

Lecture 10

Pete Manolios
Northeastern

SAT, NP-Completeness

▶ kSAT

- ▶ Literals: variables or their negations
 - ▶ Clause: disjunction of literals
 - ▶ CNF formula (Conjunctive Normal Form): conjunction of clauses
 - ▶ kCNF: CNF formula w/ at most k literals per clause
 - ▶ kSAT: The set of satisfiable kCNF formulas
- ▶ Recall: SAT (= set of satisfiable propositional formulas) is NP-complete
- ▶ NP: languages whose membership can be verified in P-time
 - ▶ NPC: Hardest problems in NP
 - ▶ A P-time algorithm for 1 NPC problem implies P-time algorithms for every problem in NP
- ▶ 3SAT is NP-complete (Cook 1971)
- ▶ $SAT \leq_p 3SAT$: define P-time function F s.t. $x \in SAT$ iff $F(x) \in 3SAT$

Tseitin Transformation

- ▶ Goal: define a P-time function F s.t. $x \in \text{SAT}$ iff $F(x) \in \text{3SAT}$
- ▶ Is there a P-time function F that given g , returns equivalent CNF formula?
 - ▶ No: just going from DNF to CNF is *provably* exponential in the worst case
- ▶ Idea: define a P-time (linear) function that preserves satisfiability
- ▶ Example: $(p \vee (q \wedge \neg r)) \wedge s$
- ▶ Introduce definition d_1 : $d_1 = q \wedge \neg r$
- ▶ Example is now in CNF: $(p \vee d_1) \wedge s$
- ▶ Turn definition into CNF: $(\neg d_1 \vee q) \wedge (\neg d_1 \vee \neg r) \wedge (\neg q \vee r \vee d_1)$
- ▶ Finally: $(p \vee d_1) \wedge s \wedge (\neg d_1 \vee q) \wedge (\neg d_1 \vee \neg r) \wedge (\neg q \vee r \vee d_1)$
- ▶ Not equivalent, but *equisatisfiable*
- ▶ Using this idea, we recursively introduce definitions as needed

2SAT is in P

- ▶ SAT and even 3SAT are NPC. What about 2SAT?
- ▶ Construct a graph corresponding to a 2CNF formula F
 - ▶ Vertices: $\{ x, \neg x : x \text{ a variable} \}$
 - ▶ Edges: $\{ (u, v), (\neg v, \neg u) : (\neg u, v) \text{ in } F \}$ (u, v are literals)
- ▶ Lemma 1: $u \rightarrow v$ (a path from u to v) implies $\neg v \rightarrow \neg u$, $u \Rightarrow v$
- ▶ Lemma 2: F is unsatisfiable iff there is a variable x , st:
 - ▶ there is a path from x to $\neg x$, and
 - ▶ there is a path from $\neg x$ to x
- ▶ Proof: Pong: $(A \Leftarrow B)$ (lemma 1) Ping: $(\neg A \Leftarrow \neg B)$ (which is equivalent to $(A \Rightarrow B)$)
 - ▶ choose unassigned u s.t. no path to $\neg u$ exists; set u , reachable vertices to true; repeat; notice if $u \rightarrow x, \neg x$, then $\neg x \rightarrow \neg u$, so $u \rightarrow \neg u$
- ▶ 2SAT can be solved in linear time
- ▶ Run SCC: SAT iff no component contains a literal and its negation

