SAT-based Computation of Minimal Cut Sets

Weilin Luo **Ou Wei**

Department of Computer Science
Nanjing University of Aeronautics and Astronautics, China

ISSRE'17 Toulouse, France

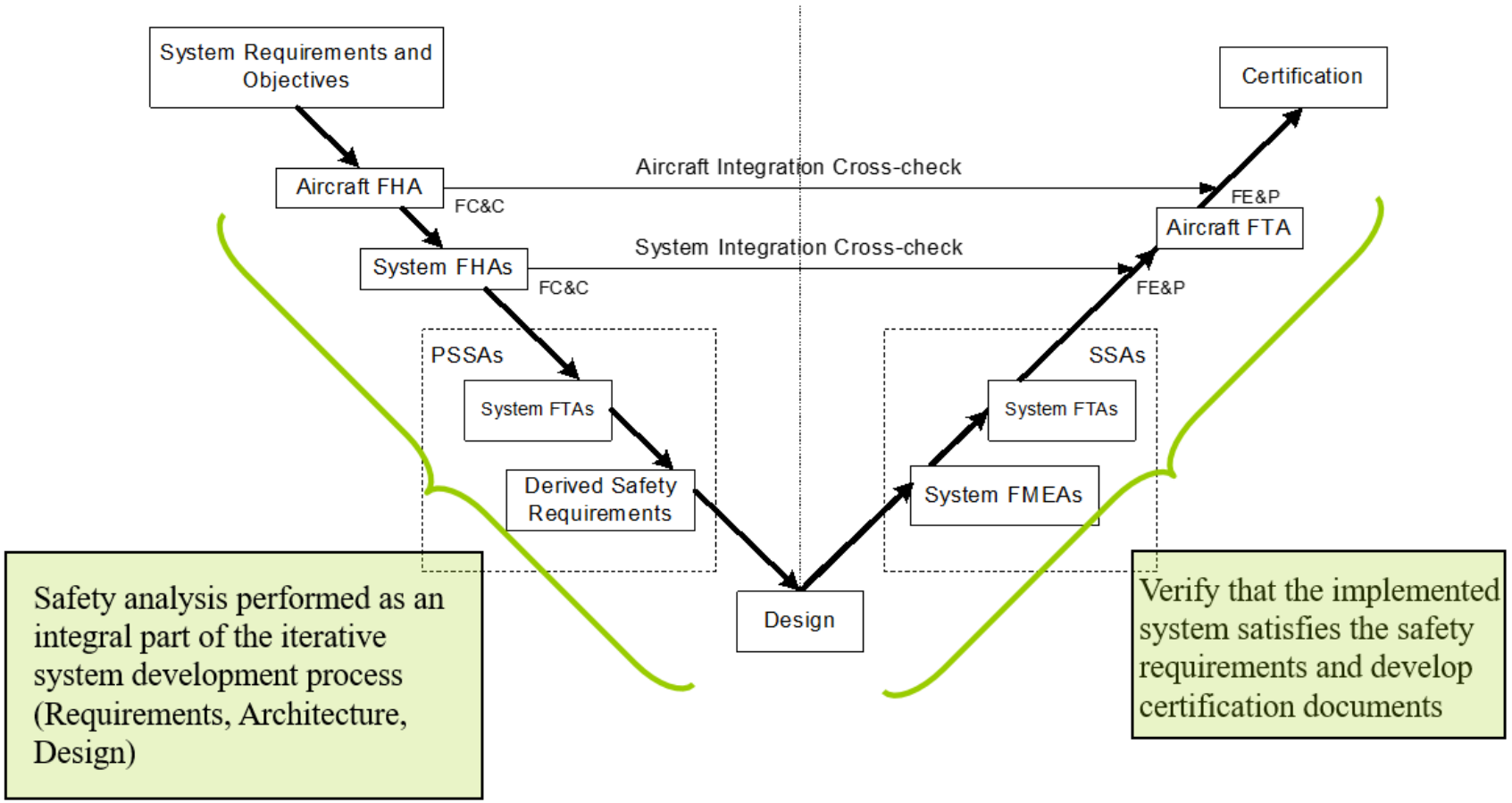
Outline

Background

- Fault Tree Analysis
- Computation of Minimal Cut Sets

Overview of FTA

- Fault Tree Analysis (FTA) is a widely-used safety analysis activity.
- Typically needed for certification of safety-critical systems (e.g., SAE ARP4761).



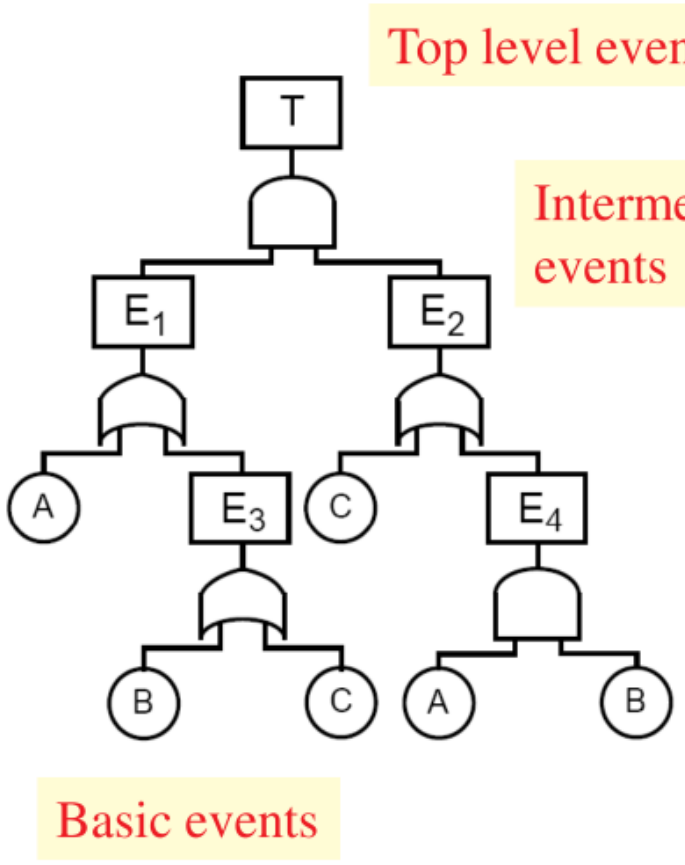
Fault Tree Basics

Fault Tree

- It specifies a top level event (TLE) representing an undesired state.
- It uses logical gates to represent the interrelationships between events and TLE, e.g. AND, OR.
- It is a systematic representation of such chains of events, i.e. cut set, that may cause the TLE to occur.

Coherent Fault Tree

- In our study, FTs are *coherent*, i.e. if a cut set S leads to a failure, then every superset S' of S also leads to failure.



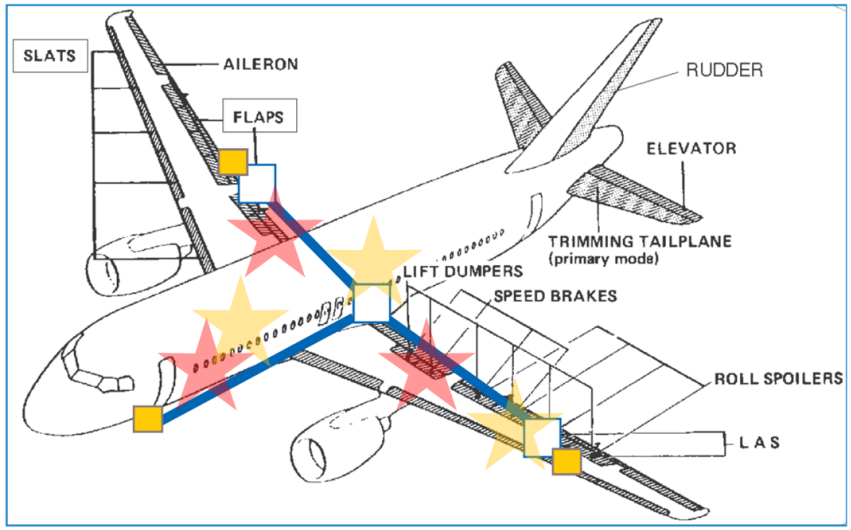
An example fault tree

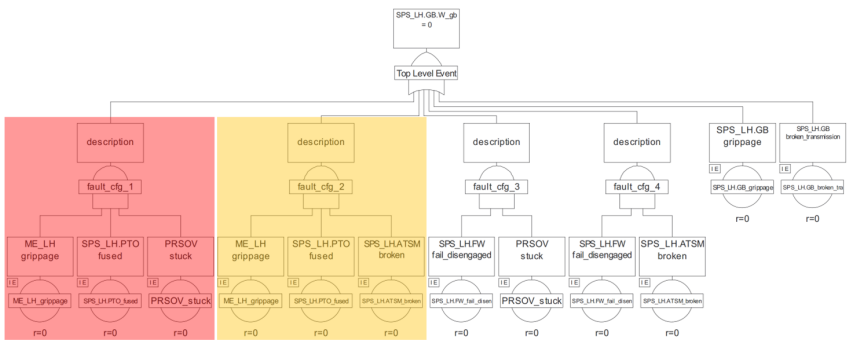
Logically: $(A \vee (B \vee C) \wedge (C \vee (A \wedge B))$

Fault Tree Basics

Why FTA is useful

- Qualitative analysis, e.g., detection of common cause failure;
- Quantitative analysis, e.g., calculation of failure probability and importance measures.
- Helps understanding the system under analysis, and reveal safety and reliability issues early in the design process.





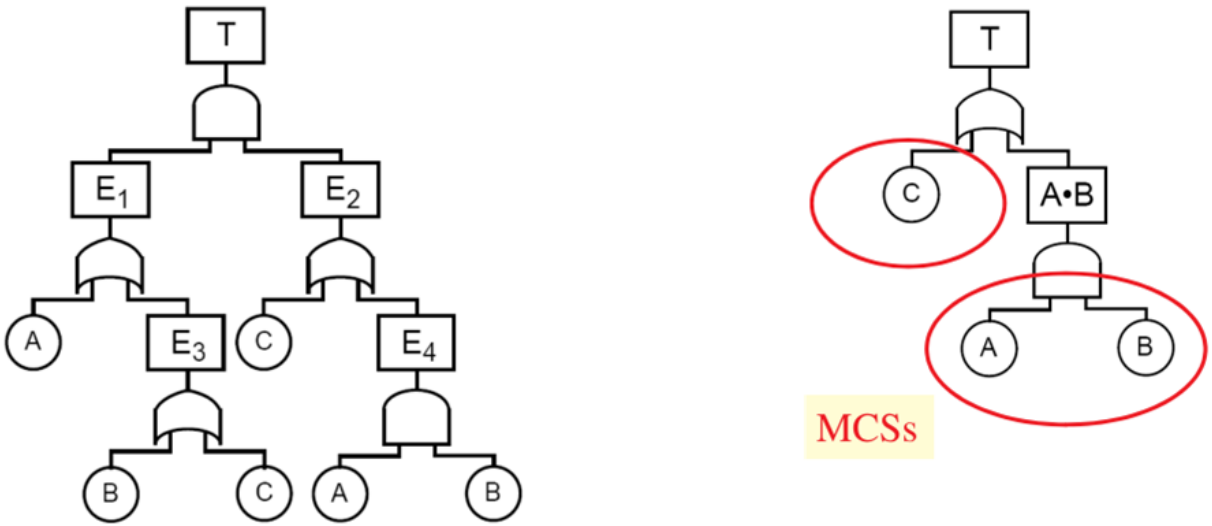
Background

- Fault Tree Analysis
- Computation of Minimal Cut Sets

Minimal cut set (MCS)

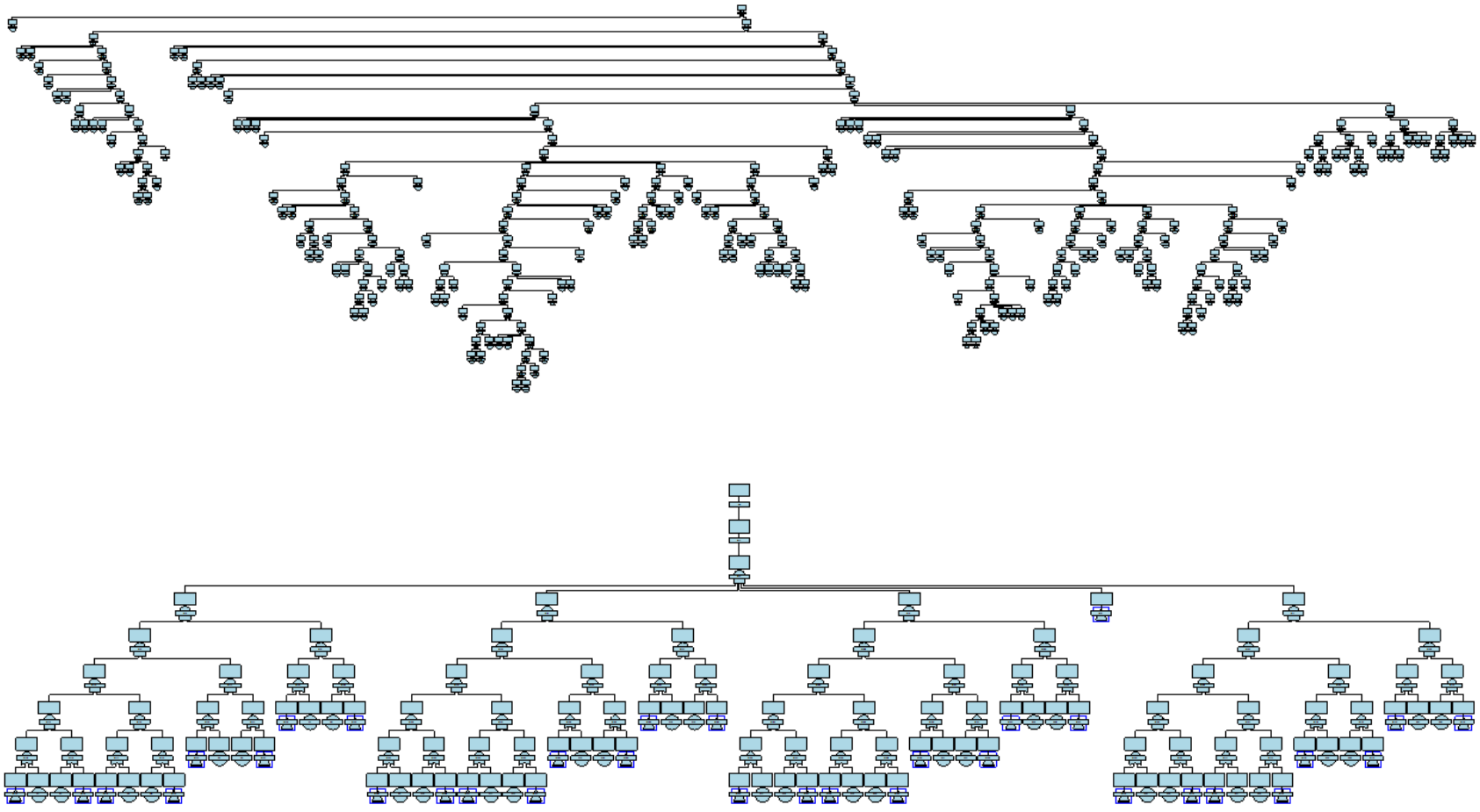
A minimal cut set (MCS) is a smallest set of basic events causing the top level event to occur.

E.g., $(A \vee (B \vee C) \wedge (C \vee (A \wedge B)) \equiv (C \vee (A \wedge B)$



A Hard Problem

Computing MCSs of a fault tree is a problem that is exponential with respect to the number of basic events in the tree.



Traditional Methods for Computing MCSs

- Based on Boolean Manipulation
- Based on Binary Decision Diagrams (BDDs)

Based on Boolean Manipulation

Using bottom-up or top-down algorithms. [J. B. Fussell and W. E. Vesely, 1972]

- These represent each gate as a Boolean expression;
- These expressions are simplified into an expression that relates the top event to the DNF without any gates;

Tool

- MOCUS
- MICSUP
- FaultTree+

Based on Binary Decision Diagrams (BDDs)

Representing a fault tree with a BDD. [O. Coudert and J. C. Madre, 1993; A. Rauzy and Y. Dutuit, 1997]

- The size of the BDD can be greatly reduced based on an appropriate variable ordering;
- Efficient minimization algorithms can be applied to the BDD to compute MCSs;

Tool

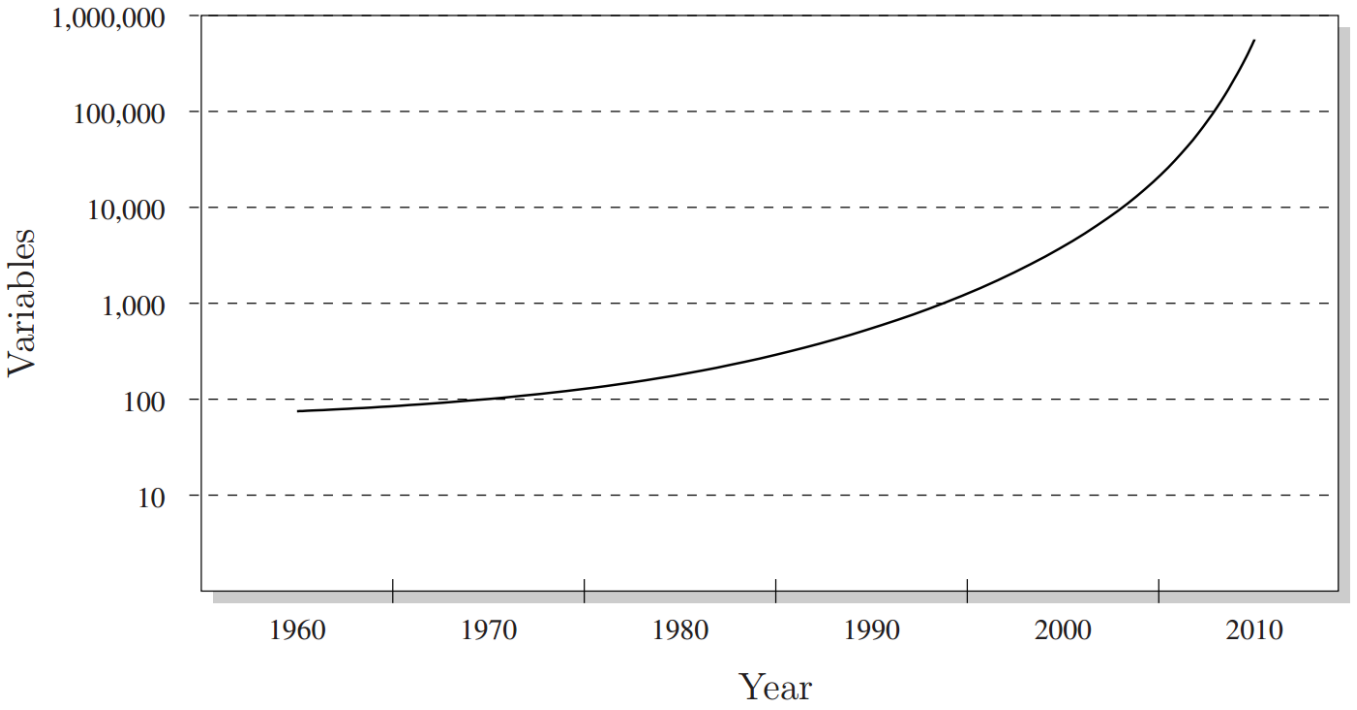
- MetaPrime
- Aralia
- CAFTA

Our Work

- Overview of SATMCS
- Tseitin Encoding of Fault Trees
- Extracting MCSs using Local Fault Graph
- Clause Learning with Dynamic Deletion
- Experiments

Motivation

- Developing new methods over different paradigms remains to be an interesting research direction.
- Recent progress on modern solver of Boolean Satisfiability Problem (SAT) based on the Davis-Putnam-Loveland-Logemann (DPLL) framework - a search algorithm, unlike the algorithms based on Boolean manipulation and BDDs.



