{b'_2} \wedge \overline{b'_1} \wedge \overline{b'_0}))$
  $\wedge (\overline{b_2} \wedge \overline{b_1} \wedge b_0) \Rightarrow ((\overline{b'_2} \wedge \overline{b'_1} \wedge b'_0) \vee (\overline{b'_2} \wedge b'_1 \wedge \overline{b'_0}))$
  $\wedge \ldots$
- Can state 6 be reached?

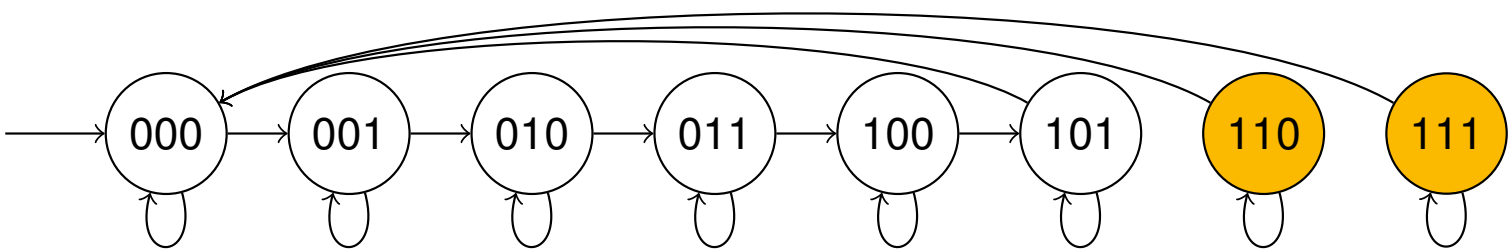
$(\overline{b_2} \vee \overline{b_1} \vee b_0) \wedge T \wedge b'_2 \wedge b'_1 \wedge \overline{b'_0}$ UNSAT $\Rightarrow T \wedge (\overline{b'_2} \vee \overline{b'_1} \vee b'_0)$



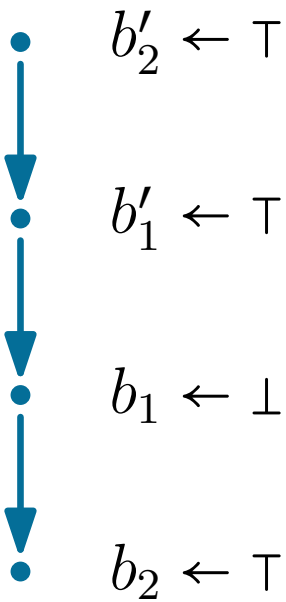
Introduction

- 3-bit ($b_2 b_1 b_0$) counter
- Property: Counter does not exceed 5
- $T = \quad (\overline{b_2} \wedge \overline{b_1} \wedge \overline{b_0}) \Rightarrow ((\overline{b_2'} \wedge \overline{b_1'} \wedge \overline{b_0'}) \vee (\overline{b_2'} \wedge \overline{b_1'} \wedge \overline{b_0'}))$
  $\wedge\ (\overline{b_2} \wedge \overline{b_1} \wedge b_0) \Rightarrow ((\overline{b_2'} \wedge \overline{b_1'} \wedge b_0') \vee (\overline{b_2'} \wedge b_1' \wedge \overline{b_0'}))$
  $\wedge \ldots$
- Can state 7 be reached?

$$(\overline{b_2} \vee \overline{b_1} \vee \overline{b_0}) \wedge T \wedge b_2' \wedge b_1' \wedge b_0' \quad ?$$



Conflict Driven Clause Learning



$b_2' \leftarrow \top$

$b_1' \leftarrow \top$

$b_1 \leftarrow \bot$

$b_2 \leftarrow \top$

C_1 $\overline{b_2} \vee b_1 \vee b_0 \vee \overline{b_1'}$

C_2 $b_2 \vee b_1 \vee \overline{b_0} \vee \overline{b_2'}$

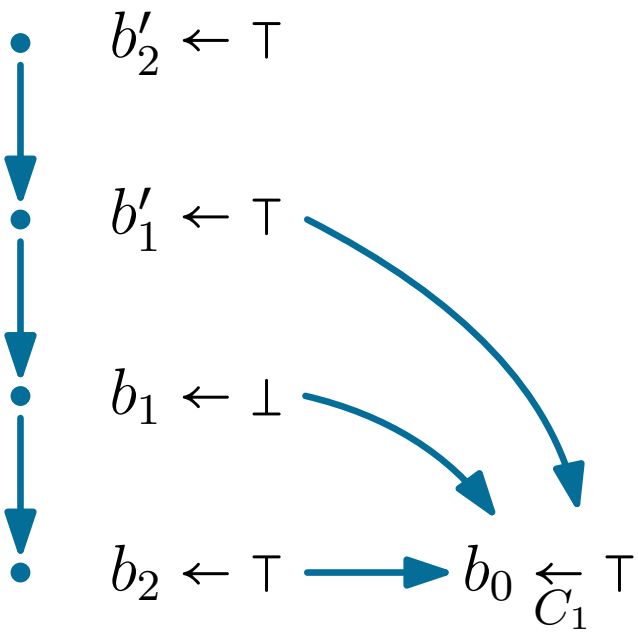
C_3 $\overline{b_2} \vee b_1 \vee \overline{b_0} \vee \overline{b_2'} \vee \overline{b_1'}$