2SAT Example

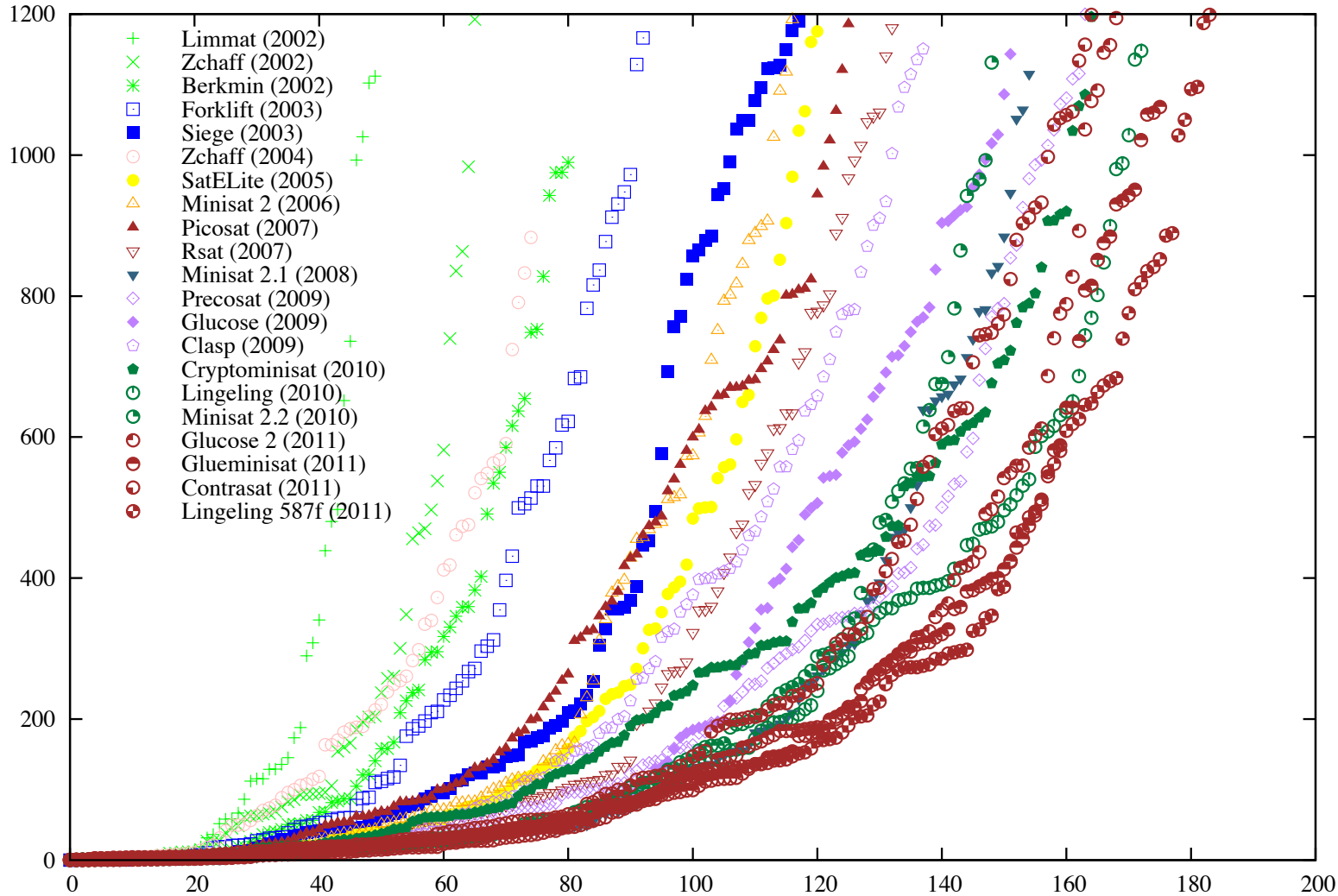
- ▶ $(p \vee q) \wedge (p \vee \neg q) \wedge \neg p$?
- ▶ Construct a graph corresponding to a 2CNF formula F
 - ▶ Vertices: $\{ x, \neg x : x \text{ a variable} \}$
 - ▶ Edges: $\{ (u, v), (\neg v, \neg u) : (\neg u, v) \text{ in } F \}$ (u, v are literals)
- ▶ Vertices: $\{ p, \neg p, q, \neg q \}$
- ▶ Try it!
- ▶ Note: for unit clause $\neg p$, treat it as $\neg p \vee \neg p$
- ▶ Edges: $\{ (\neg p, q), (\neg q, p), (\neg p, \neg q), (q, p), (p, \neg p) \}$
- ▶ SCC: $\{ \{ \neg p, q, p, \neg p \} \}$, so UNSAT

