ULTIMATELY INCREMENTAL SAT

Alexander Nadel[1], Vadim Ryvchin[1,2], and Ofer Strichman[2]

1 – Intel, Haifa, Israel
2 – Technion, Haifa, Israel

SAT'14, Vienna, Austria

Introduction

- Incremental SAT is at the core of a variety of applications
- Assumptions are widely used in Incremental SAT
- Preprocessing is essential for performance

Recall: SatELite Preprocessing

1. Variable Elimination

   $\phi = \phi \cup Res_v(\phi_v, \phi_{\neg v}) \backslash (\phi_v \cup \phi_{\neg v})$

2. Subsumption

   $c_1=(a \vee b)$ $c_2=(a \vee b \vee c)$ ➔ c_1 subsumes c_2

3. Self-subsuming resolution

   $c_1=(a \vee b)$ $c_2=(a \vee \neg b \vee c)$ ➔ $c_2=(a \vee c)$

Recall: Clause Database Simplification

1. Propagation of unit clauses
2. Elimination of satisfied clauses
3. Removal of falsified literals from clauses

Incremental SAT under Assumptions

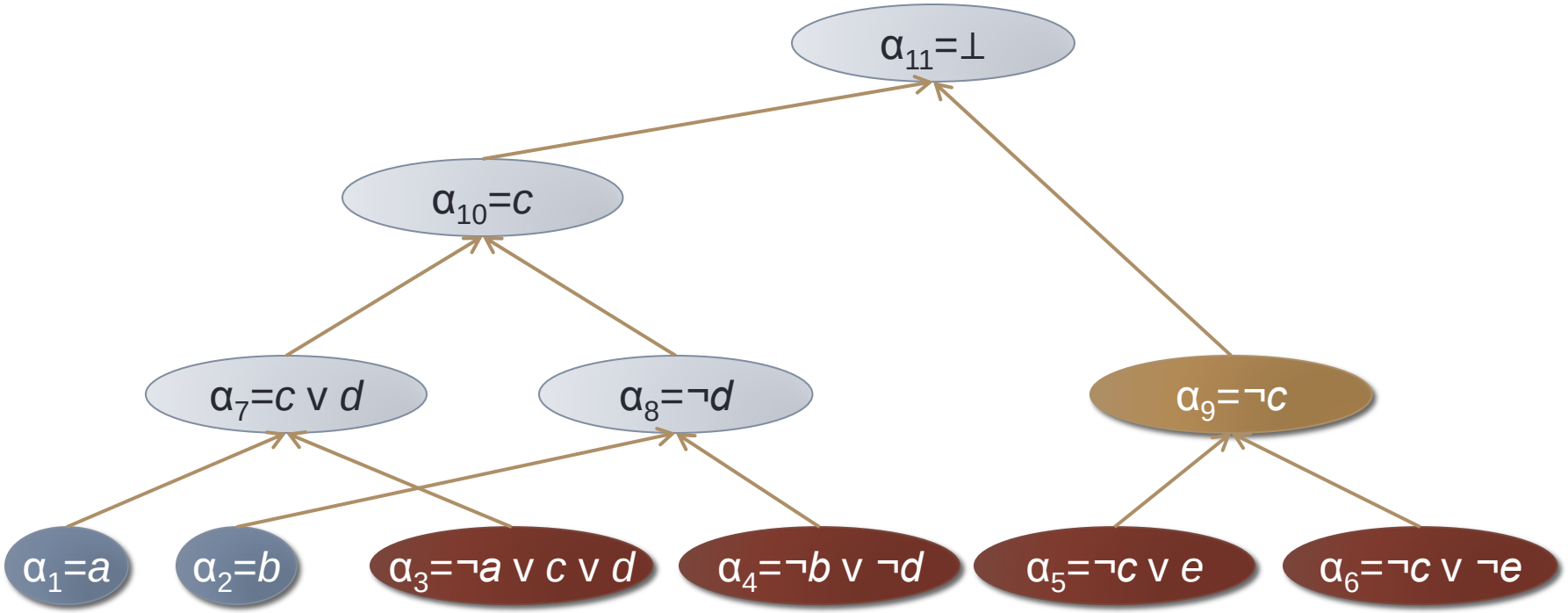
- Initial Formula + Assumptions: $(\phi_0 = \Delta_0) \wedge A_0$
- 1st incremental Call: $(\phi_1 = \phi_0 + \Delta_1) \wedge A_1$

- Nth incremental call: $(\phi_n = \phi_{n-1} + \Delta_n) \wedge A_n$

Temporary vs. Pervasive clauses

- We say that a clause is *temporary* if it is either an assumption or was derived from one or more assumptions, and *pervasive* otherwise.