C_4 $b_2 \vee b_1 \vee b_0 \vee \overline{b_2'} \vee \overline{b_0'}$

C_5 $b_2 \vee b_1 \vee b_0 \vee \overline{b_1'} \vee b_0'$

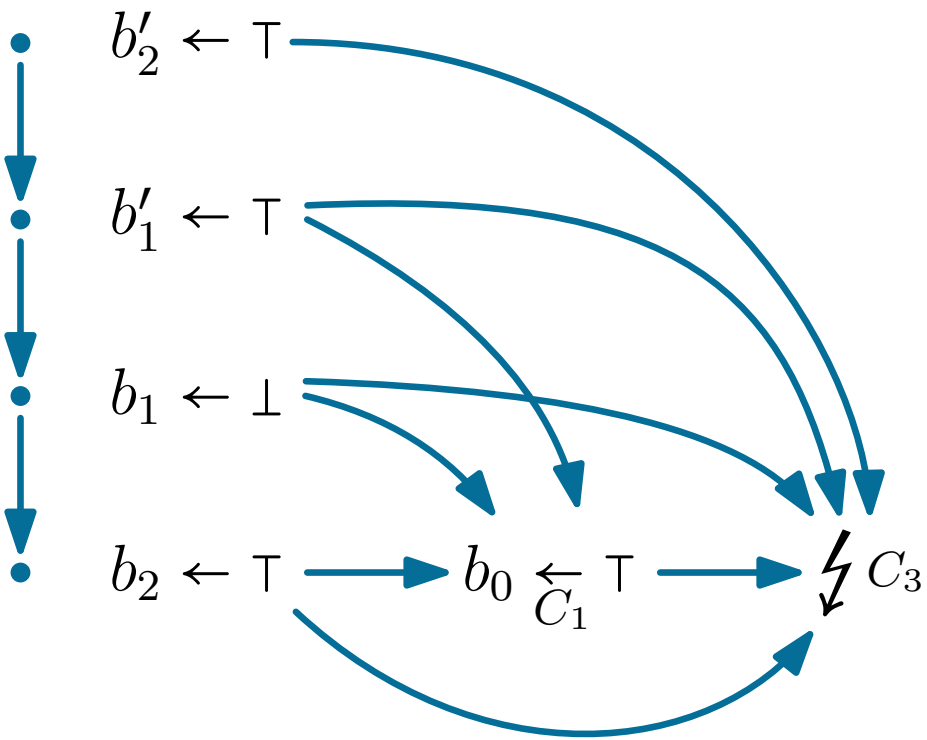
$\ldots$

Conflict Driven Clause Learning



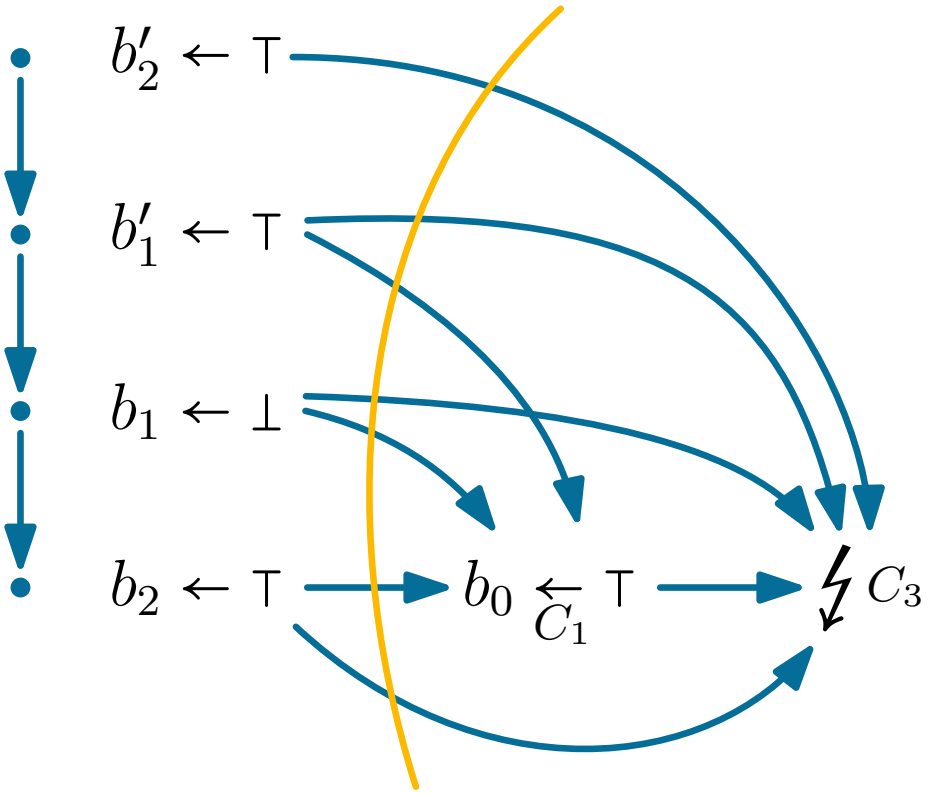
C_1 $\overline{b_2} \vee b_1 \vee b_0 \vee \overline{b'_1}$
C_2 $b_2 \vee b_1 \vee \overline{b_0} \vee \overline{b'_2}$
C_3 $\overline{b_2} \vee b_1 \vee \overline{b_0} \vee \overline{b'_2} \vee \overline{b'_1}$
C_4 $b_2 \vee b_1 \vee b_0 \vee \overline{b'_2} \vee \overline{b'_0}$
C_5 $b_2 \vee b_1 \vee b_0 \vee \overline{b'_1} \vee b'_0$
$\ldots$

Conflict Driven Clause Learning



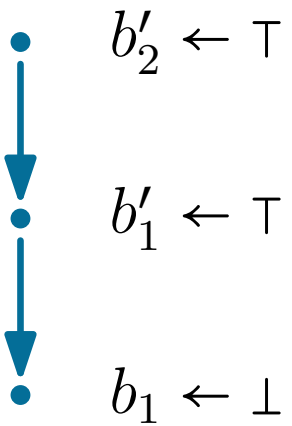
C_1 $\overline{b_2} \vee b_1 \vee b_0 \vee \overline{b_1'}$
C_2 $b_2 \vee b_1 \vee \overline{b_0} \vee \overline{b_2'}$
C_3 $\overline{b_2} \vee b_1 \vee \overline{b_0} \vee \overline{b_2'} \vee \overline{b_1'}$
C_4 $b_2 \vee b_1 \vee b_0 \vee \overline{b_2'} \vee \overline{b_0'}$
C_5 $b_2 \vee b_1 \vee b_0 \vee \overline{b_1'} \vee b_0'$
$\ldots$

Conflict Driven Clause Learning



C_1 $\overline{b_2} \vee b_1 \vee b_0 \vee \overline{b_1'}$
C_2 $b_2 \vee b_1 \vee \overline{b_0} \vee \overline{b_2'}$
C_3 $\overline{b_2} \vee b_1 \vee \overline{b_0} \vee \overline{b_2'} \vee \overline{b_1'}$
C_4 $b_2 \vee b_1 \vee b_0 \vee \overline{b_2'} \vee \overline{b_0'}$
C_5 $b_2 \vee b_1 \vee b_0 \vee \overline{b_1'} \vee b_0'$
C_6 $\overline{b_2'} \vee \overline{b_1'} \vee b_1 \vee \overline{b_2}$
$\ldots$

Conflict Driven Clause Learning



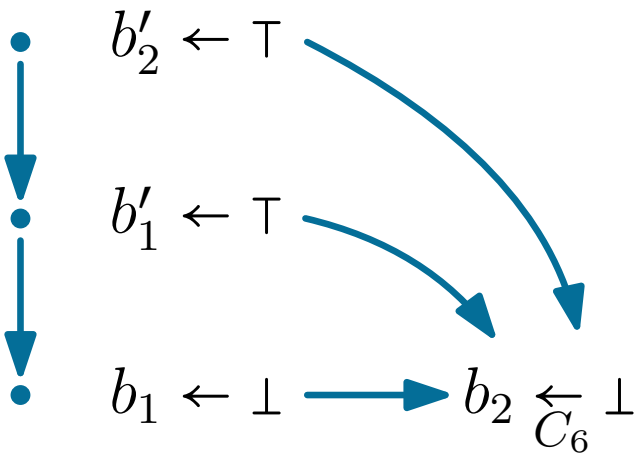
$b'_2 \leftarrow \top$

$b'_1 \leftarrow \top$

$b_1 \leftarrow \bot$

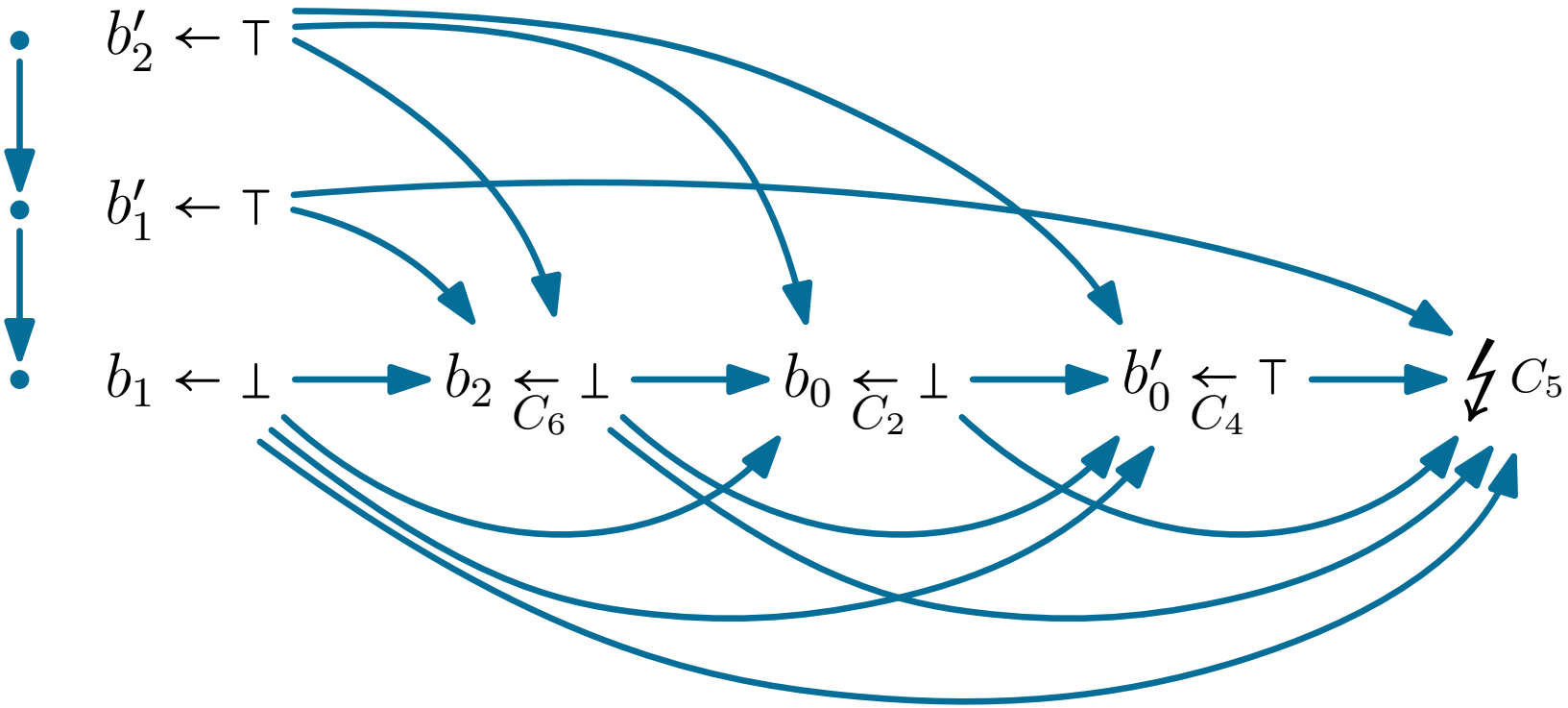
C_1 $\overline{b_2} \vee b_1 \vee b_0 \vee \overline{b'_1}$
C_2 $b_2 \vee b_1 \vee \overline{b_0} \vee \overline{b'_2}$
C_3 $\overline{b_2} \vee b_1 \vee \overline{b_0} \vee \overline{b'_2} \vee \overline{b'_1}$
C_4 $b_2 \vee b_1 \vee b_0 \vee \overline{b'_2} \vee \overline{b'_0}$
C_5 $b_2 \vee b_1 \vee b_0 \vee \overline{b'_1} \vee b'_0$
$\ldots$
C_6 $\overline{b'_2} \vee \overline{b'_1} \vee b_1 \vee \overline{b_2}$