SAT Algorithms

- ▶ SAT used everywhere: Logic, AI, SAT, scheduling, circuit design, game theory, verification (predicate abstraction), reliability theory, etc.
- ▶ Theory vs practice: NPC problem, so what do we do?
 - ▶ Option1: give up; not feasible
 - ▶ Option2: come up with techniques that work on interesting problems
- ▶ If the problem is important, we go with option2
- ▶ Multi-billion dollar industries are based on SAT
- ▶ An even more extreme example: termination analysis
- ▶ When reasoning about systems, SAT is one of the simplest problems we have to deal with

Cactus Plots

Results of the SAT competition/race winners on the SAT 2009 application benchmarks, 20mn timeout



From: Le Berre&Biere 2011

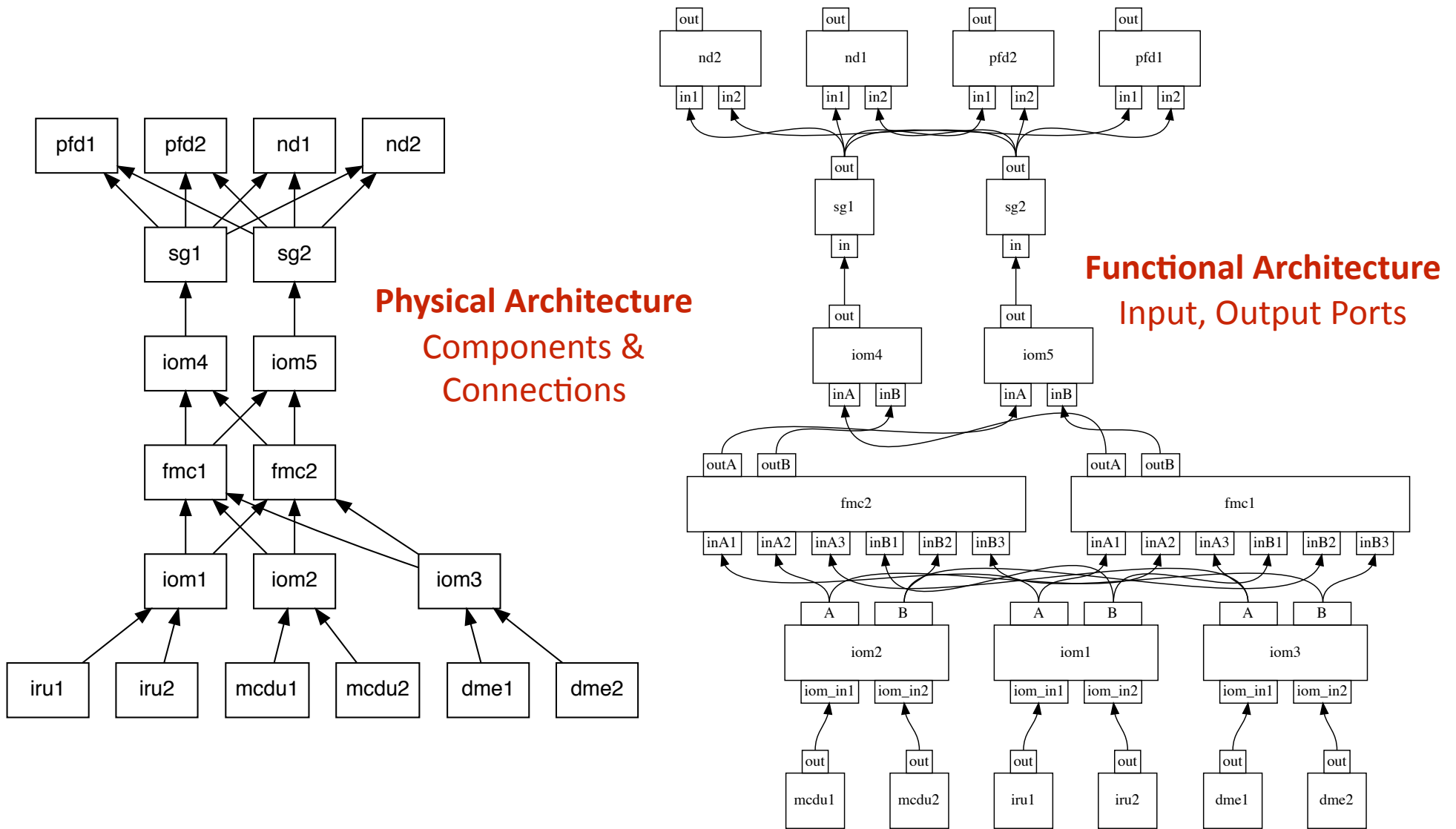
Slides by Pete Manolios for CS4820

Safety Analysis Example

- ▶ Let's look at one example in more depth which highlights some interesting issues: modeling, formula simplification, probability.
- ▶ See <https://github.com/pmanolios/safety-analysis>
- ▶ Look at Webpage
- ▶ run `make 777-example-pdf.out`