Goal and Features

Goal

We extend the DPLL framework, used by most successful complete SAT solvers, with the ability of computing MCSs.

Goal and Features

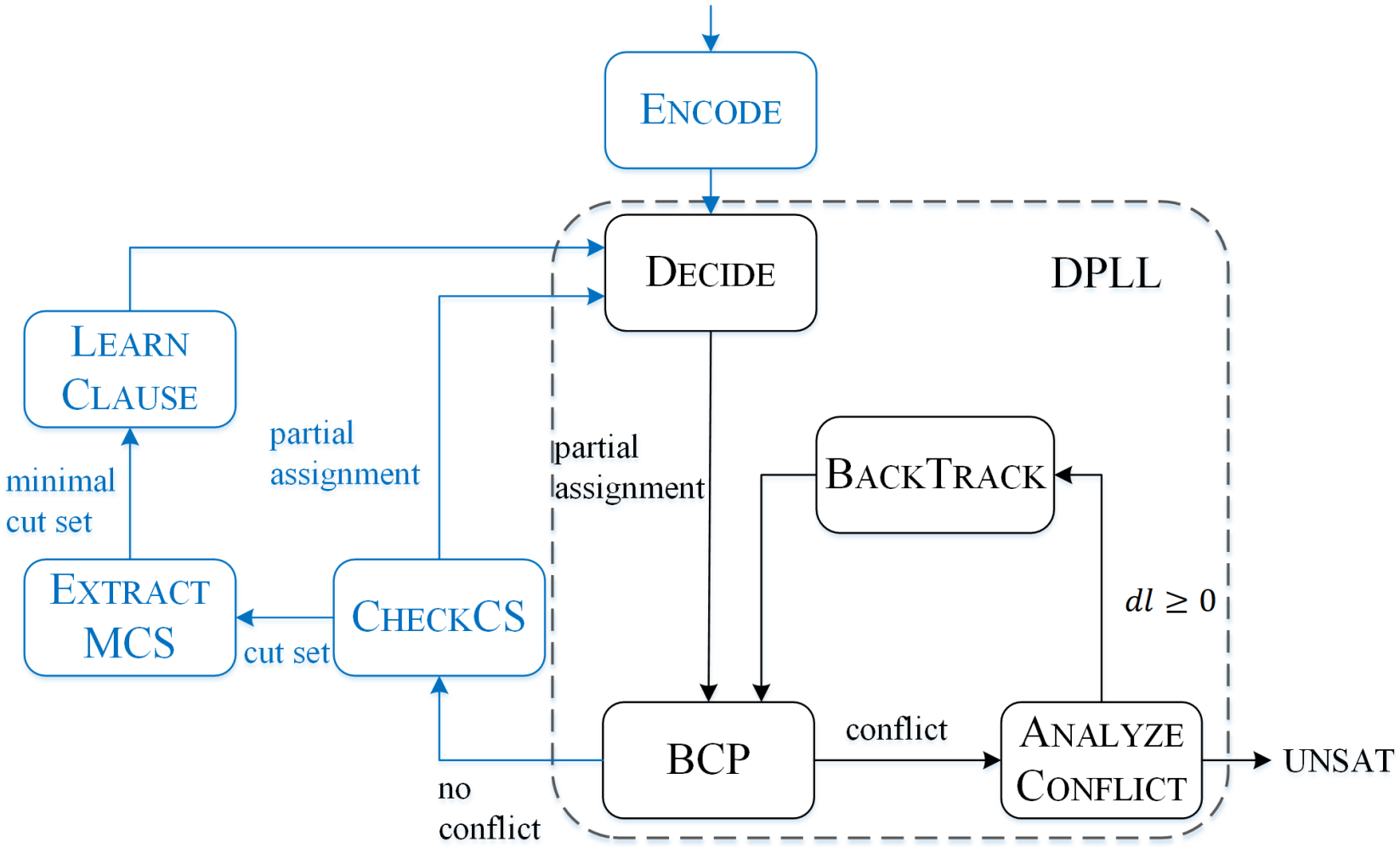
Goal

We extend the DPLL framework, used by most successful complete SAT solvers, with the ability of computing MCSs.

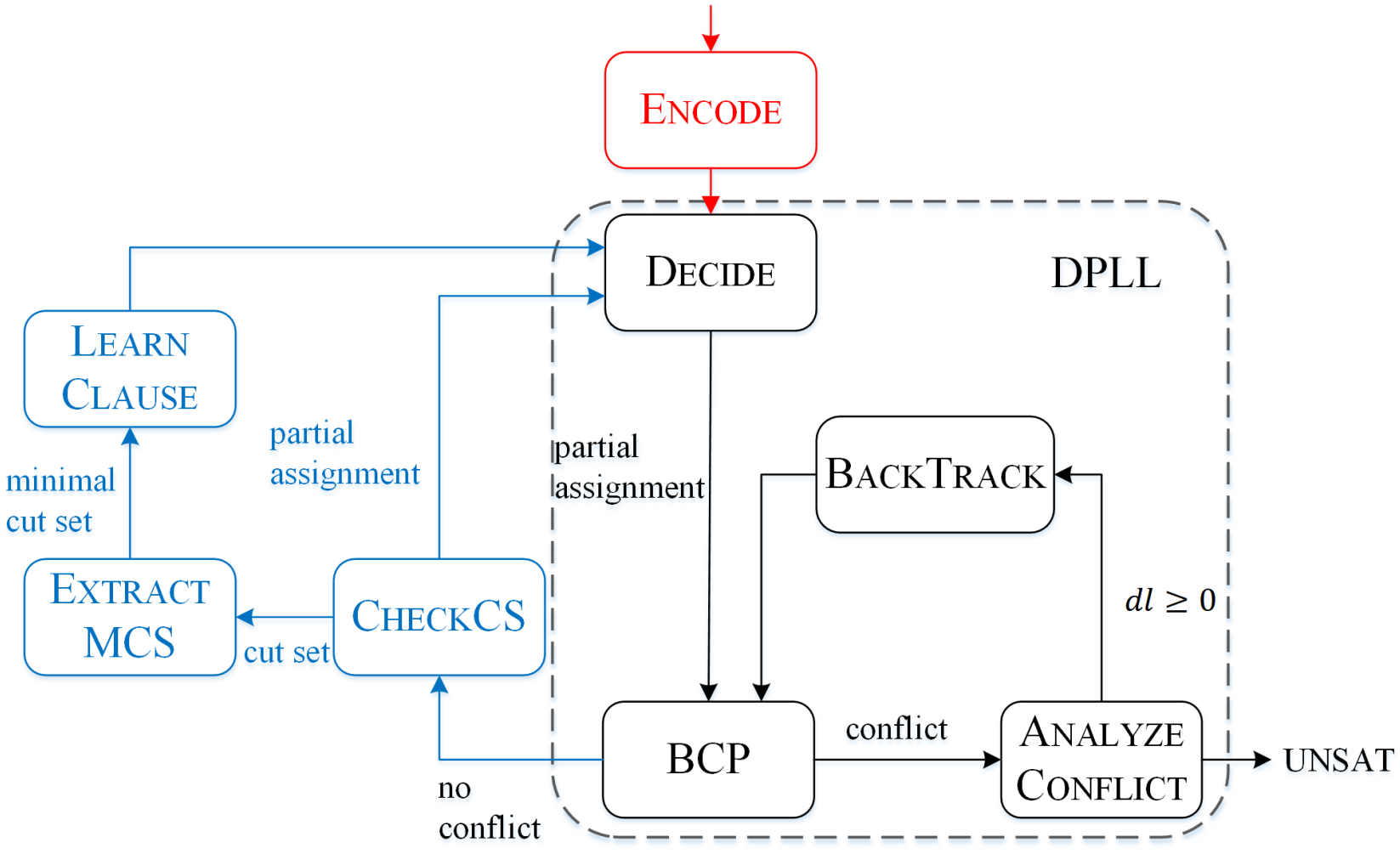
Features

- *Encoding:* DPLL requires a propositional formula presented in CNF for efficient search. The structure function of a fault tree, however, typically is a non-clausal formula.
- *ExtractMCS:* Extracting an MCS from a cut set is a key step with the goal of removing unnecessary basic events.
- *ClauseLearning:* To find all the MCSs, information of previous computed MCSs are learned as blocking clauses that prevent the same MCSs being computed again.

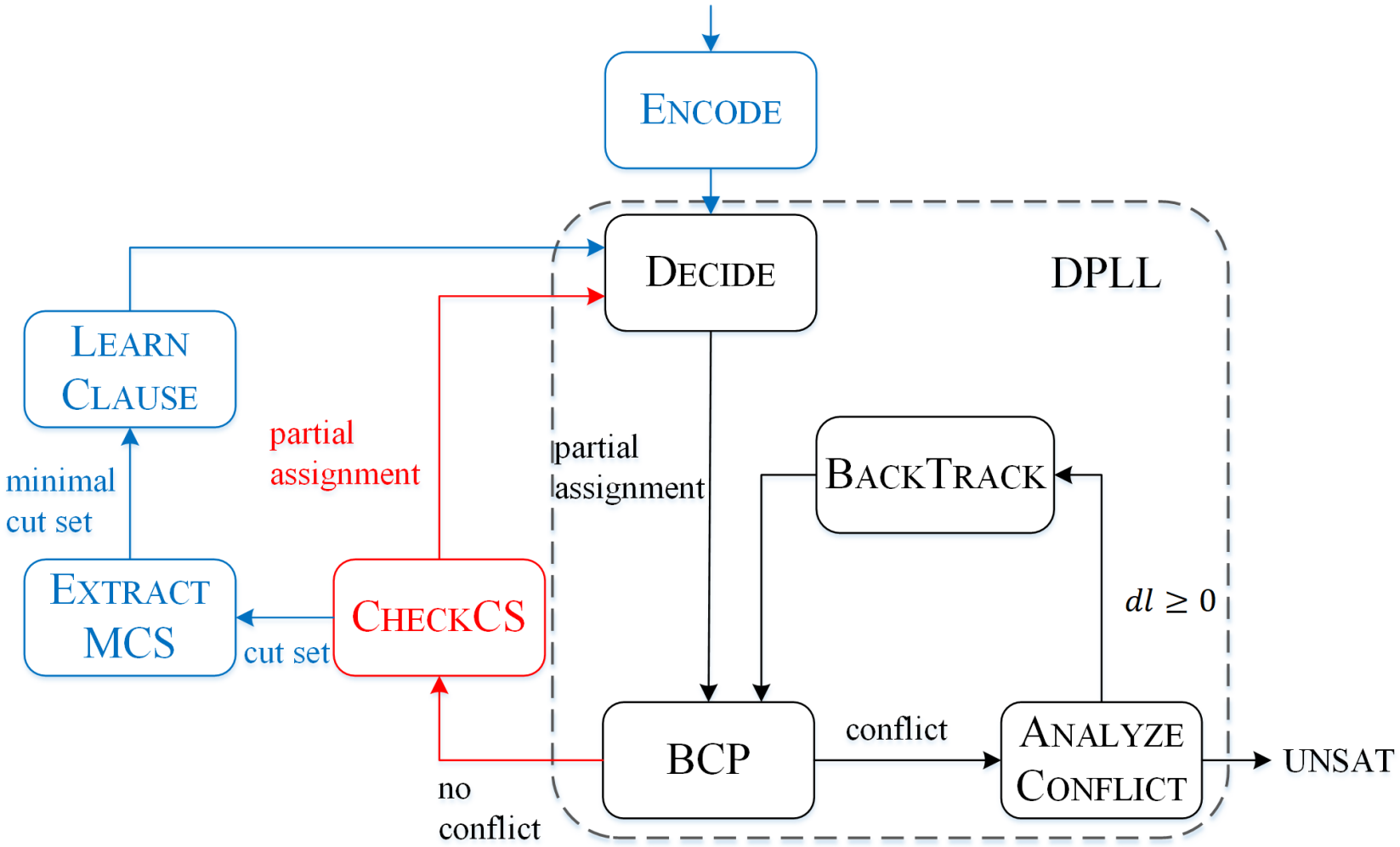
SATMCS Framework



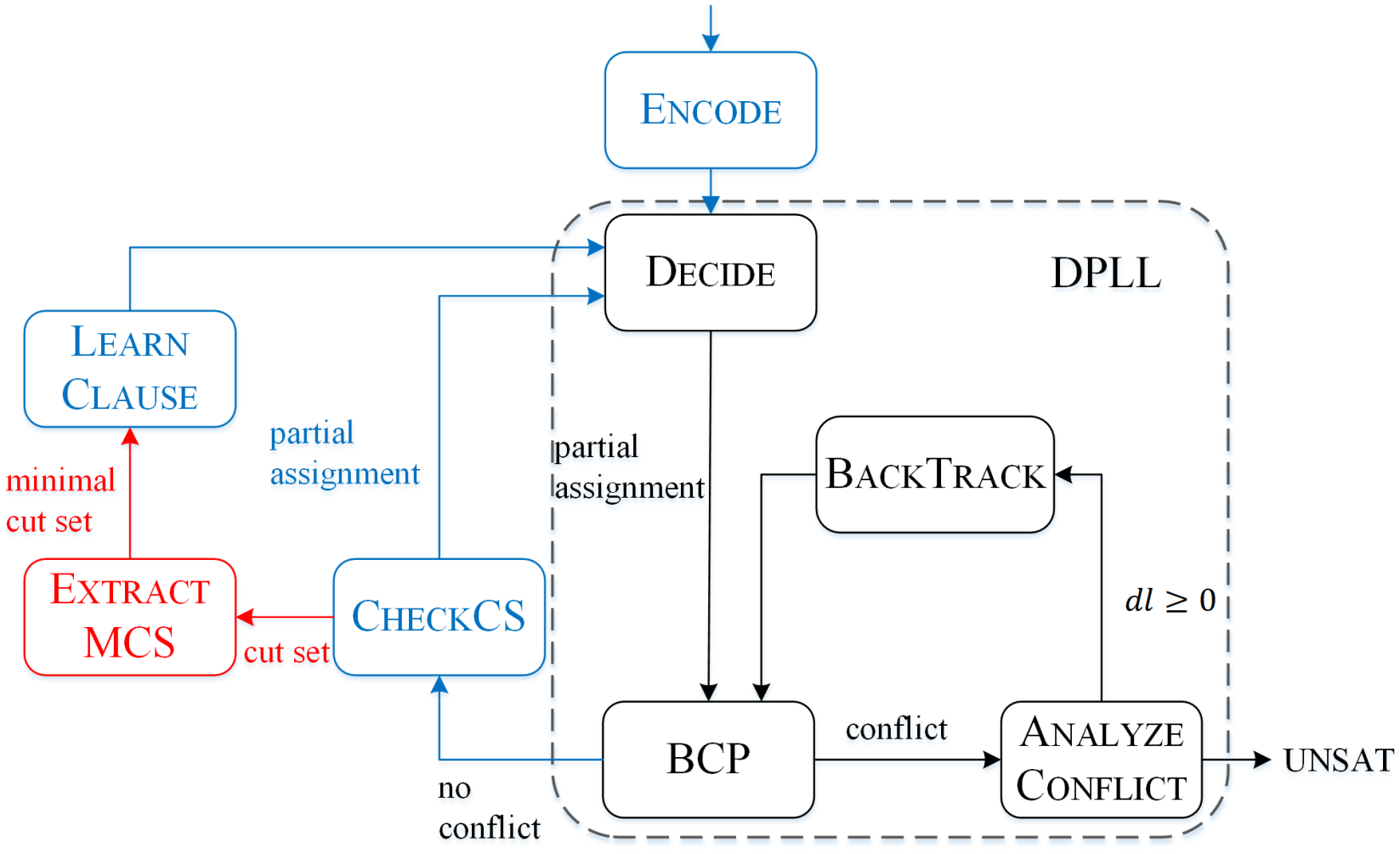
SATMCS Framework



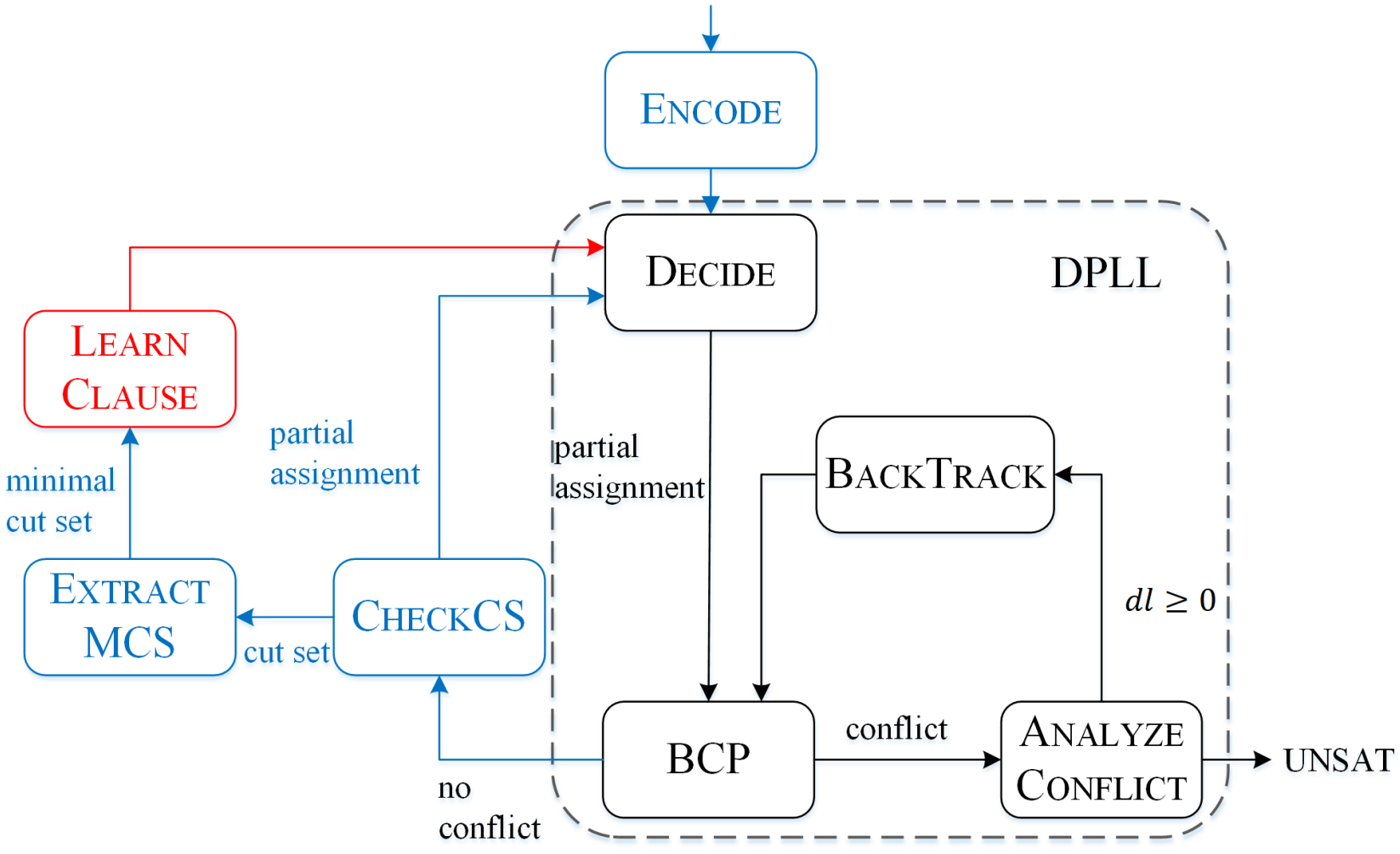
SATMCS Framework



SATMCS Framework



SATMCS Framework



Our Work

- Overview of SATMCS
- Tseitin Encoding of Fault Trees
- Extracting MCSs using Local Fault Graph
- Clause Learning with Dynamic Deletion
- Experiments

Tseitin Encoding

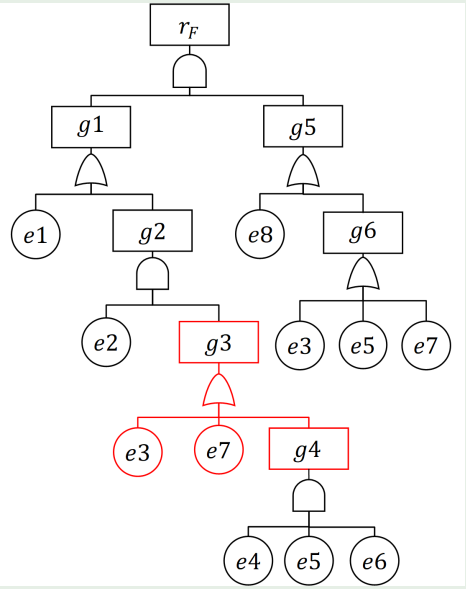
In our work, given the set of boolean equations $\mathcal{Q}$ of F, we directly encode F as the CNF formula $\mathcal{F}$ using Tseitin encoding. [G. Tseitin, 1968]

Coding

- For each boolean equation $Q_{\hat{g}} = \langle Op, \{x_1, \ldots, x_n\}, g \rangle$, $Q_{\hat{g}}$ is transformed to a set of clauses $U_{\hat{g}} = L_{\hat{g}} \cup R_{\hat{g}}$, where $L_{\hat{g}}$ denotes the clauses transformed from the formula $g \Rightarrow Op(\{x_1, \ldots, x_n\})$, and $R_{\hat{g}}$ – from the formula $g \Leftarrow Op(\{x_1, \ldots, x_n\})$.
  - If Op is OR, $L_{\hat{g}} = \{\neg g \vee \bigvee_{i=1}^{n} x_i\}$ and $R_{\hat{g}} = \bigcup_{i=1}^{n} \{\neg x_i \vee g\}$;
  - If Op is AND, $L_{\hat{g}} = \bigcup_{i=1}^{n} \{\neg g \vee x_i\}$ and $R_{\hat{g}} = \{g \vee \bigvee_{i=1}^{n} \neg x_i\}$;
- $\mathcal{F}$ is then defined by $\bigcup_{g \in G_F} U_{\hat{g}} \cup \{r_F\}$.

