Interpretable Verification of Distributed Protocols by Inductive Proof Decomposition

Will Schultz, Stavros Tripakis
MongoDB Research, Northeastern University

Northeastern University

Heidi Howard, Eddy Ashton
Microsoft Research

Microsoft

Distributed systems are everywhere.

[1] George Pirlea, List of bugs found in distributed protocols, https://github.com/dranov/protocol-bugs-list

Distributed systems are everywhere.

Distributed systems are hard.

Table of errors

Protocol	Reference	Violation	Counter-example
PBFT[1]	[Castro and Liskov 1999]	liveness	[Berger et al. 2021]
Chord	[Stoica et al. 2001; Liben-Nowell et al. 2002]	liveness[2]	[Zave 2012; Zave 2017]
Pastry	[Rowstron and Druschel 2001]	safety	[Azmy et al. 2016; Azmy et al. 2018]
Generalised Paxos	[Lamport 2005]	non-triviality[3]	[Sutra and Shapiro 2010]
FaB Paxos	[Martin and Alvisi 2005; Martin and Alvisi 2006]	liveness	[Abraham et al. 2017]
Multi-Paxos[4]	[Chandra et al. 2007]	safety	[Michael et al. 2017]
Zyzzyva	[Kotla et al. 2007; Kotla et al. 2010]	safety	[Abraham et al. 2017]
CRAQ	[Terrace and Freedman 2009]	safety[5]	[Whittaker 2020]
JPaxos	[Kończak et al. 2011]	safety	[Michael et al. 2017]
VR Revisited	[Liskov and Cowling 2012]	safety	[Michael et al. 2017]
EPaxos	[Moraru et al. 2013]	safety	[Sutra 2020]
EPaxos	[Moraru et al. 2013]	safety	[Whittaker 2021]
Raft	[Ongaro and Ousterhout 2014]	liveness[6]	[Hoch 2014]
Raft	[Ongaro 2014]	safety[7]	[Amos and Zhang 2015; Ongaro 2015]
Raft	[Ongaro and Ousterhout 2014; Ongaro 2014]	liveness	[Howard and Abraham 2020; Jensen et al. 2021]
hBFT	[Duan et al. 2015]	safety	[Shrestha et al. 2019]
Tendermint	[Buchman 2016]	liveness	[Cachin and Vukolić 2017]
CAESAR	[Arun et al. 2017]	liveness	[Enes et al. 2021]
DPaxos	[Nawab et al. 2018]	safety	[Whittaker et al. 2021]
Sync HotStuff	[Abraham et al. 2019]	safety & liveness	[Momose and Cruz 2019]
Gasper	[Buterin et al. 2020]	safety & liveness	[Neu et al. 2021]

On the correctness of Egalitarian Paxos

Pierre Sutra

Télécom SudParis
9, rue Charles Fourier
91000 Évry, France

Abstract

This paper identifies a problem in both the TLA^+ specification and the implementation of the Egalitarian Paxos protocol. It is related to how replicas switch from one ballot to another when computing the dependencies of a command. The problem may lead replicas to diverge and break the linearizability of the replicated service.

bug in single-server membership changes 4974 views

Subscribe

onga...@gmail.com
to raft...@googlegroups.com

Jul 10, 2015, 12:58:53 AM

Hi raft-dev,

Unfortunately, I need to announce a bug in the dissertation version of membership changes (the single-server changes, not joint consensus). The bug is potentially severe, but the fix I'm proposing is easy to implement.

When Huanchen Zhang and Brandon Amos were working on a class project at CMU to formalize single-server membership changes, Huanchen found the bug and the counter-example below by hand. They contacted me over email on May 14th, and I chose to keep this quiet for a while until we had agreed upon a solution to propose to the list. After several incorrect and/or ugly attempts, I came up with the solution proposed below. I apologize for keeping this information from you for so long.

[1] George Pirlea, List of bugs found in distributed protocols, https://github.com/dranov/protocol-bugs-list

Need for formal models and verification.

DOI:10.1145/2699417

Engineers use TLA+ to prevent serious but subtle bugs from reaching production.

BY CHRIS NEWCOMBE, TIM RATH, FAN ZHANG, BOGDAN MUNTEANU, MARC BROOKER, AND MICHAEL DEARDEUFF

How Amazon Web Services Uses Formal Methods

S3 is just one of many AWS services that store and process data our customers have entrusted to us. To safeguard that data, the core of each service relies on fault-tolerant distributed algorithms for replication, consistency, concurrency control, auto-scaling, load balancing, and other coordination tasks. There are many such algorithms in the literature, but combining them into a cohesive system is a challenge, as the algorithms must usually be modified to interact properly in a real-world system. In addition, we have found it necessary to invent algorithms of our own. We work hard to avoid unnecessary complexity, but the essential complexity of the task remains high.

Complexity increases the probability of human error in design, code, and operations. Errors in the core of the system could cause loss or corruption of data, or violate other interface contracts on which our customers depend. So, before launching a service, we need to reach extremely high con-

Planning for Change in a Formal Verification of the Raft Consensus Protocol

Doug Woos James R. Wilcox Steve Anton
Zachary Tatlock Michael D. Ernst Thomas Anderson
University of Washington, USA
{dwoos, jrw12, santon, ztatlock, mernst, tom}@cs.washington.edu

Abstract

We present the first formal verificat
Raft consensus protocol, a critica
systems. We connected our proof
end-to-end guarantee that our imp

IronFleet: Proving Practical Distributed Systems Correct

Chris Hawblitzel, Jon Howell, Manos Kapritsos, Jacob R. Lorch,
Bryan Parno, Michael L. Roberts, Srinath Setty, Brian Zill
Microsoft Research

Abstract

Distributed systems are notorious for harboring subtle bugs. Verification can, in principle, eliminate these bugs a priori,

This paper presents IronFleet, the first methodology for automated machine-checked verification of the safety and liveness of non-trivial distributed system implementations.

Finding Invariants of Distributed Systems: It's a Small (Enough) World After All

Travis Hance, Marijn Heule, Ruben Martins, and Bryan Parno
Carnegie Mellon University

Abstract

Today's distributed systems are increasingly complex, leading to subtle bugs that are difficult to detect with standard testing methods. Formal verification can provably rule out such bugs,

To reduce this cost, some work has developed specialized languages that either restrict the kinds of systems that can be encoded [12, 13, 36, 53] (e.g., the protocol must proceed in synchronous rounds), or restrict the language used to describe the systems' properties [42]. In exchange for these restrictions,

Formal Safety Verification

Model protocols as *state transition systems* $M = (Init, Next)$.

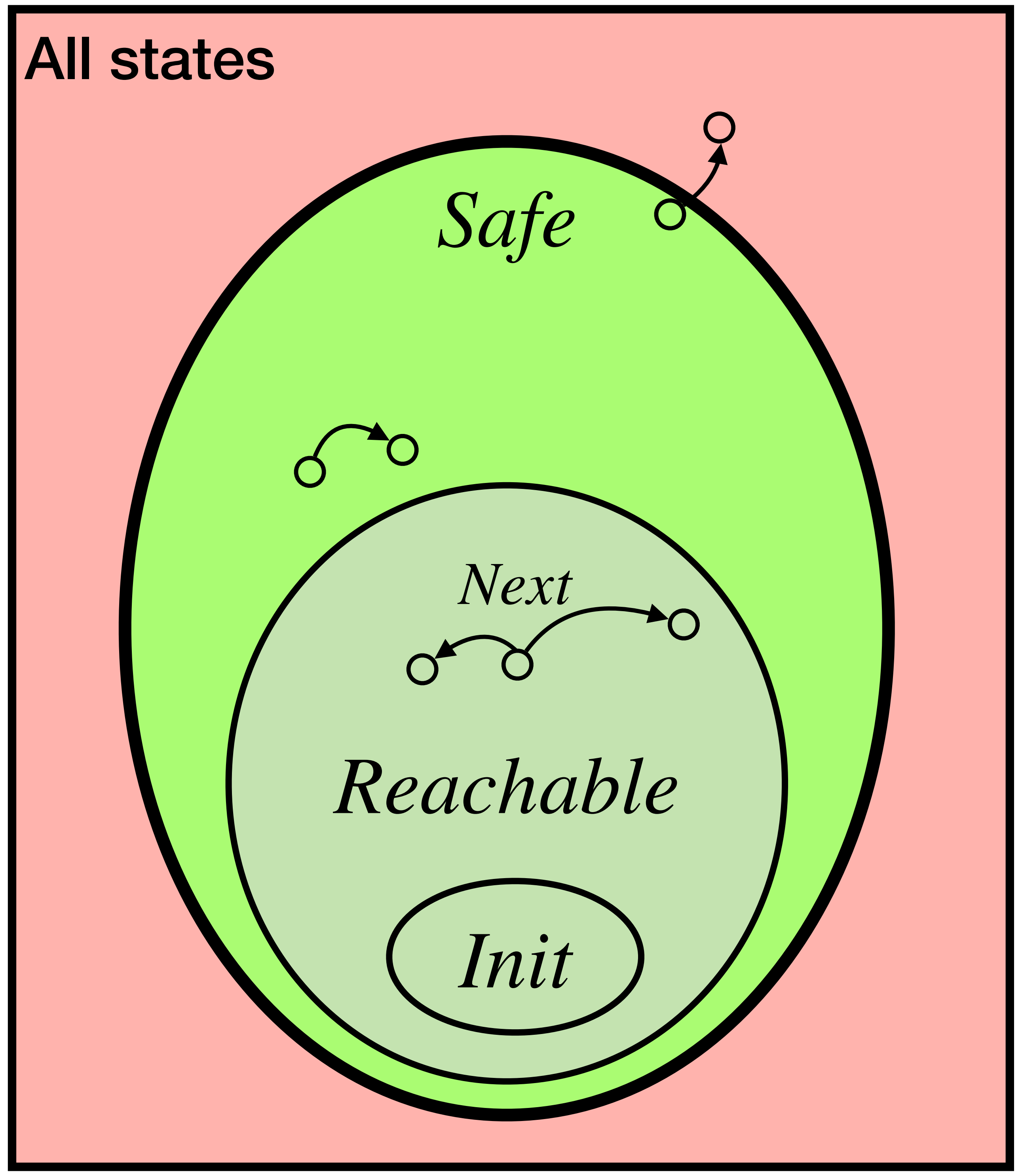
Initial states.

$$Init \triangleq x = 0$$

Allowed transitions.

$$Next \triangleq x' = x + 1$$

Verification Goal: prove that $Safe$ is an invariant of M.



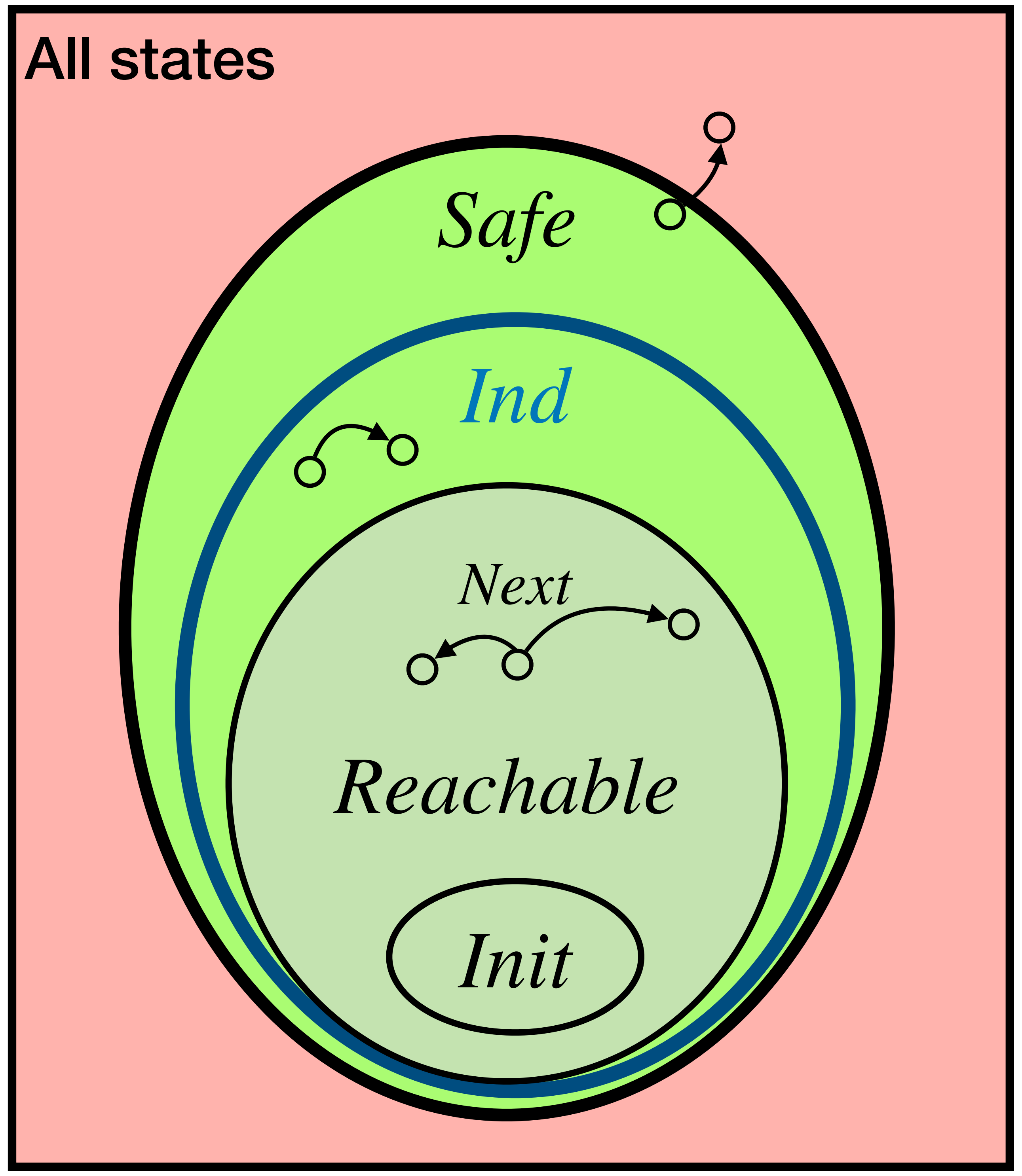
Formal Safety Verification

Model protocols as *state transition systems* $M = (Init, Next)$.

Initial states. $Init \triangleq x = 0$

Allowed transitions. $Next \triangleq x' = x + 1$

Verification Goal: prove that $Safe$ is an invariant of M.



To prove that $Safe$ is an invariant of M, find an ***inductive invariant***,

$$Ind = Safe \land L_1 \land \ldots \land L_k,$$

Lemma invariants

satisfying:

$$Init \Rightarrow Ind$$

$$Ind \land Next \Rightarrow Ind'$$

Induction rule

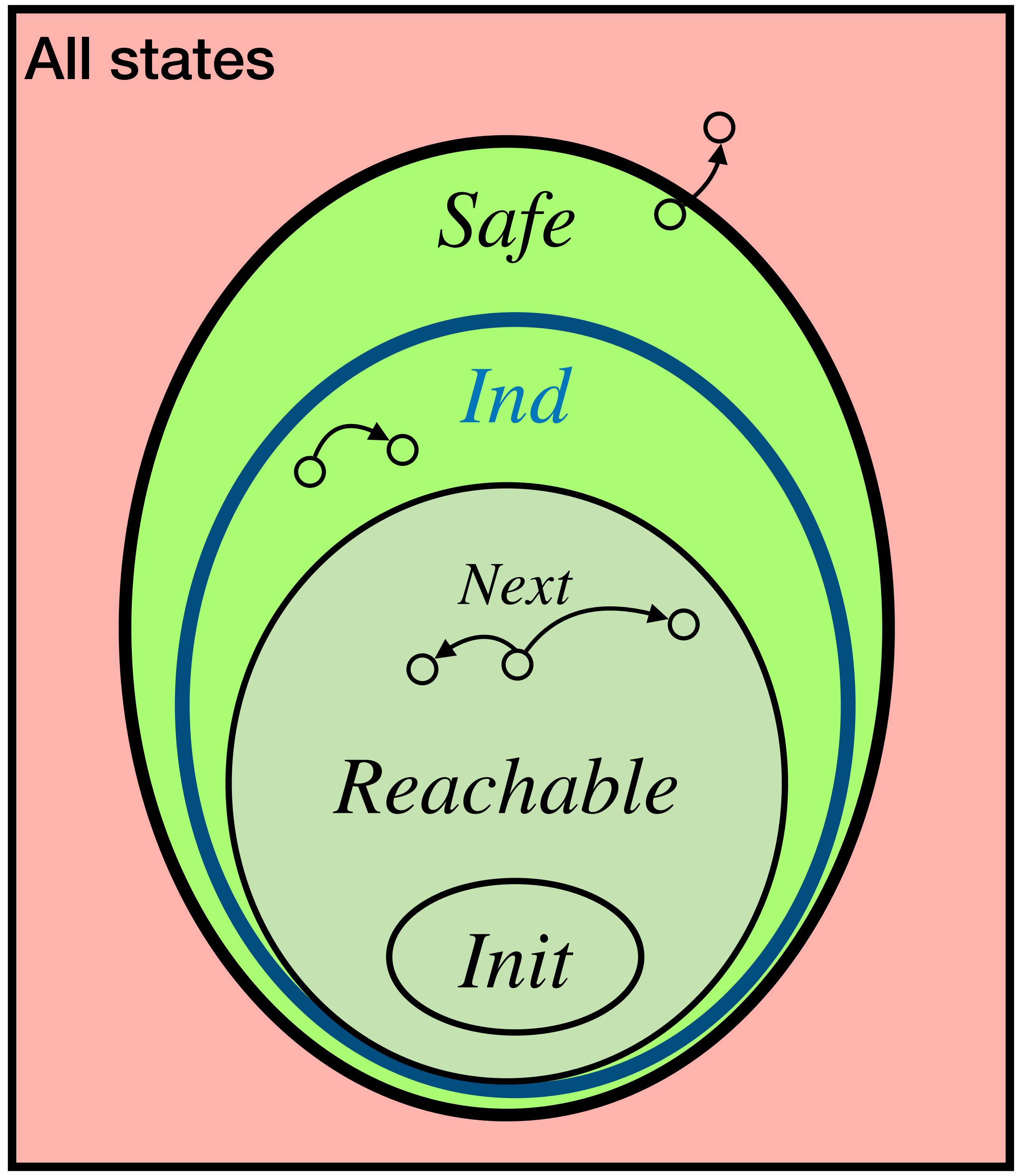
Formal Safety Verification

Model protocols as *state transition systems* $M = (Init, Next)$.