Conflict Driven Clause Learning



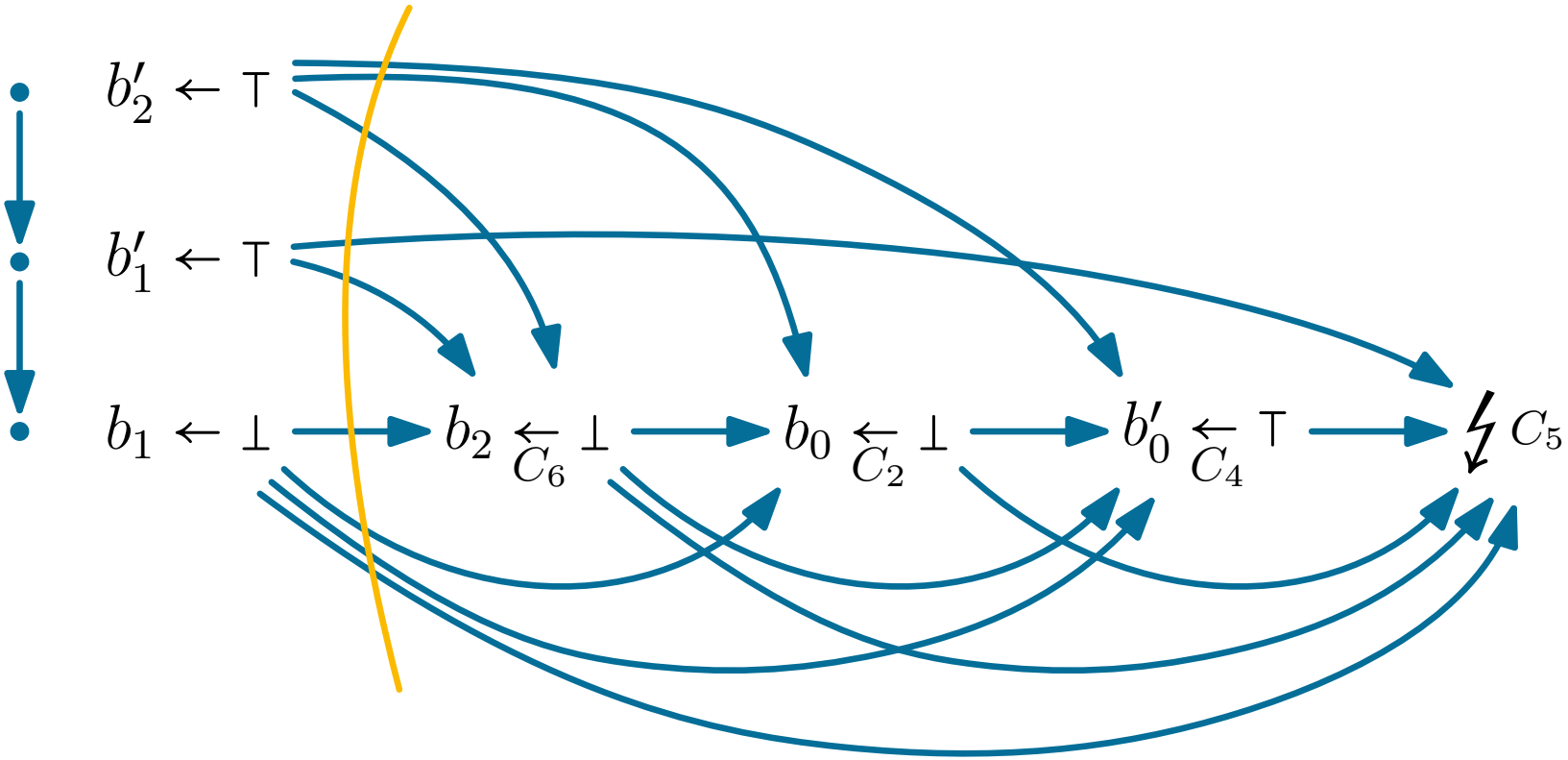
C_1 $\overline{b_2} \vee b_1 \vee b_0 \vee \overline{b_1'}$
C_2 $b_2 \vee b_1 \vee \overline{b_0} \vee \overline{b_2'}$
C_3 $\overline{b_2} \vee b_1 \vee \overline{b_0} \vee \overline{b_2'} \vee \overline{b_1'}$
C_4 $b_2 \vee b_1 \vee b_0 \vee \overline{b_2'} \vee \overline{b_0'}$
C_5 $b_2 \vee b_1 \vee b_0 \vee \overline{b_1'} \vee b_0'$
C_6 $\overline{b_2'} \vee \overline{b_1'} \vee b_1 \vee \overline{b_2}$
$\ldots$

Conflict Driven Clause Learning



C_1 $\overline{b_2} \vee b_1 \vee b_0 \vee \overline{b'_1}$
C_2 $b_2 \vee b_1 \vee \overline{b_0} \vee \overline{b'_2}$
C_3 $\overline{b_2} \vee b_1 \vee \overline{b_0} \vee \overline{b'_2} \vee \overline{b'_1}$
C_4 $b_2 \vee b_1 \vee b_0 \vee \overline{b'_2} \vee \overline{b'_0}$
C_5 $b_2 \vee b_1 \vee b_0 \vee \overline{b'_1} \vee b'_0$
C_6 $\overline{b'_2} \vee \overline{b'_1} \vee b_1 \vee \overline{b_2}$
$\ldots$

Conflict Driven Clause Learning

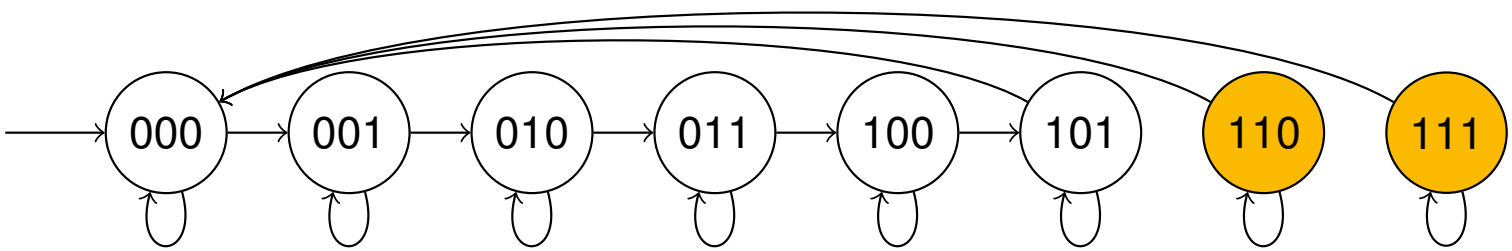


C_1 $\overline{b_2} \vee b_1 \vee b_0 \vee \overline{b'_1}$
C_2 $b_2 \vee b_1 \vee \overline{b_0} \vee \overline{b'_2}$
C_3 $\overline{b_2} \vee b_1 \vee \overline{b_0} \vee \overline{b'_2} \vee \overline{b'_1}$
C_4 $b_2 \vee b_1 \vee b_0 \vee \overline{b'_2} \vee \overline{b'_0}$
C_5 $b_2 \vee b_1 \vee b_0 \vee \overline{b'_1} \vee b'_0$
C_6 $\overline{b'_2} \vee \overline{b'_1} \vee b_1 \vee \overline{b_2}$
C_7 $\overline{b'_2} \vee \overline{b'_1} \vee b_1$
$\ldots$