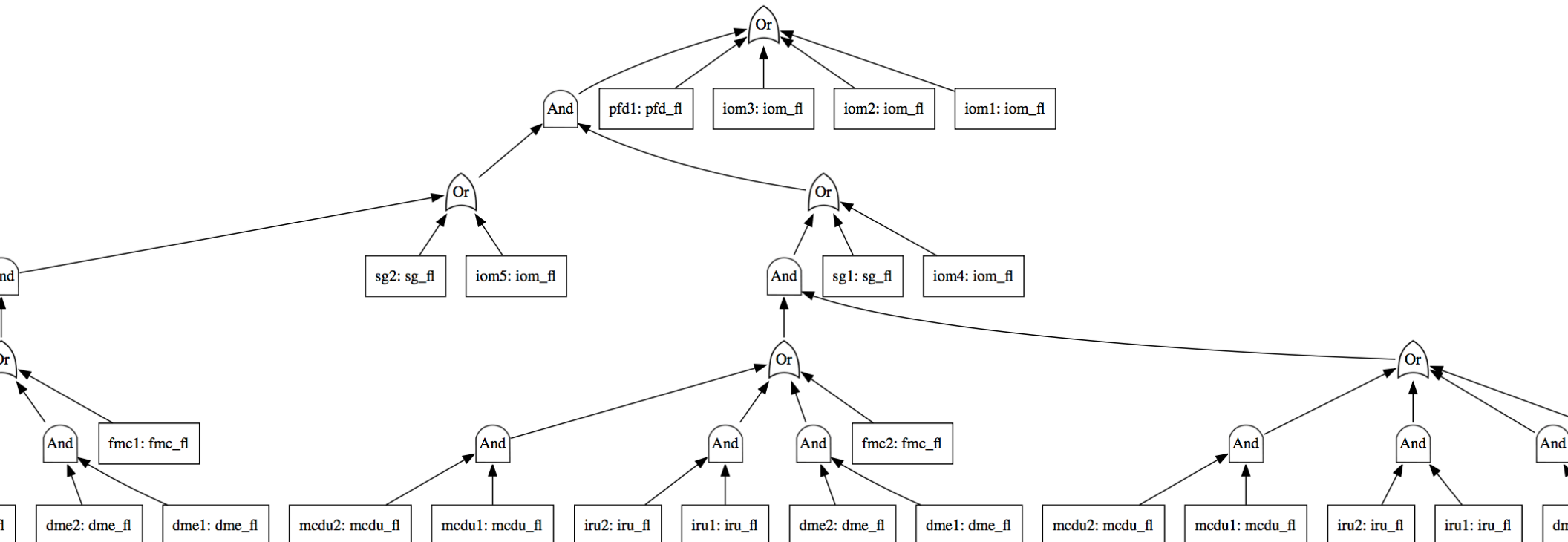
777-Inspired Architecture

Modeling loss of display of navigation information



Automated Fault Tree Synthesis

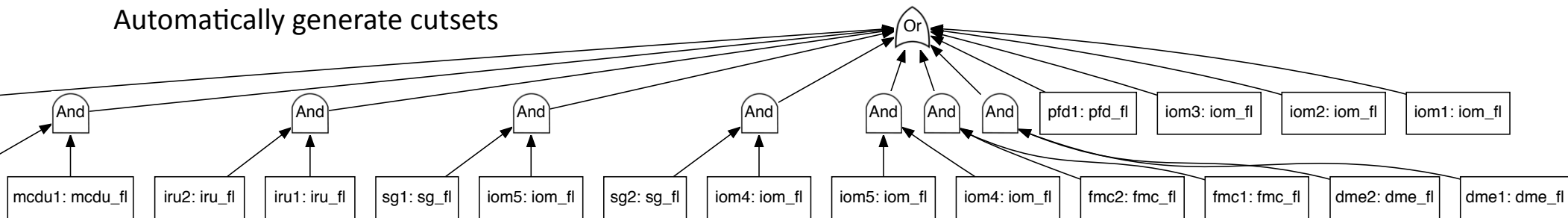
Fault trees automatically synthesized from models
 Model update, reuse, maintenance, verification, analysis capabilities



Architectural Analysis

Quantitative analysis:
Top level fault probability 3.0002e-06

Qualitative analysis:
Automatically generate cutsets



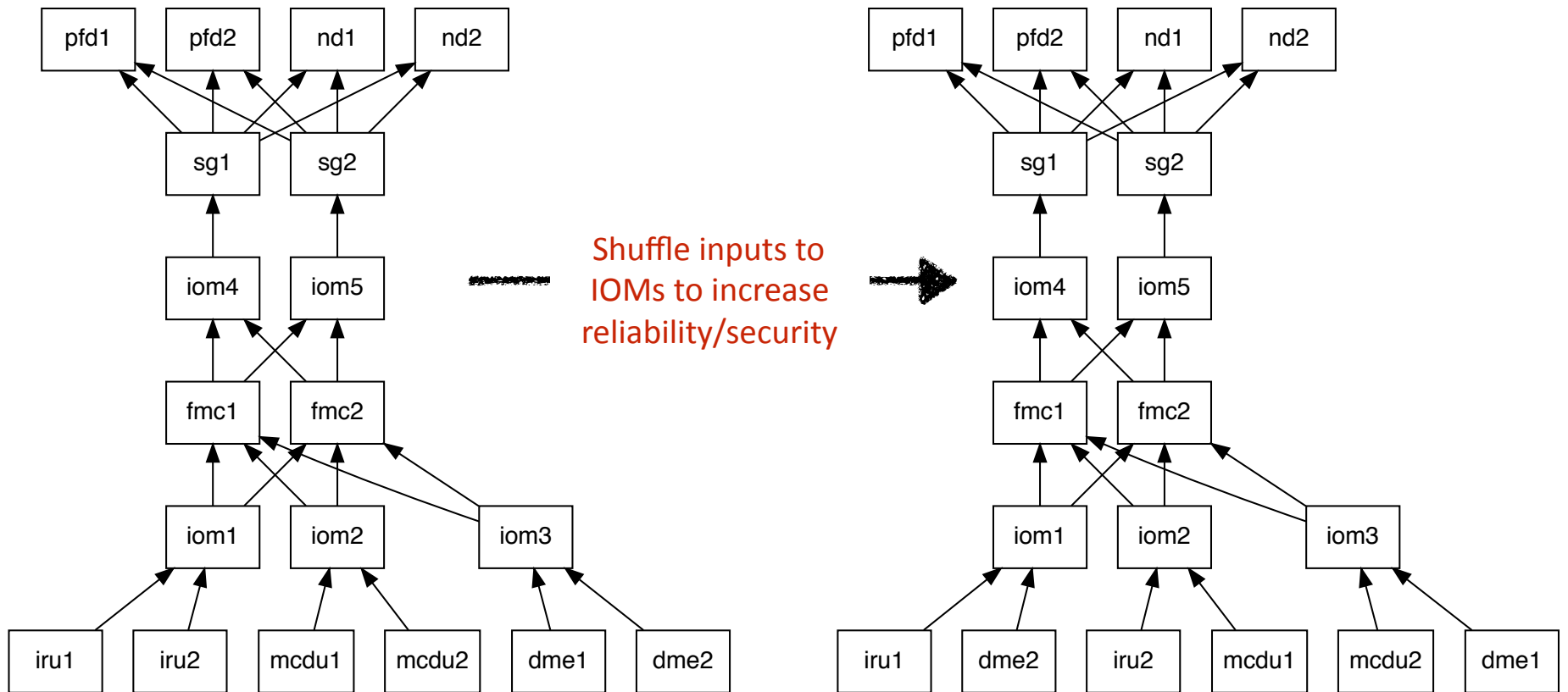
Components	Probability	Importance Measure
iom1:iom_fl	9.9999E-07	0.3333
iom2:iom_fl	9.9999E-07	0.3333
iom3:iom_fl	9.9999E-07	0.3333
pfd1:pfd_fl	2.0000E-10	6.6662E-05
dme1:dme_fl; dme2:dme_fl	9.9999E-13	3.3331E-07