Tseitin Encoding

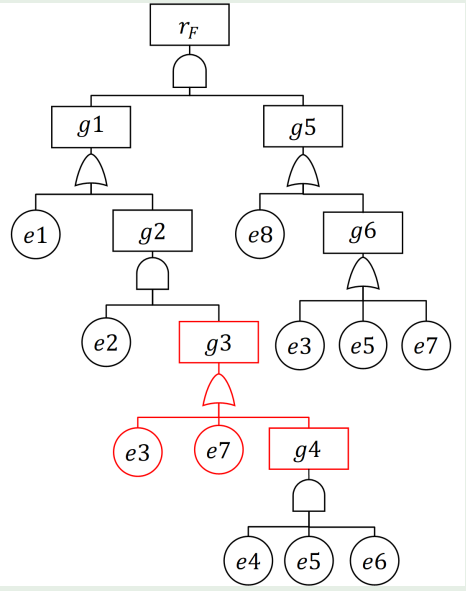
Example



$$Q_{\hat{g3}} = \langle \mathrm{OR}, \{e3, e7, g4\}, g3 \rangle$$

Tseitin Encoding

Example



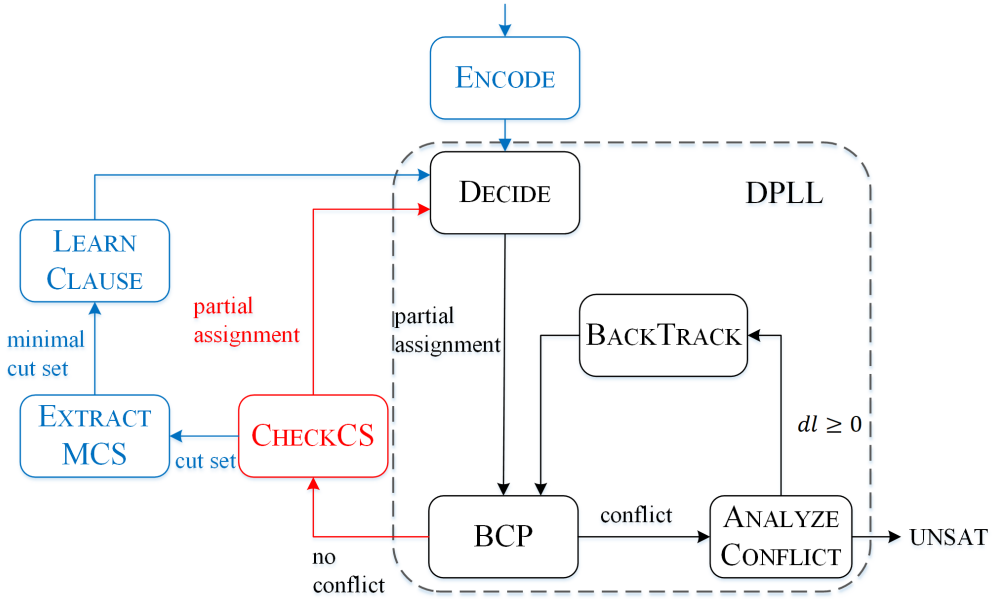
$$Q_{\hat{g3}} = \langle \mathrm{OR}, \{e3, e7, g4\}, g3 \rangle$$

1. $L_{\hat{g3}} = \{\neg g3 \vee e3 \vee e7 \vee g4\}$
2. $R_{\hat{g3}} = \{\{\neg e3 \vee g3\}, \{\neg e7 \vee g3\}, \{\neg g4 \vee g3\}\}$
3. $U_{\hat{g3}} = L_{\hat{g3}} \cup R_{\hat{g3}}$

Our Work

- Overview of SATMCS
- Tseitin Encoding of Fault Trees
- Extracting MCSs using Local Fault Graph
- Clause Learning with Dynamic Deletion
- Experiments

Check CS



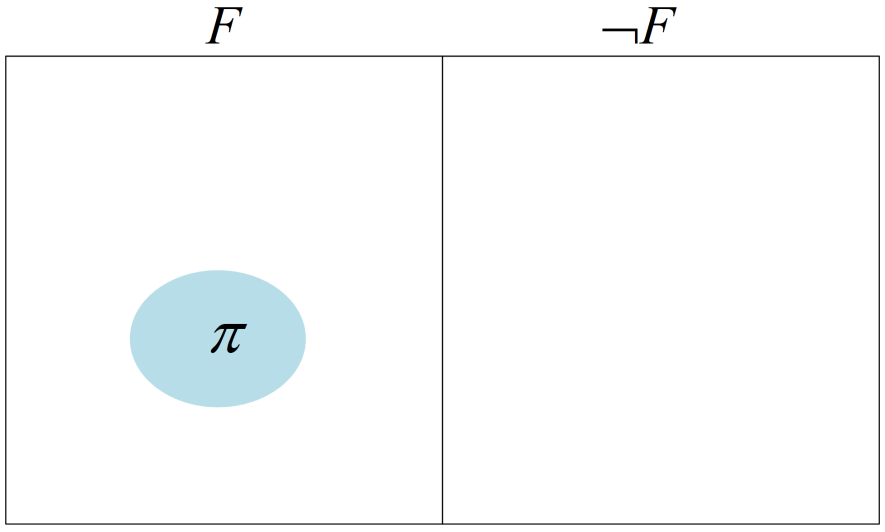
Based on the monotonic satisfiability of CNF

- keeping a list for the clauses in $\mathcal{F}$, and watching the first unresolved clause in the list for each decision level in DPLL.
- checking whether the new assignments in *dl* can satisfy more clauses than that in the previous decision level.
- updating the first unresolved clause for decision level until all the clauses are satisfied.

Extract MCS

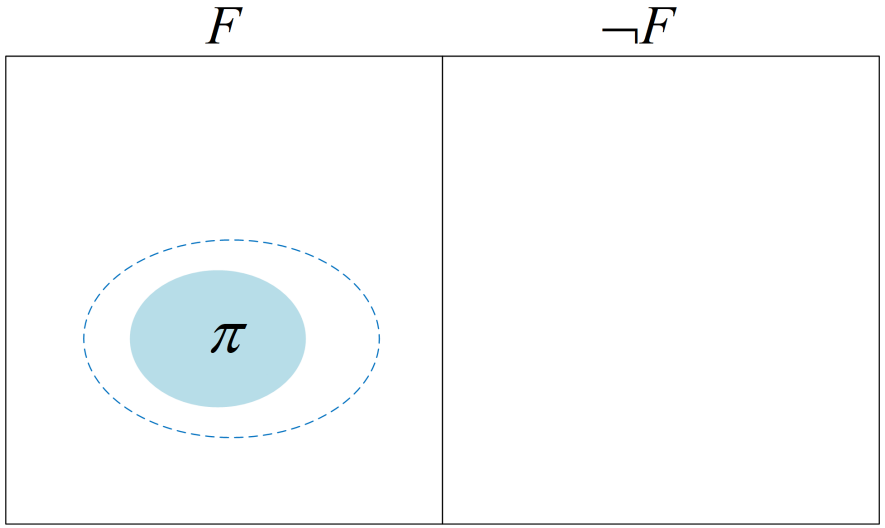
- π is identified as a cut set of F.
- EXTRACTMCS conducts a sequence of queries on F.
- The query tests whether $\pi' \wedge \neg F$ can be satisfied.

$$\pi = \{e1, e2, e3\}$$



Extract MCS

- π is identified as a cut set of F.
- EXTRACTMCS conducts a sequence of queries on F.
- The query tests whether $\pi' \wedge \neg F$ can be satisfied.

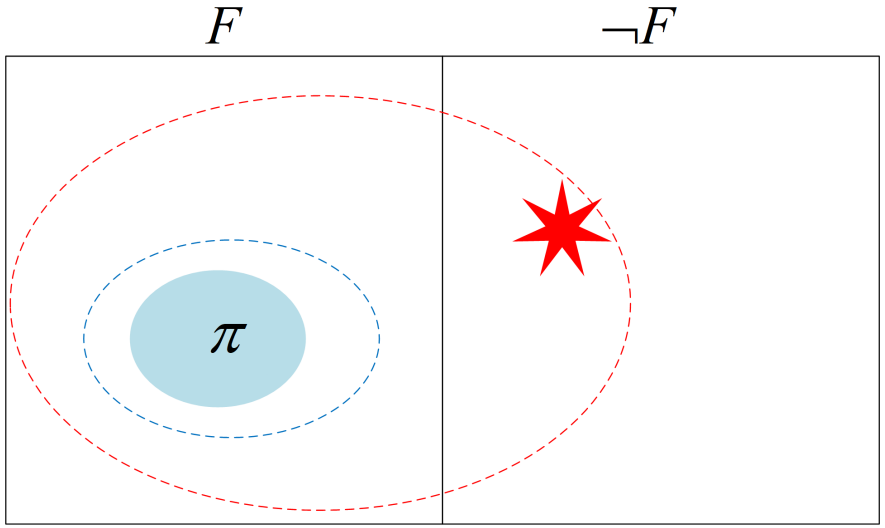


$\pi = \{e1, e2, e3\}$

1. CHECK $e1$: $\pi' = \{e2, e3\} \Rightarrow$ $\pi = \{e2, e3\}$

Extract MCS

- π is identified as a cut set of F.
- ExtractMCS conducts a sequence of queries on F.
- The query tests whether $\pi' \wedge \neg F$ can be satisfied.

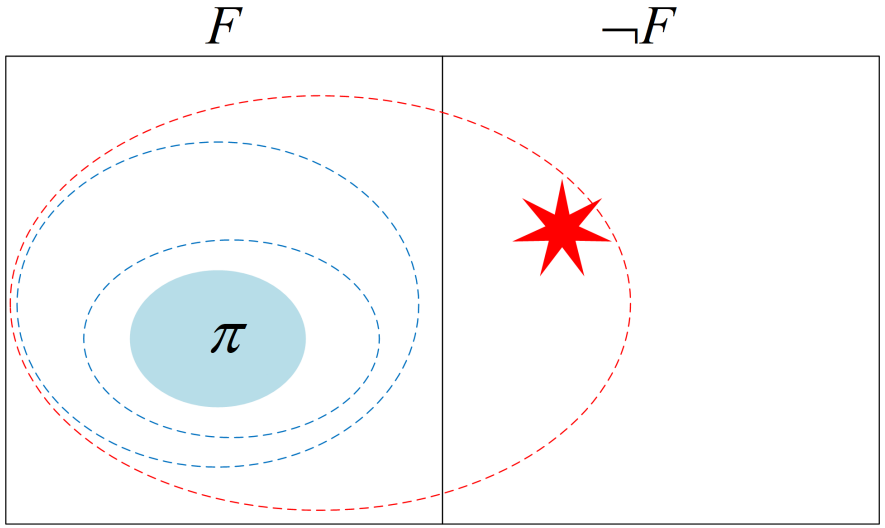


$\pi = \{e1, e2, e3\}$

1. CHECK $e1$: $\pi' = \{e2, e3\} \Rightarrow \pi = \{e2, e3\}$
2. CHECK $e2$: $\pi' = \{e3\} \Rightarrow \pi = \{e2, e3\}$

Extract MCS

- π is identified as a cut set of F.
- ExtractMCS conducts a sequence of queries on F.
- The query tests whether $\pi' \wedge \neg F$ can be satisfied.

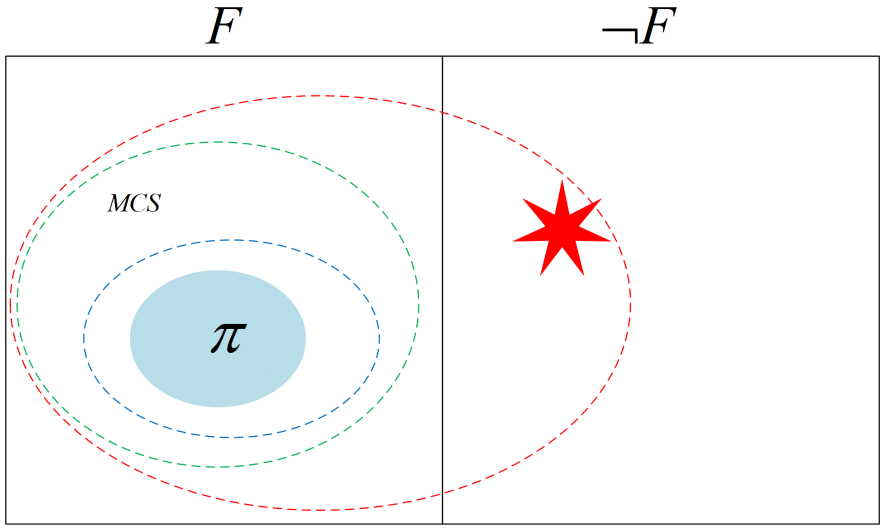


$\pi = \{e1, e2, e3\}$

1. CHECK $e1$: $\pi' = \{e2, e3\} \Rightarrow \pi = \{e2, e3\}$
2. CHECK $e2$: $\pi' = \{e3\} \Rightarrow \pi = \{e2, e3\}$
3. CHECK $e3$: $\pi' = \{e2\} \Rightarrow \pi = \{e2\}$

Extract MCS

- π is identified as a cut set of F.
- ExtractMCS conducts a sequence of queries on F.
- The query tests whether $\pi' \wedge \neg F$ can be satisfied.



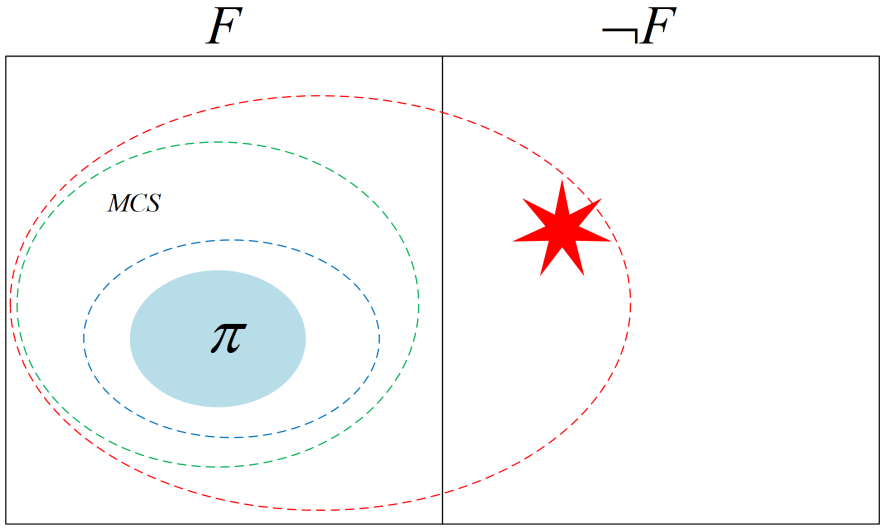
$\pi = \{e1, e2, e3\}$

1. CHECK $e1$: $\pi' = \{e2, e3\} \Rightarrow \pi = \{e2, e3\}$
2. CHECK $e2$: $\pi' = \{e3\} \Rightarrow \pi = \{e2, e3\}$
3. CHECK $e3$: $\pi' = \{e2\} \Rightarrow \pi = \{e2\}$

MCS is $\{e2\}$

Extract MCS

- π is identified as a cut set of F.
- ExtractMCS conducts a sequence of queries on F.
- The query tests whether $\pi' \wedge \neg F$ can be satisfied.



$\pi = \{e1, e2, e3\}$

1. CHECK $e1$: $\pi' = \{e2, e3\} \Rightarrow \pi = \{e2, e3\}$
2. CHECK $e2$: $\pi' = \{e3\} \Rightarrow \pi = \{e2, e3\}$
3. CHECK $e3$: $\pi' = \{e2\} \Rightarrow \pi = \{e2\}$

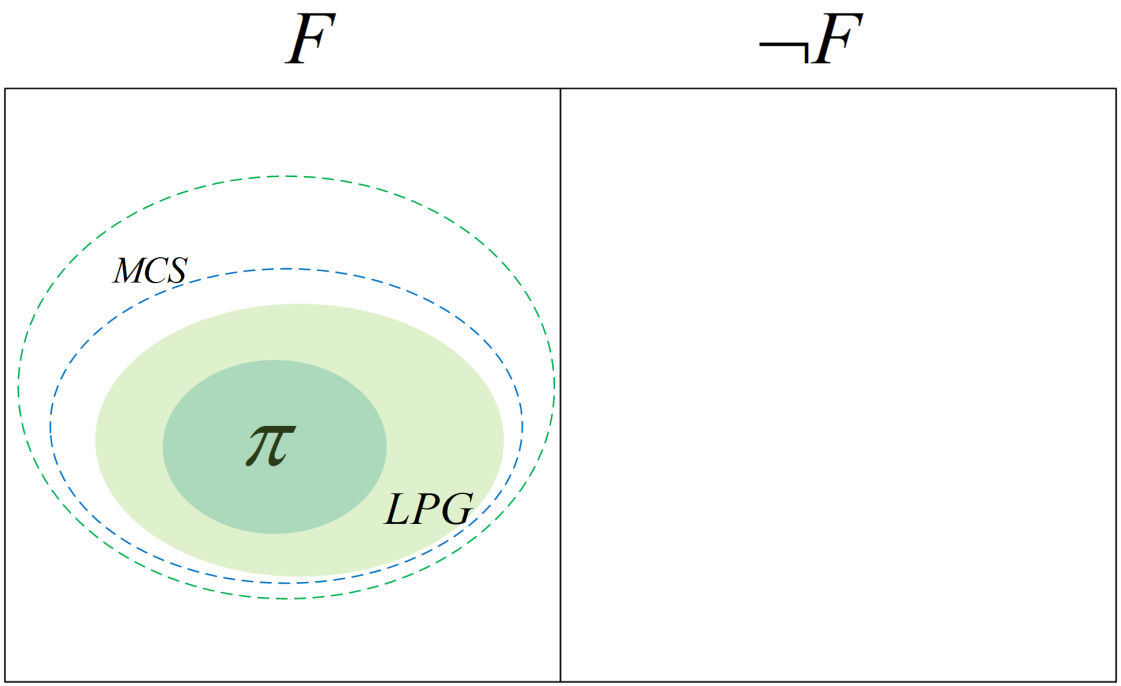
MCS is $\{e2\}$

- To be more efficient...