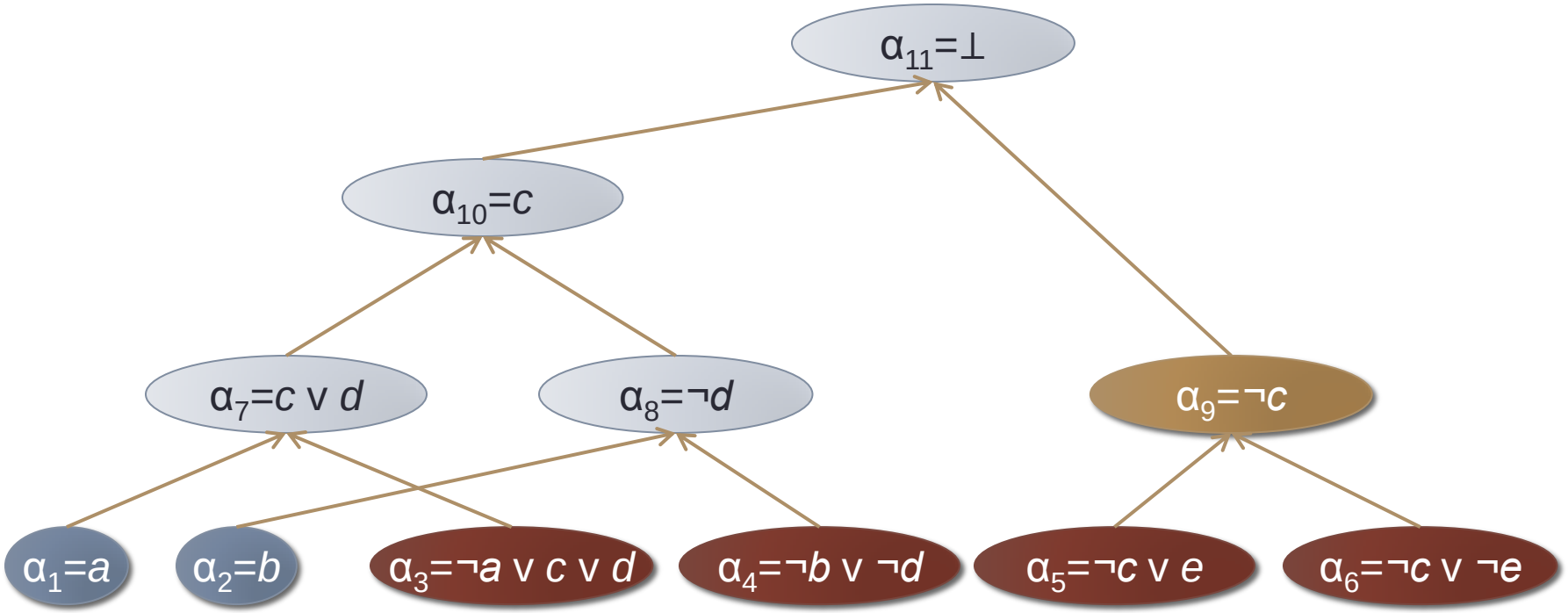
Temporary vs. Pervasive



Legend:

Input clauses	Pervasive conflict clauses	Assumptions	Temporary conflict clauses

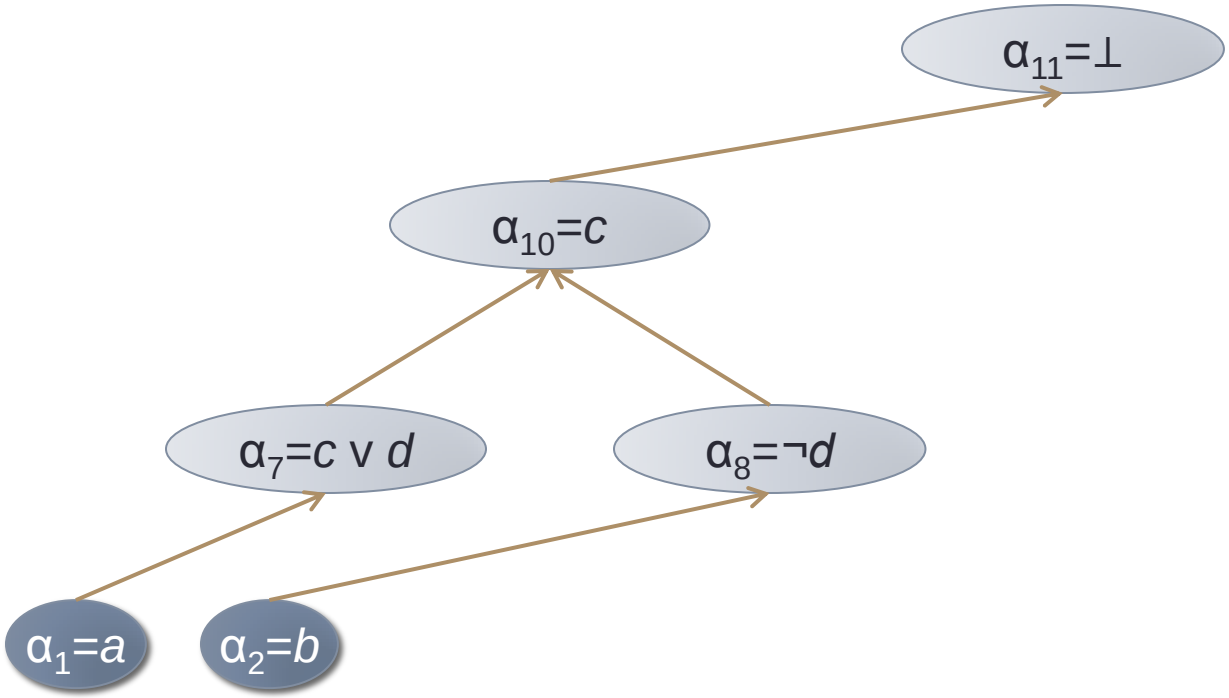
Temporary Partial Resolution



Legend:

Input clauses	Pervasive conflict clauses	Assumptions	Temporary conflict clauses

Temporary Partial Resolution



Legend:

Assumptions

Temporary conflict clauses

Assumptions

	.. as decision	.. as unit clauses
Compatible with Inc. SAT	+	-
Conflict clauses are pervasive	+	-
Simplification	-	+
Preprocessing	-	+

Previous Approaches

Algorithm	Instances	Assumption as units	SatELite	Assumption -dep. Clauses
Clause Sharing	Multiple	Yes	No	Discard
Minisat-Alg	One	No	No	Keep All
Assumption prop.	Multiple	Yes	Full	T2P
Incremental SatELite	One	No	Incremental	Keep All

Previous Approaches

Algorithm	Instances	Assumption as units	SatELite	Assumption -dep. Clauses
Clause Sharing	Multiple	Yes	No	Discard
Minisat-Alg	One	No	No	Keep All
Assumption prop.	Multiple	Yes	Full	T2P
Incremental SatELite	One	No	Incremental	Keep All

Ofer Strichman. Sharing information between SAT instances, Dec 2000, Patent
Ofer Strichman. Pruning techniques for the SAT-based bounded model checking problem. CHARME'01.
Jesse Whittemore, Joonyoung Kim, and Karem A. Sakallah. SATIRE: A new incremental satisfiability engine, *DAC'01*

Previous Approaches

Algorithm	Instances	Assumption as units	SatELite	Assumption -dep. Clauses
Clause Sharing	Multiple	Yes	No	Discard

For every incremental call i:

- Create a new solver S_i
- Add ϕ_i to S_i
- Add A_i as temporary unit clauses to S_i
- Add pervasive conflict clauses from S_{i-1} to S_i
- Solve S_i

Ofer Strichman. Sharing information between SAT instances, Dec 2000, Patent
Ofer Strichman. Pruning techniques for the SAT-based bounded model checking problem. CHARME'01.
Jesse Whittemore, Joonyoung Kim, and Karem A. Sakallah. SATIRE: A new incremental satisfiability engine, *DAC'01*

Previous Approaches

Algorithm	Instances	Assumption as units	SatELite	Assumption -dep. Clauses
Clause Sharing	Multiple	Yes	No	Discard
Minisat-Alg	One	No	No	Keep All
Assumption prop.	Multiple	Yes	Full	T2P
Incremental SatELite	One	No	Incremental	Keep All

Niklas Een and Niklas Sorensson. An extensible SAT-solver, SAT'03.