Initial states. $Init \triangleq x = 0$

Allowed transitions. $Next \triangleq x' = x + 1$

Verification Goal: prove that $Safe$ is an invariant of M.



To prove that $Safe$ is an invariant of M, find an ***inductive invariant***,

$$Ind = Safe \land L_1 \land \ldots \land L_k,$$

Lemma invariants

satisfying:

$$Init \Rightarrow Ind$$

$$Ind \land Next \Rightarrow Ind'$$

Induction rule

But we need to find Ind.

(HARD)

Automatic Safety Verification

Recent protocol verification techniques try to solve this **inductive invariant synthesis** problem automatically.

IC3PO [FMCAD21]

endive [FMCAD22]

DistAI [OSDI21, OSDI22]

SWISS [NSDI21]

Automatic Safety Verification

Recent protocol verification techniques try to solve this **inductive invariant synthesis** problem automatically.

IC3PO [FMCAD21]

endive [FMCAD22]

DistAI [OSDI21, OSDI22]

SWISS [NSDI21]

?

Distributed Protocol	Domains	Relations	Variables	Literals	Refinements	Invariants	DistAI time(s)	I4 time(s) final	I4 time(s) total	FOL-IC3 time(s) ∀	FOL-IC3 time(s) default
asynchronous lock server [5]	2	5	3	2	0	12	1.1	generalize fail[§]		6.9	-*
chord ring maintenance [24]	1	8	3	4	48	163	52.8	586.1‡	594.4	-*	-*
database chain replication [24]	4	13	7	4	158	66	58.8	20.2‡	63.1	-*	Z3 fail
decentralized lock [9]	2	2	4	2	150	16	9.4	generalize fail[§]		37.1[d]	Z3 fail
distributed lock [24]	2	4	4	3	82	45	12.6	152.1‡	204.7	1451.3	Z3 fail
hashed sharding [11]	3	3	5	2	0	15	1.1	nondet fail[§]		9.2	-*
leader election [24]	2	3	6	3	0	17	1.9	4.9‡	4.9	26.3	-*
learning switch [24]	2	4	4	3	8	71	27.6	10.5‡	12.4	-*	-*
lock server [24]	2	2	2	2	0	1	0.8	0.5‡	0.8	0.5	2.1
Paxos [13, 15, 23]	4	9	-	-	-	-	-*	-*	-*	-*	-*
permissioned blockchain [17]	4	10	6	3	2	13	4.9	blackbox fail[§]		21.2	-*
Ricart-Agrawala [26]	1	3	2	2	0	6	0.9	0.8	0.8	0.7	3.2
simple consensus [11]	3	8	5	3	19	50	23.3	41.8	68.7	-*	-*
two-phase commit [18]	1	7	2	3	3	30	1.9	3.1‡	8.0	3.4	7.9

* Time out after 1 week.
‡ I4 runtimes on our machine are similar (6 out of 7 protocols slightly faster) to those previously reported for I4 [18].
§ "generalize fail" means I4's implementation fails to convert invariants from the AVR model checker to generalized universally quantified invariants. "nondet fail" means failed on nondeterministic initialization. "blackbox fail" means error triggered on reasoning of blackbox functions.
d FOL-IC3 initially completed in less than a second, but this turned out to be incorrect due to a bug in the mypyvy protocol specification used by FOL-IC3, which does not exist in the Ivy protocol specification used by DistAI and I4.

Table 1: Evaluation results on 14 distributed protocols from multiple sources.

DistAI [Yao2021]

?

	Protocol	S.A.	Time (seconds) IC3PO	SWISS	fol-ic3	DistAI	I4	UPDR	Inv IC3PO	Inv Human	SMT IC3PO	SMT I4
EPR	epr-paxos	∅	**568**	15950*	timeout	error	memout	timeout	6	11	5680	1701556
EPR	epr-flexible_paxos	∅	**561**	18232*	timeout	error	memout	failure	6	11	1509	1761504
EPR	epr-multi_paxos	∅	timeout	timeout	timeout	error	memout	timeout	—	12	—	1902621
ORIGINAL	*Voting*	∅	**64**	timeout	timeout	error	memout	timeout	3	3	1057	1714170
ORIGINAL	*SimplePaxos*	A_{1-2}	**51**	timeout	timeout	error	failure	timeout	5	5	618	158470
ORIGINAL	*ImplicitPaxos*	A_{1-6}	**2008**	timeout	timeout	error	failure	timeout	7	7	18329	69715
ORIGINAL	*Paxos*	A_{1-8}	**98**	timeout	timeout	error	failure	timeout	10	10	668	76030
ORIGINAL	*MultiPaxos*	A_{1-11}	**340**	timeout	timeout	error	timeout	timeout	10	10	161	—
ORIGINAL	*FlexiblePaxos*	A_{1-11}	**1408**	timeout	timeout	error	failure	timeout	10	10	161	6983

TABLE II: Comparison of IC3PO against other state-of-the-art verifiers
ORIGINAL problems employ hierarchical strengthening (as detailed in Section VI), while EPR problems do not.

While introducing *SimplePaxos* and *ImplicitPaxos* to get the four-level Paxos hierarchy was quite easy, these intermediate levels were still added manually. It is appealing to explore

Towards an Automatic Proof of Lamport's Paxos [Goel2022]

Problem: Jagged boundary between *automation* and *interactivity*.

Our Work

Decompose inductive invariants into **inductive proof graphs**, and use this to guide their development.

$$\text{Safety} \triangleq \forall rm_i, rm_j \in \text{RM} : \neg(\text{rmState}[rm_i] = \text{"aborted"} \land \text{rmState}[rm_j] = \text{"committed"})$$

$$Inv73 \triangleq \forall rm_i \in RM : \forall rm_j \in RM : \neg(rmState[rm_i] = \text{"committed"}) \lor \neg(rmState[rm_j] = \text{"working"})$$

$$Inv11 \triangleq \forall rm_j \in RM : \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg(rmState[rm_j] = \text{"committed"})$$

$$Inv23 \triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(rmState[rm_i] = \text{"aborted"})$$

$$Inv2 \triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(rmState[rm_i] = \text{"working"})$$

$$Inv1 \triangleq \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg([type \mapsto \text{"Commit"}] \in msgsCommit)$$

$$Inv53 \triangleq \forall rm_i \in RM : \neg(rmState[rm_i] = \text{"committed"}) \lor \neg(tmState = \text{"init"})$$

$$Inv1140 \triangleq \forall rm_i \in RM : (rmState[rm_i] = \text{"prepared"}) \lor \neg(tmPrepared = RM) \lor \neg(tmState = \text{"init"})$$

$$Inv16 \triangleq \forall rm_i \in RM : \neg(rmState[rm_i] = \text{"working"}) \lor \neg(tmPrepared = RM)$$

$$Inv4 \triangleq \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(tmState = \text{"init"})$$

$$Inv7 \triangleq \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg(tmState = \text{"init"})$$

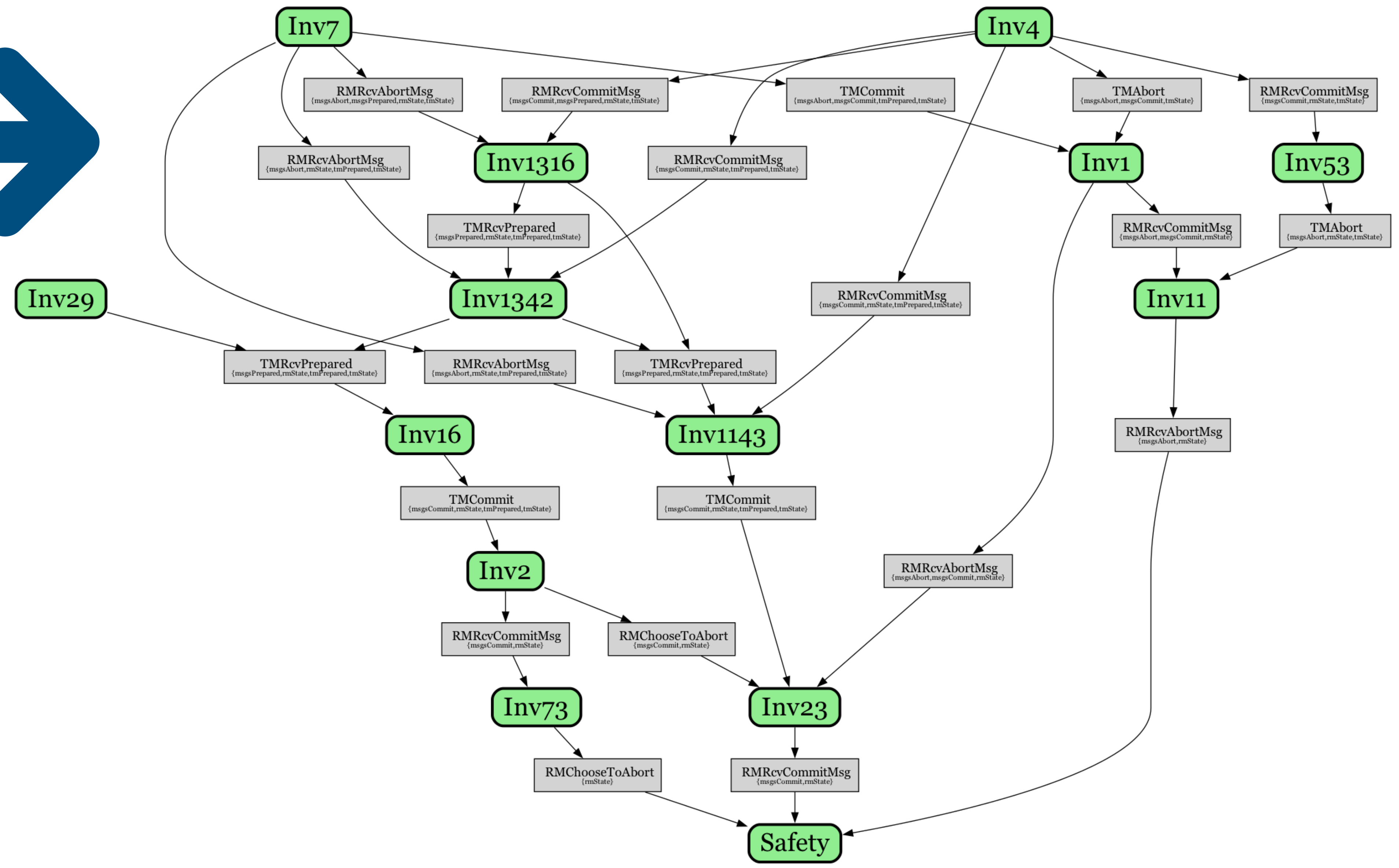
$$Inv1325 \triangleq \forall rm_j \in RM : (rmState[rm_j] = \text{"prepared"}) \lor \neg(tmPrepared = tmPrepared \cup \{rm_j\}) \lor \neg(tmState = \text{"init"})$$

$$Inv1291 \triangleq \forall rm_j \in RM : (rmState[rm_j] = \text{"prepared"}) \lor \neg([type \mapsto \text{"Prepared"}, rm \mapsto rm_j] \in msgsPrepared) \lor \neg(tmState = \text{"init"})$$

$$Inv29 \triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Prepared"}, rm \mapsto rm_i] \in msgsPrepared) \lor \neg(rmState[rm_i] = \text{"working"})$$

Standard, monolithic inductive invariant for *two-phase commit*.

Inductive Proof Decomposition



Decomposing Inductive Invariants

$Safety \triangleq \forall rm_i, rm_j \in RM : \neg(rmState[rm_i] = \text{"aborted"} \land rmState[rm_j] = \text{"committed"})$ *Top-level safety property.*

$Inv73 \triangleq \forall rm_i \in RM : \forall rm_j \in RM : \neg(rmState[rm_i] = \text{"committed"}) \lor \neg(rmState[rm_j] = \text{"working"})$

$Inv11 \triangleq \forall rm_j \in RM : \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg(rmState[rm_j] = \text{"committed"})$

$Inv23 \triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(rmState[rm_i] = \text{"aborted"})$

$Inv2 \triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(rmState[rm_i] = \text{"working"})$

$Inv1 \triangleq \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg([type \mapsto \text{"Commit"}] \in msgsCommit)$

$Inv53 \triangleq \forall rm_i \in RM : \neg(rmState[rm_i] = \text{"committed"}) \lor \neg(tmState = \text{"init"})$

$Inv1140 \triangleq \forall rm_i \in RM : (rmState[rm_i] = \text{"prepared"}) \lor \neg(tmPrepared = RM) \lor \neg(tmState = \text{"init"})$

$Inv16 \triangleq \forall rm_i \in RM : \neg(rmState[rm_i] = \text{"working"}) \lor \neg(tmPrepared = RM)$

$Inv4 \triangleq \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(tmState = \text{"init"})$

$Inv7 \triangleq \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg(tmState = \text{"init"})$

$Inv1325 \triangleq \forall rm_j \in RM : (rmState[rm_j] = \text{"prepared"}) \lor \neg(tmPrepared = tmPrepared \cup \{rm_j\}) \lor \neg(tmState = \text{"init"})$

$Inv1291 \triangleq \forall rm_j \in RM : (rmState[rm_j] = \text{"prepared"}) \lor \neg([type \mapsto \text{"Prepared"}, rm \mapsto rm_j] \in msgsPrepared) \lor \neg(tmState = \text{"init"})$

$Inv29 \triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Prepared"}, rm \mapsto rm_i] \in msgsPrepared) \lor \neg(rmState[rm_i] = \text{"working"})$

$$
\begin{aligned}
Next \triangleq\ & \lor\ TMCommitAction \\
& \lor\ TMAbortAction \\
& \lor\ TMRcvPreparedAction \\
& \lor\ RMPrepareAction \\
& \lor\ RMChooseToAbortAction \\
& \lor\ RMRcvCommitMsgAction \\
& \lor\ RMRcvAbortMsgAction
\end{aligned}
$$

Inductive invariant and transition relation for *two-phase commit* protocol.

Decomposing Inductive Invariants

Basic induction verification step.

$$Ind \quad \wedge \quad Next \quad \Rightarrow \quad Ind'$$

Decomposing Inductive Invariants

Basic induction verification step.

$$Ind \quad \land \quad Next \quad \Rightarrow \quad Ind'$$

(1) Decompose by lemma conjunct.

Decomposing Inductive Invariants

Basic induction verification step.

$$Ind \quad \land \quad Next \quad \Rightarrow$$

$$\begin{aligned}
\text{Safety} &\triangleq \forall rm_i, rm_j \in \text{RM} : \neg(\text{rmState}[rm_i] = \text{"aborted"} \land \text{rmState}[rm_j] = \text{"committed"}) \\
Inv73 &\triangleq \forall rm_i \in RM : \forall rm_j \in RM : \neg(rmState[rm_i] = \text{"committed"}) \lor \neg(rmState[rm_j] = \text{"working"}) \\
Inv11 &\triangleq \forall rm_j \in RM : \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg(rmState[rm_j] = \text{"committed"}) \\
Inv23 &\triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(rmState[rm_i] = \text{"aborted"}) \\
Inv2 &\triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(rmState[rm_i] = \text{"working"}) \\
Inv1 &\triangleq \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \\
Inv53 &\triangleq \forall rm_i \in RM : \neg(rmState[rm_i] = \text{"committed"}) \lor \neg(tmState = \text{"init"}) \\
Inv1140 &\triangleq \forall rm_i \in RM : (rmState[rm_i] = \text{"prepared"}) \lor \neg(tmPrepared = RM) \lor \neg(tmState = \text{"init"}) \\
Inv16 &\triangleq \forall rm_i \in RM : \neg(rmState[rm_i] = \text{"working"}) \lor \neg(tmPrepared = RM) \\
Inv4 &\triangleq \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(tmState = \text{"init"}) \\
Inv7 &\triangleq \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg(tmState = \text{"init"}) \\
Inv1325 &\triangleq \forall rm_j \in RM : (rmState[rm_j] = \text{"prepared"}) \lor \neg(tmPrepared = tmPrepared \cup \{rm_j\}) \lor \neg(tmState = \text{"init"}) \\
Inv1291 &\triangleq \forall rm_j \in RM : (rmState[rm_j] = \text{"prepared"}) \lor \neg([type \mapsto \text{"Prepared"}, rm \mapsto rm_j] \in msgsPrepared) \lor \neg(tmState = \text{"init"}) \\
Inv29 &\triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Prepared"}, rm \mapsto rm_i] \in msgsPrepared) \lor \neg(rmState[rm_i] = \text{"working"})
\end{aligned}$$

(1) Decompose by lemma conjunct.

Decomposing Inductive Invariants

$$Ind \land Next \Rightarrow \text{Safety} \triangleq \forall rm_i, rm_j \in \text{RM} : \neg(\text{rmState}[rm_i] = \text{"aborted"} \land \text{rmState}[rm_j] = \text{"committed"})$$

$$\vdots$$

$$Ind \land Next \Rightarrow Inv29 \triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Prepared"}, rm \mapsto rm_i] \in msgsPrepared) \lor \neg(rmState[rm_i] = \text{"working"})$$

(1) Decompose by lemma conjunct.