Counter Example

- Three-bit ($b_2 b_1 b_0$) counter
- Property: Counter does not exceed 5
- $T = \quad (\overline{b_2} \wedge \overline{b_1} \wedge \overline{b_0}) \Rightarrow ((\overline{b_2'} \wedge \overline{b_1'} \wedge \overline{b_0'}) \vee (\overline{b_2'} \wedge \overline{b_1'} \wedge \overline{b_0'}))$
  $\wedge\ (\overline{b_2} \wedge \overline{b_1} \wedge b_0) \Rightarrow ((\overline{b_2'} \wedge \overline{b_1'} \wedge b_0') \vee (\overline{b_2'} \wedge b_1' \wedge \overline{b_0'}))$
  $\wedge \ldots$
- Can state 7 be reached?

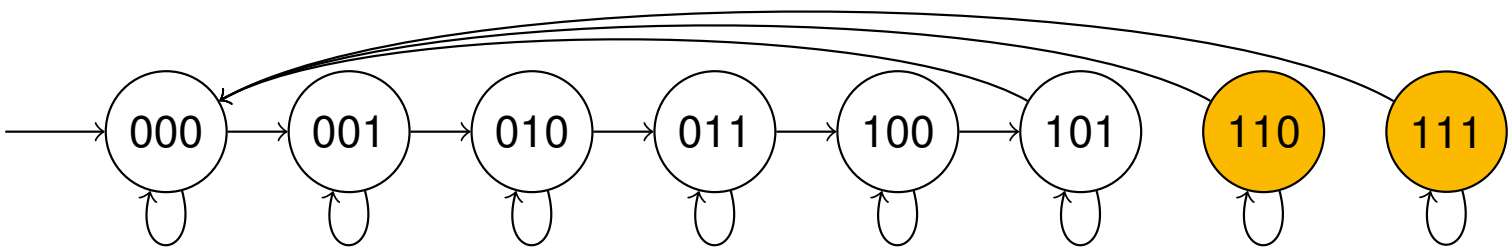
  $(\overline{b_2} \vee \overline{b_1} \vee \overline{b_0}) \wedge T \wedge b_2' \wedge b_1' \wedge b_0'$?



Counter Example

- Three-bit ($b_2 b_1 b_0$) counter
- Property: Counter does not exceed 5
- $T = \quad (\overline{b_2} \wedge \overline{b_1} \wedge \overline{b_0}) \Rightarrow ((\overline{b_2'} \wedge \overline{b_1'} \wedge \overline{b_0'}) \vee (\overline{b_2'} \wedge \overline{b_1'} \wedge \overline{b_0'}))$
  $\wedge\ (\overline{b_2} \wedge \overline{b_1} \wedge b_0) \Rightarrow ((\overline{b_2'} \wedge \overline{b_1'} \wedge b_0') \vee (\overline{b_2'} \wedge b_1' \wedge \overline{b_0'}))$
  $\wedge \ldots$
- Can the cube be reached?

110	$(\overline{b_2} \vee \overline{b_1} \vee b_0) \wedge T \wedge b_2' \wedge b_1' \wedge \overline{b_0'}$	UNSAT
11_	$(\overline{b_2} \vee \overline{b_1}) \quad \wedge T \wedge b_2' \wedge b_1'$	UNSAT



Assumptions

$$\underbrace{(\overline{b_2} \vee \overline{b_1} \vee b_0)}_{\text{Constraint}} \quad \wedge \quad T \quad \wedge \quad \underbrace{b_2' \wedge b_1' \wedge \overline{b_0'}}_{\text{Assumptions}}$$

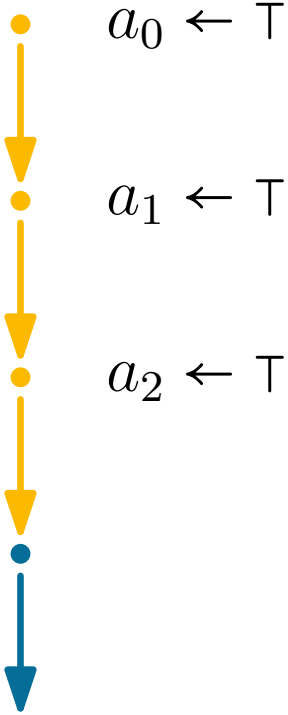
addClause	(`Clause` c)
solve	(`list` < `Literal` > assumptions)

"[. . .] we propose the following interface [. . .] very simple to implement [. . .] expressive enough to encompass several interesting incremental SAT-problems [. . .]"

Temporal Induction by Incremental SAT Solving

[Eén, Niklas and Sörensson, Niklas, 2003]

Assumptions



DECIDE ( )

```
1  if level < |assumptions|
2      ℓ = assumptions[level]
3      if val(ℓ) = false analyzeFinal(ℓ)
4      else trail[level++] = ℓ

5  else trail[level++] = heuristic()
```

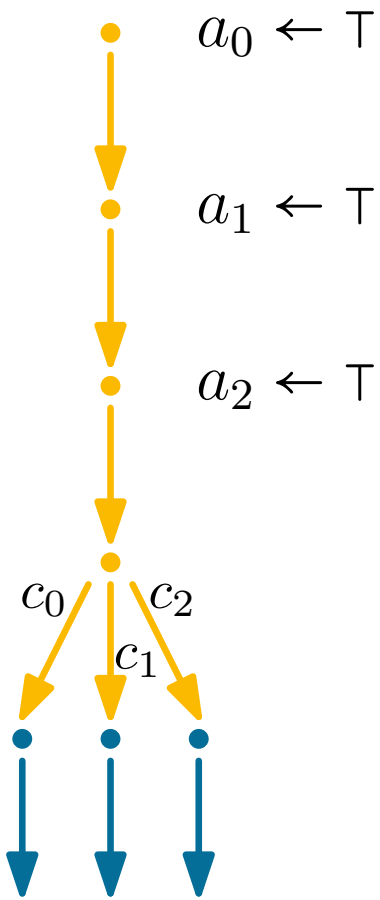
Our Contribution

$$\underbrace{(\overline{b_2} \vee \overline{b_1} \vee b_0)}_{\text{Constraint}} \quad \wedge \quad T \quad \wedge \quad \underbrace{b_2' \wedge b_1' \wedge \overline{b_0'}}_{\text{Assumptions}}$$

addClause	(`Clause` c)
solve	(`list < Literal >` assumptions, `Clause` c)

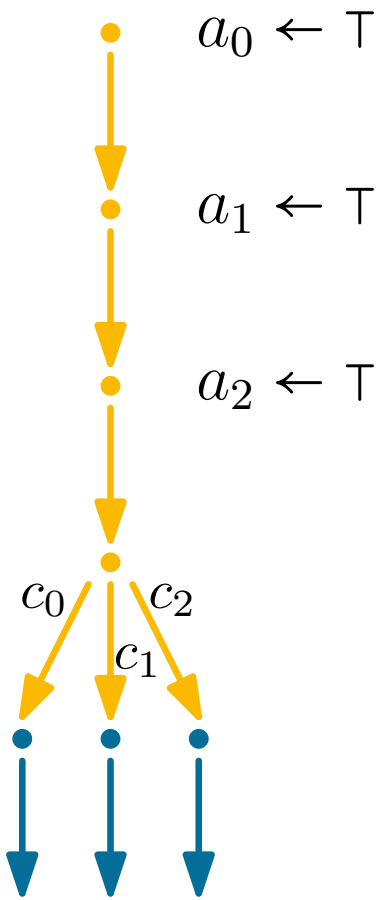
- Easy to implement
- Expressive enough for IC3
- Learned clauses are save to keep