Previous Approaches

Algorithm	Instances	Assumption as units	SatELite	Assumption -dep. Clauses
Clause Sharing	Multiple	Yes	No	Discard
Minisat-Alg	One	No	No	Keep All
Assu				2P
Incr Sa				p All

Create a solver S
For every incremental call i:
Add Δ_i to S
Add A_i as assumptions (decisions) to S
Solve S

Niklas Een and Niklas Sorensson. An extensible SAT-solver, SAT'03.

Previous Approaches

Algorithm	Instances	Assumption as units	SatELite	Assumption -dep. Clauses
Clause Sharing	Multiple	Yes	No	Discard
Minisat-Alg	One	No	No	Keep All
Assumption prop.	Multiple	Yes	Full	T2P
Incremental SatELite	One	No	Incremental	Keep All

Alexander Nadel and Vadim Ryvchin. Efficient SAT solving under assumptions, SAT'12.

Previous Approaches

Algorithm	Instances	Assumption as units	SatELite	Assumption -dep. Clauses
Clause Sharing	Multiple	Yes	No	Discard
Minisat-Alg	One	No	No	Keep All
Assumption prop.	Multiple	Yes	Full	T2P
Incremental SatELite	One	No	Incremental	Keep All

Alexander Nadel and Vadim Ryvchin. Efficient SAT solving under assumptions, SAT'12.

Previous Approaches

Algorithm	Instances	Assumption as units	SatELite	Assumption -dep. Clauses

For every incremental call i:

- In S_{i-1} convert learnt temporary clauses to pervasive
- Create a new solver S_i
- Add ϕ_i to S_i
- Copy all learnt clauses from S_{i-1} to S_i
- Add A_i as unit temporary clauses to S_i
- Run preprocessor over S_i
- Solve S_i

Alexander Nadel and Vadim Ryvchin. Efficient SAT solving under assumptions, SAT'12.

Previous Approaches

Algorithm	Instances	Assumption as units	SatELite	Assumption -dep. Clauses
Clause Sharing	Multiple	Yes	No	Discard
Minisat-Alg	One	No	No	Keep All
Assumption prop.	Multiple	Yes	Full	T2P
Incremental SatELite	One	No	Incremental	Keep All

Alexander Nadel, Vadim Ryvchin, and Ofer Strichman. Preprocessing in incremental SAT, SAT'12.

Previous Approaches

Algorithm	Instances	Assumption as units	SatELite	Assumption -dep. Clauses
Clause Sharing	Multiple	Yes	No	Discard
Minisat-Alg	One	No	No	Keep All
Assumption prop.	Multiple	Yes	Full	T2P
Incremental SatELite	One	No	Incremental	Keep All

Alexander Nadel, Vadim Ryvchin, and Ofer Strichman. Preprocessing in incremental SAT, SAT'12.

Previous Approaches

Algorithm	Instances	Assumption as units	SatELite	Assumption -dep. Clauses
Clause				Discard
Mini				Keep All
Assu p				T2P
Incremental SatELite	One	No	Incremental	Keep All

Create a new solver S
For every incremental call i:
Add Δ_i to S
Add A_i as assumptions to S
Run preprocessor over S (Incremental)
Solve S

Alexander Nadel, Vadim Ryvchin, and Ofer Strichman. Preprocessing in incremental SAT, SAT'12.

Our New Approach

Algorithm	Instances	Assumption as units	SatELite	Assumption -dep. Clauses
Clause Sharing	Multiple	Yes	No	Discard
Minisat-Alg	One	No	No	Keep All
Assumption prop.	Multiple	Yes	Full	T2P
Incremental SatELite	One	No	Incremental	Keep All
UI-SAT	One	Yes	Incremental	Incremental T2P

Our New Approach

Algorithm	Instances	Assumption as units	SatELite	Assumption -dep. Clauses
Clause Sharing	Multiple	Yes	No	Discard
Minisat-Alg	One	No	No	Keep All
Assumption prop.	Multiple	Yes	Full	T2P
Incremental SatELite	One	No	Incremental	Keep All
UI-SAT	One	Yes	Incremental	Incremental T2P

Our New Approach

Algorithm	Instances	Assumption as units	SatELite	Assumption -dep. Clauses
Clause S				rd
Minisa				All
Assum pro				
Increm SatE				All
UI-SAT	One	Yes	Incremental	Incremental T2P

Create a new solver S
For every incremental call i:
Undo assumptions in S (Inc. T2P)
Add A_i as temporary unit clauses to S
Add Δ_i clauses to S
Run preprocessor over S (Incremental)
Solve S

Incremental SatELite (NRS'12)

Freeze Assumptions // For current call
Preprocess
Solve // might run in-processing
UnFreeze Assumptions

Incremental SatELite (NRS'12)

```
Remove subsumptions.
For each eliminated variable in elimination order:
    if (*) // decide heuristically
        Re-Eliminate
    else
        Re-Introduce
Freeze Assumptions // For current call
Preprocess
Solve // might run in-processing
UnFreeze Assumptions
```

Incremental SatELite (NRS'12)

Core of Incremental Preprocessing

Remove subsumptions.
For each eliminated variable in elimination order:
  if (*) // decide heuristically
    Re-Eliminate
  else
    Re-Introduce
Freeze Assumptions // For current call
Preprocess
Solve // might run in-processing
UnFreeze Assumptions