Automatically generate importance measures

IOMs contribute most to top-level probability. Remove the single points of failure.



Architectural Trades/Synthesis



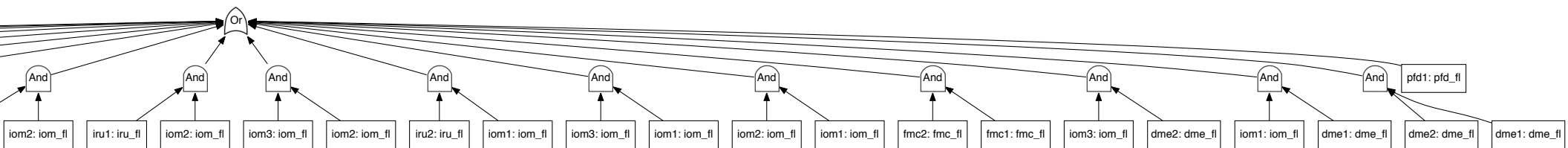
Architectural Analysis

Quantitative analysis:

Top level fault probability 2.1599e-10 (before: 3.0002e-06)

Qualitative analysis:

Automatically generate cutsets

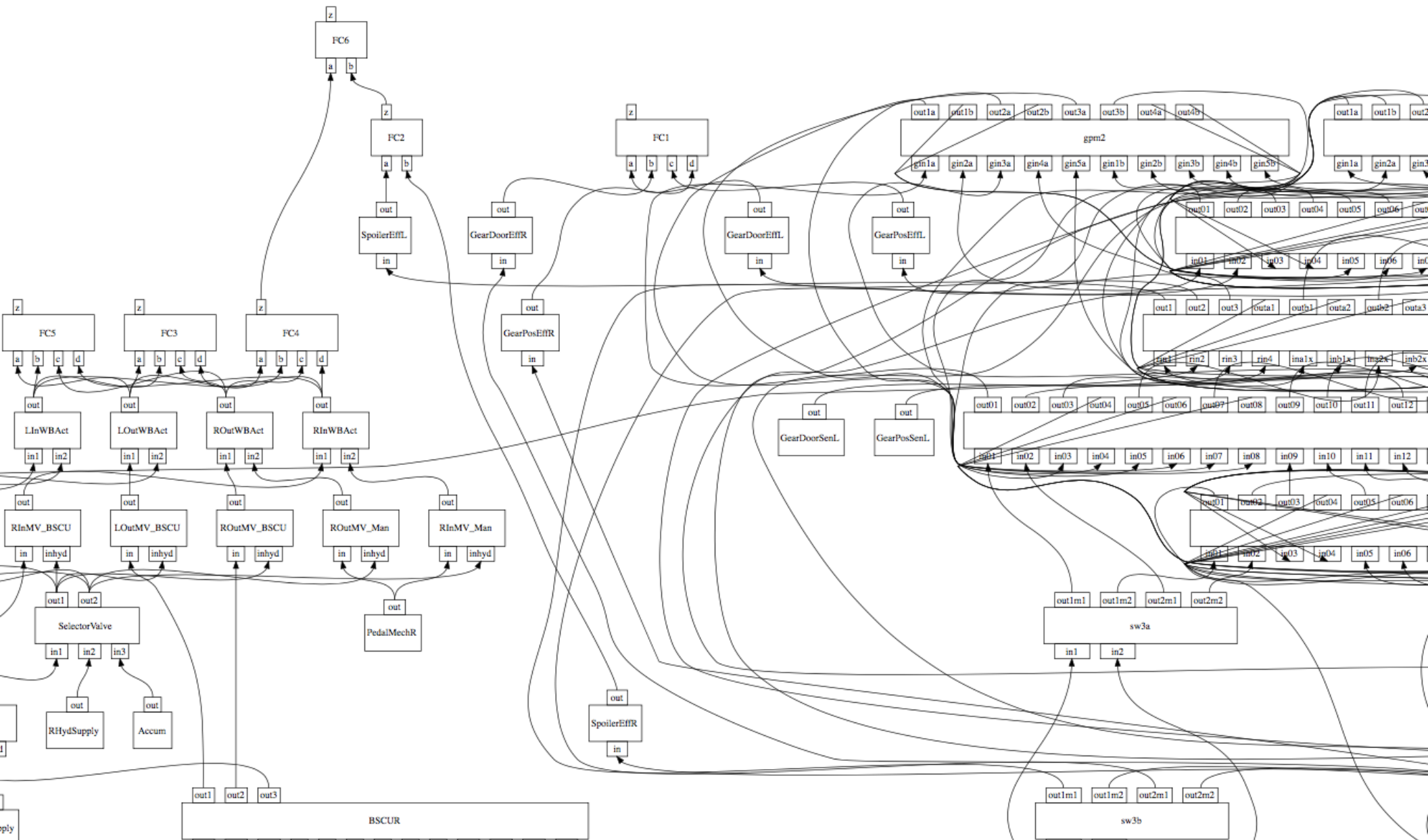


Components	Probability	Importance Measure
pfd1:pfd_fl	2.0000E-10	0.9259
dme1:dme_fl; dme2:dme_fl	9.9999E-13	0.0046
dme1:dme_fl; iom1:iom_fl	9.9999E-13	0.0046
iom1:iom_fl; iom3:iom_fl	9.9999E-13	0.0046
iru1:iru_fl; iru2:iru_fl	9.9999E-13	0.0046