Constraint




```
DECIDE ()
1  if level < |assumptions|
2      ℓ = assumptions[level]
3      if val(ℓ) = false analyzeFinal(ℓ)
4      else trail[level++] = ℓ
5  else if level = |assumptions|
6      unassignedLit = 0
7      for ℓ in constraint
8          if val(ℓ) = true level++
9          else if val(ℓ) = unassigned
10             unassigendLit = ℓ
11     if unassigendLit = 0
12         analyzeFinal(constraint)
13     else trail[level++] = unassigendLit
14 else trail[level++] = heuristic()
```

Constraint




```
DECIDE ()
1  if level < |assumptions|
2      ℓ = assumptions[level]
3      if val(ℓ) = false analyzeFinal(ℓ)
4      else trail[level++] = ℓ
5  else if level = |assumptions|
6      unassignedLit = 0
7      for ℓ in constraint
8          if val(ℓ) = true level++
9          else if val(ℓ) = unassigned
10             unassigendLit = ℓ
11     if unassigendLit = 0
12         analyzeFinal(constraint)
13     else trail[level++] = unassigendLit
14 else trail[level++] = heuristic()
```

Analyze Final

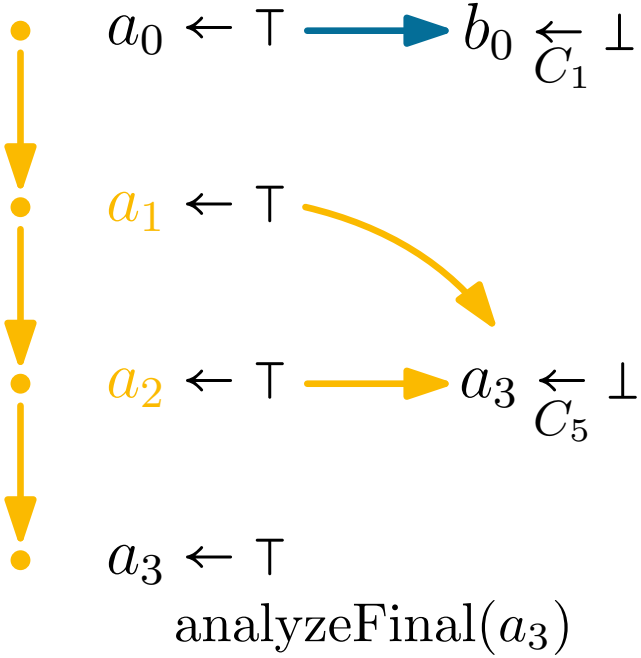
failed(`Literal` ℓ)

Queries if ℓ was used to proof unsatisfiability

If only b_2' and b_1' failed the second cube is free

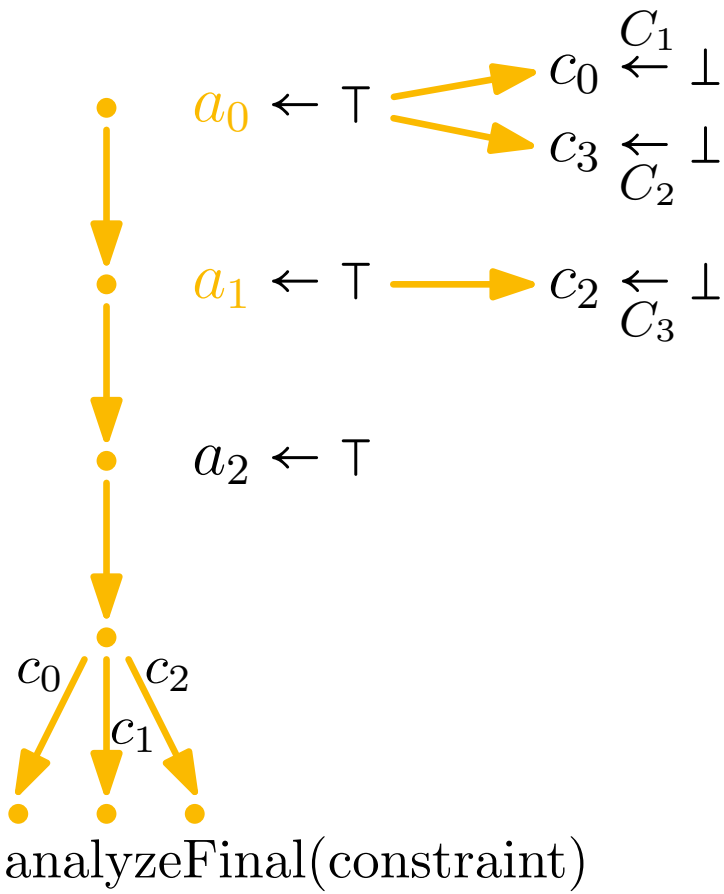
110	$(\overline{b_2} \vee \overline{b_1} \vee b_0) \wedge T \wedge b_2' \wedge b_1' \wedge \overline{b_0'}$	UNSAT
		$\Downarrow$
11_	$(\overline{b_2} \vee \overline{b_1}) \wedge T \wedge b_2' \wedge b_1'$	UNSAT

Analyze Final




```
DECIDE ()
1  if level < |assumptions|
2      ℓ = assumptions[level]
3      if val(ℓ) = false analyzeFinal(ℓ)
4      else trail[level++] = ℓ
5  else if level = |assumptions|
6      unassignedLit = 0
7      for ℓ in constraint
8          if val(ℓ) = true level++
9          else if val(ℓ) = unassigned
10             unassigendLit = ℓ
11     if unassigendLit = 0
12         analyzeFinal(constraint)
13     else trail[level++] = unassigendLit
14 else trail[level++] = heuristic()
```

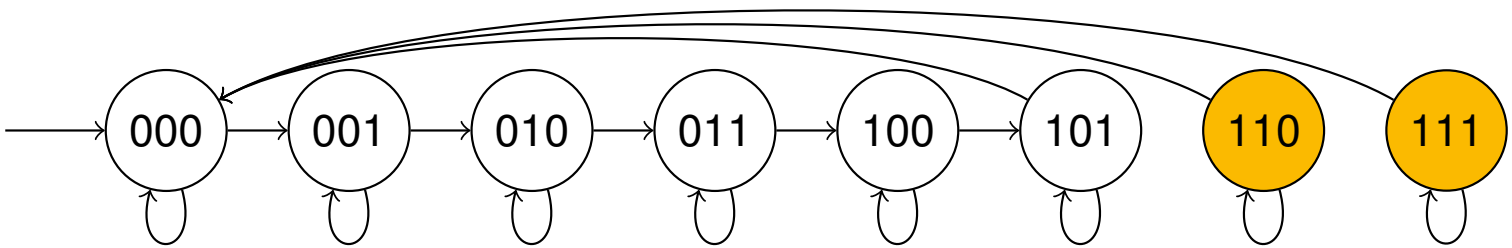
Analyze Final




```
DECIDE ()
1  if level < |assumptions|
2      ℓ = assumptions[level]
3      if val(ℓ) = false analyzeFinal(ℓ)
4      else trail[level++] = ℓ
5  else if level = |assumptions|
6      unassignedLit = 0
7      for ℓ in constraint
8          if val(ℓ) = true level++
9          else if val(ℓ) = unassigned
10             unassigendLit = ℓ
11     if unassigendLit = 0
12         analyzeFinal(constraint)
13     else trail[level++] = unassigendLit
14 else trail[level++] = heuristic()
```