Incremental SatELite (SAT'12)

Assumptions as Unit Clauses

Remove subsumptions.
For each eliminated variable in elimination order:
    if (*) // decide heuristically
        Re-Eliminate
    else
        Re-Introduce
Freeze Assumptions // For current call
Preprocess
Solve // might run in-processing
UnFreeze Assumptions

Incremental SatELite (SAT'12)

Assumptions as Unit Clauses

Remove subsumptions.
For each eliminated variable in elimination order:
    if (*) // decide heuristically
        Re-Eliminate
    else
        Re-Introduce

Preprocess
Solve // might run in-processing

Incremental SatELite (SAT'12)

Assumptions as Unit Clauses

Undo Previous Assumptions
Add assumptions as temporary unit clauses
Remove subsumptions.
For each eliminated variable in elimination order:
    if (*) // decide heuristically
        Re-Eliminate
    else
        Re-Introduce

Preprocess
Solve // might run in-processing

Data Structures

- We need to keep relevant data to undo assumptions.
- SubsumedClauses
  - Per clause – set of clauses subsumed by it

Subsumption (Example)

- Two clauses: $c_1=(a \vee b)$ $c_2=(a \vee b \vee c)$
- Normal subsumption: Delete c_2
- Our Subsumption: if c_1 marked as temporary:
  - c_1 subsumes c_2 → Add(c_2) to SubsumedClauses[c_1]
  - Delete c_2

Data Structures

- We need to keep relevant data to undo assumptions.
- SubsumedClauses
  - Per clause – set of clauses subsumed by it

- Resol - Partial Resolution for temporary clauses
  - If at least one of the parent clauses is temporary, then a new vertex is added to the resolution graph :
    - Conflict Analyze
    - Variable Elimination
    - Self Subsumption

Self-Subsumption (Example)

Two clauses: $c_1=(a \vee b)$ $c_2=(a \vee \neg b \vee c)$

- Normal self-subsumption:
  - Create a new clause $c_3= Resol(c_1, c_2) = (a \vee c)$
  - Delete c_2
  - (Normally implemented just as removal of $\neg b$ from c_2)
- In our case (in addition to normal):
  - c_3 subsumes c_2 → Add c_2 to $SubsumedClauses[c_3]$

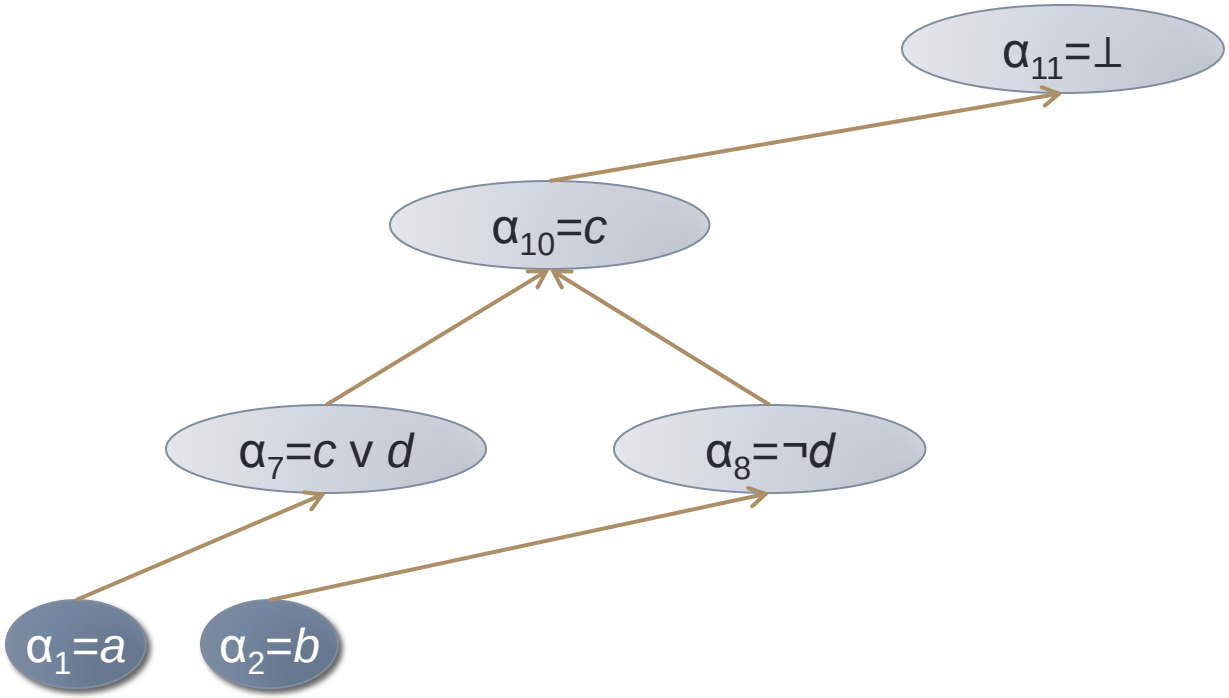
Undo Assumptions

Assumptions:

Iteration i : b, a

Iteration i+1: b

Undo Assumptions (Incremental T2P)