Decomposing Inductive Invariants

$$\text{Safety} \triangleq \forall rm_i, rm_j \in \text{RM} : \neg(\text{rmState}[rm_i] = \text{"aborted"} \land \text{rmState}[rm_j] = \text{"committed"})$$
$$Inv73 \triangleq \forall rm_i \in RM : \forall rm_j \in RM : \neg(rmState[rm_i] = \text{"committed"}) \lor \neg(rmState[rm_j] = \text{"working"})$$
$$Inv11 \triangleq \forall rm_j \in RM : \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg(rmState[rm_j] = \text{"committed"})$$
$$Inv23 \triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(rmState[rm_i] = \text{"aborted"})$$
$$Inv2 \triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(rmState[rm_i] = \text{"working"})$$
$$Inv1 \triangleq \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg([type \mapsto \text{"Commit"}] \in msgsCommit)$$
$$Inv53 \triangleq \forall rm_i \in RM : \neg(rmState[rm_i] = \text{"committed"}) \lor \neg(tmState = \text{"init"})$$
$$Inv1140 \triangleq \forall rm_i \in RM : (rmState[rm_i] = \text{"prepared"}) \lor \neg(tmPrepared = RM) \lor \neg(tmState = \text{"init"})$$
$$Inv16 \triangleq \forall rm_i \in RM : \neg(rmState[rm_i] = \text{"working"}) \lor \neg(tmPrepared = RM)$$
$$Inv4 \triangleq \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(tmState = \text{"init"})$$
$$Inv7 \triangleq \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg(tmState = \text{"init"})$$
$$Inv1325 \triangleq \forall rm_j \in RM : (rmState[rm_j] = \text{"prepared"}) \lor \neg(tmPrepared = tmPrepared \cup \{rm_j\}) \lor \neg(tmState = \text{"init"})$$
$$Inv1291 \triangleq \forall rm_j \in RM : (rmState[rm_j] = \text{"prepared"}) \lor \neg([type \mapsto \text{"Prepared"}, rm \mapsto rm_j] \in msgsPrepared) \lor \neg(tmState = \text{"init"})$$
$$Inv29 \triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Prepared"}, rm \mapsto rm_i] \in msgsPrepared) \lor \neg(rmState[rm_i] = \text{"working"})$$

$\land$ *Next* $\Rightarrow$

$$\text{Safety} \triangleq \forall rm_i, rm_j \in \text{RM} : \neg(\text{rmState}[rm_i] = \text{"aborted"} \land \text{rmState}[rm_j] = \text{"committed"})$$

⋮

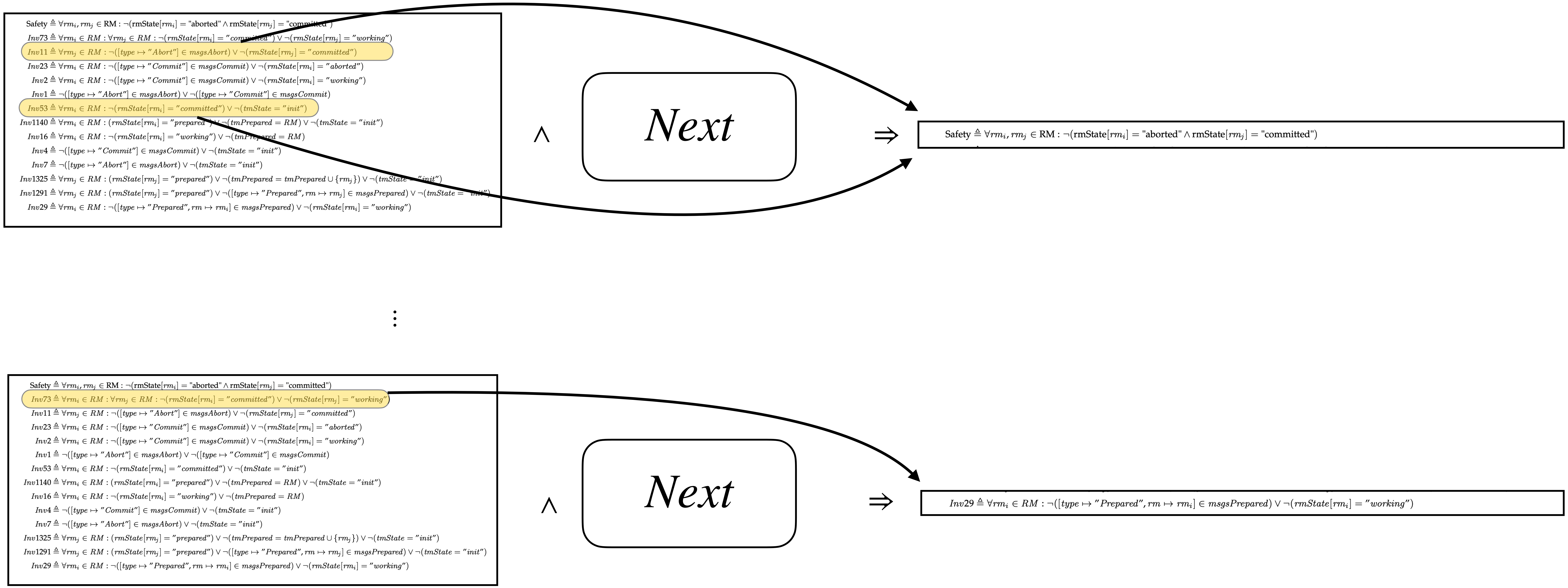
$$\text{Safety} \triangleq \forall rm_i, rm_j \in \text{RM} : \neg(\text{rmState}[rm_i] = \text{"aborted"} \land \text{rmState}[rm_j] = \text{"committed"})$$
$$Inv73 \triangleq \forall rm_i \in RM : \forall rm_j \in RM : \neg(rmState[rm_i] = \text{"committed"}) \lor \neg(rmState[rm_j] = \text{"working"})$$
$$Inv11 \triangleq \forall rm_j \in RM : \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg(rmState[rm_j] = \text{"committed"})$$
$$Inv23 \triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(rmState[rm_i] = \text{"aborted"})$$
$$Inv2 \triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(rmState[rm_i] = \text{"working"})$$
$$Inv1 \triangleq \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg([type \mapsto \text{"Commit"}] \in msgsCommit)$$
$$Inv53 \triangleq \forall rm_i \in RM : \neg(rmState[rm_i] = \text{"committed"}) \lor \neg(tmState = \text{"init"})$$
$$Inv1140 \triangleq \forall rm_i \in RM : (rmState[rm_i] = \text{"prepared"}) \lor \neg(tmPrepared = RM) \lor \neg(tmState = \text{"init"})$$
$$Inv16 \triangleq \forall rm_i \in RM : \neg(rmState[rm_i] = \text{"working"}) \lor \neg(tmPrepared = RM)$$
$$Inv4 \triangleq \neg([type \mapsto \text{"Commit"}] \in msgsCommit) \lor \neg(tmState = \text{"init"})$$
$$Inv7 \triangleq \neg([type \mapsto \text{"Abort"}] \in msgsAbort) \lor \neg(tmState = \text{"init"})$$
$$Inv1325 \triangleq \forall rm_j \in RM : (rmState[rm_j] = \text{"prepared"}) \lor \neg(tmPrepared = tmPrepared \cup \{rm_j\}) \lor \neg(tmState = \text{"init"})$$
$$Inv1291 \triangleq \forall rm_j \in RM : (rmState[rm_j] = \text{"prepared"}) \lor \neg([type \mapsto \text{"Prepared"}, rm \mapsto rm_j] \in msgsPrepared) \lor \neg(tmState = \text{"init"})$$
$$Inv29 \triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Prepared"}, rm \mapsto rm_i] \in msgsPrepared) \lor \neg(rmState[rm_i] = \text{"working"})$$

$\land$ *Next* $\Rightarrow$

$$Inv29 \triangleq \forall rm_i \in RM : \neg([type \mapsto \text{"Prepared"}, rm \mapsto rm_i] \in msgsPrepared) \lor \neg(rmState[rm_i] = \text{"working"})$$

(1) Decompose by lemma conjunct.

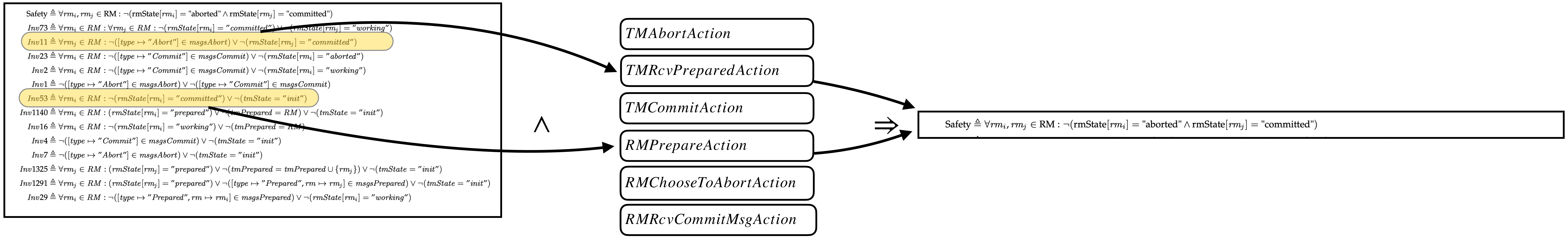
Decomposing Inductive Invariants



(2) Find inductive support lemmas.

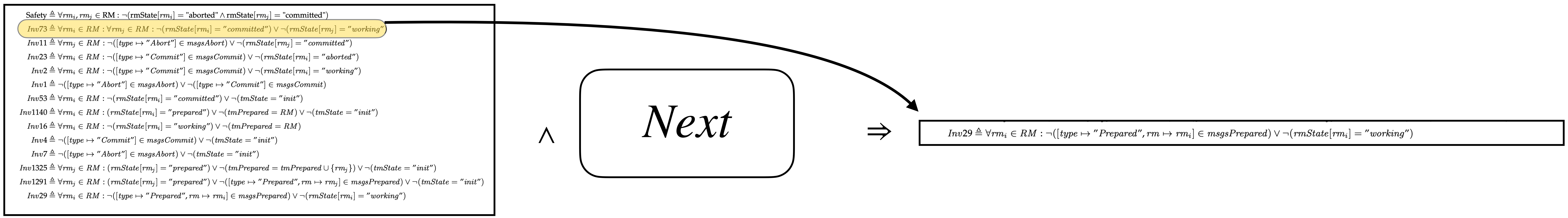
(1) Decompose by lemma conjunct.

Decomposing Inductive Invariants



(3) Decompose by action disjunct.

⋮



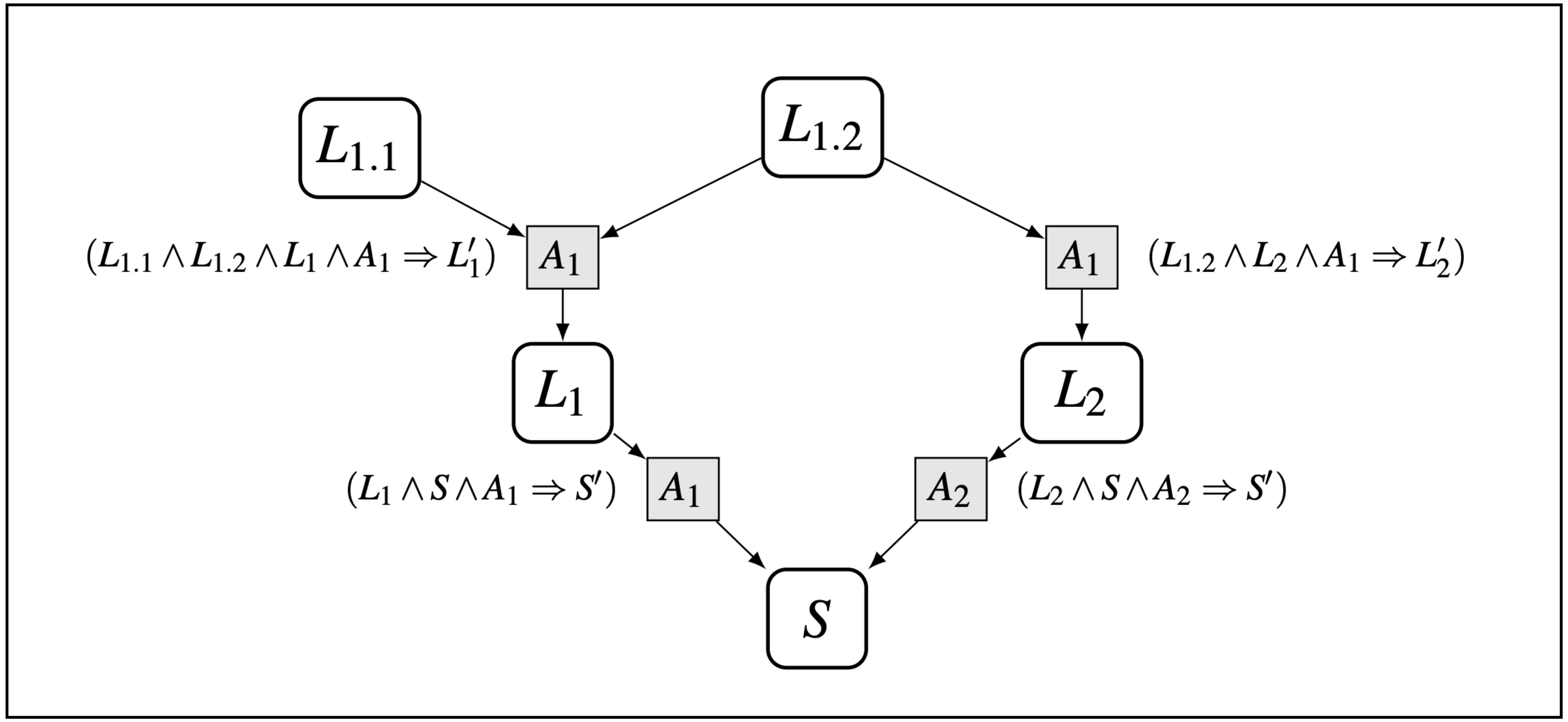
(2) Find inductive support lemmas.

(1) Decompose by lemma conjunct.

Inductive Proof Graphs

Inductive proof graph: lemma and action nodes, with induction support edges.

An inductive proof graph is *valid* when all lemma nodes are *locally valid*.



Inductive Proof Graphs

Cycles are permissible in these proof graphs.

"Ring counter" system.

VARIABLES a, b, c

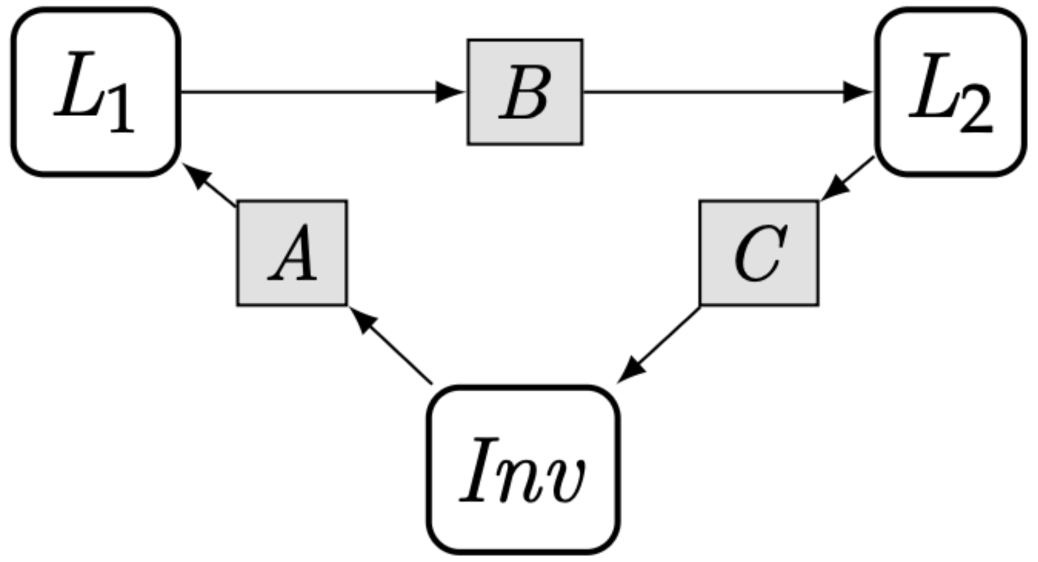
Init $\triangleq a = 1 \land b = 0 \land c = 0$

$A \triangleq a > 0 \land b' = a \land a' = 0 \land c' = c$
$B \triangleq b > 0 \land c' = b \land b' = 0 \land a' = a$
$C \triangleq c > 0 \land a' = c \land c' = 0 \land b' = b$