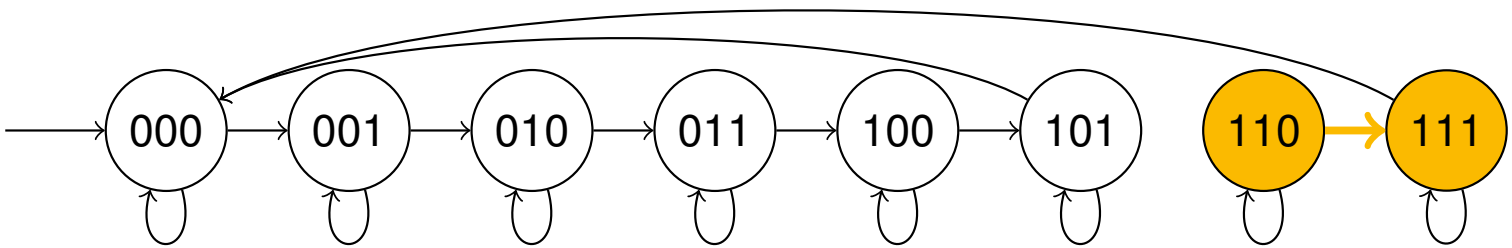
IC3

$$\overline{s} \wedge \quad T \quad \wedge s'$$

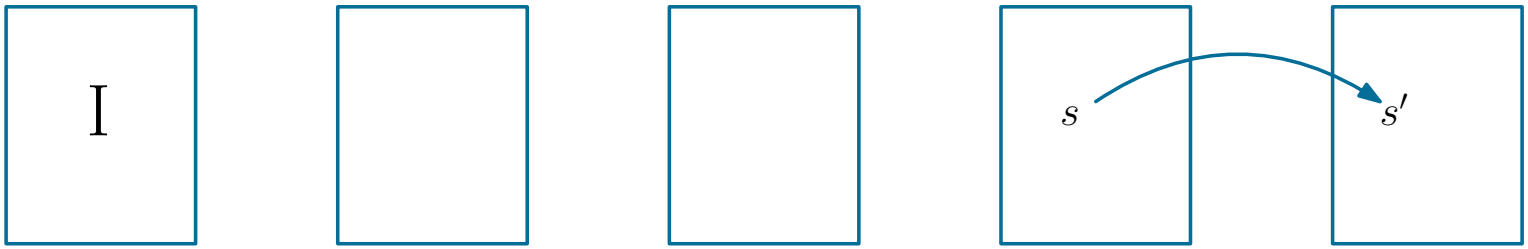


IC3

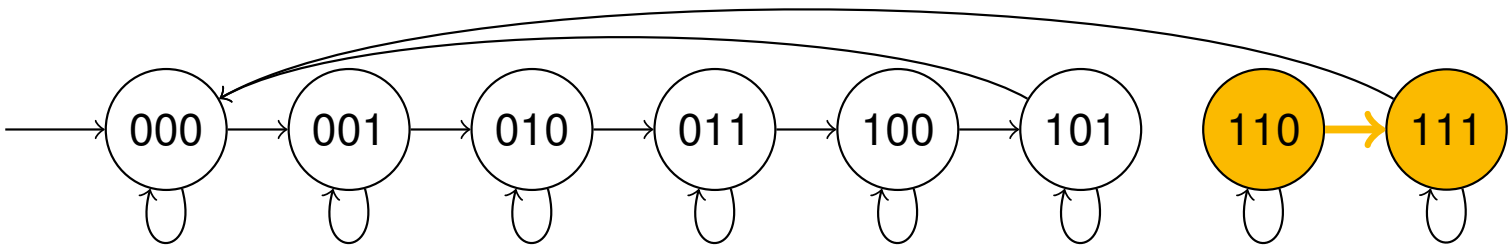
$$\overline{s} \wedge \quad T \quad \wedge s'$$



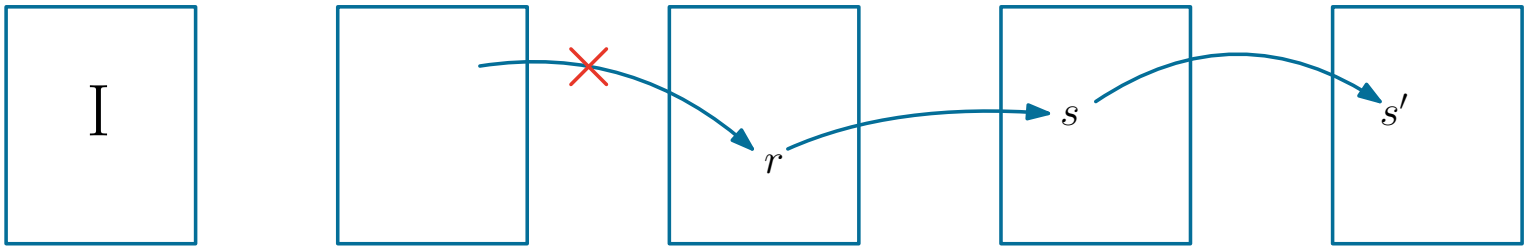
IC3



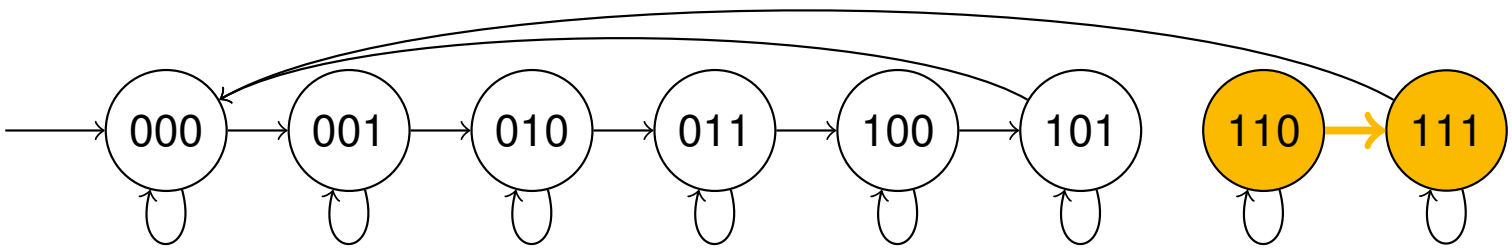
$$\underbrace{\overline{s}}_{\text{Constraint}} \wedge \underbrace{Frame_{i-1} \wedge T}_{\text{Clauses}} \wedge \underbrace{s'}_{\text{Assumptions}}$$



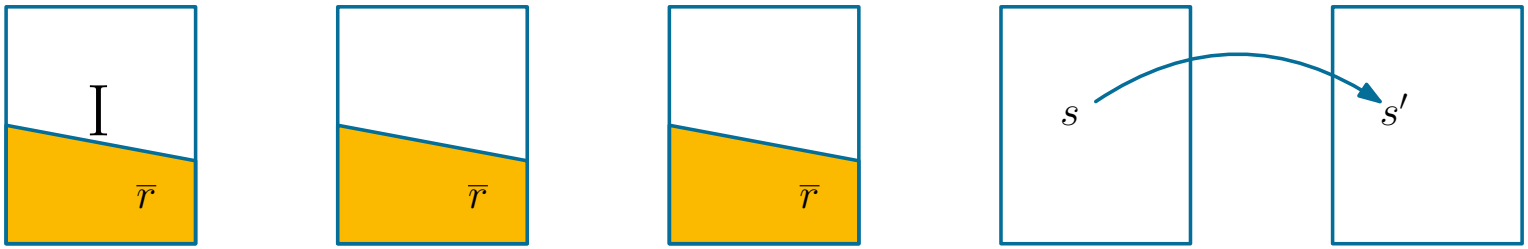
IC3



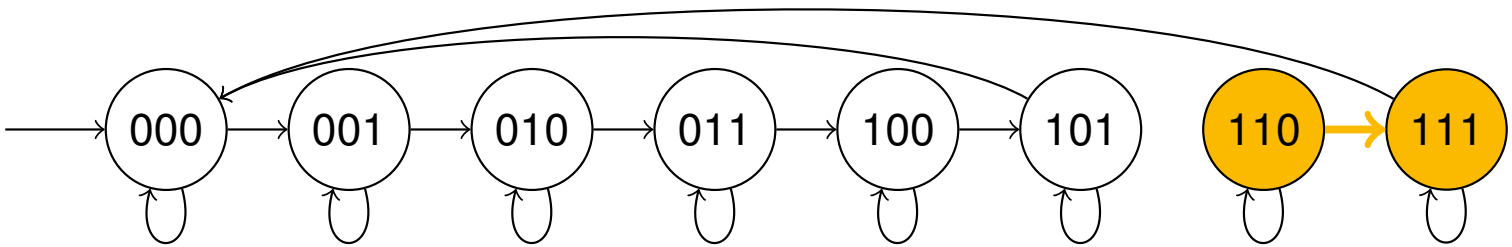
$$\underbrace{\overline{s}}_{\text{Constraint}} \wedge \underbrace{Frame_{i-1} \wedge T}_{\text{Clauses}} \wedge \underbrace{s'}_{\text{Assumptions}}$$