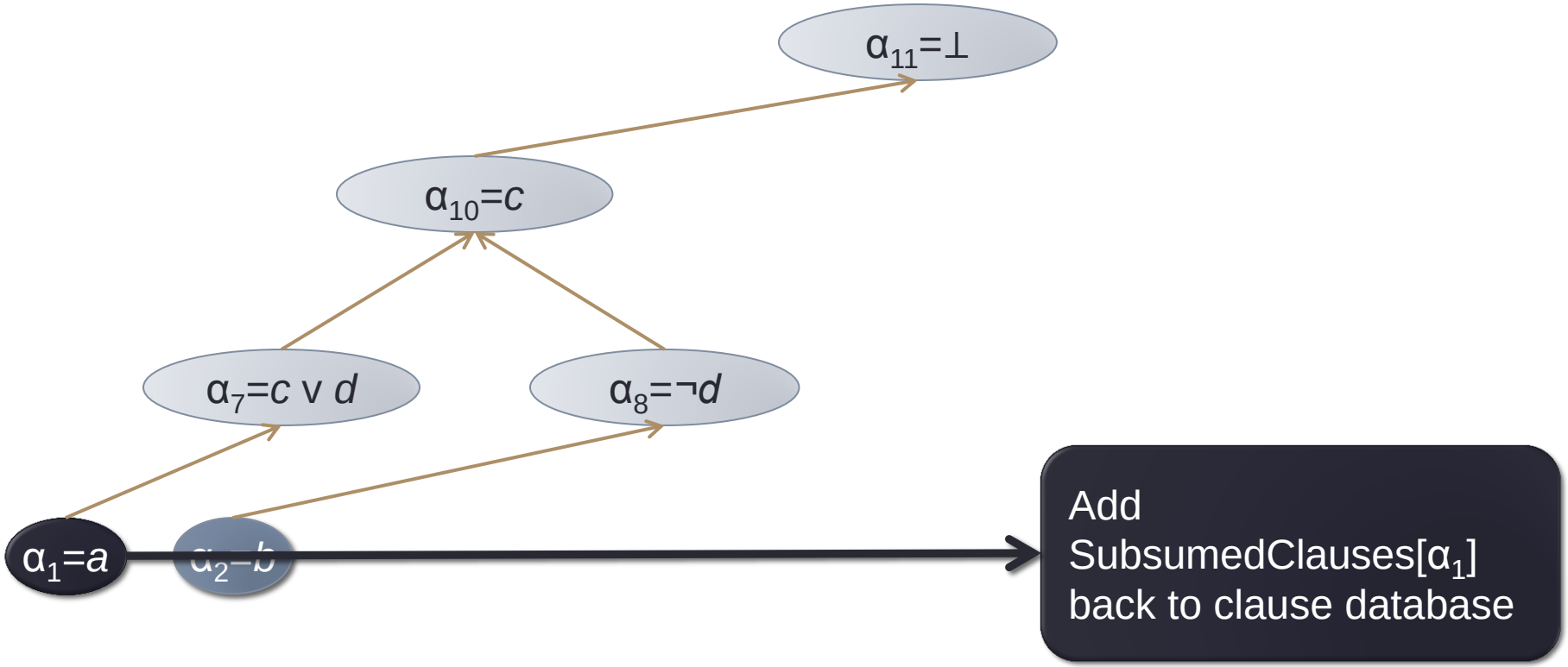


Legend:

Assumptions

Temporary conflict clauses

Undo Assumptions (Incremental T2P)

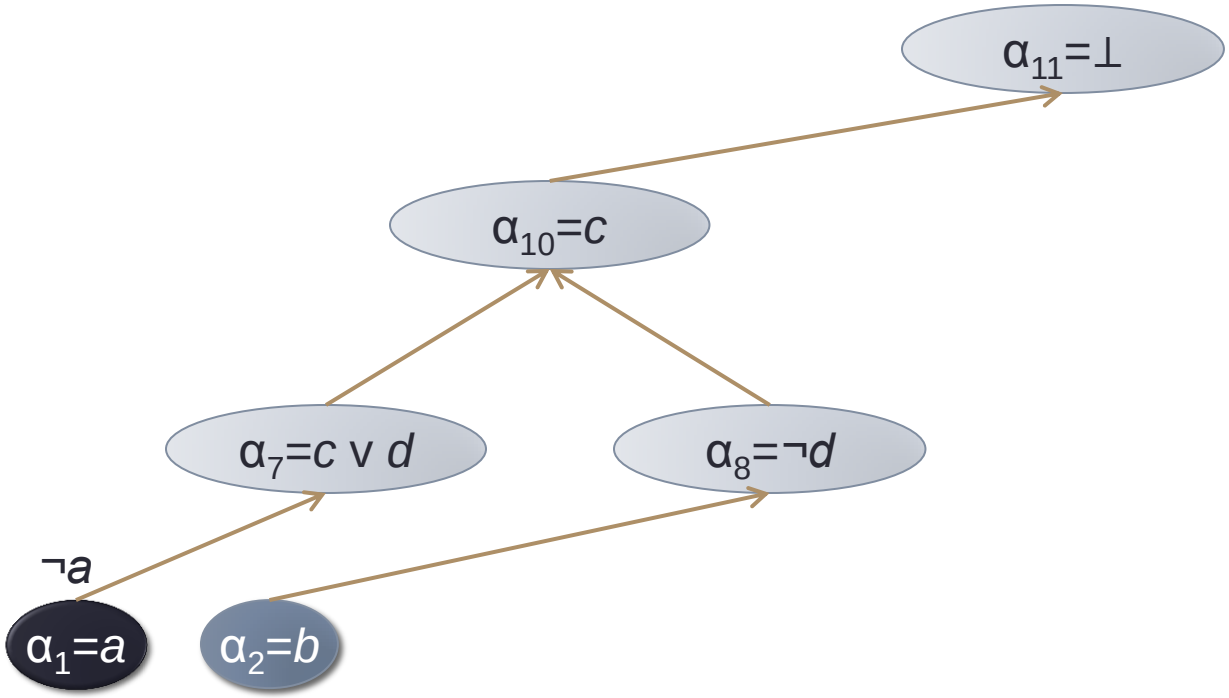


Legend:

Assumptions

Temporary conflict clauses

Undo Assumptions (Incremental T2P)