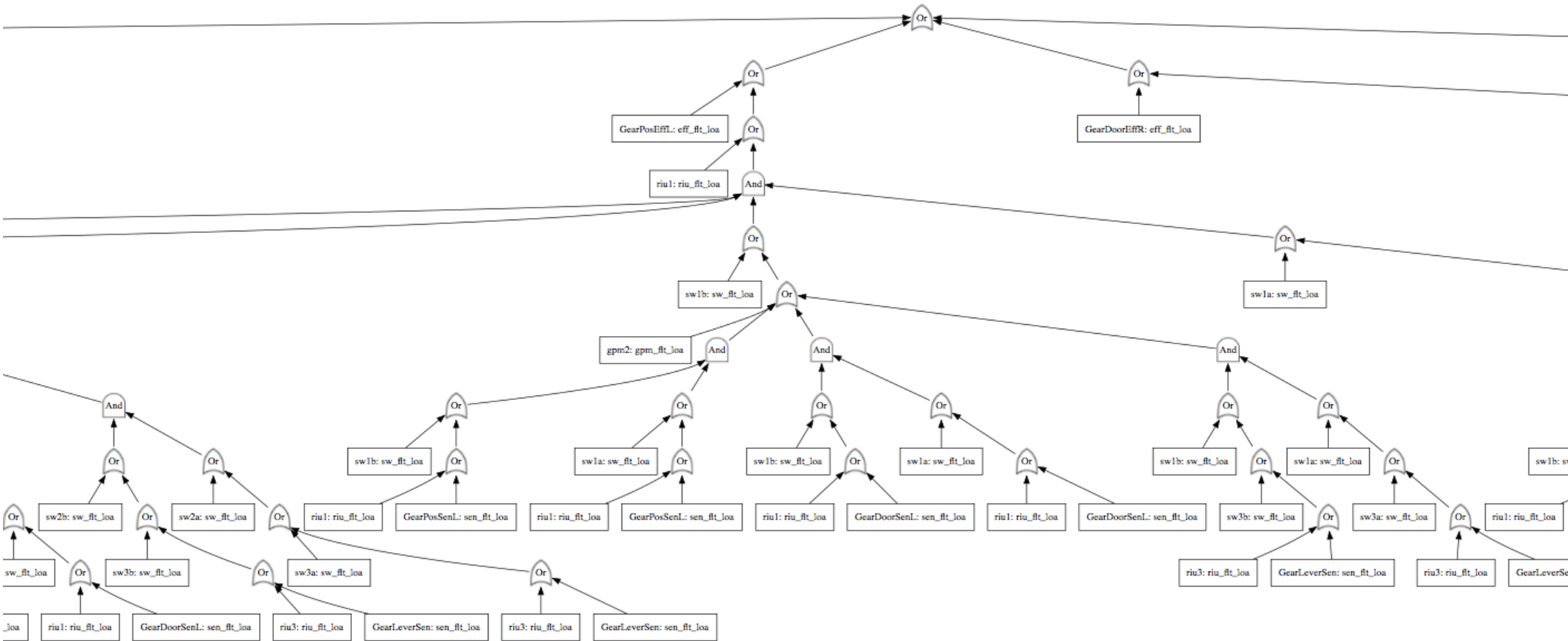
Automatically generate importance measures

IOMs are no longer single points of failure.
You need to compromise at least one other system for a failure.

Functional Architecture



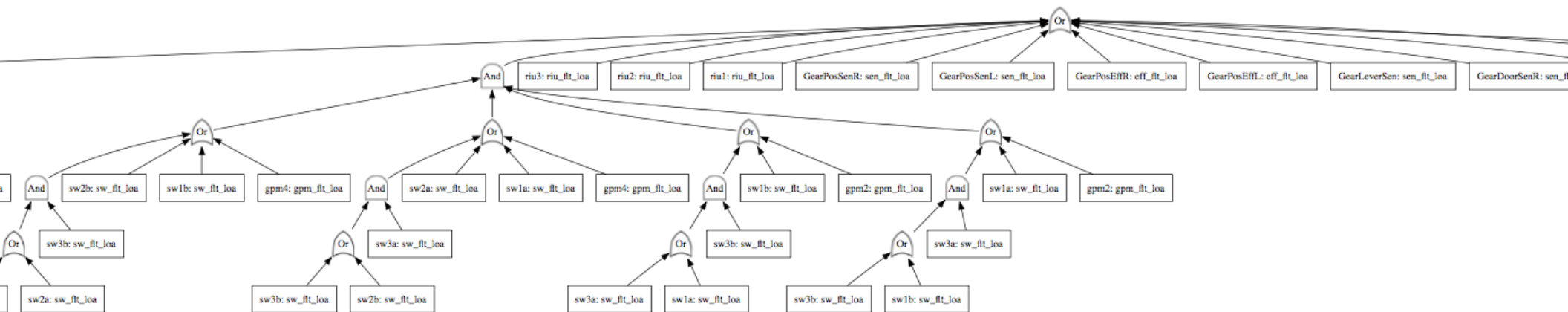
Fault Tree Synthesized



Large fault tree: 549 nodes, 261 Gates, Depth 11

The problem: safety engineer has to perform mental gymnastics to understand
The challenge: can we generate fault trees that a human might generate?

Fault Tree Reduction Engine



Nodes: 549 \rightarrow 83, Gates: 261 \rightarrow 27, Depth: Depth 11 \rightarrow 5

Notes: reduction engine does not increase depth

returns semantically equivalent tree

yields significant simplification of fault tree

fault trees now look like something a human generated