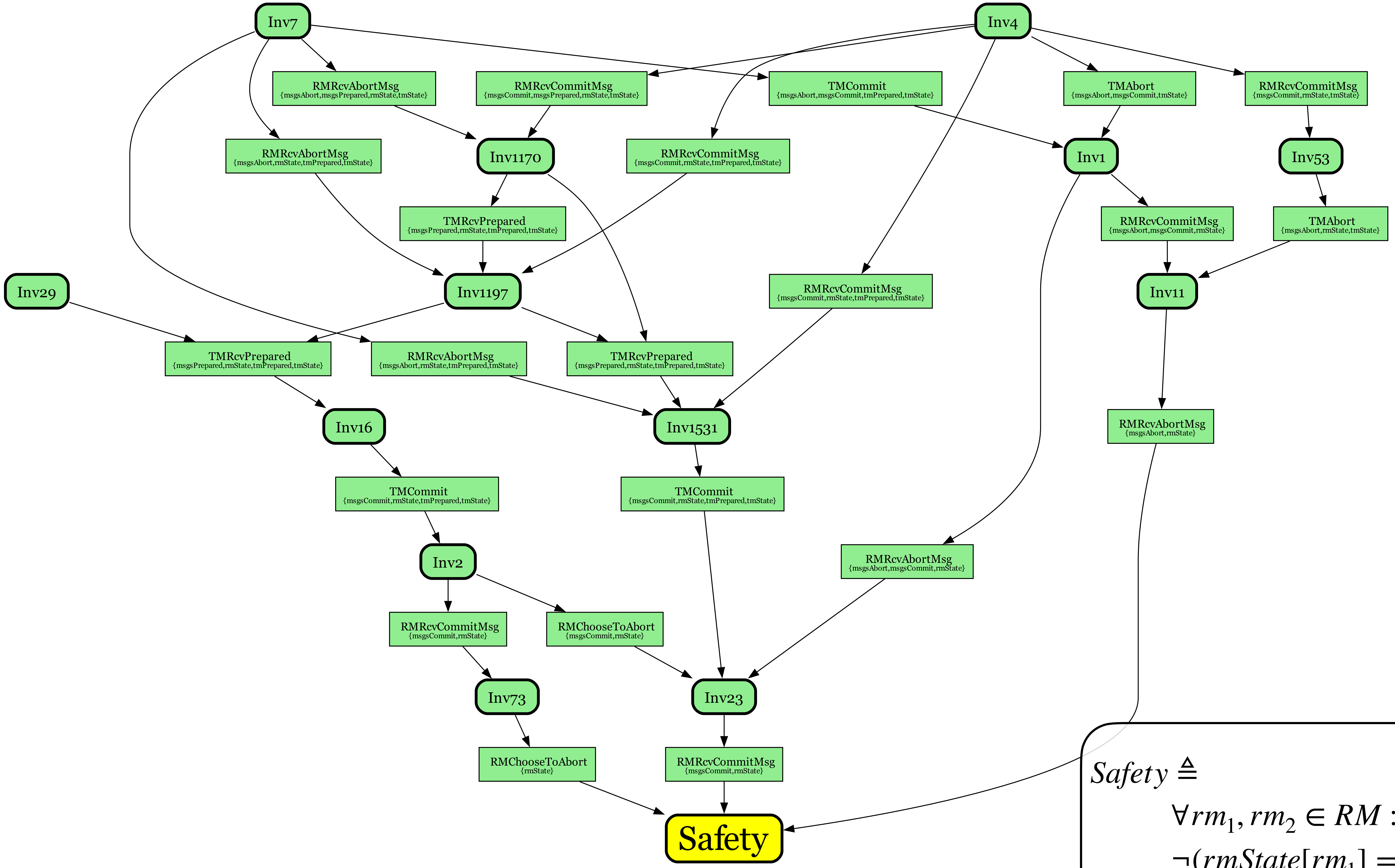
Next $\triangleq A \lor B \lor C$

Inv $\triangleq a \in \{0, 1\}$ (* top-level invariant. *)

$L_1 \triangleq b \in \{0, 1\}$
$L_2 \triangleq c \in \{0, 1\}$

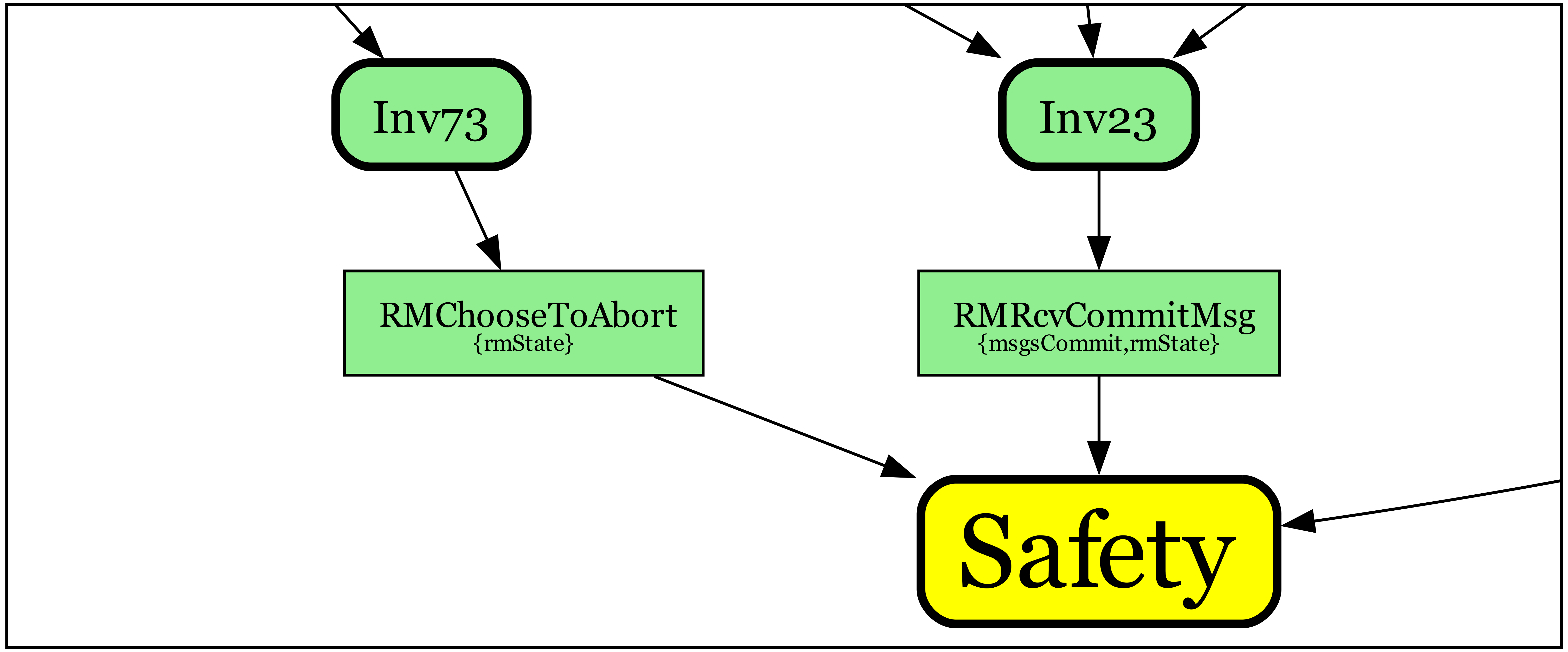


Inductive Proof Graph: Two-Phase Commit



$$Safety \triangleq$$
$$\forall rm_1, rm_2 \in RM :$$
$$\neg(rmState[rm_1] = aborted \land rmState[rm_2] = committed)$$

Variable Slices



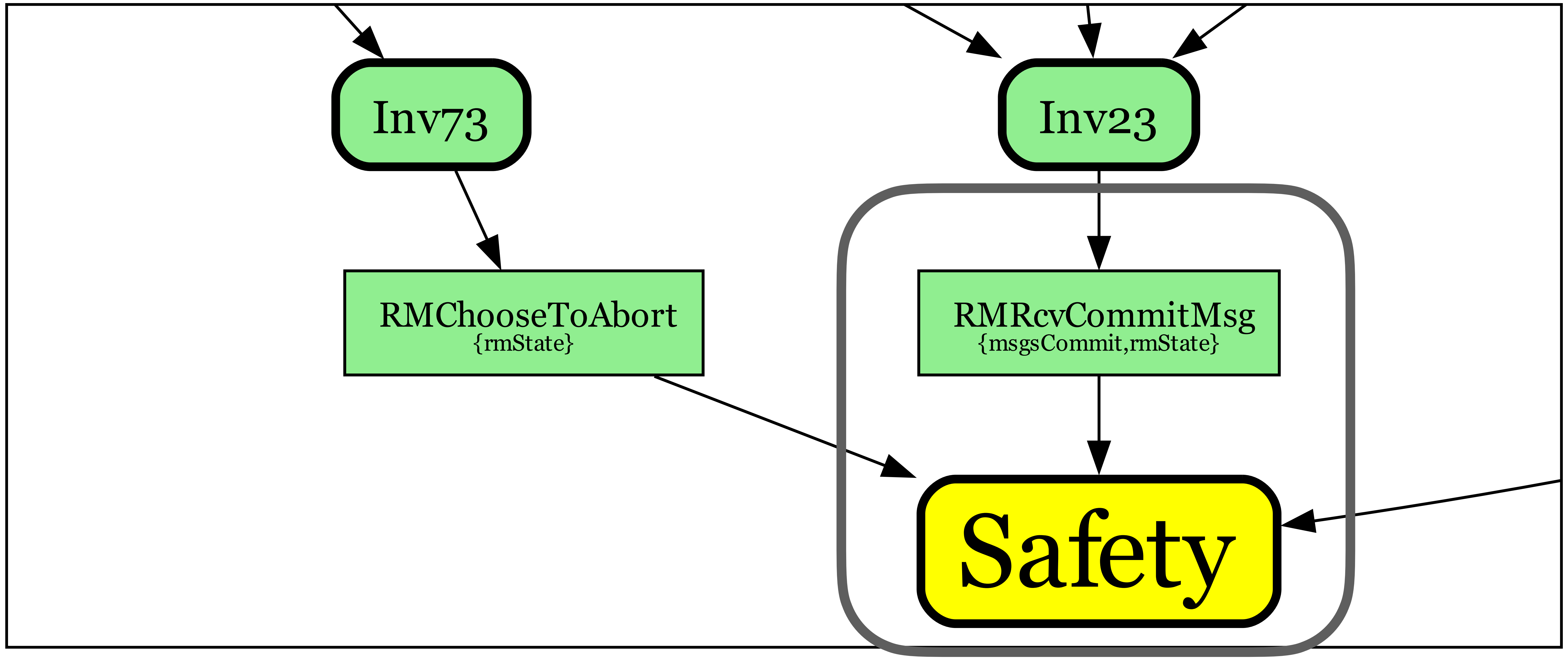
VARIABLES
rmState,
tmState,
tmPrepared,
msgsPrepared,
msgsCommit,
msgsAbort

Variable **slices** can be statically computed at each node based on (lemma,action) pair (L, A).

Subset of state variables needed for support lemmas to discharge inductive node.

$$Safety \triangleq \forall rm_1, rm_2 \in RM : \neg(rmState[rm_1] = aborted \land rmState[rm_2] = committed)$$

Variable Slices

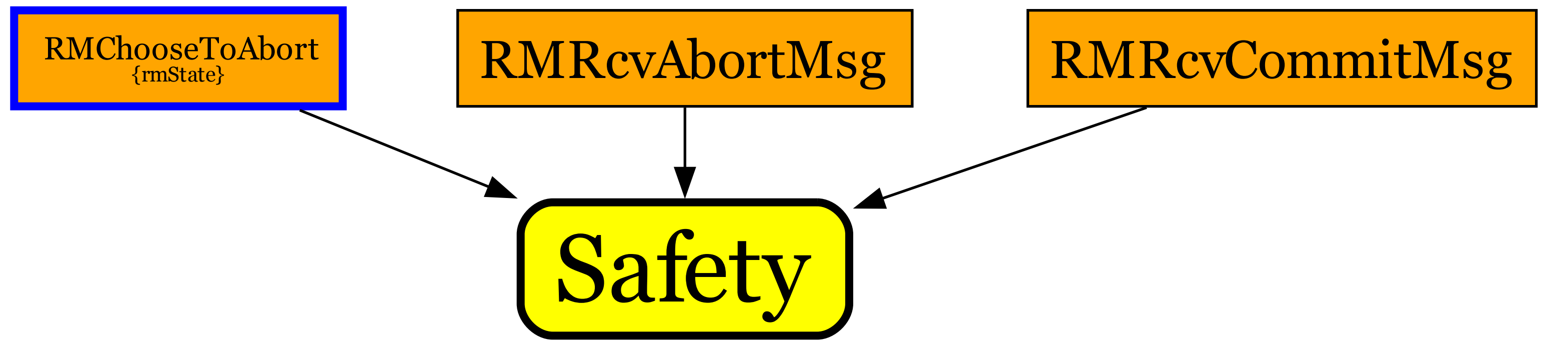


VARIABLES
rmState,
tmState,
tmPrepared,
msgsPrepared,
msgsCommit,
msgsAbort

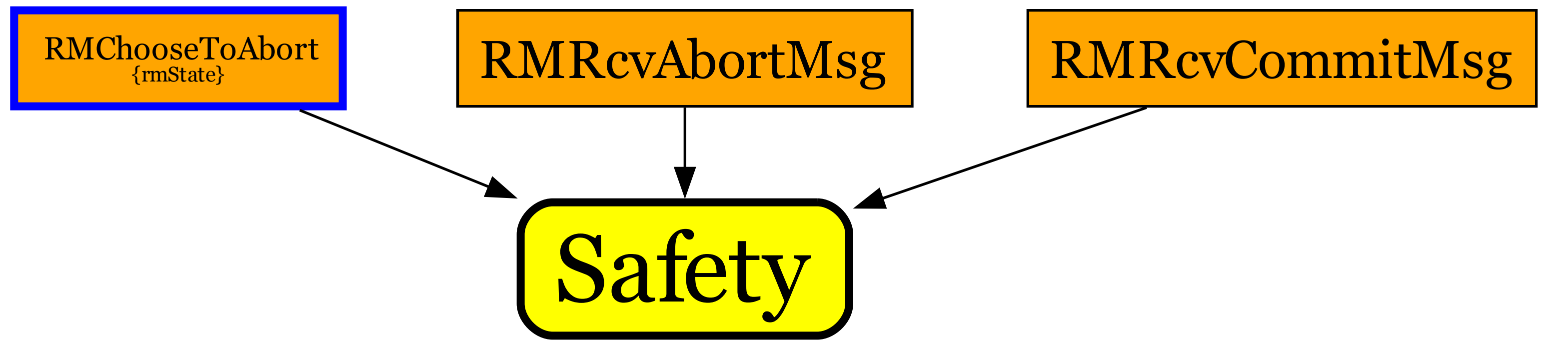
Variable **slices** can be statically computed at each node based on (lemma,action) pair (L, A).

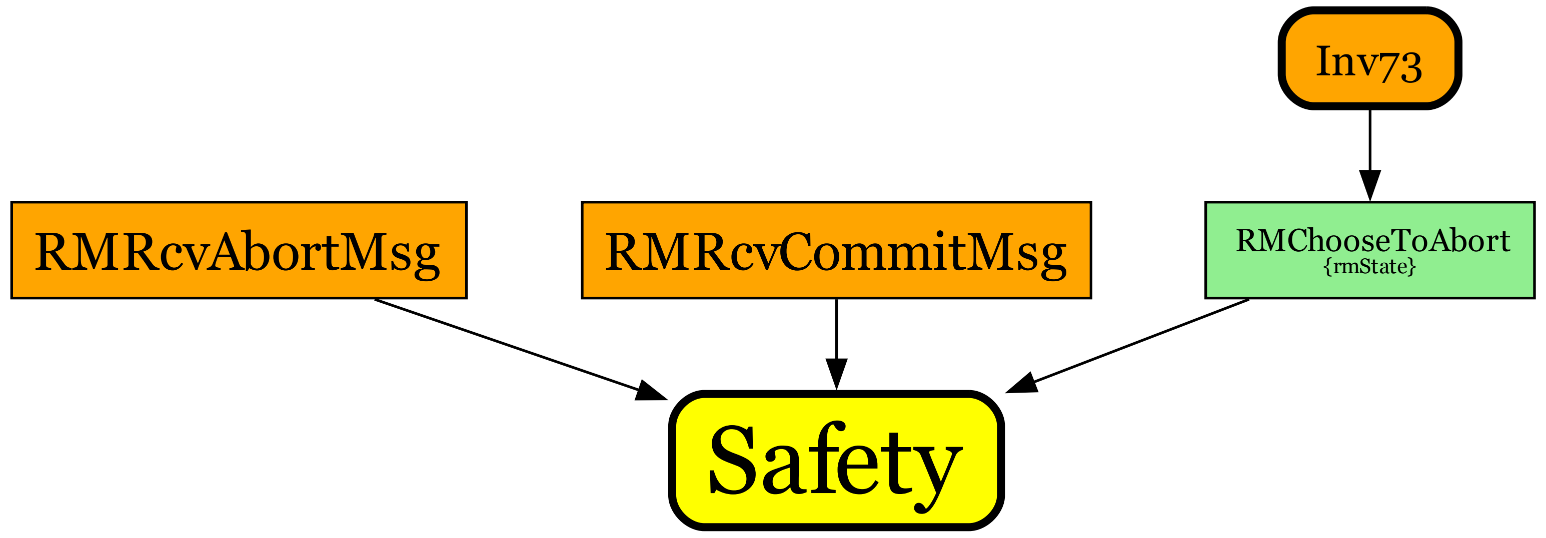
Subset of state variables needed for support lemmas to discharge inductive node.