Extract MCS

We exploit the structures of F, providing an incremental extraction algorithm based on a *Local Propagation Graph* (LPG) induced by a cut set.

An LPG can be considered as a projection of F over a cut set π, characterizing the failure propagation between the corresponding events.



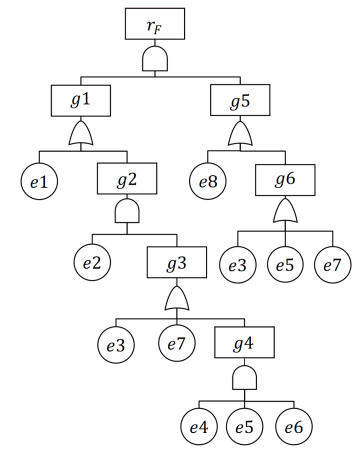
Extract MCS

Build LPG

- The LPG over π^+ is a directed hyper-graph $P_{\pi^+} = \langle V, R \rangle$, where $V = \pi^+$ is a set of events and $R \subseteq 2^V \times V$ is a set of hyper-edges.
- An edge $(S, v) \in R$ denotes that occurrence of all the events in S causes the event v to occur.
  - if there exists a $\langle \text{AND}, In, v \rangle \in \mathcal{Q}$, add (In, v) into R;
  - if there exists a $v' \in \pi^+$ and a $\langle \text{OR}, In, v' \rangle \in \mathcal{Q}$ such that $v \in In$, then $(\{v\}, v')$ into R.

Extract MCS

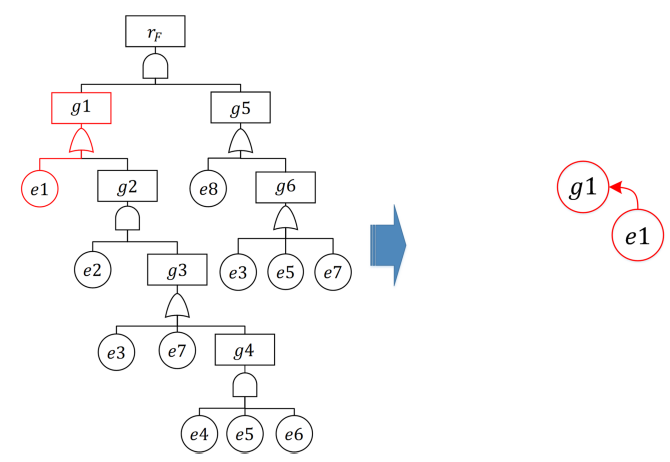
Example



1. $\mathcal{Q} = \{\langle \text{AND}, \{g1, g5\}, r_F\rangle,$
   $\langle \text{OR}, \{e1, g2\}, g1\rangle, \langle \text{AND}, \{e2, g3\}, g2\rangle,$
   $\langle \text{OR}, \{e3, e7, g4\}, g3\rangle,$
   $\langle \text{OR}, \{e6, e5, e6\}, g4\rangle,$
   $\langle \text{OR}, \{e8, g6\}, g5\rangle,$
   $\langle \text{OR}, \{e3, e5, e7\}, g6\rangle\}$
2. $\tilde{\pi}^+ =$
   $\{r_F, g1, g5, g2, g3, g6, e1, e2, e3, e8\}$

Extract MCS

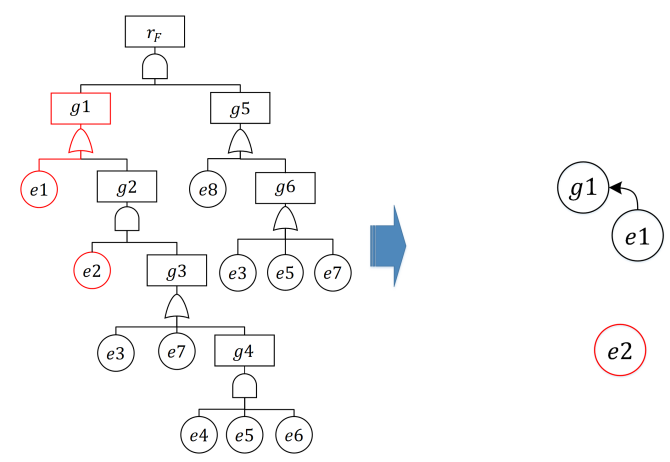
Example



1. $\mathcal{Q} = \{\langle \text{AND}, \{g1, g5\}, r_F\rangle,$
   $\langle \text{OR}, \{e1, g2\}, g1\rangle, \langle \text{AND}, \{e2, g3\}, g2\rangle,$
   $\langle \text{OR}, \{e3, e7, g4\}, g3\rangle,$
   $\langle \text{OR}, \{e6, e5, e6\}, g4\rangle,$
   $\langle \text{OR}, \{e8, g6\}, g5\rangle,$
   $\langle \text{OR}, \{e3, e5, e7\}, g6\rangle\}$
2. $\tilde{\pi}^+ =$
   $\{r_F, g1, g5, g2, g3, g6, e1, e2, e3, e8\}$

Extract MCS

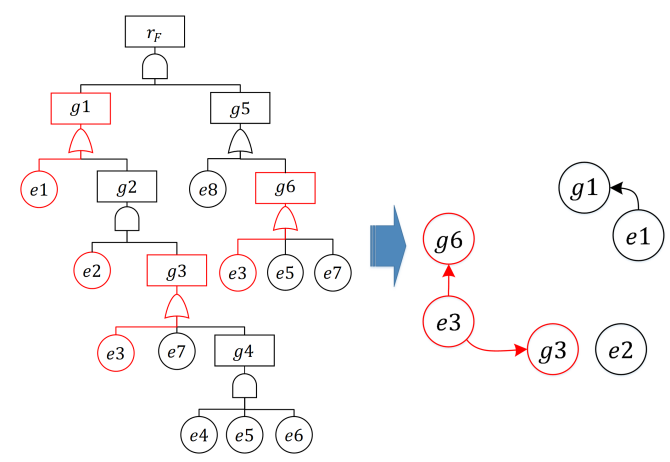
Example



1. $\mathcal{Q} = \{\langle \text{AND}, \{g1, g5\}, r_F\rangle,$
   $\langle \text{OR}, \{e1, g2\}, g1\rangle, \langle \text{AND}, \{e2, g3\}, g2\rangle,$
   $\langle \text{OR}, \{e3, e7, g4\}, g3\rangle,$
   $\langle \text{OR}, \{e6, e5, e6\}, g4\rangle,$
   $\langle \text{OR}, \{e8, g6\}, g5\rangle,$
   $\langle \text{OR}, \{e3, e5, e7\}, g6\rangle\}$
2. $\tilde{\pi}^+ =$
   $\{r_F, g1, g5, g2, g3, g6, e1, e2, e3, e8\}$

Extract MCS

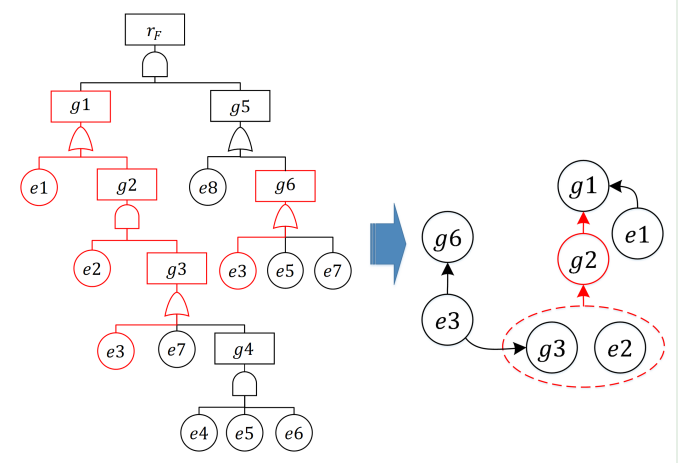
Example



1. $\mathcal{Q} = \{\langle \text{AND}, \{g1, g5\}, r_F\rangle,$
   $\langle \text{OR}, \{e1, g2\}, g1\rangle, \langle \text{AND}, \{e2, g3\}, g2\rangle,$
   $\langle \text{OR}, \{e3, e7, g4\}, g3\rangle,$
   $\langle \text{OR}, \{e6, e5, e6\}, g4\rangle,$
   $\langle \text{OR}, \{e8, g6\}, g5\rangle,$
   $\langle \text{OR}, \{e3, e5, e7\}, g6\rangle\}$
2. $\tilde{\pi}^+ =$
   $\{r_F, g1, g5, g2, g3, g6, e1, e2, e3, e8\}$

Extract MCS

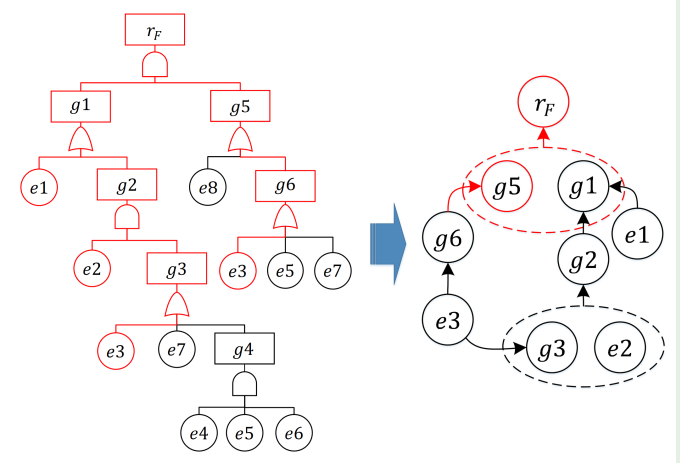
Example



1. $\mathcal{Q} = \{\langle \text{AND}, \{g1, g5\}, r_F\rangle,$
   $\langle \text{OR}, \{e1, g2\}, g1\rangle, \langle \text{AND}, \{e2, g3\}, g2\rangle,$
   $\langle \text{OR}, \{e3, e7, g4\}, g3\rangle,$
   $\langle \text{OR}, \{e6, e5, e6\}, g4\rangle,$
   $\langle \text{OR}, \{e8, g6\}, g5\rangle,$
   $\langle \text{OR}, \{e3, e5, e7\}, g6\rangle\}$
2. $\tilde{\pi}^+ =$
   $\{r_F, g1, g5, g2, g3, g6, e1, e2, e3, e8\}$

Extract MCS

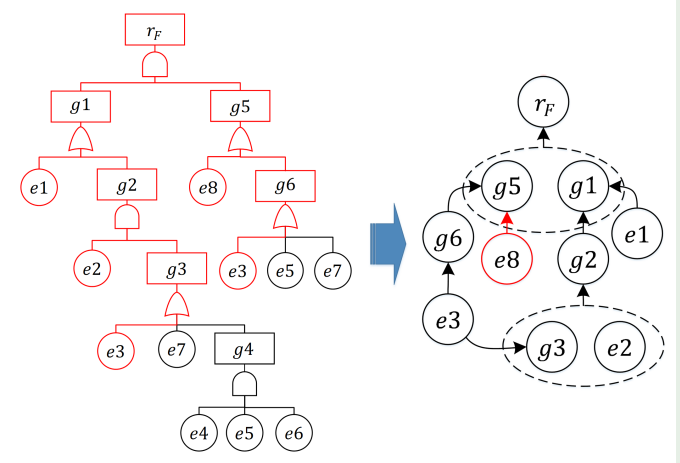
Example



1. $\mathcal{Q} = \{\langle \text{AND}, \{g1, g5\}, r_F\rangle,$
   $\langle \text{OR}, \{e1, g2\}, g1\rangle, \langle \text{AND}, \{e2, g3\}, g2\rangle,$
   $\langle \text{OR}, \{e3, e7, g4\}, g3\rangle,$
   $\langle \text{OR}, \{e6, e5, e6\}, g4\rangle,$
   $\langle \text{OR}, \{e8, g6\}, g5\rangle,$
   $\langle \text{OR}, \{e3, e5, e7\}, g6\rangle\}$
2. $\tilde{\pi}^+ =$
   $\{r_F, g1, g5, g2, g3, g6, e1, e2, e3, e8\}$

Extract MCS

Example



1. $\mathcal{Q} = \{\langle \text{AND}, \{g1, g5\}, r_F\rangle,$
   $\langle \text{OR}, \{e1, g2\}, g1\rangle, \langle \text{AND}, \{e2, g3\}, g2\rangle,$
   $\langle \text{OR}, \{e3, e7, g4\}, g3\rangle,$
   $\langle \text{OR}, \{e6, e5, e6\}, g4\rangle,$
   $\langle \text{OR}, \{e8, g6\}, g5\rangle,$
   $\langle \text{OR}, \{e3, e5, e7\}, g6\rangle\}$
2. $\tilde{\pi}^+ =$
   $\{r_F, g1, g5, g2, g3, g6, e1, e2, e3, e8\}$

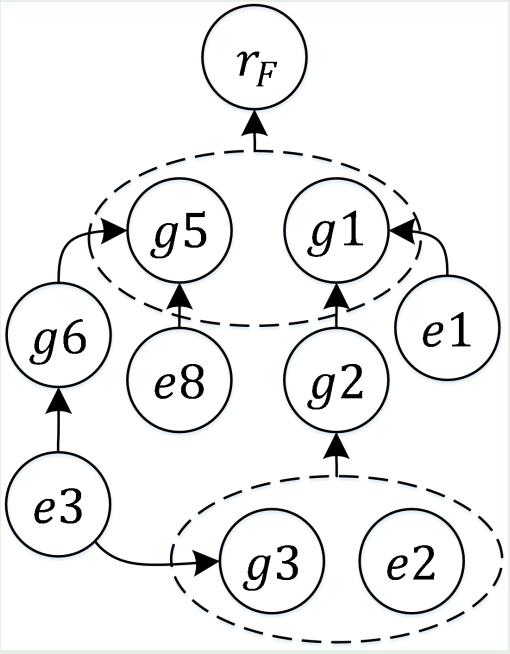
Extract MCS

Extract MCS over LPG

- Next, we will test the necessity of each basic event $v \in \pi^{+}_{E_F}$ by analyzing the failure propagation paths in P_{π^+} triggered by v.
  - If removing those paths from P_{π^+} does not cause the top event r_F to be disabled, v is not necessary;
  - otherwise, v is included into an MCS.

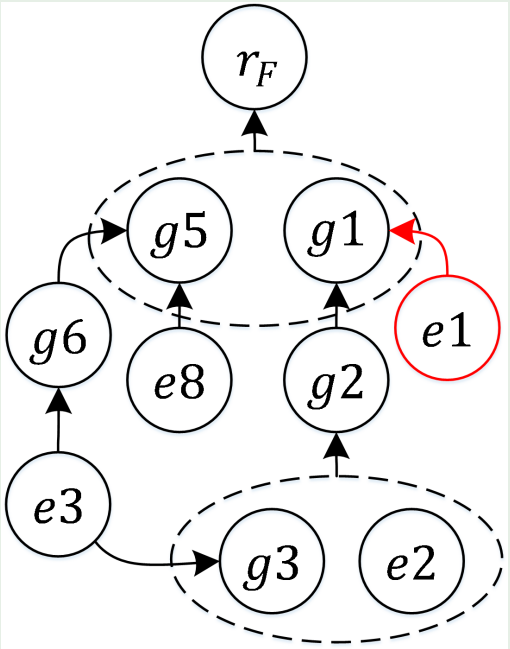
Extract MCS

Example



Extract MCS

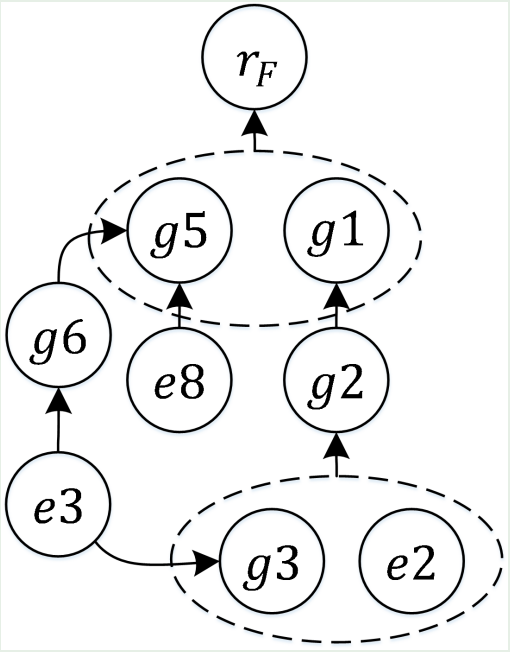
Example



1. $e1$ √

Extract MCS

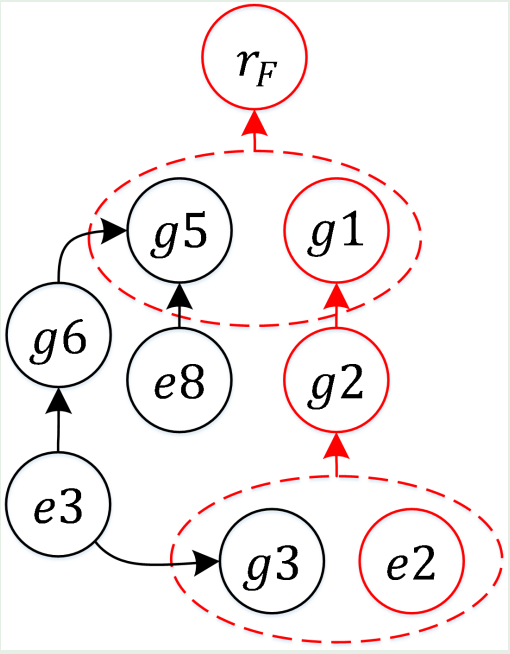
Example



1. $e1$ √

Extract MCS

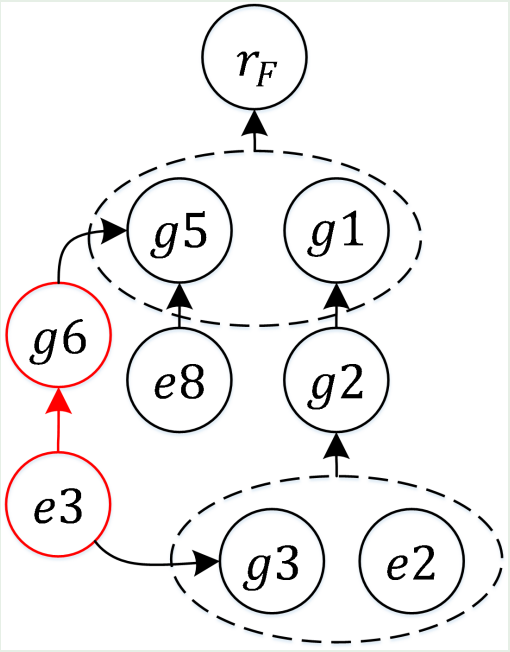
Example



1. $e1$ ✓
2. $e2 \rightarrow g2 \rightarrow g1 \rightarrow r_F$ ×

Extract MCS

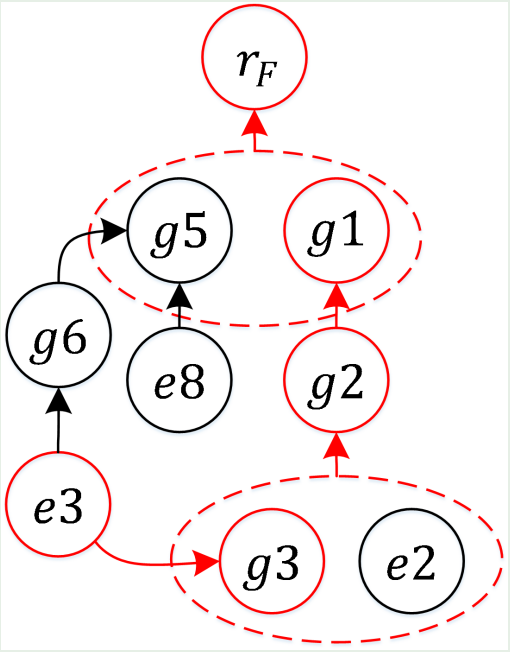
Example



1. $e1$ ✓
2. $e2 \rightarrow g2 \rightarrow g1 \rightarrow r_F$ ×
3. $e3 \rightarrow g6$ ✓