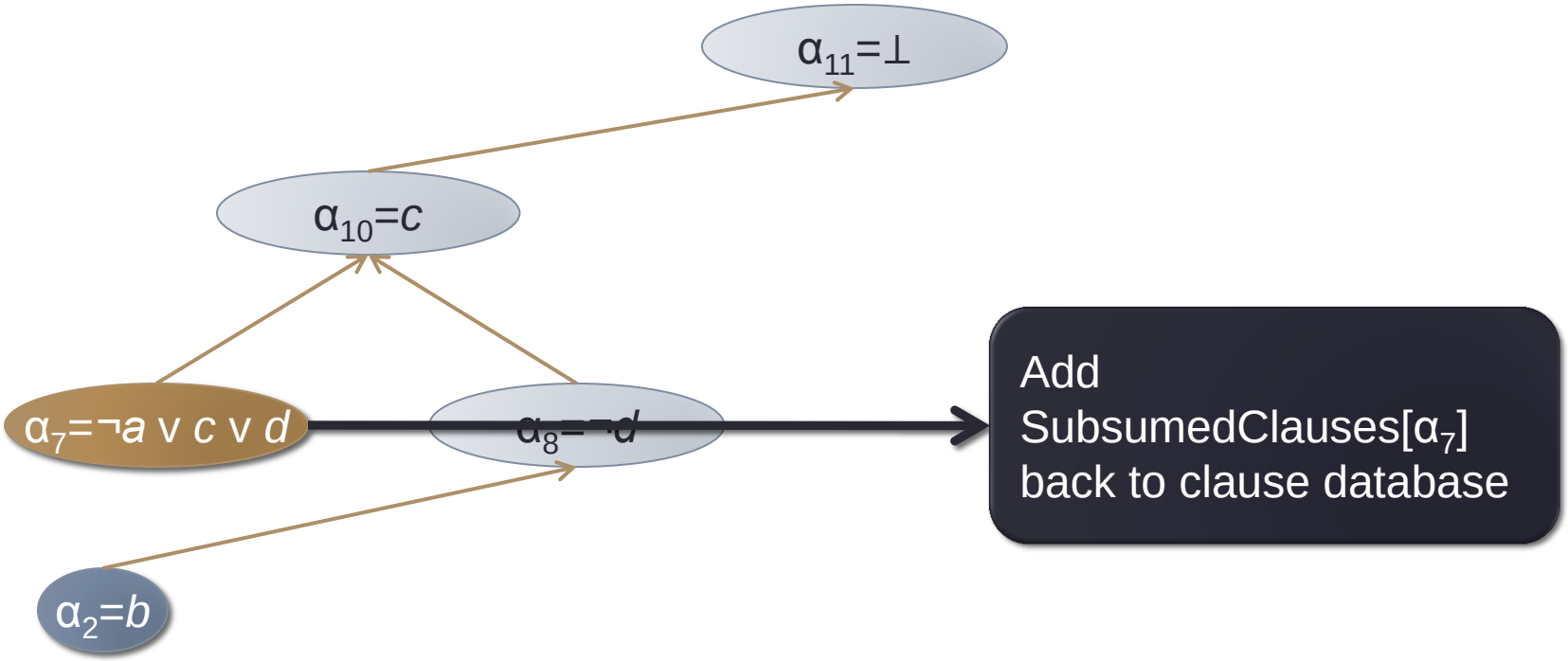


Legend:

Assumptions

Temporary conflict clauses

Undo Assumptions (Incremental T2P)



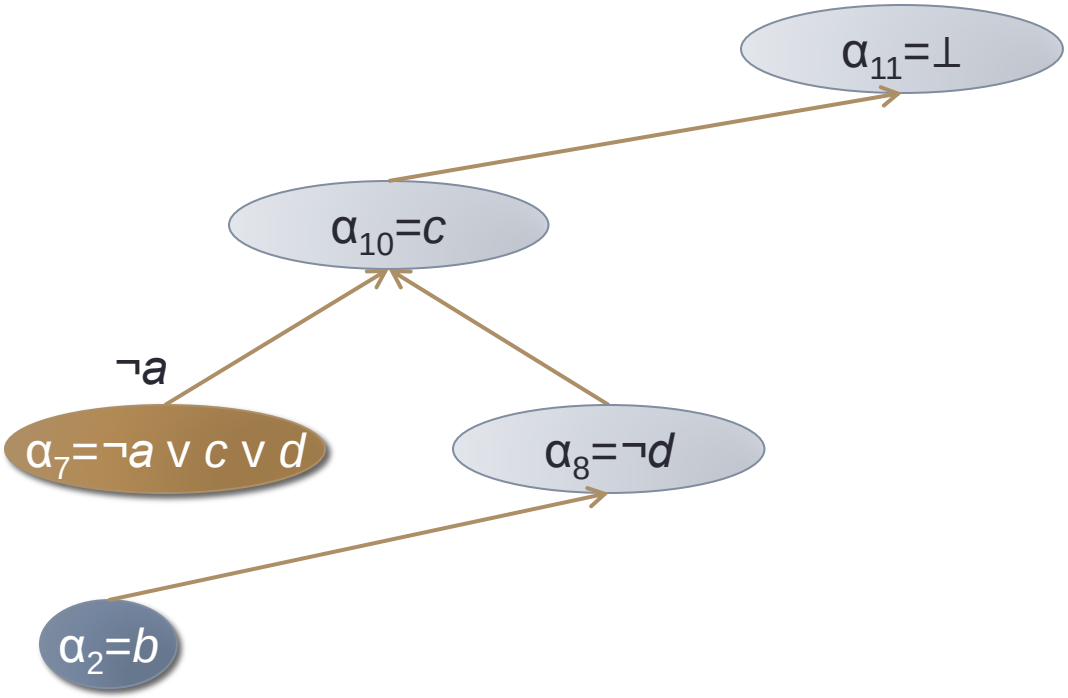
Legend:

Pervasive conflict clauses

Assumptions

Temporary conflict clauses

Undo Assumptions (Incremental T2P)