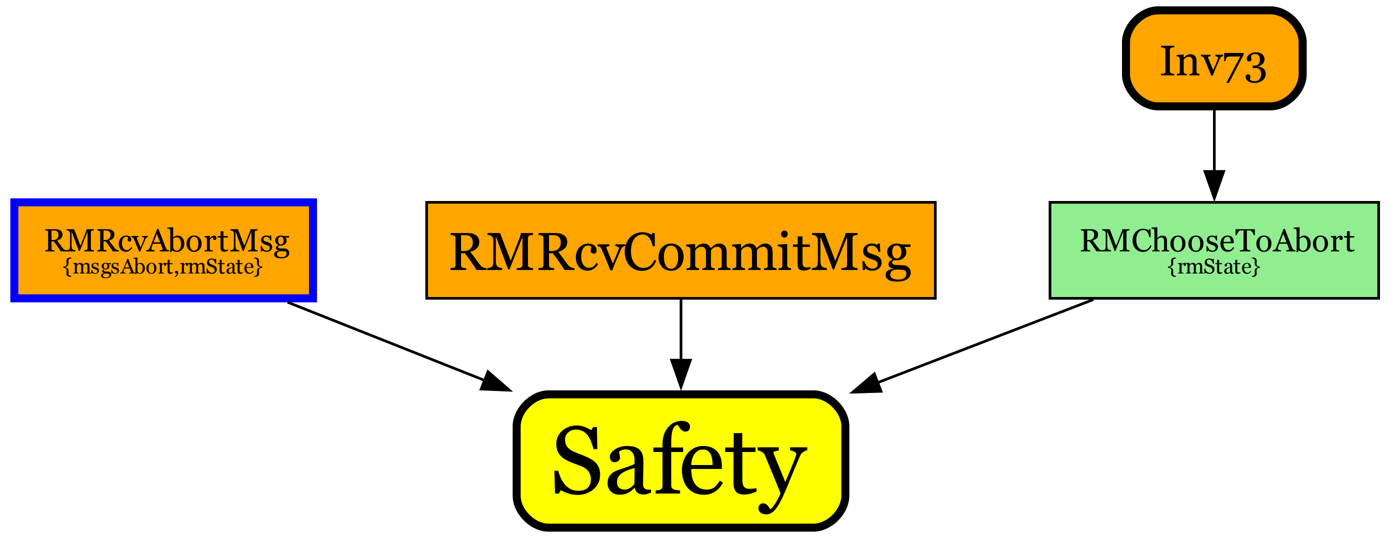
$$Safety \triangleq \forall rm_1, rm_2 \in RM : \neg(rmState[rm_1] = aborted \land rmState[rm_2] = committed)$$

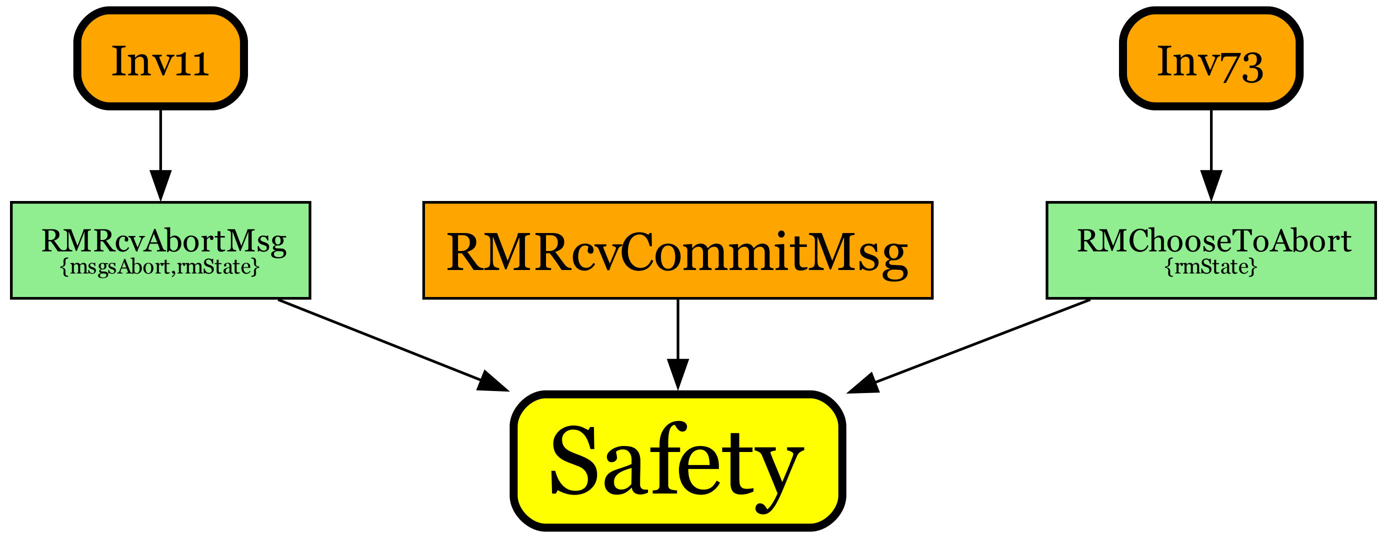


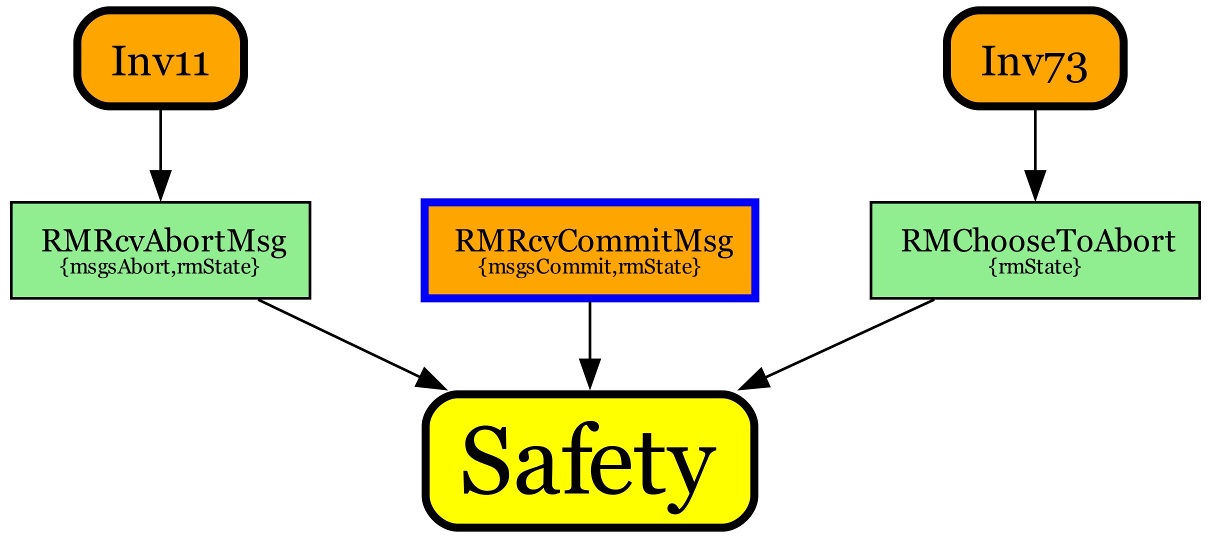
RMChooseToAbort
{rmState}
RMRcvAbortMsg
RMRcvCommitMsg
Safety

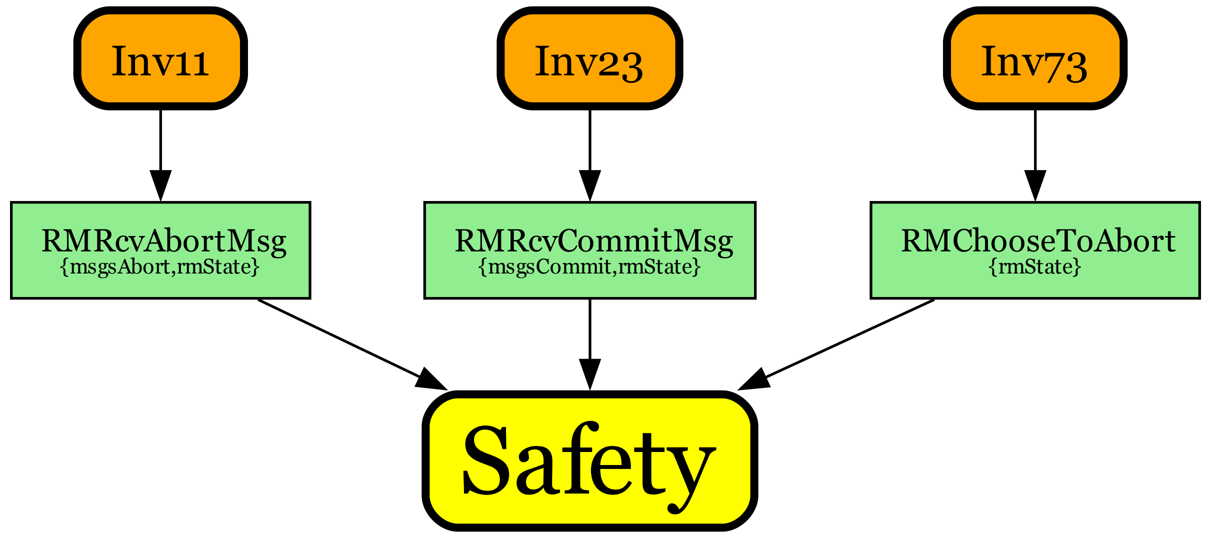

RMChooseToAbort
{rmState}
RMRcvAbortMsg
RMRcvCommitMsg
Safety


Inv73
RMRcvAbortMsg
RMRcvCommitMsg
RMChooseToAbort
{rmState}
Safety

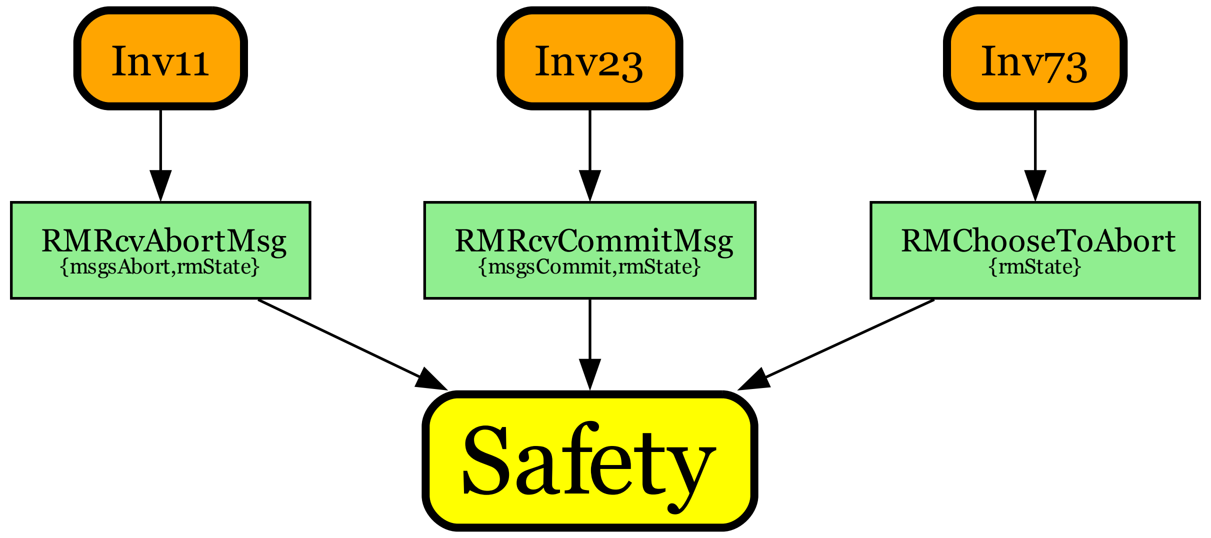

Inv73
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
RMChooseToAbort
{rmState}
Safety

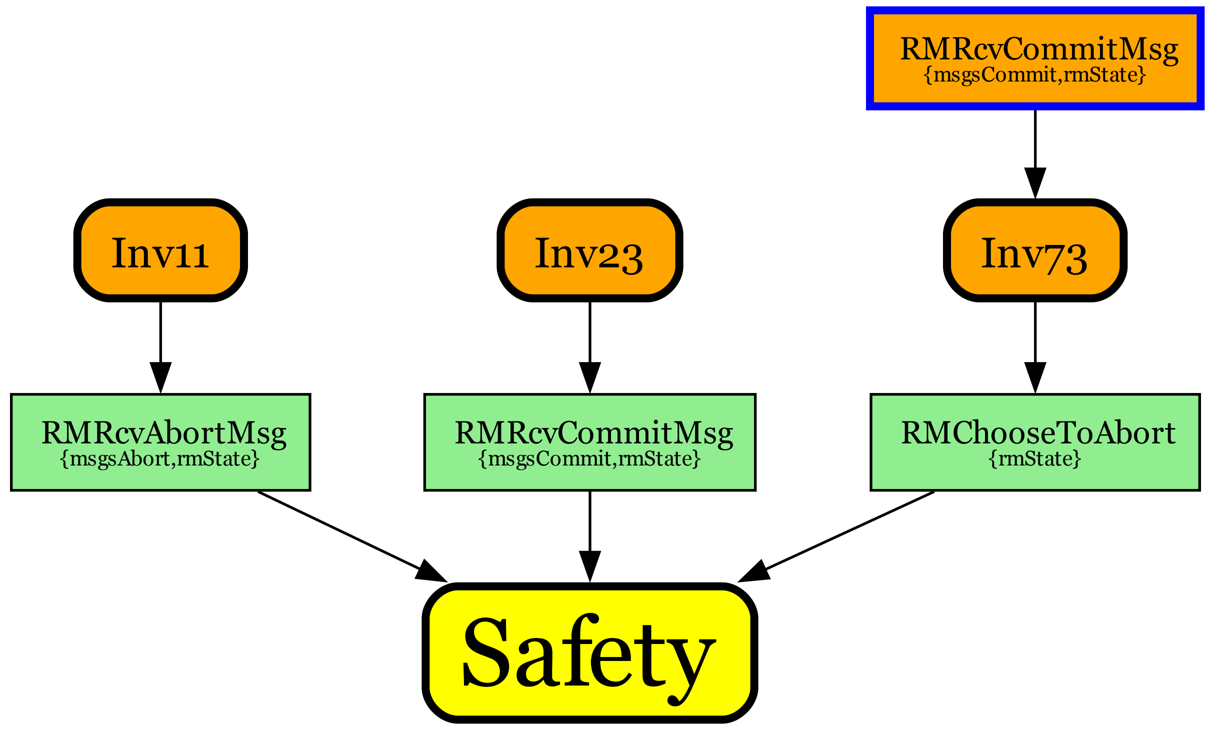

Inv11
Inv73
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
RMChooseToAbort
{rmState}
Safety

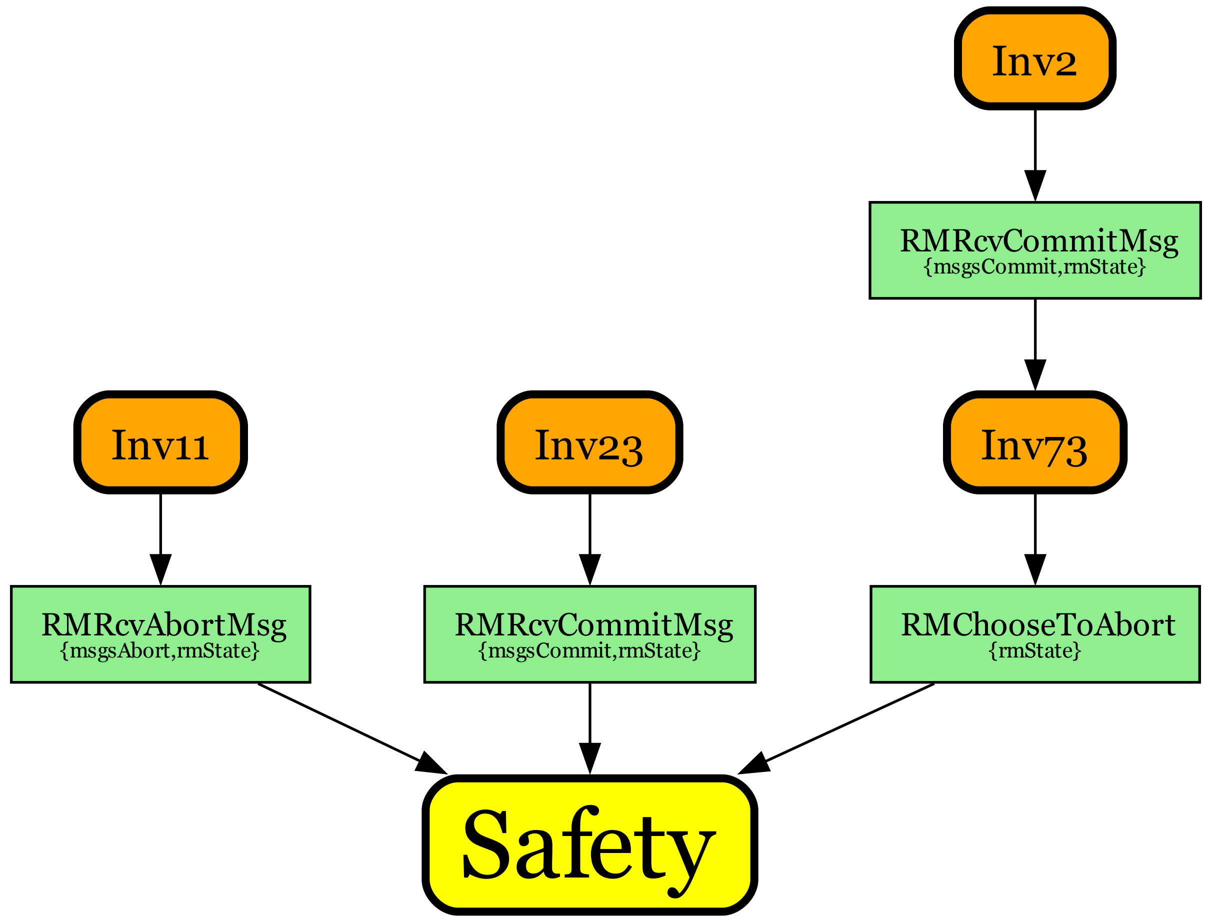

Inv11
Inv73
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{rmState}
Safety