Extract MCS

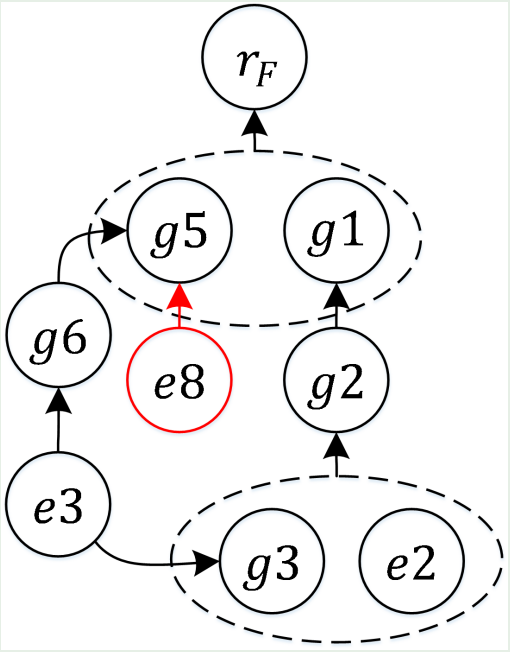
Example



1. $e1$ $\surd$
2. $e2 \rightarrow g2 \rightarrow g1 \rightarrow r_F$ $\times$
3. $e3 \rightarrow g6$ $\surd$
4. $e3 \rightarrow g3 \rightarrow g2 \rightarrow g1 \rightarrow r_F$ $\times$

Extract MCS

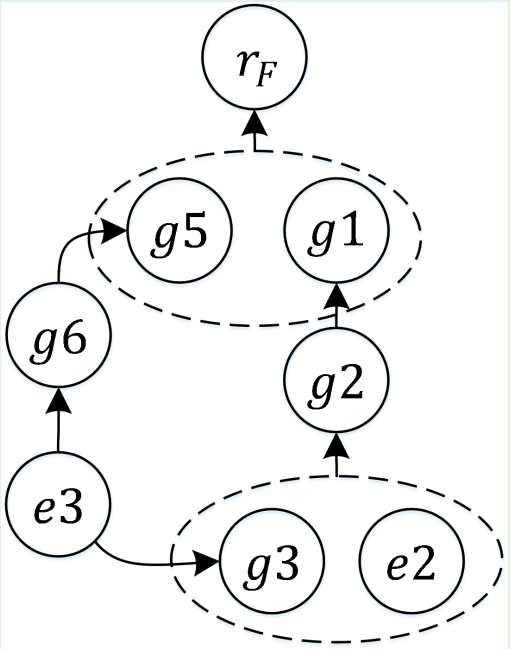
Example



1. $e1$ $\surd$
2. $e2 \rightarrow g2 \rightarrow g1 \rightarrow r_F$ $\times$
3. $e3 \rightarrow g6$ $\surd$
4. $e3 \rightarrow g3 \rightarrow g2 \rightarrow g1 \rightarrow r_F$ $\times$
5. $e8$ $\surd$

Extract MCS

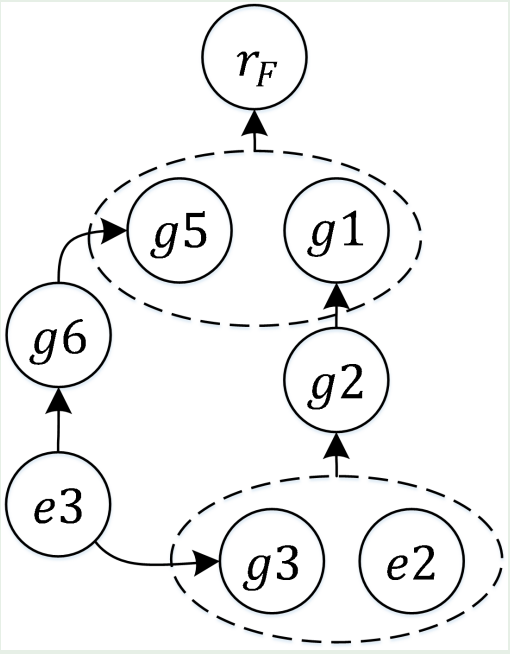
Example



1. $e1$ √
2. $e2 \rightarrow g2 \rightarrow g1 \rightarrow r_F$ ×
3. $e3 \rightarrow g6$ √
4. $e3 \rightarrow g3 \rightarrow g2 \rightarrow g1 \rightarrow r_F$ ×
5. $e8$ √

Extract MCS

Example



1. $e1$ √
2. $e2 \rightarrow g2 \rightarrow g1 \rightarrow r_F$ ×
3. $e3 \rightarrow g6$ √
4. $e3 \rightarrow g3 \rightarrow g2 \rightarrow g1 \rightarrow r_F$ ×
5. $e8$ √

MCS is $\{e2, e3\}$

Extract MCS

Failure Propagation

For each edge (S, I'), if an event $I \in S$, we conclude that I influences I' in two cases.

1. $|S| \geq 2$: by the construction of P_{π^+}, I' is associated with an AND gate; hence, I' is influenced by I.
2. $\neg(\exists \tilde{I} \neq I \cdot (\{\tilde{I}\}, I') \in R')$: note that $|S| = 1$ implies that I' corresponds to an OR gate; therefore, if I is the event in π^+ that causes I' to occur, I' is influenced by I.

Correctness

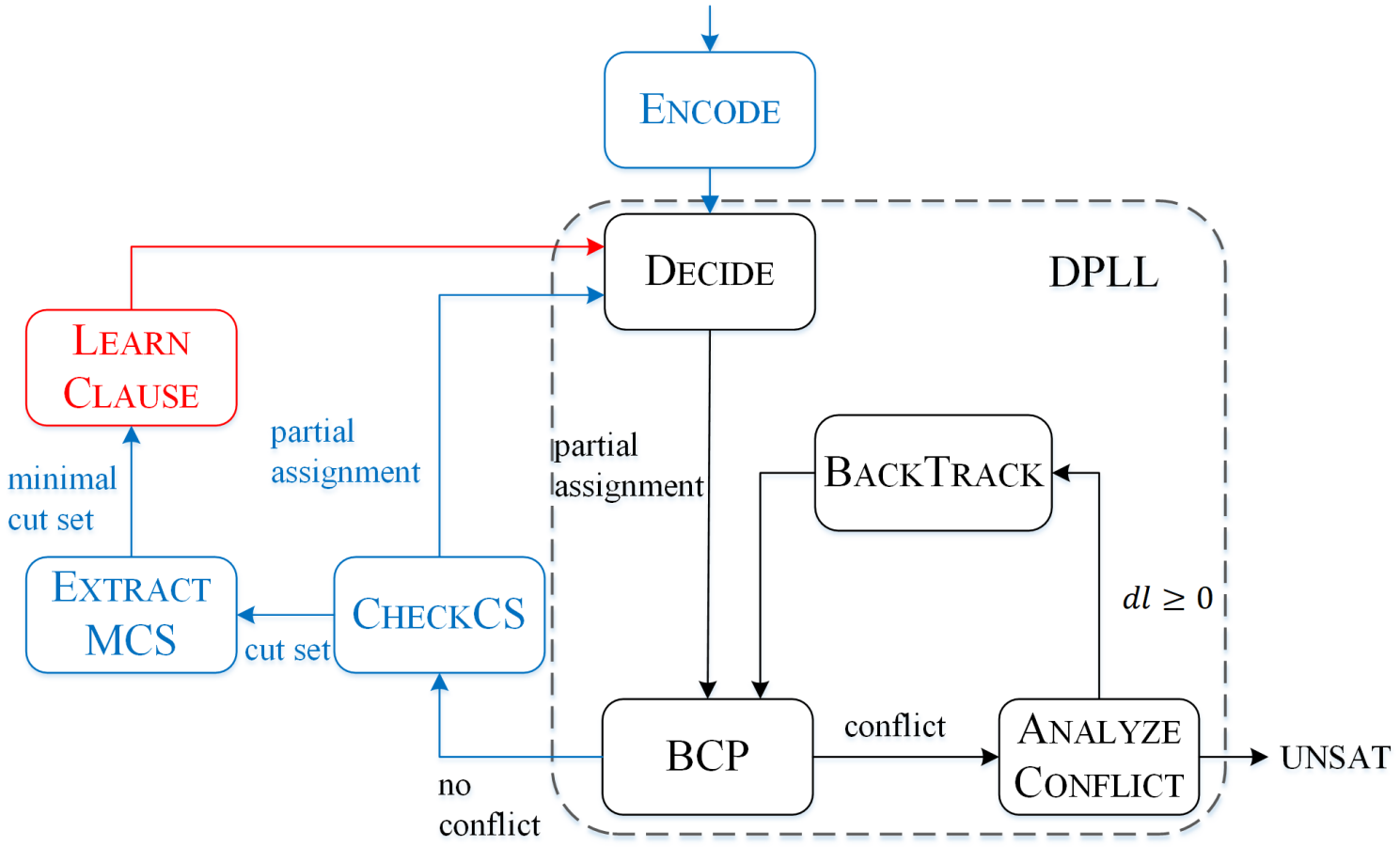
Fixpoint computation: checking the necessity of $v \in \pi^+_{E_F}$ through chaotic iteration.

Our Work

- Overview of SATMCS
- Tseitin Encoding of Fault Trees
- Extracting MCSs using Local Fault Graph
- Clause Learning with Dynamic Deletion
- Experiments

Clause Learning with Dynamic Deletion

SATMCS maintains a CNF formula $\mathcal{H} = \mathcal{F} \cup \mathcal{B} \cup \mathcal{C}$ that consists of three sets of clauses: $\mathcal{F}$ encodes F, $\mathcal{B}$ blocks already computed MCSs, and $\mathcal{C}$ contains the conflict clauses from DPLL.



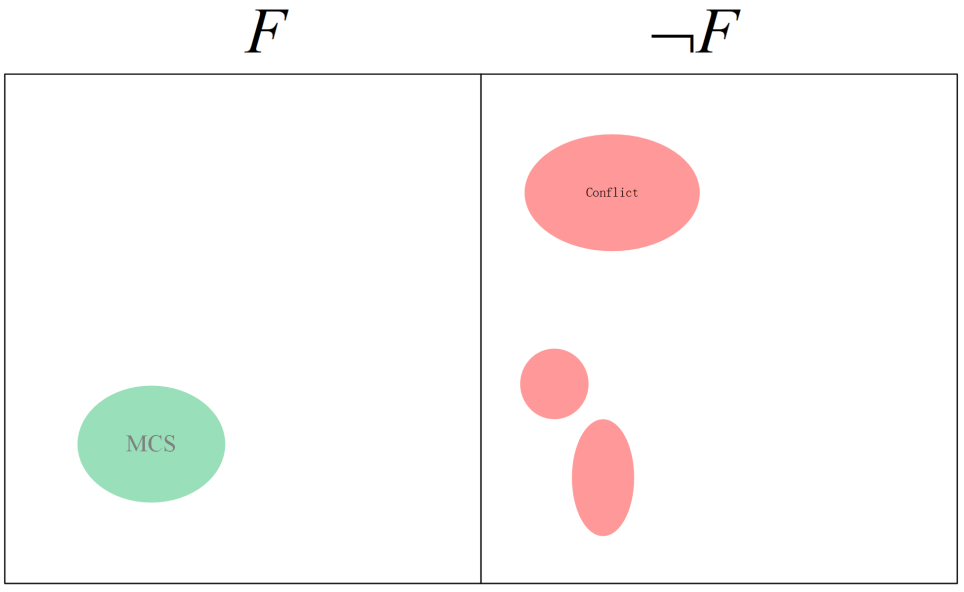
Clause Learning with Dynamic Deletion

Clause Learning

$\mathcal{H} = \mathcal{F} \cup \mathcal{B} \cup \mathcal{C}$

- A blocking clause consisting of the opposite to the literals in the MCS is added to $\mathcal{B}$;
- LEARNCLAUSE also updates $\mathcal{C}$ to include the conflict clauses generated from ANALYZECONFLICT.

Example



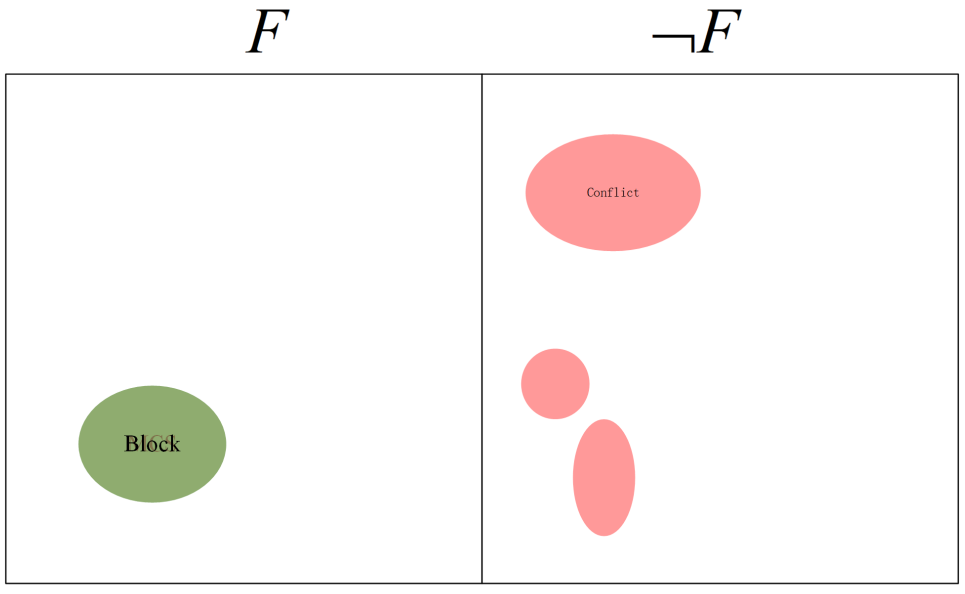
Clause Learning with Dynamic Deletion

Clause Learning

$\mathcal{H} = \mathcal{F} \cup \mathcal{B} \cup \mathcal{C}$

- A blocking clause consisting of the opposite to the literals in the MCS is added to $\mathcal{B}$;
- LEARNCLAUSE also updates $\mathcal{C}$ to include the conflict clauses generated from ANALYZECONFLICT.

Example



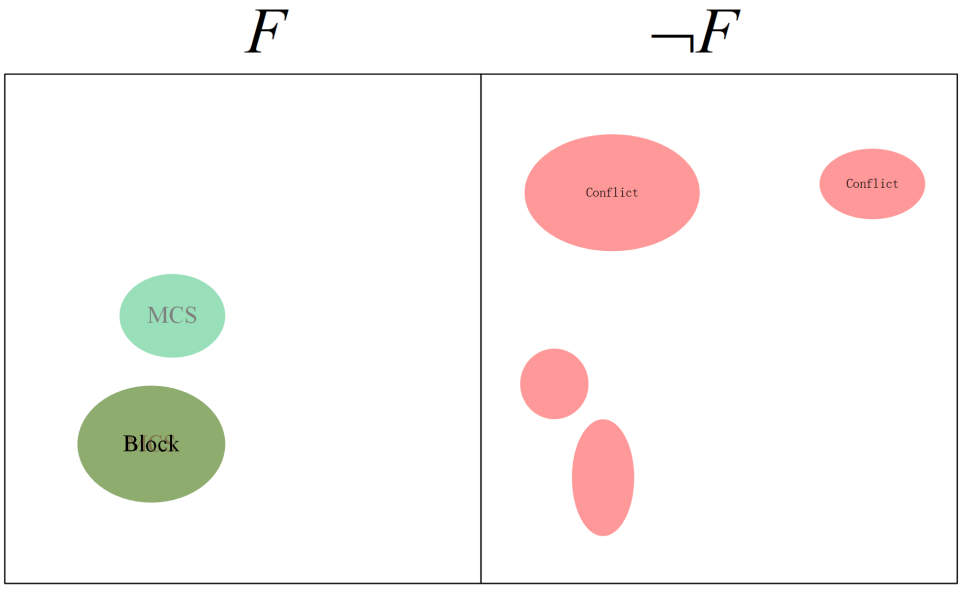
Clause Learning with Dynamic Deletion

Clause Learning

$\mathcal{H} = \mathcal{F} \cup \mathcal{B} \cup \mathcal{C}$

- A blocking clause consisting of the opposite to the literals in the MCS is added to $\mathcal{B}$;
- LEARNCLAUSE also updates $\mathcal{C}$ to include the conflict clauses generated from ANALYZECONFLICT.

Example



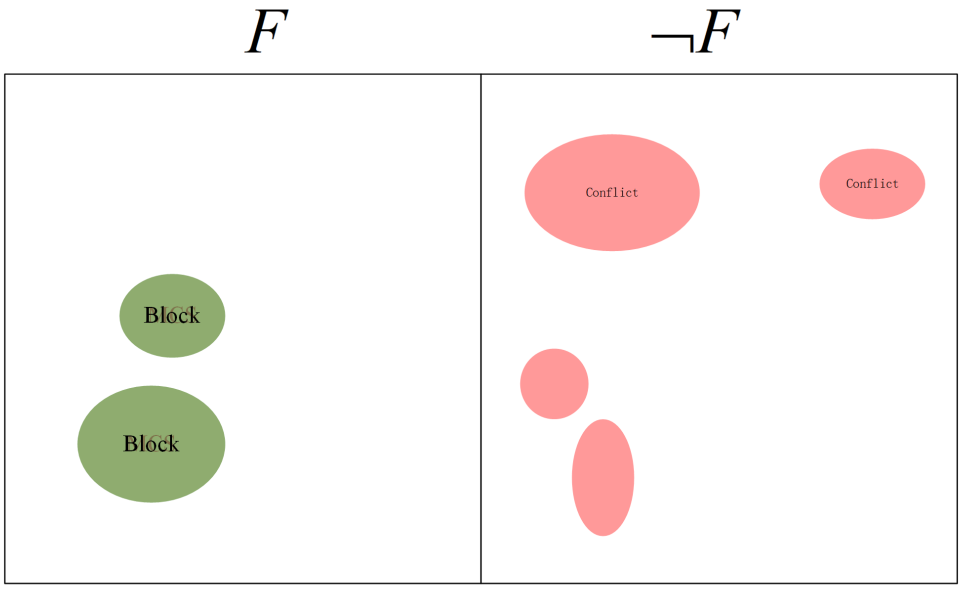
Clause Learning with Dynamic Deletion

Clause Learning

$\mathcal{H} = \mathcal{F} \cup \mathcal{B} \cup \mathcal{C}$

- A blocking clause consisting of the opposite to the literals in the MCS is added to $\mathcal{B}$;
- LEARNCLAUSE also updates $\mathcal{C}$ to include the conflict clauses generated from ANALYZECONFLICT.