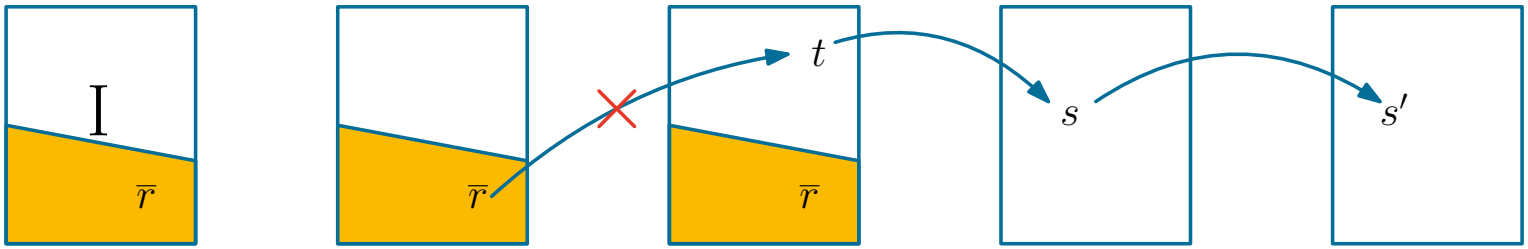
IC3



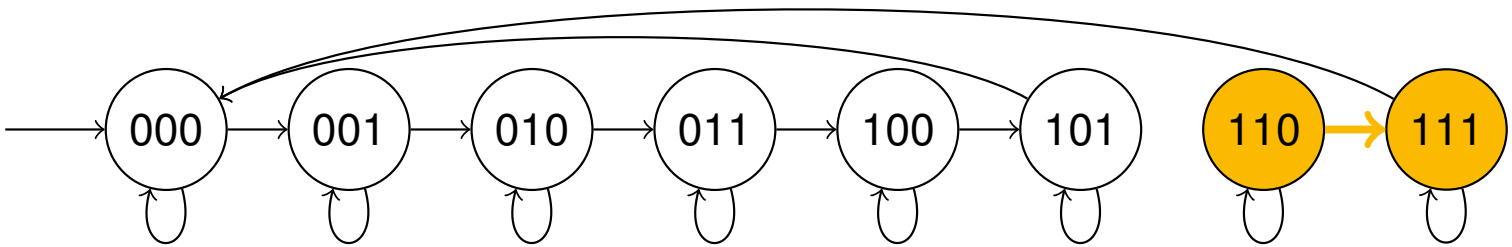
$$\underbrace{\overline{s}}_{\text{Constraint}} \wedge \underbrace{Frame_{i-1} \wedge T}_{\text{Clauses}} \wedge \underbrace{s'}_{\text{Assumptions}}$$



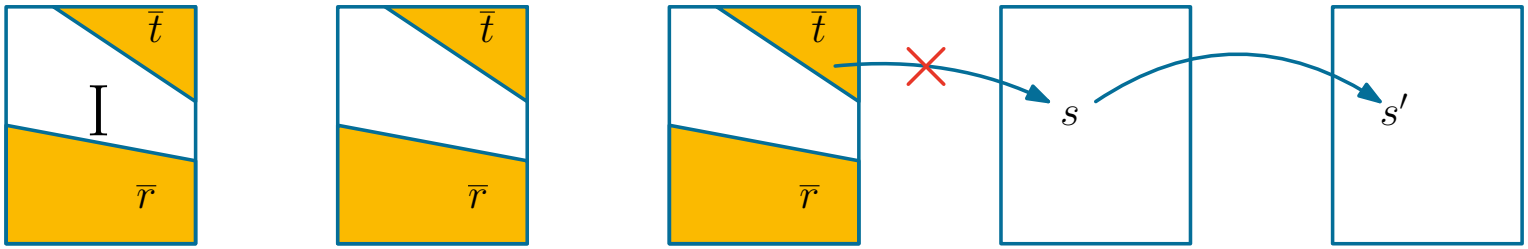
IC3



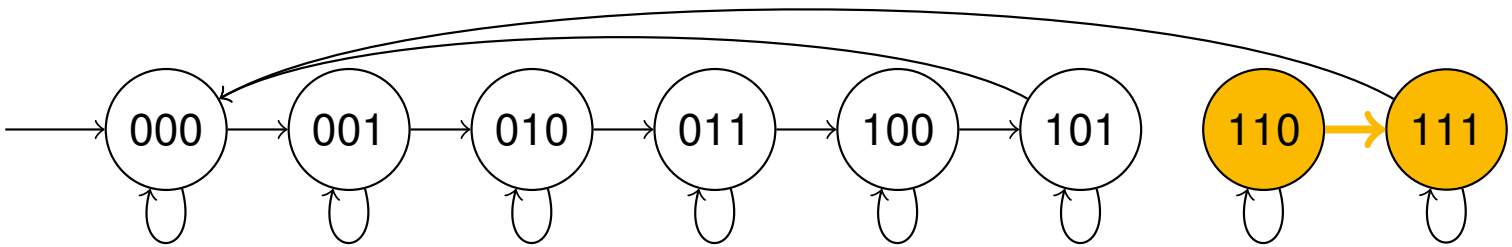
$$\underbrace{\overline{s}}_{\text{Constraint}} \wedge \underbrace{Frame_{i-1} \wedge T}_{\text{Clauses}} \wedge \underbrace{s'}_{\text{Assumptions}}$$



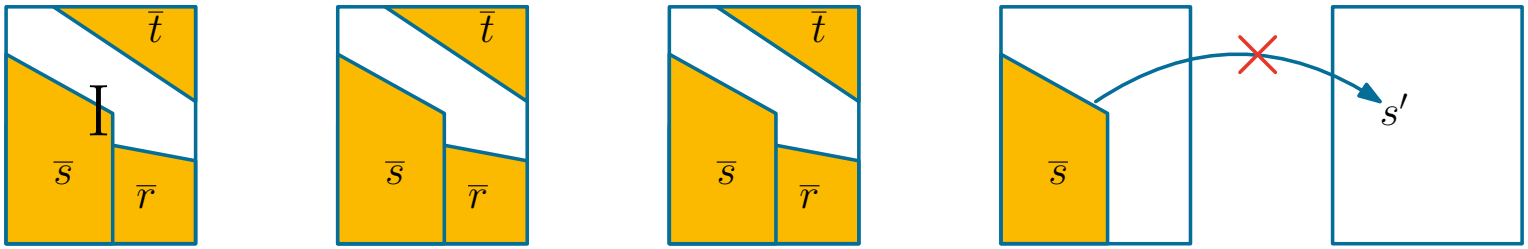
IC3



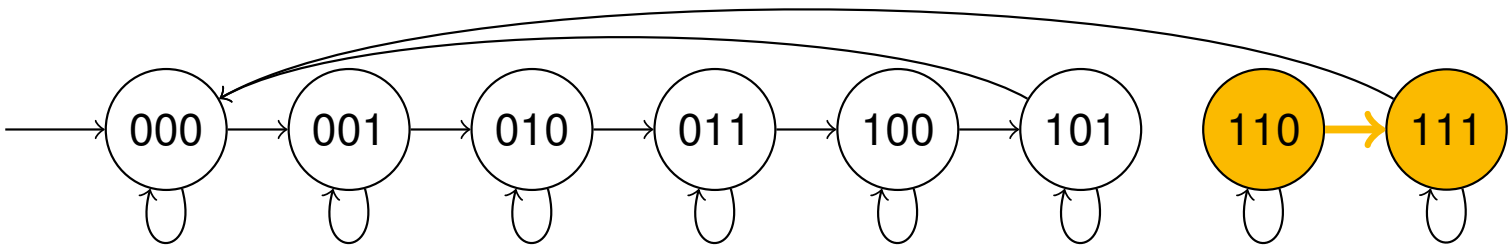
$$\underbrace{\overline{s}}_{\text{Constraint}} \wedge \underbrace{Frame_{i-1} \wedge T}_{\text{Clauses}} \wedge \underbrace{s'}_{\text{Assumptions}}$$