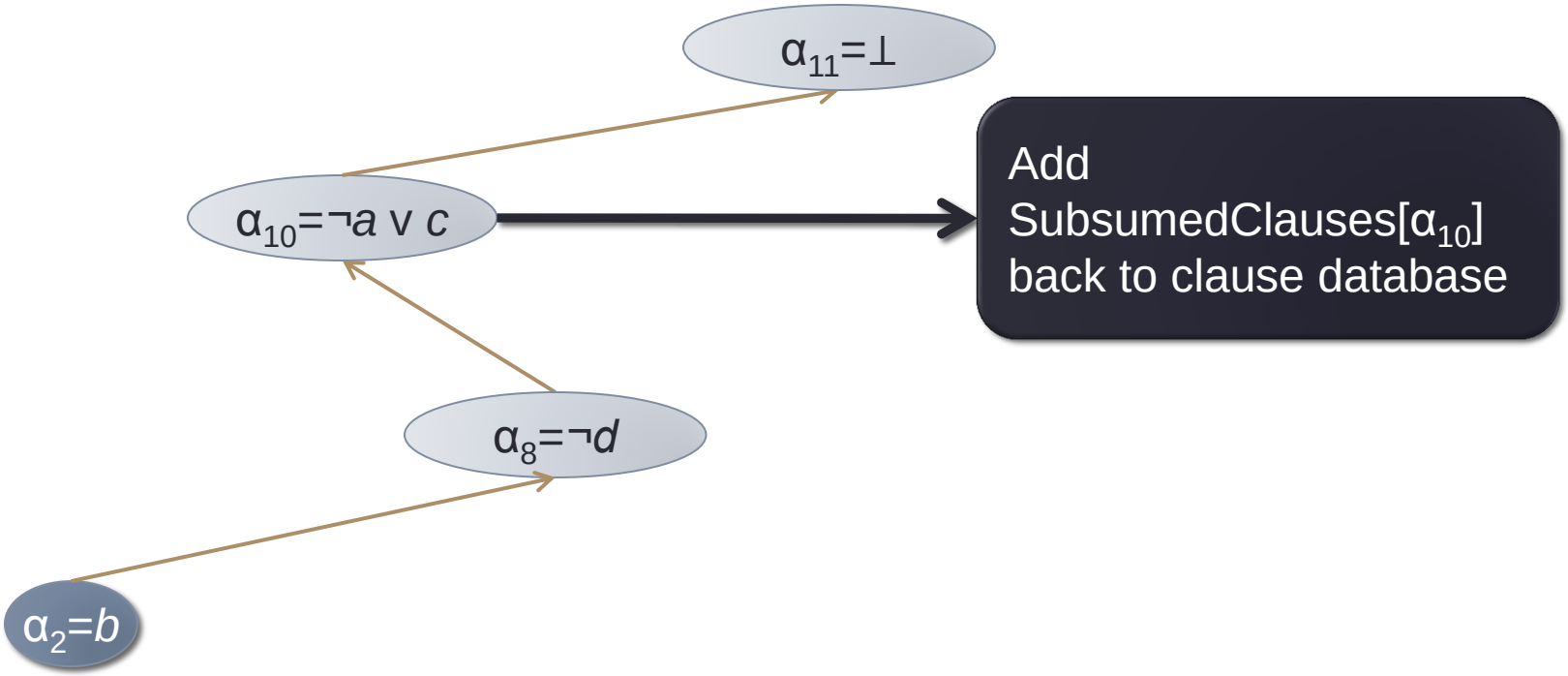
Legend:

Pervasive conflict clauses

Assumptions

Temporary conflict clauses

Undo Assumptions (Incremental T2P)

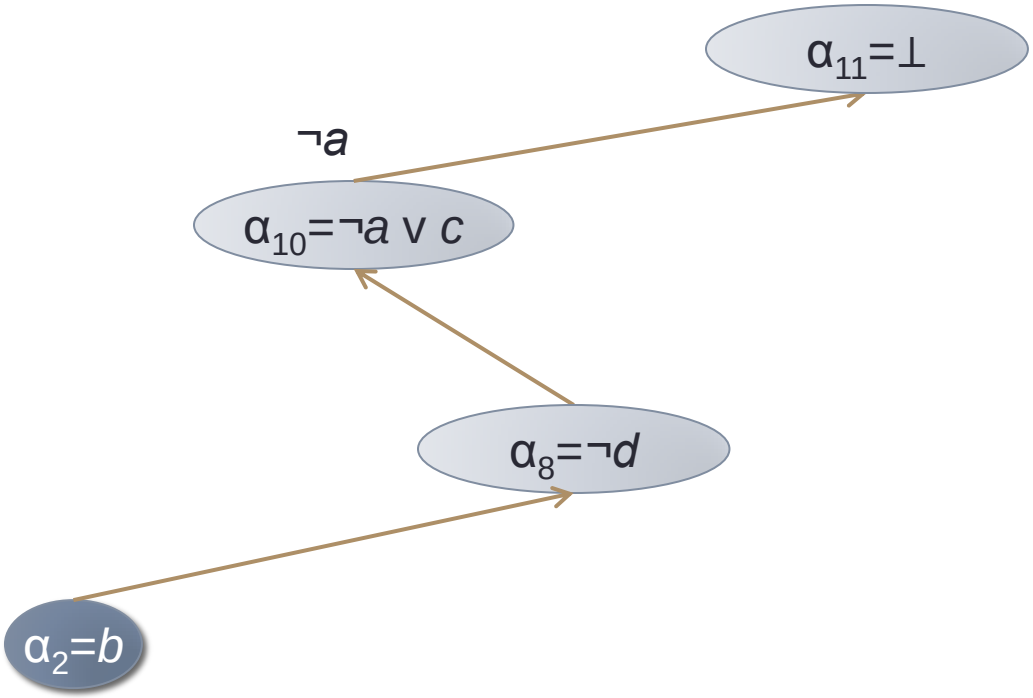


Legend:

Assumptions

Temporary conflict clauses

Undo Assumptions (Incremental T2P)