Example



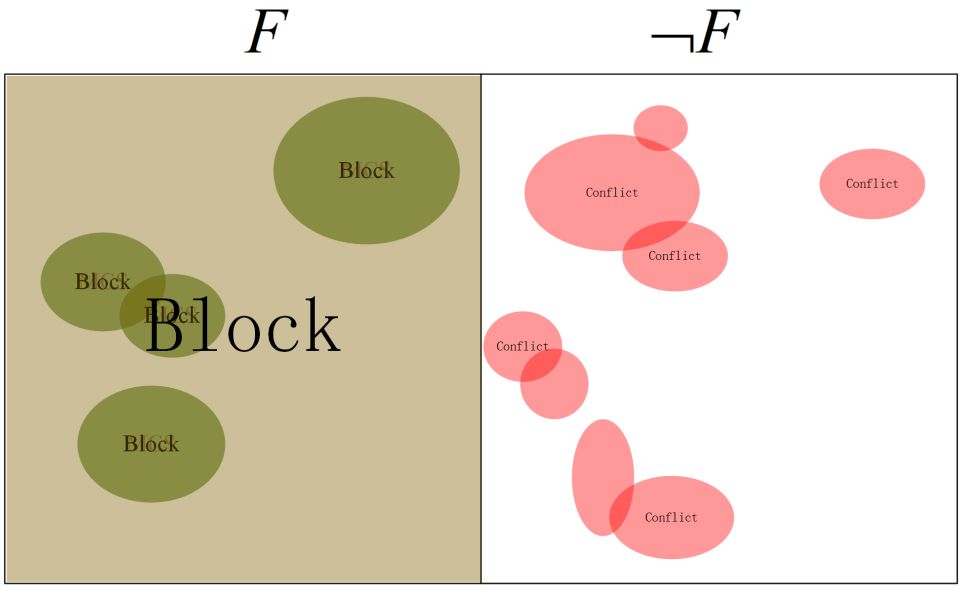
Clause Learning with Dynamic Deletion

Clause Learning

$\mathcal{H} = \mathcal{F} \cup \mathcal{B} \cup \mathcal{C}$

- A blocking clause consisting of the opposite to the literals in the MCS is added to $\mathcal{B}$;
- LEARNCLAUSE also updates $\mathcal{C}$ to include the conflict clauses generated from ANALYZECONFLICT.

Example



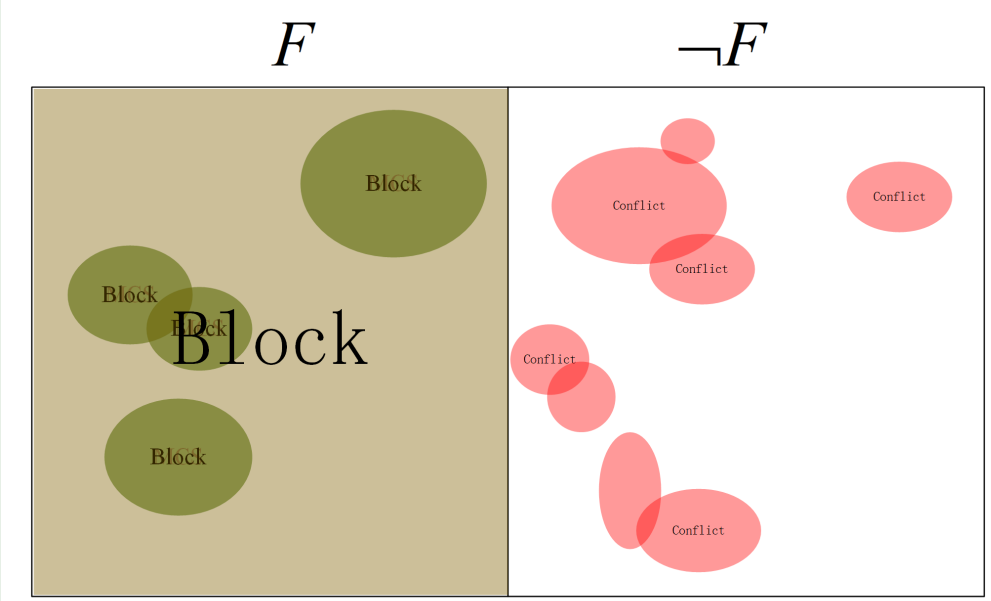
Clause Learning with Dynamic Deletion

Clause Learning

$\mathcal{H} = \mathcal{F} \cup \mathcal{B} \cup \mathcal{C}$

- A blocking clause consisting of the opposite to the literals in the MCS is added to $\mathcal{B}$;
- LEARNCLAUSE also updates $\mathcal{C}$ to include the conflict clauses generated from ANALYZECONFLICT.

Example



Dynamic Deletion

- A large number of clauses in $\mathcal{H}$ will reduce the performance of the DPLL procedure.
- The clauses in $\mathcal{C}$ can be deleted without affecting the correctness.
- We consider the ratio, i.e., $|\mathcal{C}|/|\mathcal{H}|$; when the ratio exceeds a certain threshold η, LEARNCLAUSE deletes the clauses in $\mathcal{C}$.

Our Work

- Overview of SATMCS
- Tseitin Encoding of Fault Trees
- Extracting MCSs using Local Fault Graph
- Clause Learning with Dynamic Deletion
- Experiments

Our Work

- Comparison between SATMCS and FaultTree+
- Evaluation of Extracting MCSs

Comparison between SATMCS and FaultTree+

FaultTree+

- one of the most popular FTA commercial tools;
- bottom-up boolean manipulation method.

Benchmarks

- There are 33 coherent fault tree benchmarks from real life applications such as aerospace and nuclear power industries.

Comparison between SATMCS and FaultTree+

Benchmark	#E	#G	#MCS	SATMCS		FaultTree+	
				Time (sec.)	Mem (MB)	Time (sec.)	Mem (MB)
baobab3	80	107	24386	12.7	66.4	–	–
chinese	25	36	392	<0.1	3.1	0.8	89.5
elf9601	145	242	151348	38.4	105.3	–	–
ftr10	175	94	396	<0.1	3.2	1.1	91.4
jbd9601	533	315	14007	1.6	8.5	2.8	109.2
das9205	51	20	17280	<0.1	3.4	0.8	86.1
das9203	51	30	16200	<0.1	3.1	0.7	86.4
das9204	53	30	16704	<0.1	3.3	0.6	89.7
das9202	49	36	27778	<0.1	3.3	1.6	88.9
das9209	109	73	8.2E10	<0.1	3.1	0.7	96.1
das9206	121	112	19518	0.4	5.2	3.6	92.9
das9201	122	82	14217	0.2	4.1	1.3	91.8
das9208	103	145	8060	1.9	9.8	1.5	96.2
das9207	276	324	25988	4.7	10.2	3.5	100.2
edf9205	165	142	21308	0.5	5.2	376.4	248.6
edf9201	183	132	579720	56.1	57.7	2423	723.8
edf9203	362	475	–	–	–	–	–
edf9204	323	375	–	–	–	–	–
edf9202	458	435	130112	5.5	9.7	–	–
edf9206	240	362	–	–	–	–	–
edfpa15r	88	110	26549	26.8	142.1	85.6	124.1
edfpa14r	106	132	–	–	–	–	–
edfpa15p	276	324	27870	22.5	142.3	409.8	179
edfpa14p	124	101	–	–	–	–	–
edfpa14q	311	194	–	–	–	–	–
edfpa15b	283	249	2910473	2662.6	1493.8	–	–
edfpa14o	311	173	–	–	–	–	–
edfpa14b	311	290	–	–	–	–	–
isp9606	89	41	1776	<0.1	3.1	0.9	90.2
isp9607	74	65	150436	<0.1	4.2	5.1	94.2
isp9603	91	95	3434	0.1	3.6	1.6	93.4
isp9602	116	122	5197647	1.6	12.6	16.6	94.1
isp9604	215	132	746574	<0.1	3.2	1.1	88

Results

- SATMCS can solve 25 cases of the 33 fault trees, which include all the 21 ones that FaultTree+ can successfully compute.
- SATMCS and FaultTree+ produce exactly the same set of MCSs for the 21 cases solved by both of them.

Comparison between SATMCS and FaultTree+

Benchmark	#E	#G	#MCS	SATMCS		FaultTree+	
				Time (sec.)	Mem (MB)	Time (sec.)	Mem (MB)
baobab3	80	107	24386	12.7	66.4	–	–
chinese	25	36	392	<0.1	3.1	0.8	89.5
elf9601	145	242	151348	38.4	105.3	–	–
ftr10	175	94	396	<0.1	3.2	1.1	91.4
jbd9601	533	315	14007	1.6	8.5	2.8	109.2
das9205	51	20	17280	<0.1	3.4	0.8	86.1
das9203	51	30	16200	<0.1	3.1	0.7	86.4
das9204	53	30	16704	<0.1	3.3	0.6	89.7
das9202	49	36	27778	<0.1	3.3	1.6	88.9
das9209	109	73	8.2E10	<0.1	3.1	0.7	96.1
das9206	121	112	19518	0.4	5.2	3.6	92.9
das9201	122	82	14217	0.2	4.1	1.3	91.8
das9208	103	145	8060	1.9	9.8	1.5	96.2
das9207	276	324	25988	4.7	10.2	3.5	100.2
edf9205	165	142	21308	0.5	5.2	376.4	248.6
edf9201	183	132	579720	56.1	57.7	2423	723.8
edf9203	362	475	–	–	–	–	–
edf9204	323	375	–	–	–	–	–
edf9202	458	435	130112	5.5	9.7	–	–
edf9206	240	362	–	–	–	–	–
edfpa15r	88	110	26549	26.8	142.1	85.6	124.1
edfpa14r	106	132	–	–	–	–	–
edfpa15p	276	324	27870	22.5	142.3	409.8	179
edfpa14p	124	101	–	–	–	–	–
edfpa14q	311	194	–	–	–	–	–
edfpa15b	283	249	2910473	2662.6	1493.8	–	–
edfpa14o	311	173	–	–	–	–	–
edfpa14b	311	290	–	–	–	–	–
isp9606	89	41	1776	<0.1	3.1	0.9	90.2
isp9607	74	65	150436	<0.1	4.2	5.1	94.2
isp9603	91	95	3434	0.1	3.6	1.6	93.4
isp9602	116	122	5197647	1.6	12.6	16.6	94.1
isp9604	215	132	746574	<0.1	3.2	1.1	88

Results

- SATMCS can solve 25 cases of the 33 fault trees, which include all the 21 ones that FaultTree+ can successfully compute.
- SATMCS and FaultTree+ produce exactly the same set of MCSs for the 21 cases solved by both of them.
- SATMCS generally computes much faster than FaultTree+ for the 21 cases.
- The memory used by SATMCS is significantly less than that used by FaultTree+ in most cases.

Evaluation of Extracting MCSs

Features of ExtractMCS

ExtractMCS tests the necessity of a basic event by analyzing failure propagation paths in an LPG induced by the cut set.

- The tests of the necessity of a basic event work based on LPG.
- It conducts a linear number of tests for deciding the necessity of basic events in a cut set.

SATMCS-Call

It replaces the LPG-based test in ExtractMCS with direct calling ZChaff.

SATMCS-QX

It uses the QuickXplain algorithm [U. Junker, 2004; A. R. Bradley and Z. Manna, 2007], which, based on recursively splitting the cut set, requires exponentially less tests in optimal case than the linear approach.

Evaluation of Extracting MCSs

Benchmark	SATMCS		SATMCS-Call		SATMCS-QX	
	Time (sec.)	Mem (MB)	Time (sec.)	Mem (MB)	Time (sec.)	Mem (MB)
baobab3	12.7	66.4	36.1	65.6	16.6	93.4
elf9601	38.4	105.3	118.1	104.1	51.8	136.7
edf9201	56.1	57.7	96.6	54.2	53.9	56.2
edf9202	5.5	9.7	43.9	9.5	13.3	21.1
edfpa15r	26.8	142.1	127.8	96.6	88.1	147.3
edfpa15p	22.5	142.3	114.9	150.4	43.1	238.2
edfpa15b	2662.6	1493.8	6659.5	644.8	5415.6	1005.5

Results

- SATMCS takes about less than half of the time used by SATMCS-Call.
- SATMCS-Call consumes slightly less memory than SATMCS in most cases.
- SATMCS takes about half of the time used by SATMCS-QX in most cases with comparable memory usage.

Summary

- Classical methods for computing MCSs are based on boolean manipulation and BDDs, which may still suffer from time or memory constraints in practice;
- We presented and implemented SATMCS, a novel method for computing MCSs based on SAT solving;
- We compared SATMCS with a popular FTA tool in market; preliminary experimental results are promising.

- Future Work:
  - better optimization strategies
  - more experiments: cases, tools

Evaluation of Extracting MCSs

Benchmark	SATMCS		SATMCS-Call		SATMCS-QX	
	Time (sec.)	Mem (MB)	Time (sec.)	Mem (MB)	Time (sec.)	Mem (MB)
baobab3	12.7	66.4	36.1	65.6	16.6	93.4
elf9601	38.4	105.3	118.1	104.1	51.8	136.7
edf9201	56.1	57.7	96.6	54.2	53.9	56.2
edf9202	5.5	9.7	43.9	9.5	13.3	21.1
edfpa15r	26.8	142.1	127.8	96.6	88.1	147.3
edfpa15p	22.5	142.3	114.9	150.4	43.1	238.2
edfpa15b	2662.6	1493.8	6659.5	644.8	5415.6	1005.5

Result

- SATMCS takes about less than half of the time used by SATMCS-Call.
- SATMCS-Call consumes slightly less memory than SATMCS in most cases.
- SATMCS takes about half of the time used by SATMCS-QX in most cases with comparable memory usage.

- $O\left((|\bar{S}| - 1) + |\bar{S}| \lg \frac{|S|}{|\bar{S}|}\right)$ is the upper bound of QuickXplain in [A. R. Bradley and Z. Manna, 2007].
- Because of $\frac{|S|}{|\bar{S}|} \approx 1.33$, QuickXplain requires more tests than the linear one.

Improvement of SATMCS

Benchmark	#E	#G	#MCS	SATMCS		SATMCS-pro	
				Time (sec.)	Mem (MB)	Time (sec.)	Mem (MB)
baobab3	80	107	24386	12.7	66.4	5.2	14.3
chinese	25	36	392	<0.1	3.1	<0.1	3.1
elf9601	145	242	151348	38.4	105.3	270.1	28.1
ftr10	175	94	396	<0.1	3.2	0.1	3.1
jbd9601	533	315	14007	1.6	8.5	1.2	3.9
das9205	51	20	17280	<0.1	3.4	<0.1	3
das9203	51	30	16200	<0.1	3.1	<0.1	3
das9204	53	30	16704	<0.1	3.3	<0.1	3
das9202	49	36	27778	<0.1	3.3	<0.1	3.1
das9209	109	73	8.2E10	<0.1	3.1	<0.1	3
das9206	121	112	19518	0.4	5.2	0.1	3.3
das9201	122	82	14217	0.2	4.1	<0.1	3.1
das9208	103	145	8060	1.9	9.8	0.5	4.1
das9207	276	324	25988	4.7	10.2	1	4
edf9205	165	142	21308	0.5	5.2	0.2	3.3
edf9201	183	132	579720	56.1	57.7	0.3	3.7
edf9203	362	475	–	–	–	164.2	167.7
edf9204	323	375	–	–	–	–	–
edf9202	458	435	130112	5.5	9.7	0.7	3.3
edf9206	240	362	–	–	–	–	–
edfpa15r	88	110	26549	26.8	142.1	12.4	24.1
edfpa14r	106	132	–	–	–	1802.6	722
edfpa15p	276	324	27870	22.5	142.3	12.5	31.4
edfpa14p	124	101	–	–	–	2012.9	751.3
edfpa14q	311	194	–	–	–	1558.9	659.5
edfpa15b	283	249	2910473	2662.6	1493.8	12	29.8
edfpa14o	311	173	–	–	–	1348.0	644.1
edfpa14b	311	290	–	–	–	1851.8	746.0
isp9606	89	41	1776	<0.1	3.1	<0.1	3.1
isp9607	74	65	150436	<0.1	4.2	<0.1	3.2
isp9603	91	95	3434	0.1	3.6	0.1	3.2
isp9602	116	122	5197647	1.6	12.6	0.2	3.8
isp9604	215	132	746574	<0.1	3.2	0.1	3.3

Strategies

Improvement of SATMCS

Benchmark	#E	#G	#MCS	SATMCS		SATMCS-pro	
				Time (sec.)	Mem (MB)	Time (sec.)	Mem (MB)
baobab3	80	107	24386	12.7	66.4	5.2	14.3
chinese	25	36	392	<0.1	3.1	<0.1	3.1
elf9601	145	242	151348	38.4	105.3	270.1	28.1
ftr10	175	94	396	<0.1	3.2	0.1	3.1
jbd9601	533	315	14007	1.6	8.5	1.2	3.9
das9205	51	20	17280	<0.1	3.4	<0.1	3
das9203	51	30	16200	<0.1	3.1	<0.1	3
das9204	53	30	16704	<0.1	3.3	<0.1	3
das9202	49	36	27778	<0.1	3.3	<0.1	3.1
das9209	109	73	8.2E10	<0.1	3.1	<0.1	3
das9206	121	112	19518	0.4	5.2	0.1	3.3
das9201	122	82	14217	0.2	4.1	<0.1	3.1
das9208	103	145	8060	1.9	9.8	0.5	4.1
das9207	276	324	25988	4.7	10.2	1	4
edf9205	165	142	21308	0.5	5.2	0.2	3.3
edf9201	183	132	579720	56.1	57.7	0.3	3.7
edf9203	362	475	–	–	–	164.2	167.7
edf9204	323	375	–	–	–	–	–
edf9202	458	435	130112	5.5	9.7	0.7	3.3
edf9206	240	362	–	–	–	–	–
edfpa15r	88	110	26549	26.8	142.1	12.4	24.1
edfpa14r	106	132	–	–	–	1802.6	722
edfpa15p	276	324	27870	22.5	142.3	12.5	31.4
edfpa14p	124	101	–	–	–	2012.9	751.3
edfpa14q	311	194	–	–	–	1558.9	659.5
edfpa15b	283	249	2910473	2662.6	1493.8	12	29.8
edfpa14o	311	173	–	–	–	1348.0	644.1
edfpa14b	311	290	–	–	–	1851.8	746.0
isp9606	89	41	1776	<0.1	3.1	<0.1	3.1
isp9607	74	65	150436	<0.1	4.2	<0.1	3.2
isp9603	91	95	3434	0.1	3.6	0.1	3.2
isp9602	116	122	5197647	1.6	12.6	0.2	3.8
isp9604	215	132	746574	<0.1	3.2	0.1	3.3

Strategies

- We used a more powerful preprocessing method to handle fault trees before computing MCS;

Improvement of SATMCS

Benchmark	#E	#G	#MCS	SATMCS		SATMCS-pro	
				Time (sec.)	Mem (MB)	Time (sec.)	Mem (MB)
baobab3	80	107	24386	12.7	66.4	5.2	14.3
chinese	25	36	392	<0.1	3.1	<0.1	3.1
elf9601	145	242	151348	38.4	105.3	270.1	28.1
ftr10	175	94	396	<0.1	3.2	0.1	3.1
jbd9601	533	315	14007	1.6	8.5	1.2	3.9
das9205	51	20	17280	<0.1	3.4	<0.1	3
das9203	51	30	16200	<0.1	3.1	<0.1	3
das9204	53	30	16704	<0.1	3.3	<0.1	3
das9202	49	36	27778	<0.1	3.3	<0.1	3.1
das9209	109	73	8.2E10	<0.1	3.1	<0.1	3
das9206	121	112	19518	0.4	5.2	0.1	3.3
das9201	122	82	14217	0.2	4.1	<0.1	3.1
das9208	103	145	8060	1.9	9.8	0.5	4.1
das9207	276	324	25988	4.7	10.2	1	4
edf9205	165	142	21308	0.5	5.2	0.2	3.3
edf9201	183	132	579720	56.1	57.7	0.3	3.7
edf9203	362	475	–	–	–	164.2	167.7
edf9204	323	375	–	–	–	–	–
edf9202	458	435	130112	5.5	9.7	0.7	3.3
edf9206	240	362	–	–	–	–	–
edfpa15r	88	110	26549	26.8	142.1	12.4	24.1
edfpa14r	106	132	–	–	–	1802.6	722
edfpa15p	276	324	27870	22.5	142.3	12.5	31.4
edfpa14p	124	101	–	–	–	2012.9	751.3
edfpa14q	311	194	–	–	–	1558.9	659.5
edfpa15b	283	249	2910473	2662.6	1493.8	12	29.8
edfpa14o	311	173	–	–	–	1348.0	644.1
edfpa14b	311	290	–	–	–	1851.8	746.0
isp9606	89	41	1776	<0.1	3.1	<0.1	3.1
isp9607	74	65	150436	<0.1	4.2	<0.1	3.2
isp9603	91	95	3434	0.1	3.6	0.1	3.2
isp9602	116	122	5197647	1.6	12.6	0.2	3.8
isp9604	215	132	746574	<0.1	3.2	0.1	3.3

Strategies

- We used a more powerful preprocessing method to handle fault trees before computing MCS;
- SATMCS-pro backtracks by chronological backtracking way after LEARNCLAUSE;

All-SAT Problem

Definition

- All-SAT problem computes all the satisfying assignments of a propositional logic formula. This problem has been investigated in the context of model checking and predicate abstraction.
- Main algorithms for solving the problem are based on SAT and BDDs.