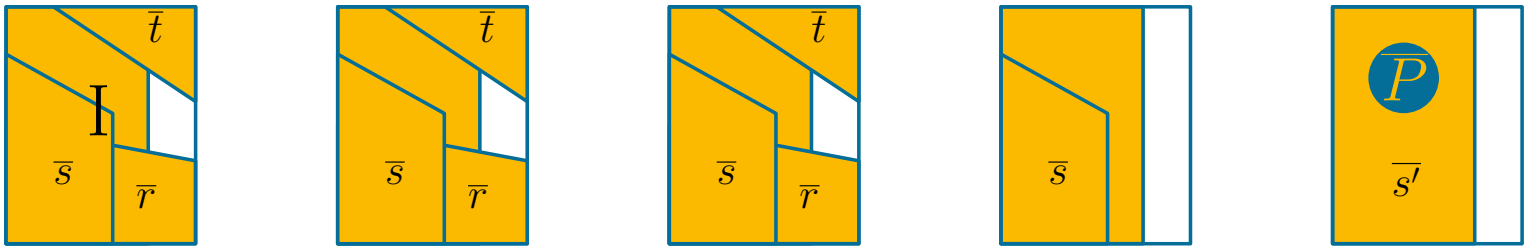
IC3



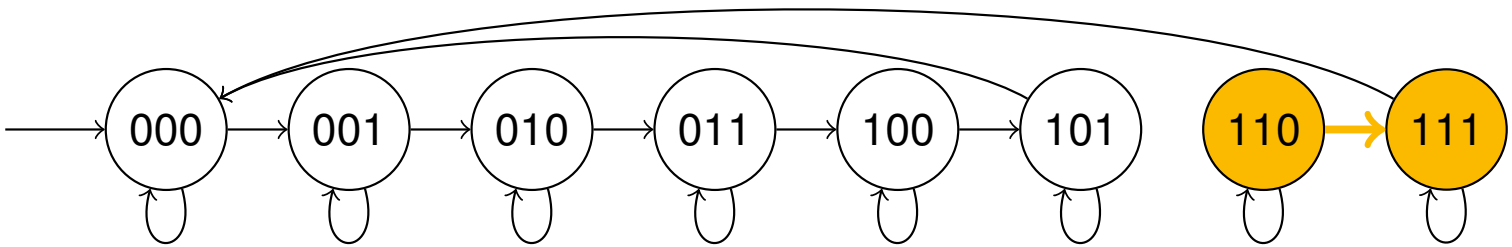
$$\underbrace{\overline{s}}_{\text{Constraint}} \wedge \underbrace{Frame_{i-1} \wedge T}_{\text{Clauses}} \wedge \underbrace{s'}_{\text{Assumptions}}$$



IC3



$$\underbrace{\overline{s}}_{\text{Constraint}} \wedge \underbrace{Frame_{i-1} \wedge T}_{\text{Clauses}} \wedge \underbrace{s'}_{\text{Assumptions}}$$



CaDiCaL

> *IC3 is the first widespread verification method that requires highly efficient incremental solvers. An incremental interface for IC3 must allow single clauses to be pushed and popped; it must also allow literal assumptions. [It poses] many thousands to millions of queries in the course of an analysis, and so speed is crucial.*
>
> *Understanding IC3 [Bradley, Aaron R., 2012]*

github.com/arminbiere/cadical

Model-based testing: 1.5 Billion runs

Activation Literal

$$\underbrace{\overline{s}}_{\text{Constraint}} \wedge \underbrace{Frame_{i-1} \wedge T}_{\text{Clauses}} \wedge \underbrace{s'}_{\text{Assumptions}}$$

$$\underbrace{(a \vee \overline{s}) \wedge Frame_{i-1} \wedge T}_{\text{Clauses}} \wedge \underbrace{s' \wedge \overline{a}}_{\text{Assumptions}}$$

- Deactivated Clauses
- Unused activation literals
- Solver restarts – SAT solver management strategies in IC3

  [Cabodi, G. and Camurati, P. E. and Mishchenko, A. and Palena, M. and Pasini, P., 2017]
- Worse Learned Clauses
  - Deleted by restarts
  - Tend to contain the activation literal

$\Rightarrow$ More failed assumptions

$\Rightarrow$ Worse generalization

Experimental Setup

- Bit-level model checker ABC
  - Multi-engine
  - Sophisticated preprocessing
- Replaced MiniSat [Eén, Niklas and Sörensson, Niklas, 2004]
  with CaDiCaL
  [Biere, Armin and Fleury, Mathias and Heisinger, Maximilian and Fazekas, Katalin, 2020]
- Replaced activation literals with constraints
- Benchmarks HWMCC'19 [Preiner, Mathias and Biere, Armin, 2019]
  - 219 instances, 15 unsolved and removed

PAR-2 Score

	Di	Og	Ca	Co	De
Mean	80	46	16	8.93	**8.21**
beemTele6Int	136	7200	**53**	181	101
toyLock4	7200	483	1731	**357**	359
visArraysField5	7200	1.6	**0.58**	51	34
nan	208	421	163	158	**140**
beemColl6Int	241	258	322	133	**108**
cal110	213	168	130	**110**	122
cal109	179	197	102	117	**86**
cal93	186	136	121	**118**	140
cal94	127	160	115	**95**	131
cal100	112	**42**	67	67	54

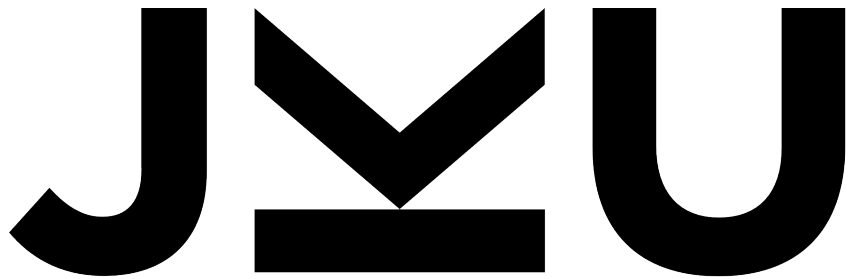
Disable restarts, **O**ri**g**inal ABC, **Ca**DiCaL,
Constraints, **De**fer model reconstruction

Time Per Call

	Restarts	Calls [10^3]		TpC [ms]	
	Ca	Ca	Co	Ca	Co
Mean	61	19	**15**	0.61	**0.51**
beemTele6Int	520	**157**	574	**0.24**	0.27
toyLock4	7459	2251	**1098**	0.42	**0.25**
visArraysField5	1	**<1**	113	0.53	**0.41**
nan	1381	**420**	423	**0.29**	0.32
beemColl6Int	398	123	**91**	2.31	**1.24**
cal110	191	59	**42**	**1.96**	2.39
cal109	110	**34**	44	2.71	**2.44**
cal93	206	63	**58**	**1.69**	1.8
cal94	171	52	**41**	**1.94**	2.1
cal100	148	45	**44**	**1.23**	1.29

Conclusion

- Our approach
  - Easy to implement in modern SAT solvers
  - Simplifies solver usage
  - Speedup of 1.84 for IC3
- Future work
  - Multiple constraints for other applications
  - Reduce number of failed literals



JOHANNES KEPLER
UNIVERSITY LINZ

References I

[Biere, Armin and Fleury, Mathias and Heisinger, Maximilian and Fazekas, Katalin, 20

Biere, Armin and Fleury, Mathias and Heisinger, Maximilian and Fazekas, Katalin (2020).
Entering the SAT Competition 2020.
page 4.

[Bradley, Aaron R., 2012] Bradley, Aaron R. (2012).
Understanding IC3.
In Cimatti, Alessandro and Sebastiani, Roberto, editor, *Theory and Applications of Satisfiability Testing – SAT 2012*, Lecture Notes in Computer Science, pages 1–14, Berlin, Heidelberg. Springer.

[Cabodi, G. and Camurati, P. E. and Mishchenko, A. and Palena, M. and Pasini, P., 20

Cabodi, G. and Camurati, P. E. and Mishchenko, A. and Palena, M. and Pasini, P. (2017).
SAT Solver Management Strategies in IC3: An Experimental Approach.
Formal Methods in System Design, 50:39–74.

References II

[Eén, Niklas and Sörensson, Niklas, 2003] Eén, Niklas and Sörensson, Niklas (2003).
Temporal Induction by Incremental SAT Solving.
Electronic Notes in Theoretical Computer Science, 89(4):543–560.

[Eén, Niklas and Sörensson, Niklas, 2004] Eén, Niklas and Sörensson, Niklas (2004).
An Extensible SAT-Solver.
In Giunchiglia, Enrico and Tacchella, Armando, editor, *Theory and Applications of Satisfiability Testing*, Lecture Notes in Computer Science, pages 502–518, Berlin, Heidelberg. Springer.

[Preiner, Mathias and Biere, Armin, 2019] Preiner, Mathias and Biere, Armin (2019).
Hardware Model Checking Competition 2019.
http://fmv.jku.at/hwmcc19/.