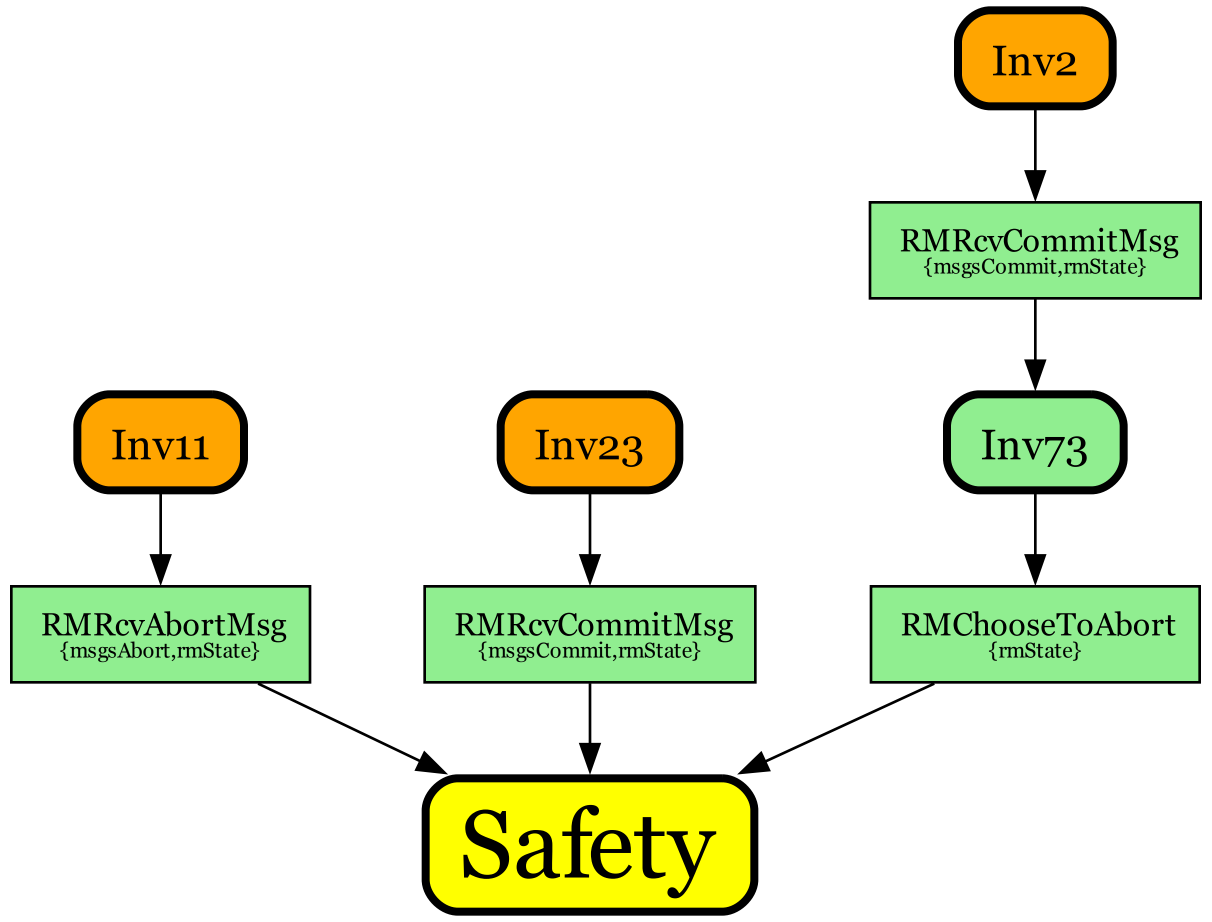
Inv11
Inv23
Inv73
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{rmState}
Safety


Inv11
Inv23
Inv73
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{rmState}
Safety

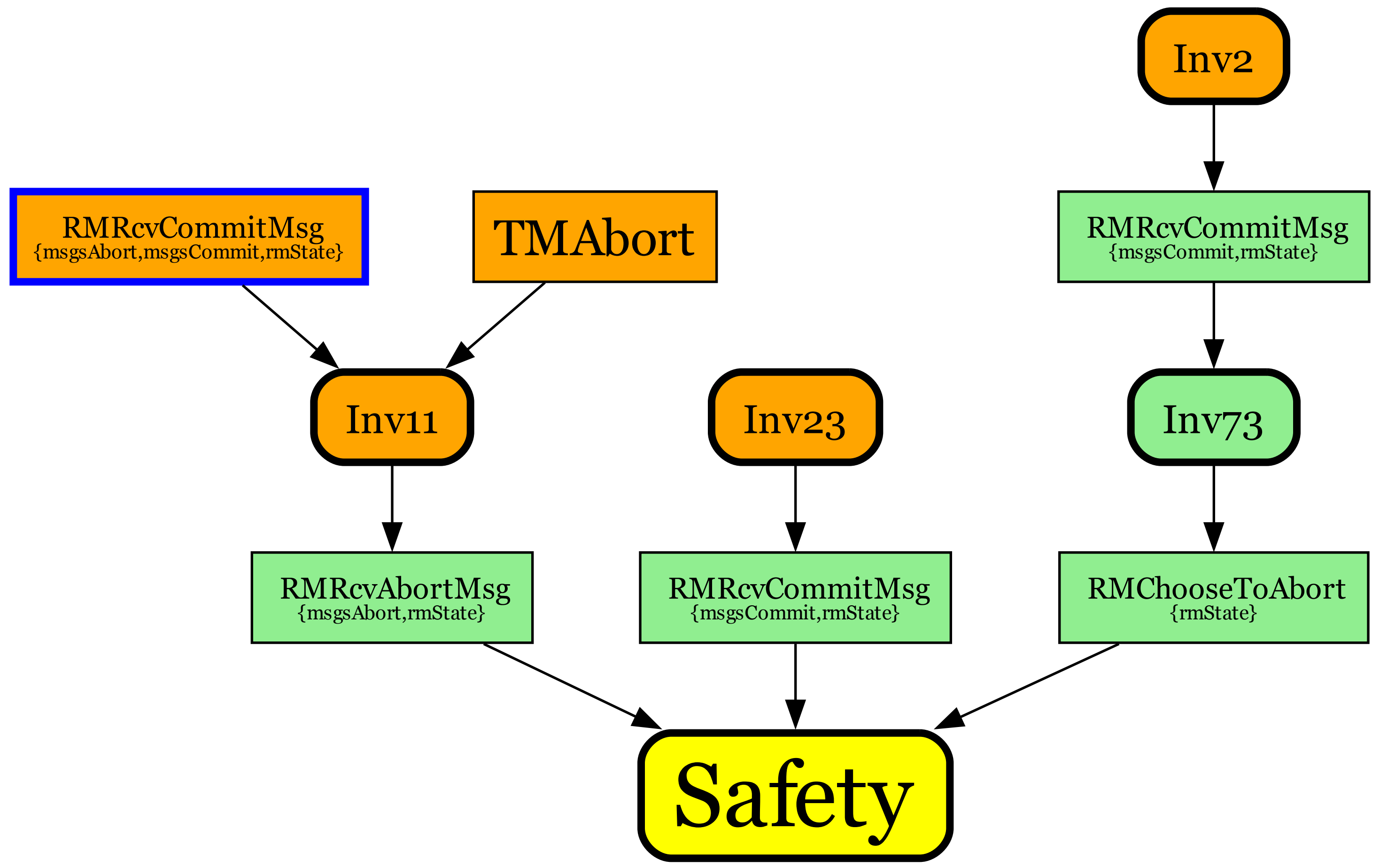

RMRcvCommitMsg
{msgsCommit,rmState}
Inv11
Inv23
Inv73
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{rmState}
Safety



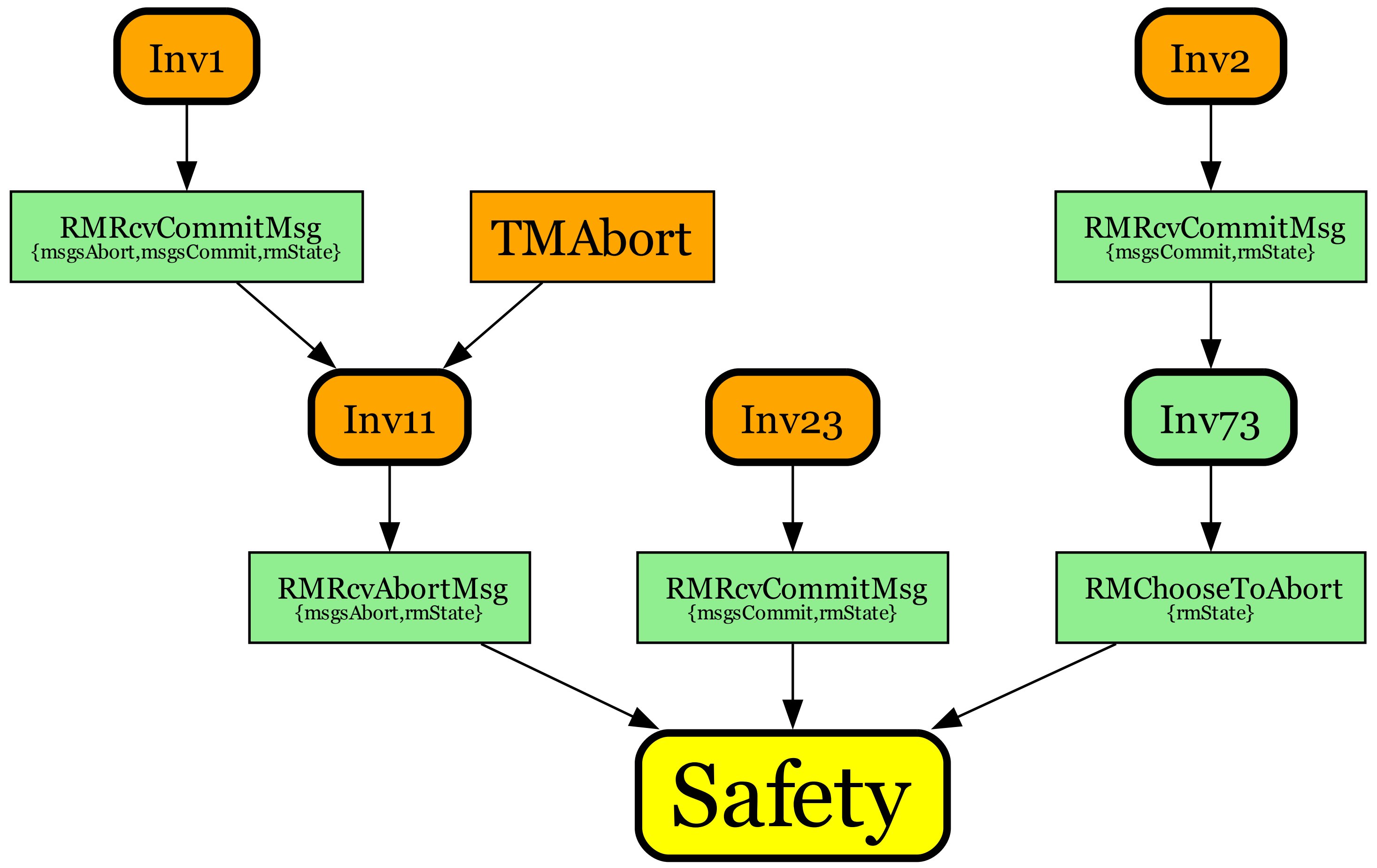
Inv2
RMRcvCommitMsg
{msgsCommit,rmState}
Inv11
Inv23
Inv73
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{rmState}
Safety

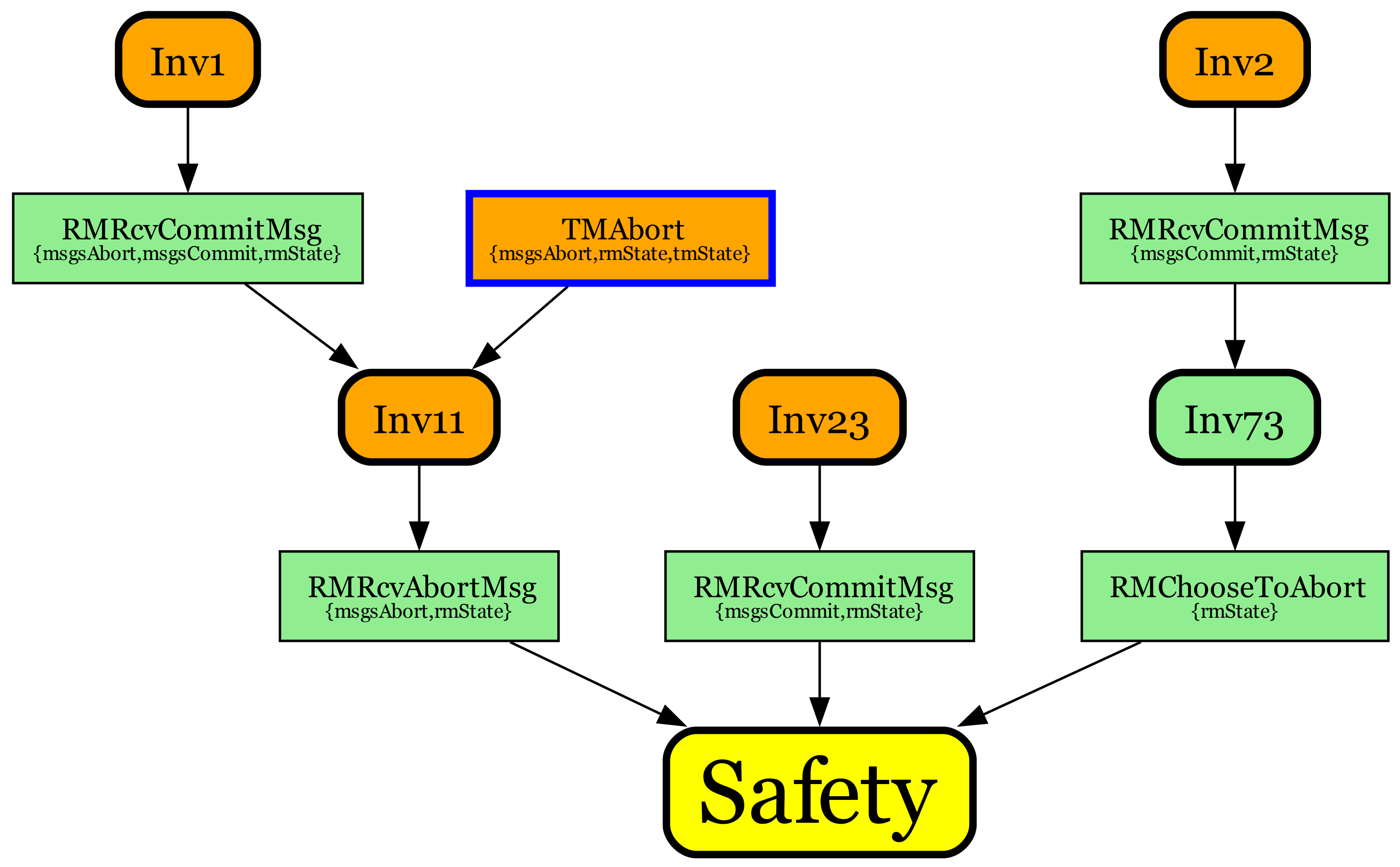


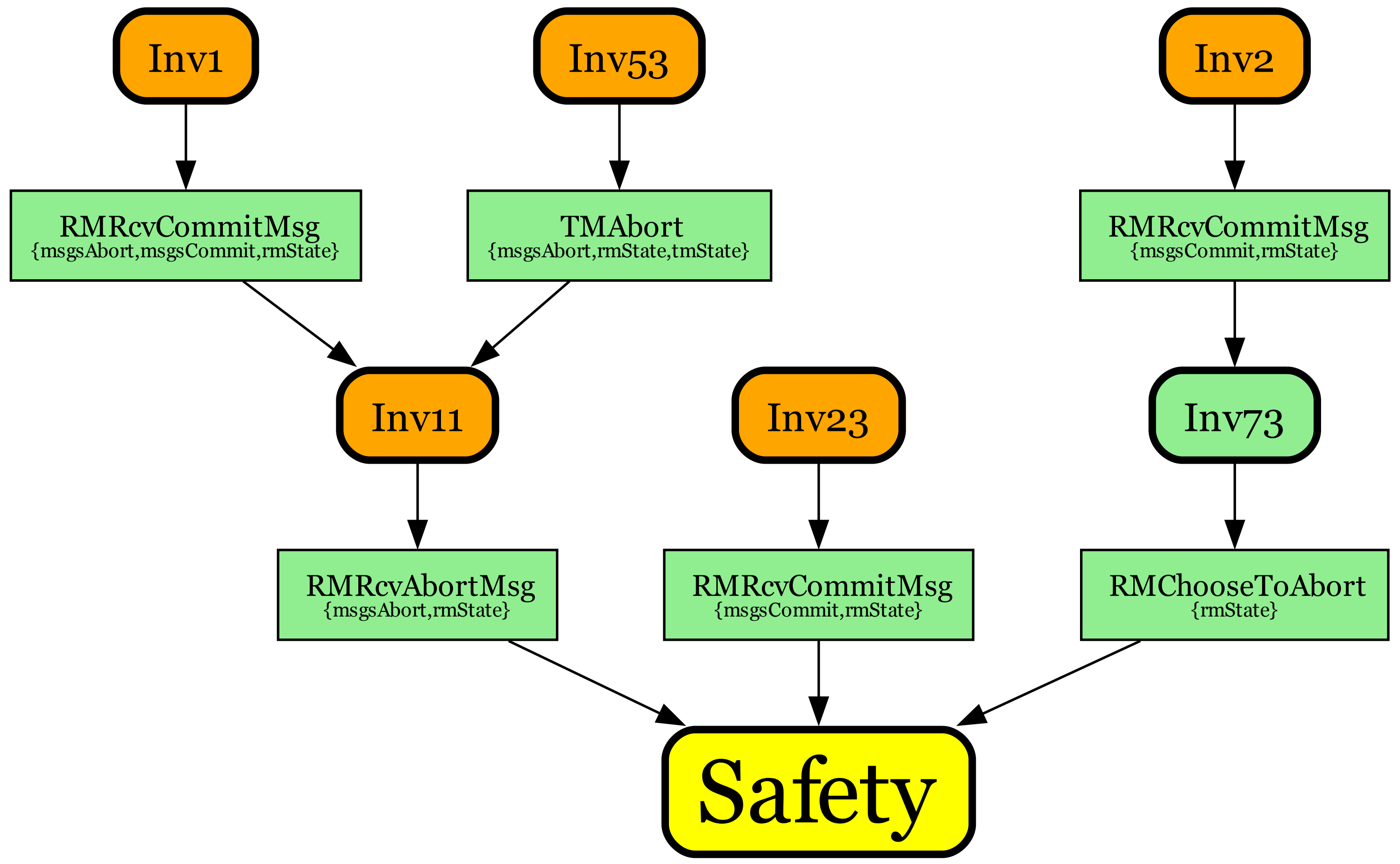
Inv2
RMRcvCommitMsg
{msgsCommit,rmState}
Inv11
Inv23
Inv73
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{rmState}
Safety