All-SAT Problem

Definition

- All-SAT problem computes all the satisfying assignments of a propositional logic formula. This problem has been investigated in the context of model checking and predicate abstraction.
- Main algorithms for solving the problem are based on SAT and BDDs.

Commonality

- The purpose of both is to find all the "assignments" that satisfy constraints.

All-SAT Problem

Definition

- All-SAT problem computes all the satisfying assignments of a propositional logic formula. This problem has been investigated in the context of model checking and predicate abstraction.
- Main algorithms for solving the problem are based on SAT and BDDs.

Commonality

- The purpose of both is to find all the "assignments" that satisfy constraints.

Difference

- It is computationally less expensive than the problem of computing MCSs, since the satisfying assignments are not necessary to be minimized.

All-SAT Problem

Definition

- All-SAT problem computes all the satisfying assignments of a propositional logic formula. This problem has been investigated in the context of model checking and predicate abstraction.
- Main algorithms for solving the problem are based on SAT and BDDs.

Commonality

- The purpose of both is to find all the "assignments" that satisfy constraints.

Difference

- It is computationally less expensive than the problem of computing MCSs, since the satisfying assignments are not necessary to be minimized.

Idea

- SATMCS can backtrack by chronological backtracking way, which is proved more efficient in All-SAT solver.

All-SAT Problem

Fault Tree	Decide time(s)	BCP time(s)	AnalyzeConflict time(s)	ExtractMCS time(s)	LearnClause time(s)
edf9203_m0	36.3406	6763.56	246.294	777.224	8771.2
edfpa14o_m0	79.3506	5836.03	177.409	130.407	6863.44
edfpa14p_m0	62.2286	5423.57	163.988	123.055	6594.63
edfpa14q_m0	54.2406	4444.27	138.508	148.67	5166.82
edfpa14r_m0	47.3522	3761.06	111.248	146.252	4184.56
edfpa15b_m1	12.1073	796.248	34.5342	128.307	1468.61

All-SAT Problem

Fault Tree	Decide time(s)	BCP time(s)	AnalyzeConflict time(s)	ExtractMCS time(s)	LearnClause time(s)
edf9203_m0	36.3406	6763.56	246.294	777.224	8771.2
edfpa14o_m0	79.3506	5836.03	177.409	130.407	6863.44
edfpa14p_m0	62.2286	5423.57	163.988	123.055	6594.63
edfpa14q_m0	54.2406	4444.27	138.508	148.67	5166.82
edfpa14r_m0	47.3522	3761.06	111.248	146.252	4184.56
edfpa15b_m1	12.1073	796.248	34.5342	128.307	1468.61

- The consumption of time in the LEARNCLAUSE accounts for a large part.

All-SAT Problem

Fault Tree	Decide time(s)	BCP time(s)	AnalyzeConflict time(s)	ExtractMCS time(s)	LearnClause time(s)
edf9203_m0	36.3406	6763.56	246.294	777.224	8771.2
edfpa14o_m0	79.3506	5836.03	177.409	130.407	6863.44
edfpa14p_m0	62.2286	5423.57	163.988	123.055	6594.63
edfpa14q_m0	54.2406	4444.27	138.508	148.67	5166.82
edfpa14r_m0	47.3522	3761.06	111.248	146.252	4184.56
edfpa15b_m1	12.1073	796.248	34.5342	128.307	1468.61

- The consumption of time in the LEARNCLAUSE accounts for a large part.

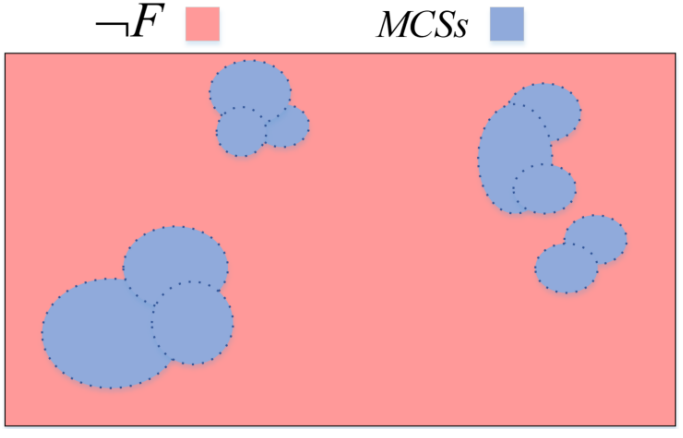
```
...
(e31 & e40 & e34)
(e31 & e40 & e36)
(e31 & e40 & e6)
(e31 & e40 & e47)
(e31 & e40 & e49)
(e31 & e40 & e88)
(e31 & e40 & e3)
(e31 & e40 & e30 & e29)
...
```

All-SAT Problem

Fault Tree	Decide time(s)	BCP time(s)	AnalyzeConflict time(s)	ExtractMCS time(s)	LearnClause time(s)
edf9203_m0	36.3406	6763.56	246.294	777.224	8771.2
edfpa14o_m0	79.3506	5836.03	177.409	130.407	6863.44
edfpa14p_m0	62.2286	5423.57	163.988	123.055	6594.63
edfpa14q_m0	54.2406	4444.27	138.508	148.67	5166.82
edfpa14r_m0	47.3522	3761.06	111.248	146.252	4184.56
edfpa15b_m1	12.1073	796.248	34.5342	128.307	1468.61

- The consumption of time in the LEARNCLAUSE accounts for a large part.

```
...
(e31 & e40 & e34)
(e31 & e40 & e36)
(e31 & e40 & e6)
(e31 & e40 & e47)
(e31 & e40 & e49)
(e31 & e40 & e88)
(e31 & e40 & e3)
(e31 & e40 & e30 & e29)
...
```

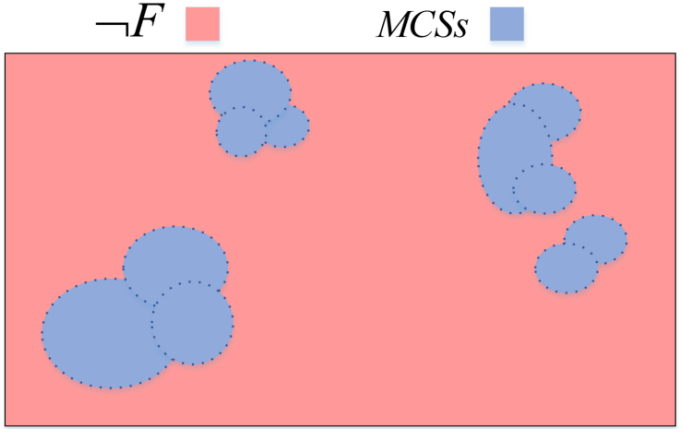


All-SAT Problem

Fault Tree	Decide time(s)	BCP time(s)	AnalyzeConflict time(s)	ExtractMCS time(s)	LearnClause time(s)
edf9203_m0	36.3406	6763.56	246.294	777.224	8771.2
edfpa14o_m0	79.3506	5836.03	177.409	130.407	6863.44
edfpa14p_m0	62.2286	5423.57	163.988	123.055	6594.63
edfpa14q_m0	54.2406	4444.27	138.508	148.67	5166.82
edfpa14r_m0	47.3522	3761.06	111.248	146.252	4184.56
edfpa15b_m1	12.1073	796.248	34.5342	128.307	1468.61

- The consumption of time in the LEARNCLAUSE accounts for a large part.

```
...
(e31 & e40 & e34)
(e31 & e40 & e36)
(e31 & e40 & e6)
(e31 & e40 & e47)
(e31 & e40 & e49)
(e31 & e40 & e88)
(e31 & e40 & e3)
(e31 & e40 & e30 & e29)
...
```



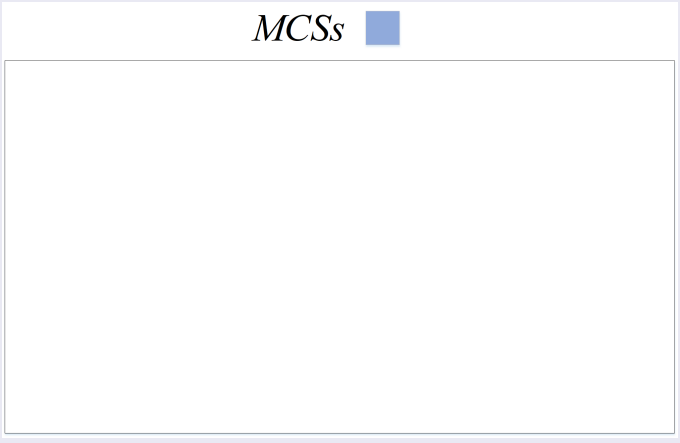
- In the search space, the distribution of MCSs are concentrated.