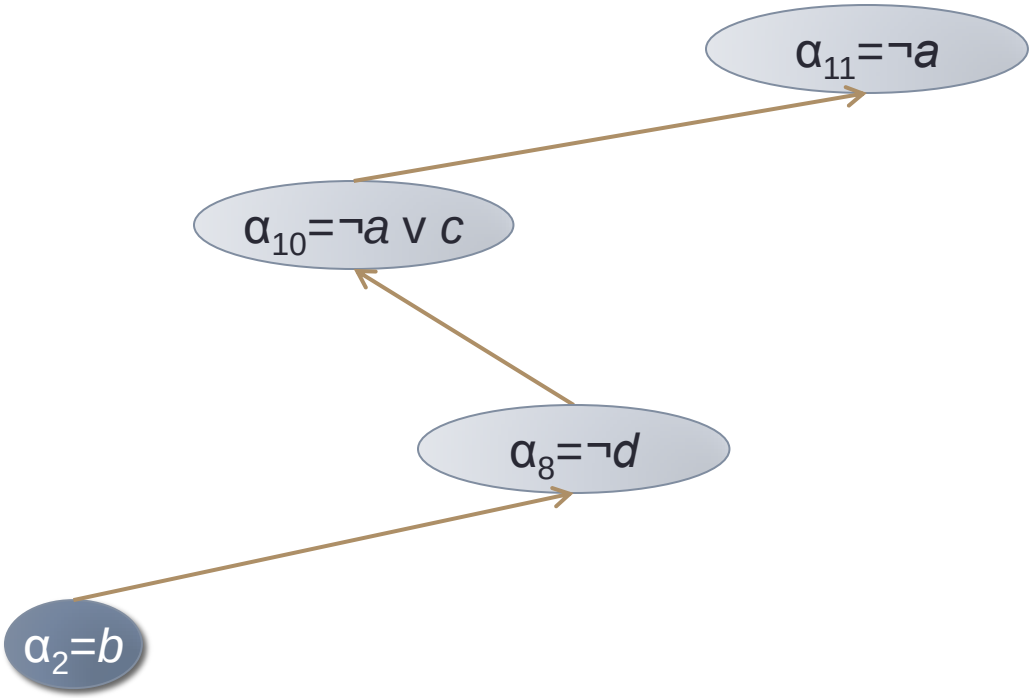


Legend:

Assumptions

Temporary conflict clauses

Undo Assumptions (Incremental T2P)



Legend:

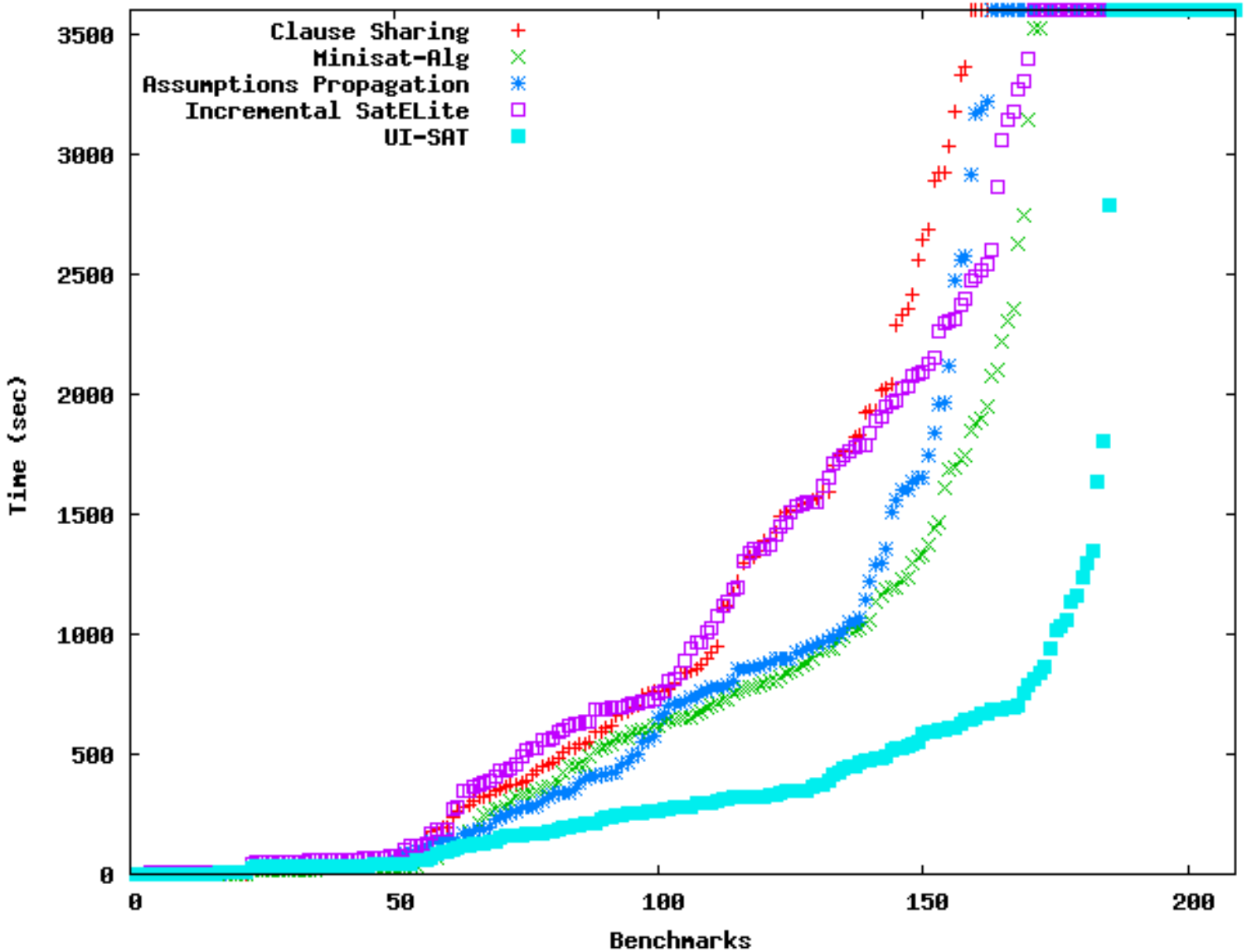
Assumptions

Temporary conflict clauses

Experimental Results

- Benchmark Set:
  - Instances generated by BMC (without look-ahead) under assumptions
    - Generated by an incremental model checker
      - May be invoked multiple times with different assumptions and properties
      - Essential to reduce the debug loop time for validation engineers
  - 3 satisfiable families – 128 instances
  - 4 unsatisfiable families – 81 instances
  - Algorithm Implementation in Intel's internal Fiver SAT Solver
  - Timeout: 3600sec
- Machines:
  - Intel® Xeon® 4Ghz 32Gb of memory

Experimental Results



Experimental Results

Method	Time-outs	Run-time
Clause Sharing	28	223,424
Minisat-Alg.	14	159,423
Assumption Prop.	24	182,530
Incremental SatELite	16	209,781
UI-SAT	**1**	**64,176**

Thank You!