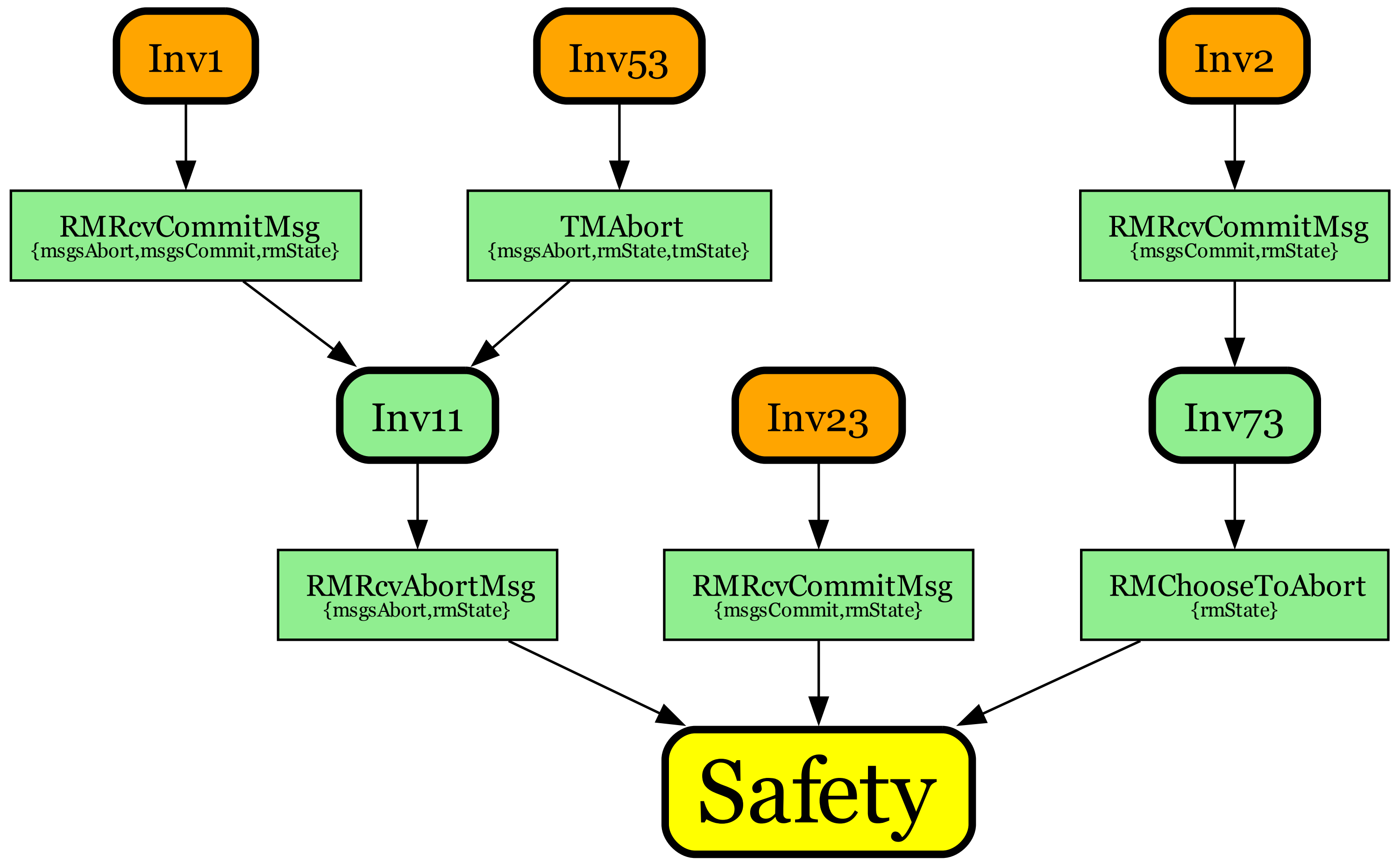


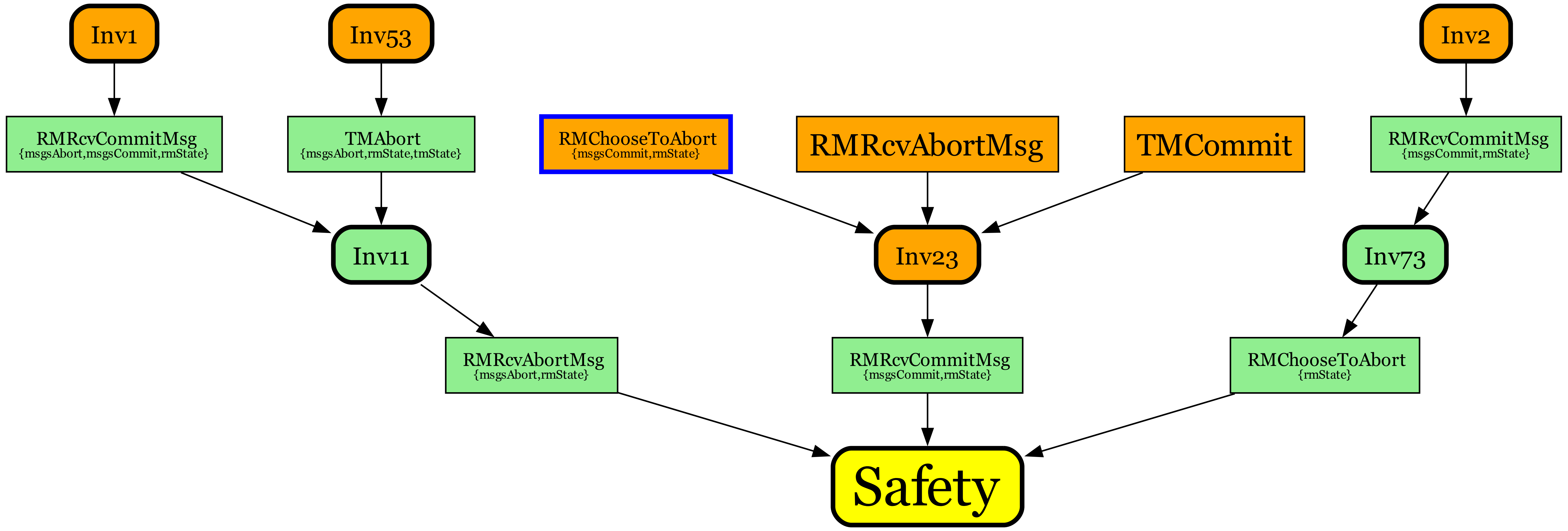
Inv2
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
RMRcvCommitMsg
{msgsCommit,rmState}
Inv11
Inv23
Inv73
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{rmState}
Safety


Inv1
Inv2
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
RMRcvCommitMsg
{msgsCommit,rmState}
Inv11
Inv23
Inv73
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{rmState}
Safety

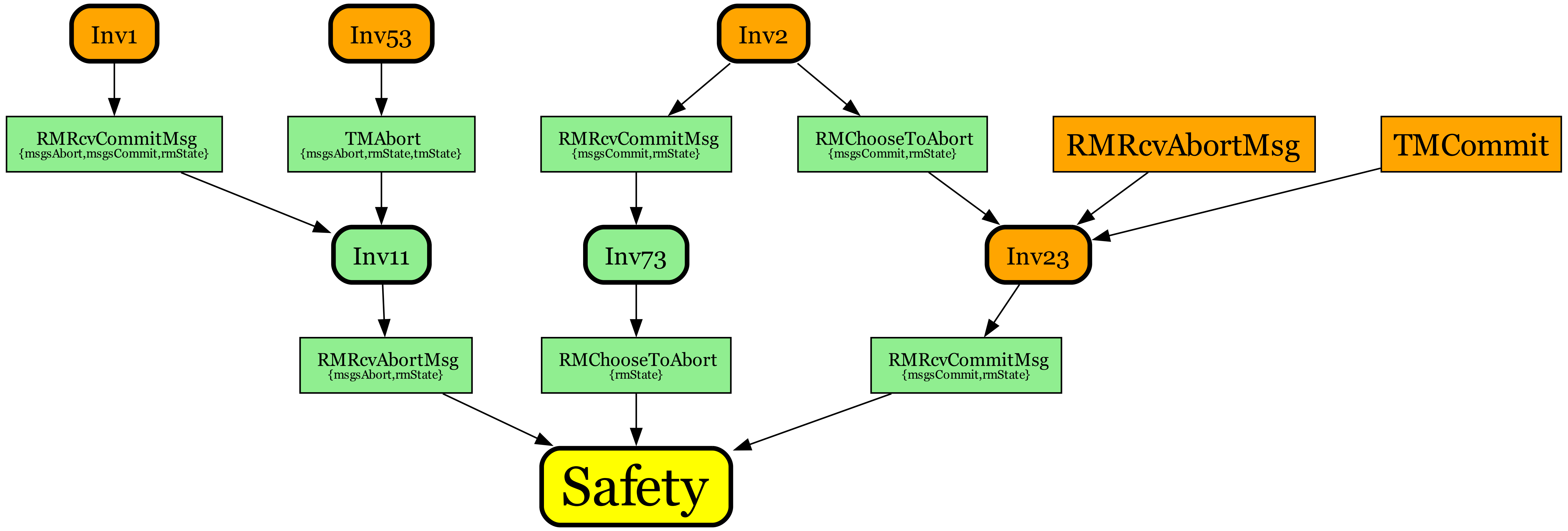

Inv1
Inv2
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
RMRcvCommitMsg
{msgsCommit,rmState}
Inv11
Inv23
Inv73
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{rmState}
Safety

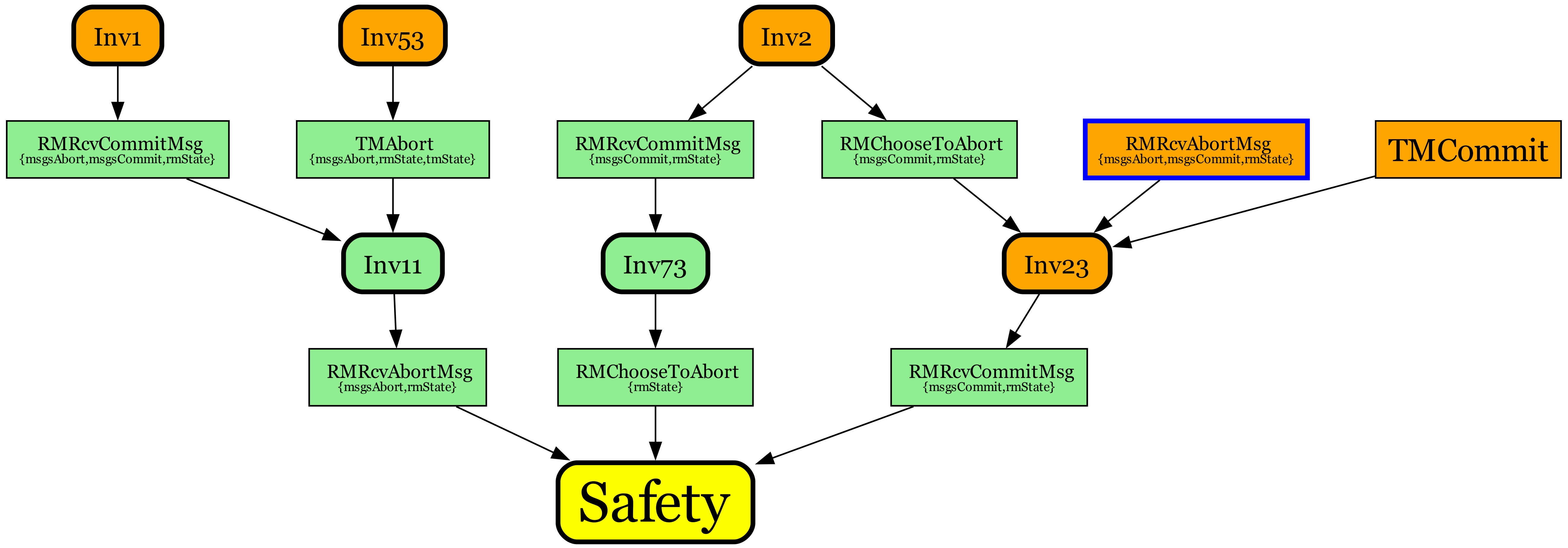

Inv1
Inv53
Inv2
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
RMRcvCommitMsg
{msgsCommit,rmState}
Inv11
Inv23
Inv73
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{rmState}
Safety

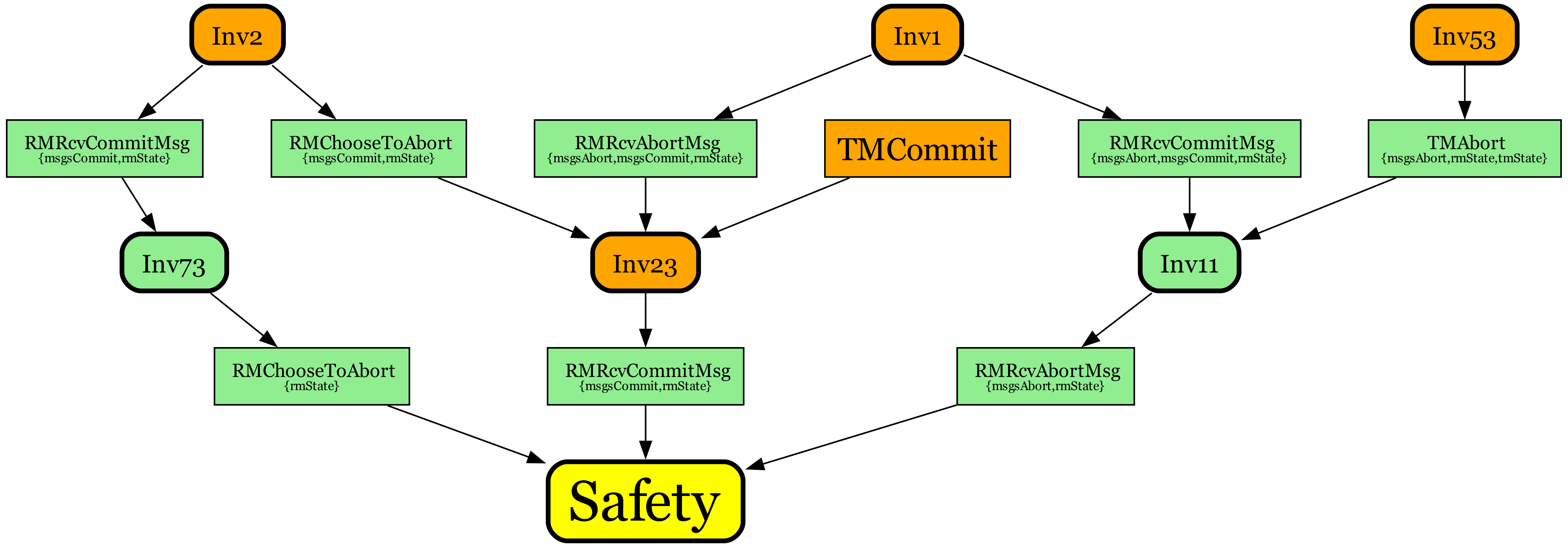

Inv1
Inv53
Inv2
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
RMRcvCommitMsg
{msgsCommit,rmState}
Inv11
Inv23
Inv73
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{rmState}
Safety



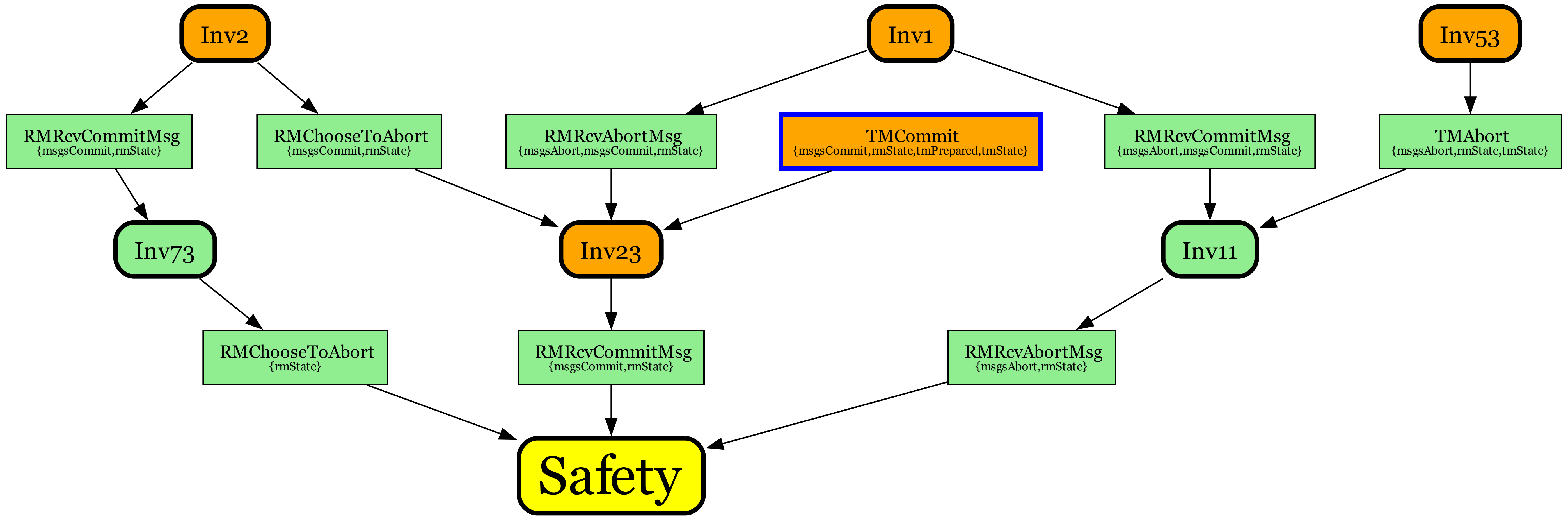
Inv1
Inv53
Inv2
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
RMChooseToAbort
{msgsCommit,rmState}
RMRcvAbortMsg
TMCommit
RMRcvCommitMsg
{msgsCommit,rmState}
Inv11
Inv23
Inv73
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{rmState}
Safety