All-SAT Problem

SATMCS

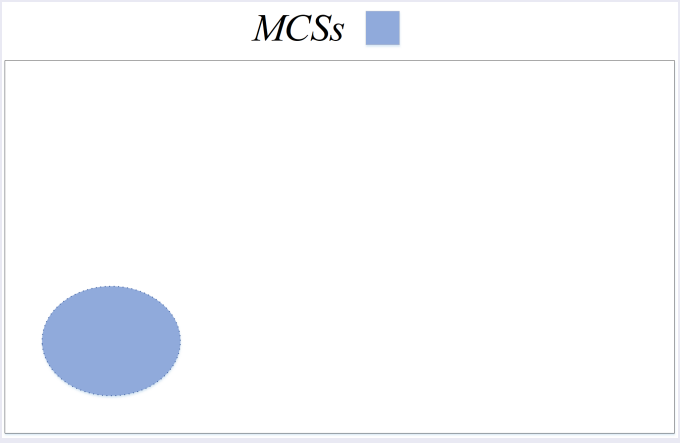
All-SAT Problem

SATMCS



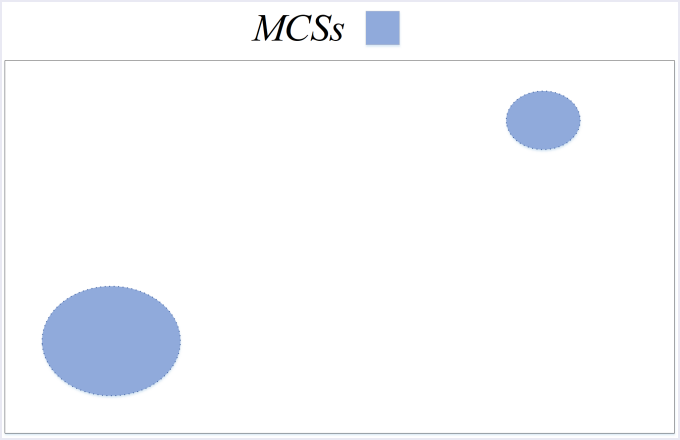
All-SAT Problem

SATMCS



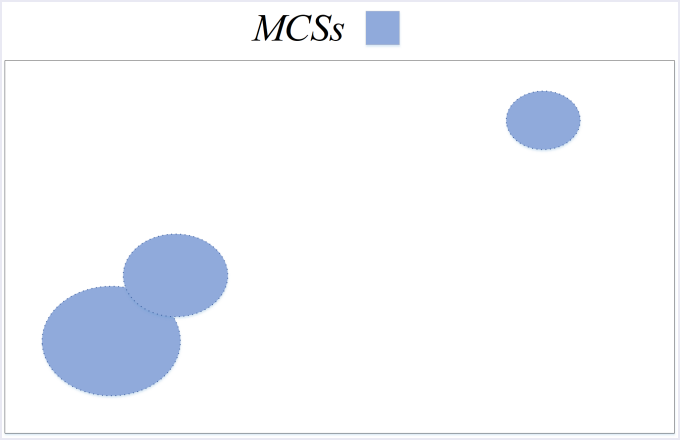
All-SAT Problem

SATMCS



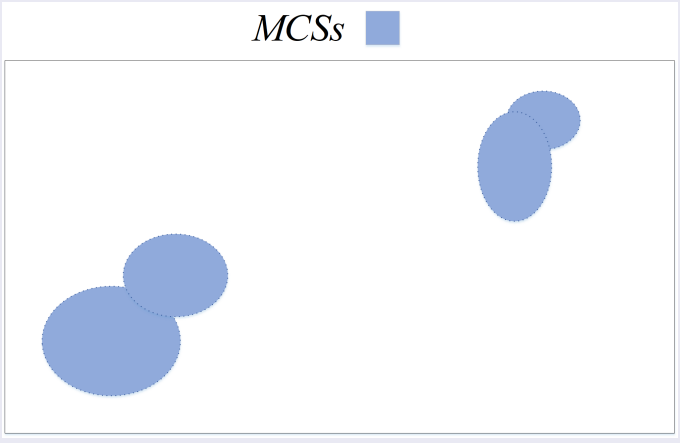
All-SAT Problem

SATMCS



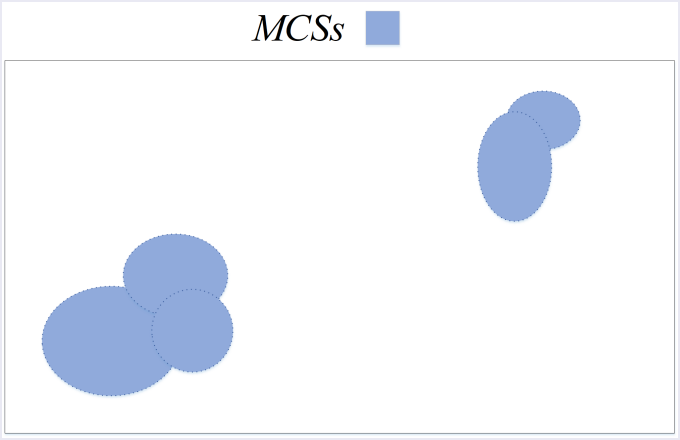
All-SAT Problem

SATMCS



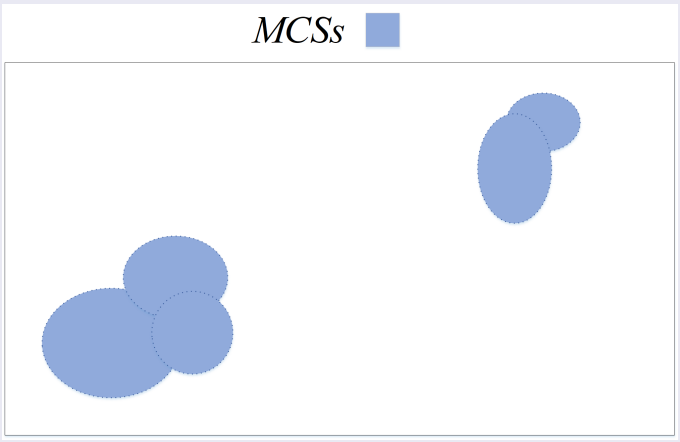
All-SAT Problem

SATMCS



All-SAT Problem

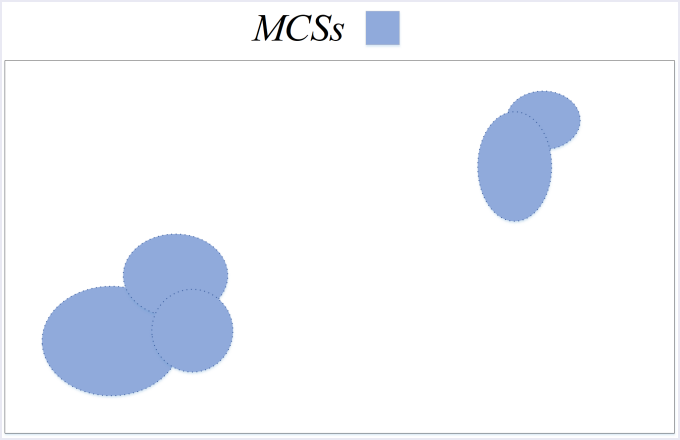
SATMCS



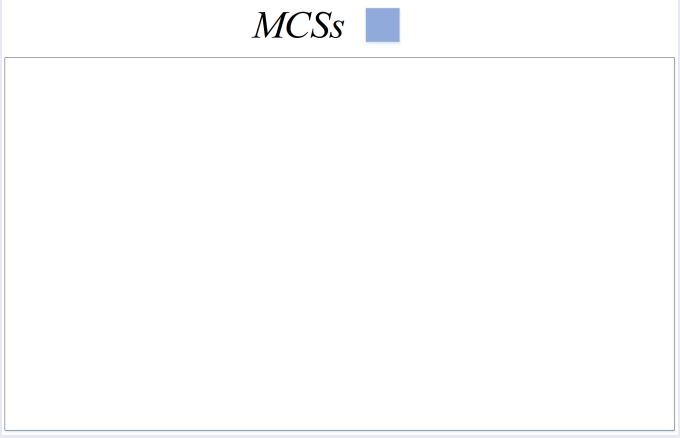
SATMCS with Chronological Backtracking

All-SAT Problem

SATMCS

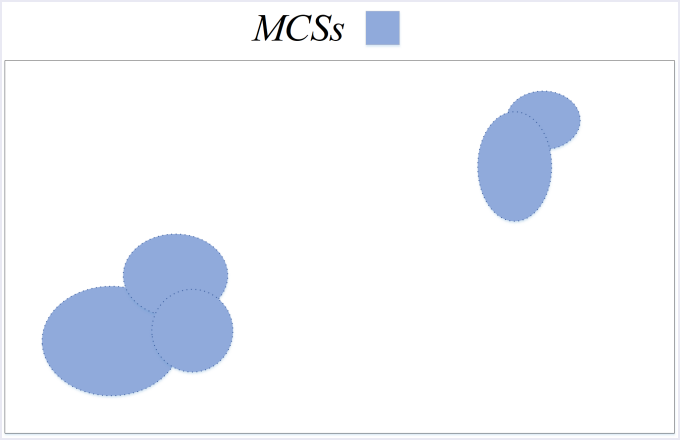


SATMCS with Chronological Backtracking

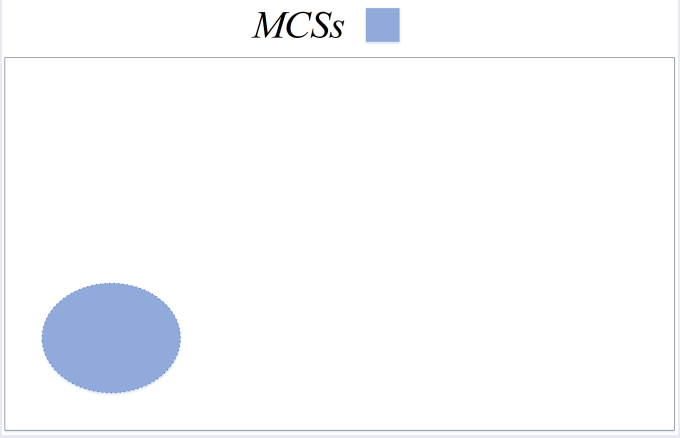


All-SAT Problem

SATMCS

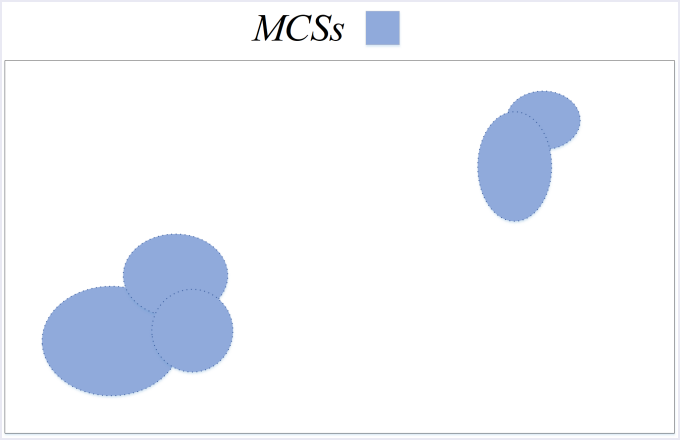


SATMCS with Chronological Backtracking

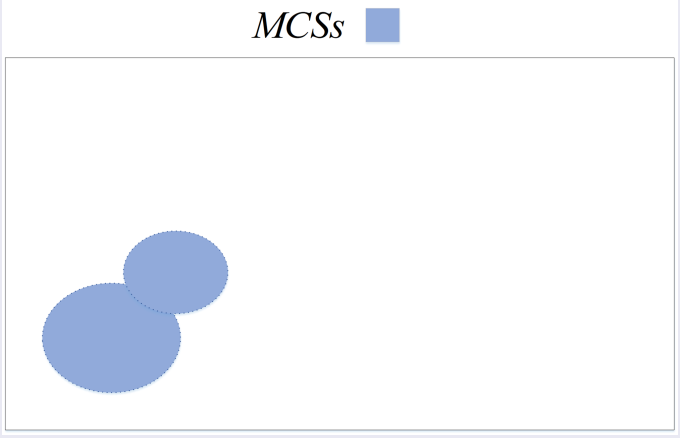


All-SAT Problem

SATMCS

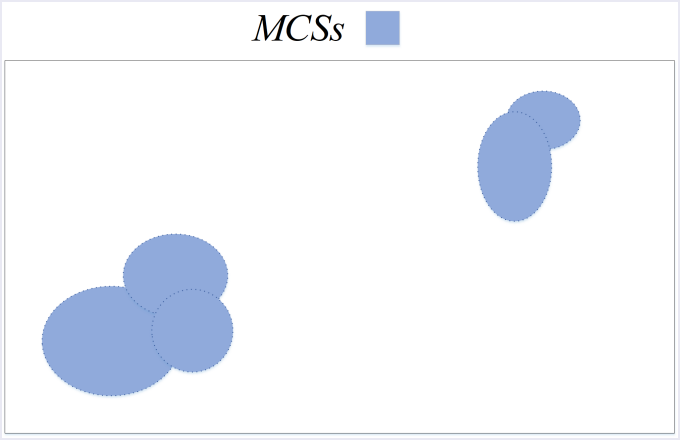


SATMCS with Chronological Backtracking

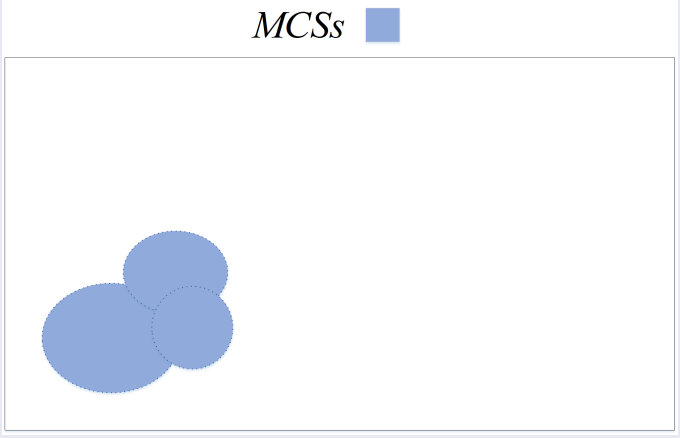


All-SAT Problem

SATMCS

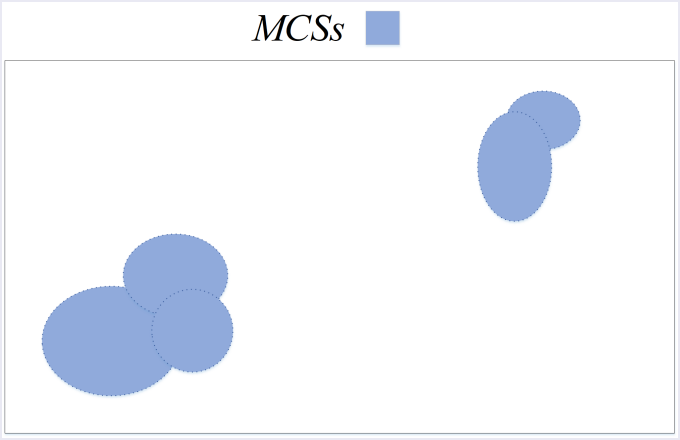


SATMCS with Chronological Backtracking

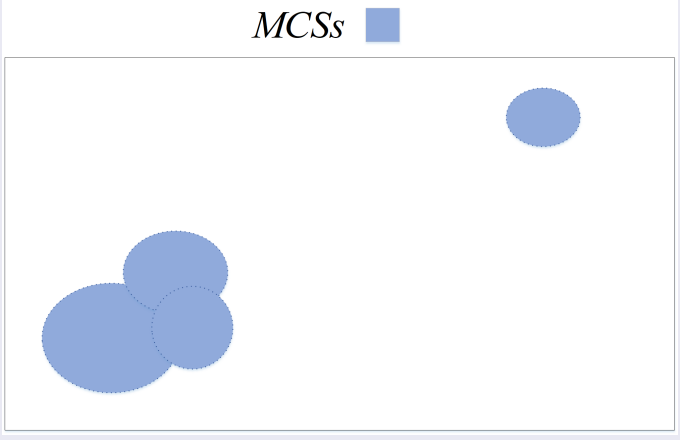


All-SAT Problem

SATMCS

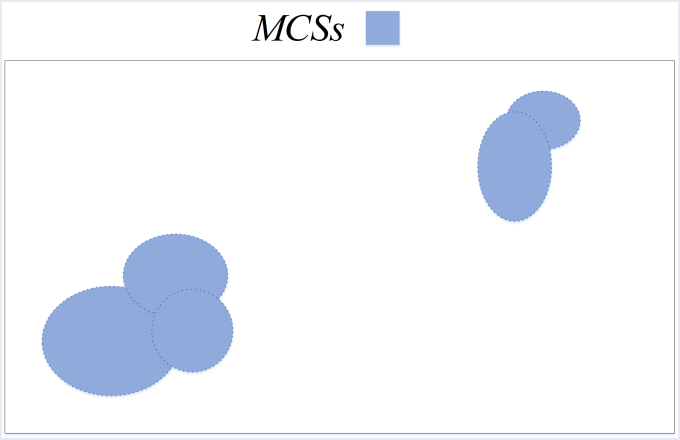


SATMCS with Chronological Backtracking

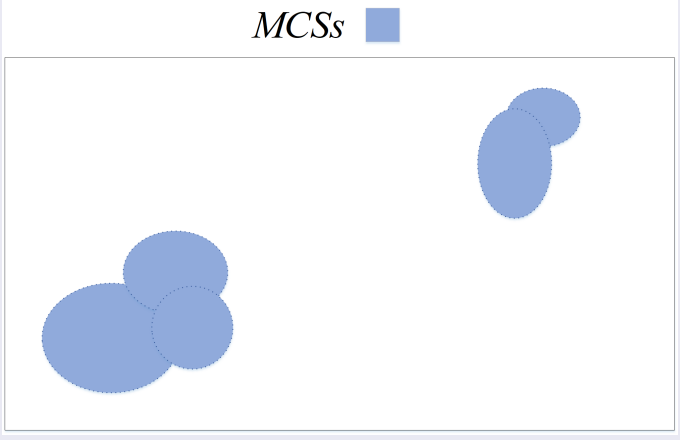


All-SAT Problem

SATMCS

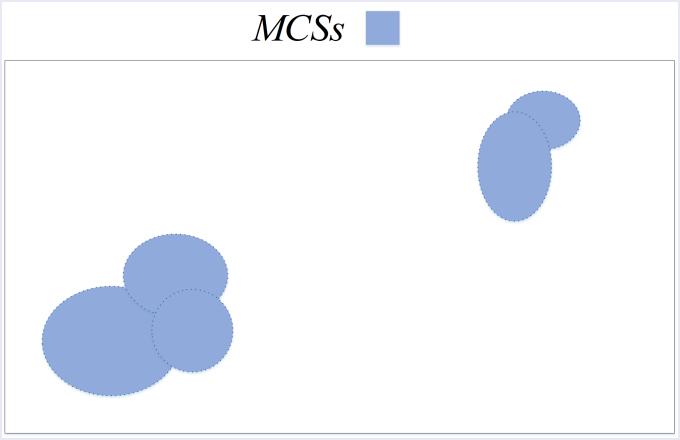


SATMCS with Chronological Backtracking

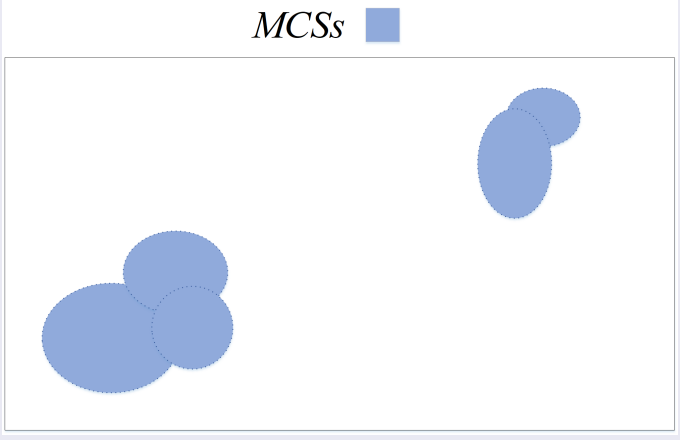


All-SAT Problem

SATMCS



SATMCS with Chronological Backtracking



Benchmark	#E	#G	#MCS	SATMCS		SATMCS-CB	
				Time (sec.)	Mem (MB)	Time (sec.)	Mem (MB)
baobab3	80	107	24386	12.7	66.4	5.4	65.6
chinese	25	36	392	<0.1	3.1	<0.1	31.3
elf9601	145	242	151348	38.4	105.3	270.2	138.9
ftr10	175	94	396	<0.1	3.2	0.1	3.1
jbd9601	533	315	14007	1.6	8.5	1.2	6.1
das9205	51	20	17280	<0.1	3.4	<0.1	3.1
das9203	51	30	16200	<0.1	3.1	<0.1	3
das9204	53	30	16704	<0.1	3.3	<0.1	3
das9202	49	36	27778	<0.1	3.3	<0.1	3.1
das9209	109	73	8.2E10	<0.1	3.1	<0.1	3
das9206	121	112	19518	0.4	5.2	0.2	3.7
das9201	122	82	14217	0.2	4.1	<0.1	3
das9208	103	145	8060	1.9	9.8	0.5	6.3
das9207	276	324	25988	4.7	10.2	1	6.1
edf9205	165	142	21308	0.5	5.2	0.2	3.6
edf9201	183	132	579720	56.1	57.7	0.3	4.5
edf9203	362	475	–	–	–	156.9	539.8
edf9204	323	375	–	–	–	–	–
edf9202	458	435	130112	5.5	9.7	0.7	3.5
edf9206	240	362	–	–	–	–	–
edfpa15r	88	110	26549	26.8	142.1	12.6	134.2
edfpa14r	106	132	–	–	–	1873.8	3009.7
edfpa15p	276	324	27870	22.5	142.3	13.2	151
edfpa14p	124	101	–	–	–	2123.7	3392.1
edfpa14q	311	194	–	–	–	1597.8	2750.2
edfpa15b	283	249	2910473	2662.6	1493.8	12.7	142.2
edfpa14o	311	173	–	–	–	1481.7	3025.1
edfpa14b	311	290	–	–	–	1919.6	3431.4
isp9606	89	41	1776	<0.1	3.1	<0.1	3.1
isp9607	74	65	150436	<0.1	4.2	0.1	3.8
isp9603	91	95	3434	0.1	3.6	0.1	3.5
isp9602	116	122	5197647	1.6	12.6	0.2	5.9
isp9604	215	132	746574	<0.1	3.2	0.2	3.5

Prime Implicant

Definition

- *Implicant*: A term I_n is called an implicant of F if $I_n \rightarrow F$.
- *Prime implicant*: An implicant I_n of F is called prime if any subset $I'_n \subset I_n$ is not an implicant of F.
- The goal is to find prime implicants of non-clausal formulae.

Prime Implicant

Definition

- *Implicant*: A term I_n is called an implicant of F if $I_n \rightarrow F$.
- *Prime implicant*: An implicant I_n of F is called prime if any subset $I'_n \subset I_n$ is not an implicant of F.
- The goal is to find prime implicants of non-clausal formulae.

Same

- Computing MCSs of fault trees can be considered as a special case of enumerating prime implicants of boolean formulae.

Prime Implicant

Definition

- *Implicant*: A term I_n is called an implicant of F if $I_n \rightarrow F$.
- *Prime implicant*: An implicant I_n of F is called prime if any subset $I'_n \subset I_n$ is not an implicant of F.
- The goal is to find prime implicants of non-clausal formulae.

Same

- Computing MCSs of fault trees can be considered as a special case of enumerating prime implicants of boolean formulae.

Method

- Based on BDD; [Coudert and Madre, 1992; Simon and del Val, 2001; Matusiewicz et al., 2009]
  - These methods can compile formulae having at most a couple of hundred variables;
  - That can encode a very large number of primes;
- Based on SAT; [Jabbour, et al., 2014; A. Previti, et al., 2015]

Prime Implicant

The work by [Previti et al., 2015] is the closet to ours, where they proposed algorithms for finding prime implicants of non-clausal formulas based on iterative SAT solving.

Prime Implicant

The work by [Previti et al., 2015] is the closet to ours, where they proposed algorithms for finding prime implicants of non-clausal formulas based on iterative SAT solving.

Feature

- This approach uses a SAT solver as a black box;
- It doesn't take the characteristic of boolean formulae into account;
- Their approach uses *dual rail encoding* to ensure the computation of the complete set of prime implicants, and can be adapted for computing MCSs of fault trees;

Prime Implicant

The work by [Previti et al., 2015] is the closet to ours, where they proposed algorithms for finding prime implicants of non-clausal formulas based on iterative SAT solving.

Feature

- This approach uses a SAT solver as a black box;
- It doesn't take the characteristic of boolean formulae into account;
- Their approach uses *dual rail encoding* to ensure the computation of the complete set of prime implicants, and can be adapted for computing MCSs of fault trees;

Dual Rail Encoding

- For each $v \in E_F$, the pair of variables $x_v, x_{\neg v}$ encodes the non-occurence, the negative or the positive occurrence of v in a possible prime implicant of F;
- $\mathcal{H} = \mathcal{F}_{\mathcal{D}} \cup \mathcal{B} \cup \mathcal{C} \cup \mathcal{L}$ where $\mathcal{L} = \{\neg x_v \vee \neg x_{\neg v} | v \in E_F\}$;

Comparison between SATMCS and Primer

Benchmark	#E	#G	#MCS	SATMCS		primer	
				Time (sec.)	Mem (MB)	Time (sec.)	Mem (MB)
baobab3	80	107	24386	12.7	66.4	–	–
chinese	25	36	392	<0.1	3.1	<0.1	1.3
elf9601	145	242	151348	38.4	105.3	–	–
ftr10	175	94	396	<0.1	3.2	10.6	8
jbd9601	533	315	14007	1.6	8.5	–	–
das9205	51	20	17280	<0.1	3.4	0.2	4.8
das9203	51	30	16200	<0.1	3.1	0.2	3.7
das9204	53	30	16704	<0.1	3.3	1.2	7.9
das9202	49	36	27778	<0.1	3.3	41.5	23
das9209	109	73	8.2E10	<0.1	3.1	–	–
das9206	121	112	19518	0.4	5.2	46.7	26.1
das9201	122	82	14217	0.2	4.1	150.3	81.2
das9208	103	145	8060	1.9	9.8	5	7
das9207	276	324	25988	4.7	10.2	–	–
edf9205	165	142	21308	0.5	5.2	56.3	39.3
edf9201	183	132	579720	56.1	57.7	632.1	244
edf9203	362	475	–	–	–	–	–
edf9204	323	375	–	–	–	–	–
edf9202	458	435	130112	5.5	9.7	–	–
edf9206	240	362	–	–	–	–	–
edfpa15r	88	110	26549	26.8	142.1	2308.3	94.4
edfpa14r	106	132	–	–	–	–	–
edfpa15p	276	324	27870	22.5	142.3	1899.1	78.9
edfpa14p	124	101	–	–	–	–	–
edfpa14q	311	194	–	–	–	–	–
edfpa15b	283	249	2910473	2662.6	1493.8	–	–
edfpa14o	311	173	–	–	–	–	–
edfpa14b	311	290	–	–	–	–	–
isp9606	89	41	1776	<0.1	3.1	56.6	53.1
isp9607	74	65	150436	<0.1	4.2	299.4	170.5
isp9603	91	95	3434	0.1	3.6	446.7	28.1
isp9602	116	122	5197647	1.6	12.6	–	–
isp9604	215	132	746574	<0.1	3.2	–	–

Result

Comparison between SATMCS and Primer

Benchmark	#E	#G	#MCS	SATMCS		primer	
				Time (sec.)	Mem (MB)	Time (sec.)	Mem (MB)
baobab3	80	107	24386	12.7	66.4	–	–
chinese	25	36	392	<0.1	3.1	<0.1	1.3
elf9601	145	242	151348	38.4	105.3	–	–
ftr10	175	94	396	<0.1	3.2	10.6	8
jbd9601	533	315	14007	1.6	8.5	–	–
das9205	51	20	17280	<0.1	3.4	0.2	4.8
das9203	51	30	16200	<0.1	3.1	0.2	3.7
das9204	53	30	16704	<0.1	3.3	1.2	7.9
das9202	49	36	27778	<0.1	3.3	41.5	23
das9209	109	73	8.2E10	<0.1	3.1	–	–
das9206	121	112	19518	0.4	5.2	46.7	26.1
das9201	122	82	14217	0.2	4.1	150.3	81.2
das9208	103	145	8060	1.9	9.8	5	7
das9207	276	324	25988	4.7	10.2	–	–
edf9205	165	142	21308	0.5	5.2	56.3	39.3
edf9201	183	132	579720	56.1	57.7	632.1	244
edf9203	362	475	–	–	–	–	–
edf9204	323	375	–	–	–	–	–
edf9202	458	435	130112	5.5	9.7	–	–
edf9206	240	362	–	–	–	–	–
edfpa15r	88	110	26549	26.8	142.1	2308.3	94.4
edfpa14r	106	132	–	–	–	–	–
edfpa15p	276	324	27870	22.5	142.3	1899.1	78.9
edfpa14p	124	101	–	–	–	–	–
edfpa14q	311	194	–	–	–	–	–
edfpa15b	283	249	2910473	2662.6	1493.8	–	–
edfpa14o	311	173	–	–	–	–	–
edfpa14b	311	290	–	–	–	–	–
isp9606	89	41	1776	<0.1	3.1	56.6	53.1
isp9607	74	65	150436	<0.1	4.2	299.4	170.5
isp9603	91	95	3434	0.1	3.6	446.7	28.1
isp9602	116	122	5197647	1.6	12.6	–	–
isp9604	215	132	746574	<0.1	3.2	–	–

Result

- Primer relies on the QuickXplain algorithm for extracting prime implicants from satisfying assignments; it may require more number of satisfiability queries than our approach;