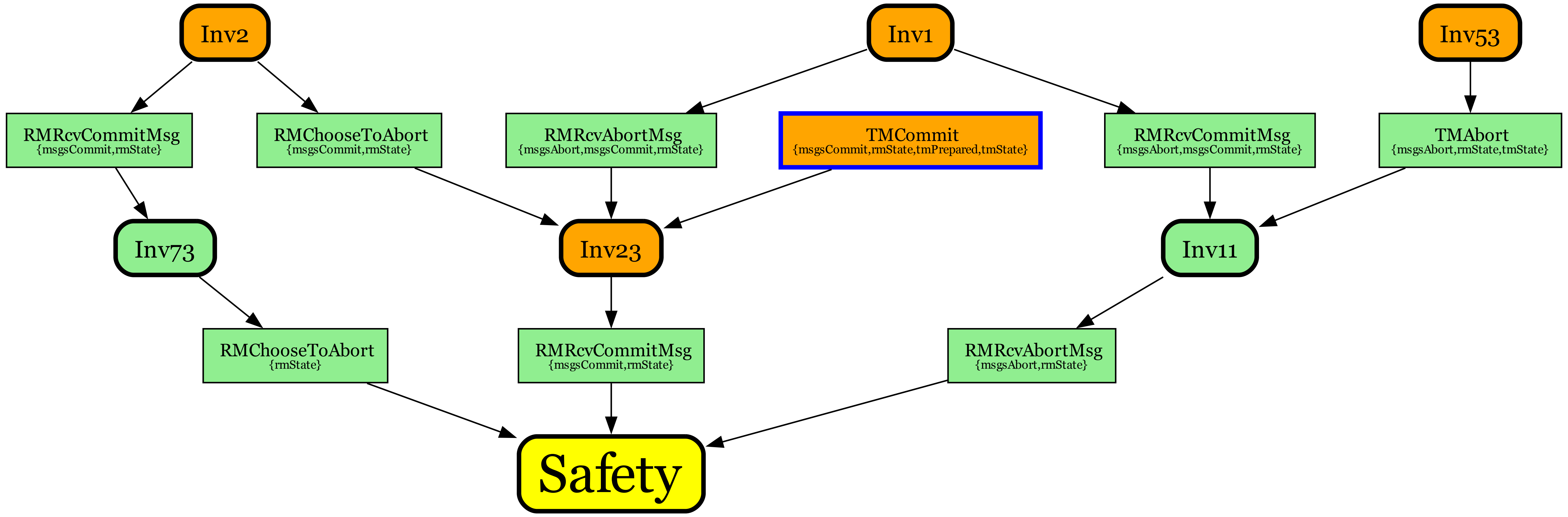

Inv1
Inv53
Inv2
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
RMRcvAbortMsg
TMCommit
Inv11
Inv73
Inv23
RMRcvAbortMsg
{msgsAbort,rmState}
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
Safety

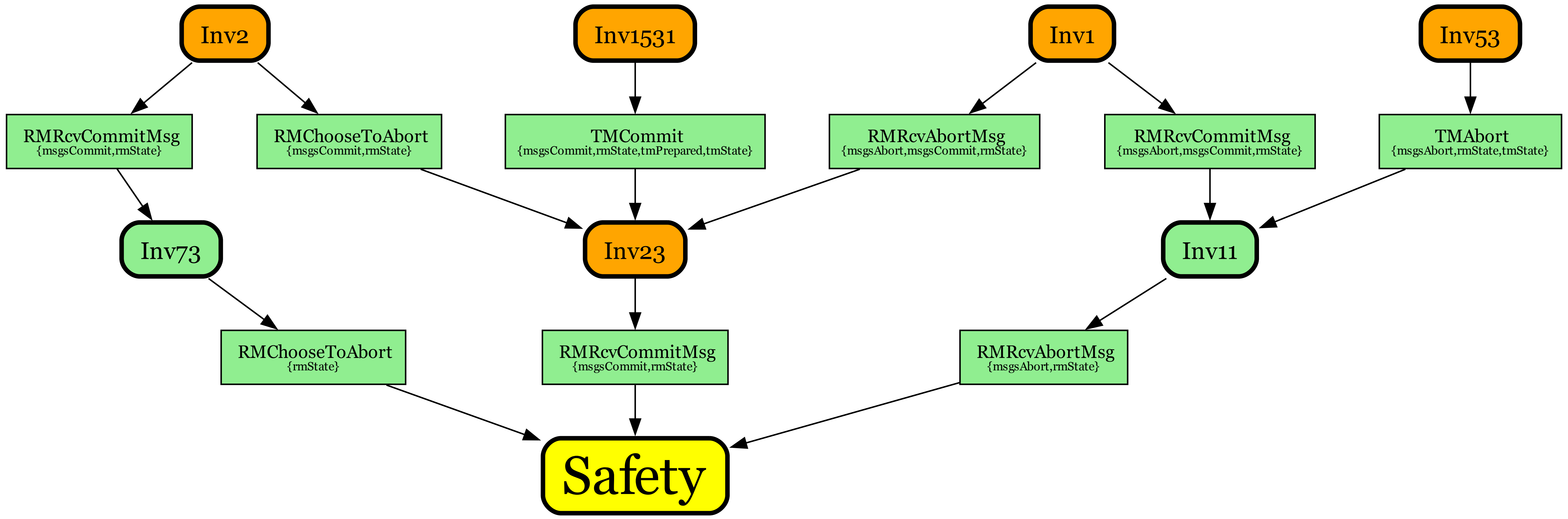

Inv1
Inv53
Inv2
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
TMCommit
Inv11
Inv73
Inv23
RMRcvAbortMsg
{msgsAbort,rmState}
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
Safety

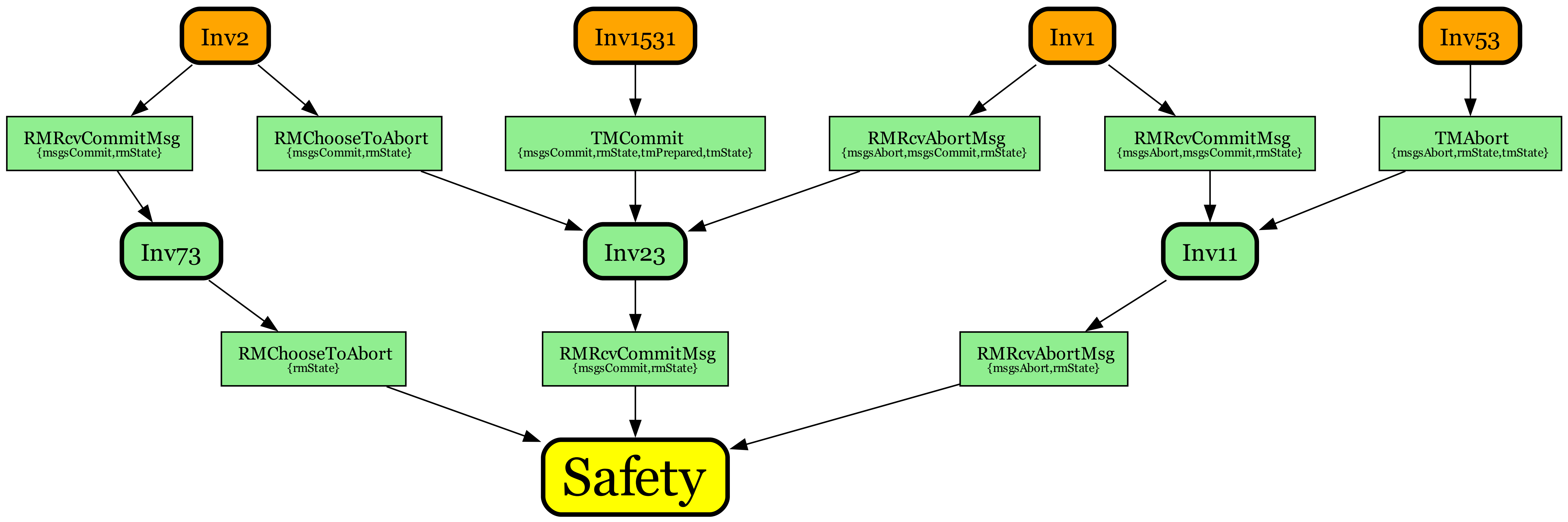


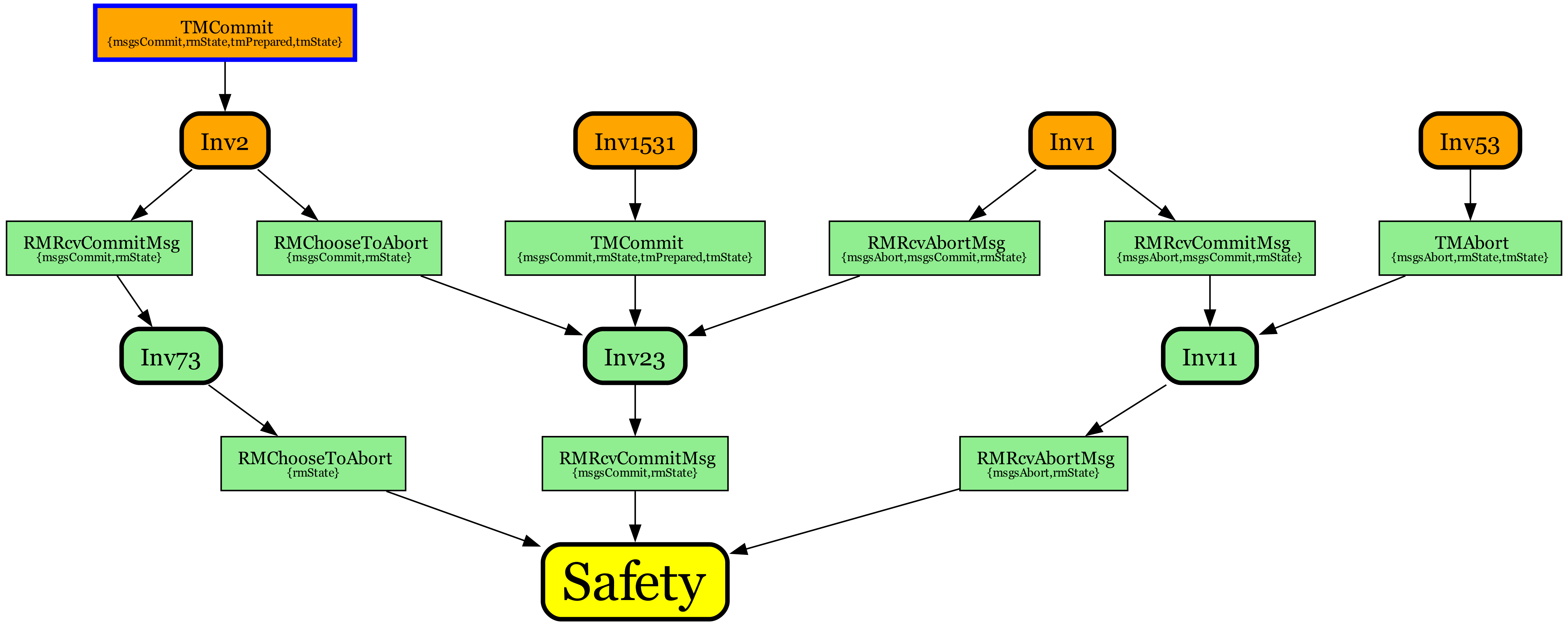
Inv2
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
TMCommit
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety


Inv2
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety

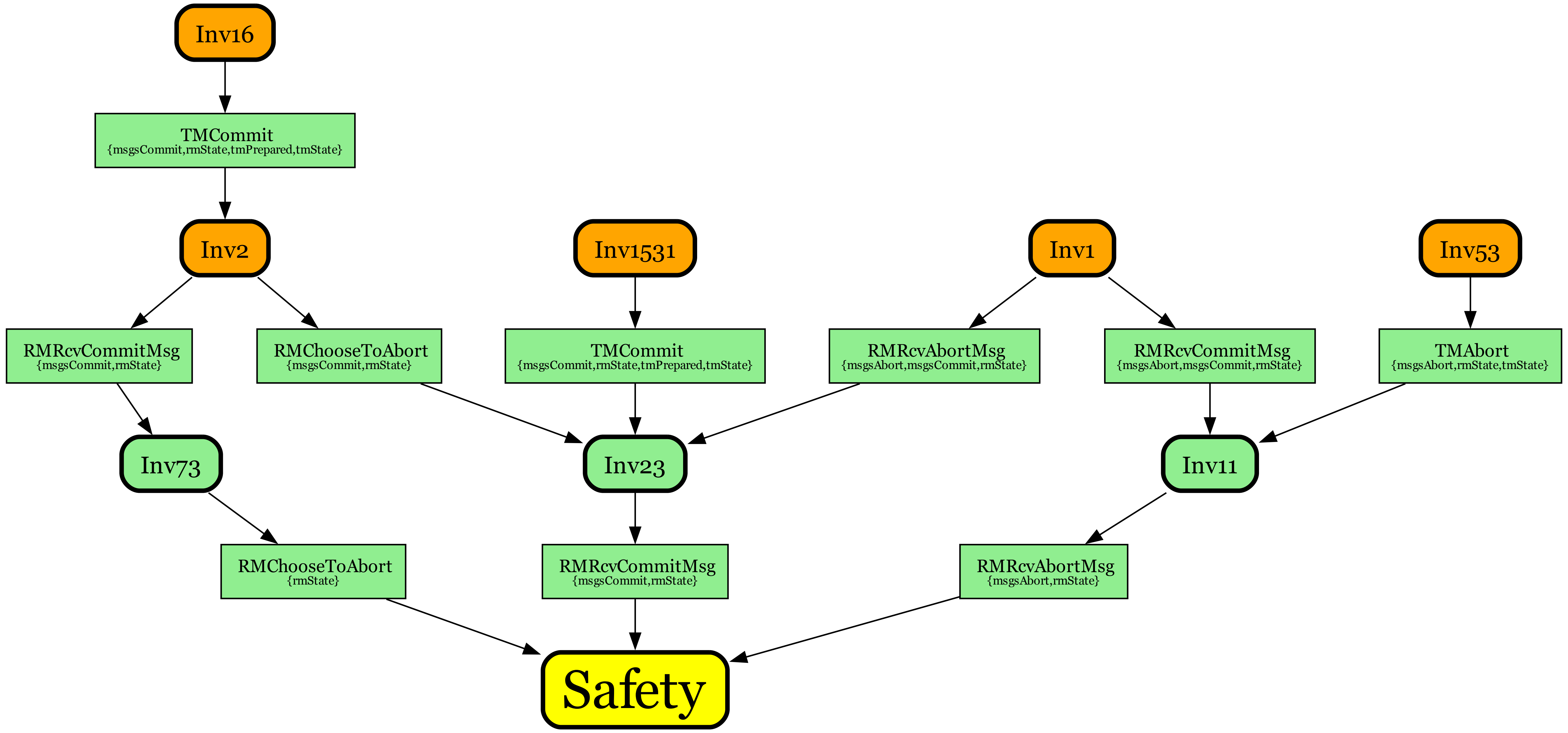

Inv2
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety

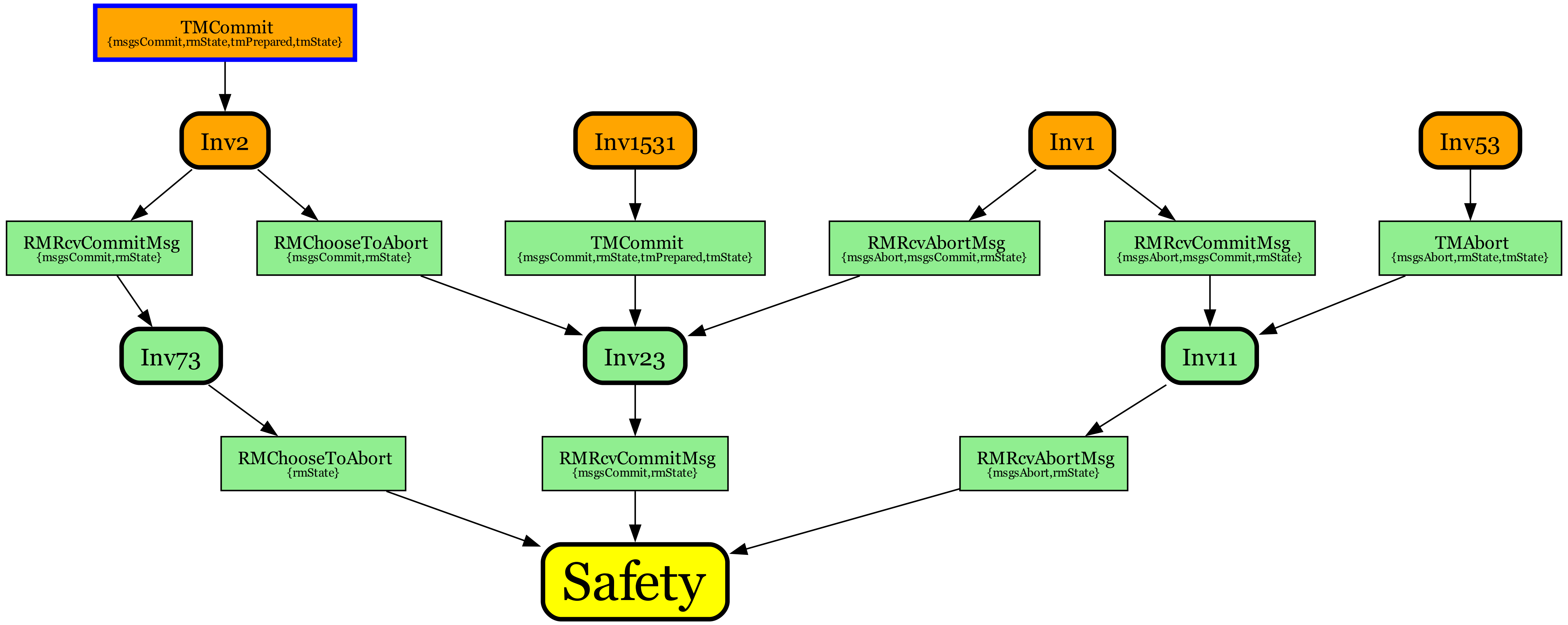

Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety


Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety

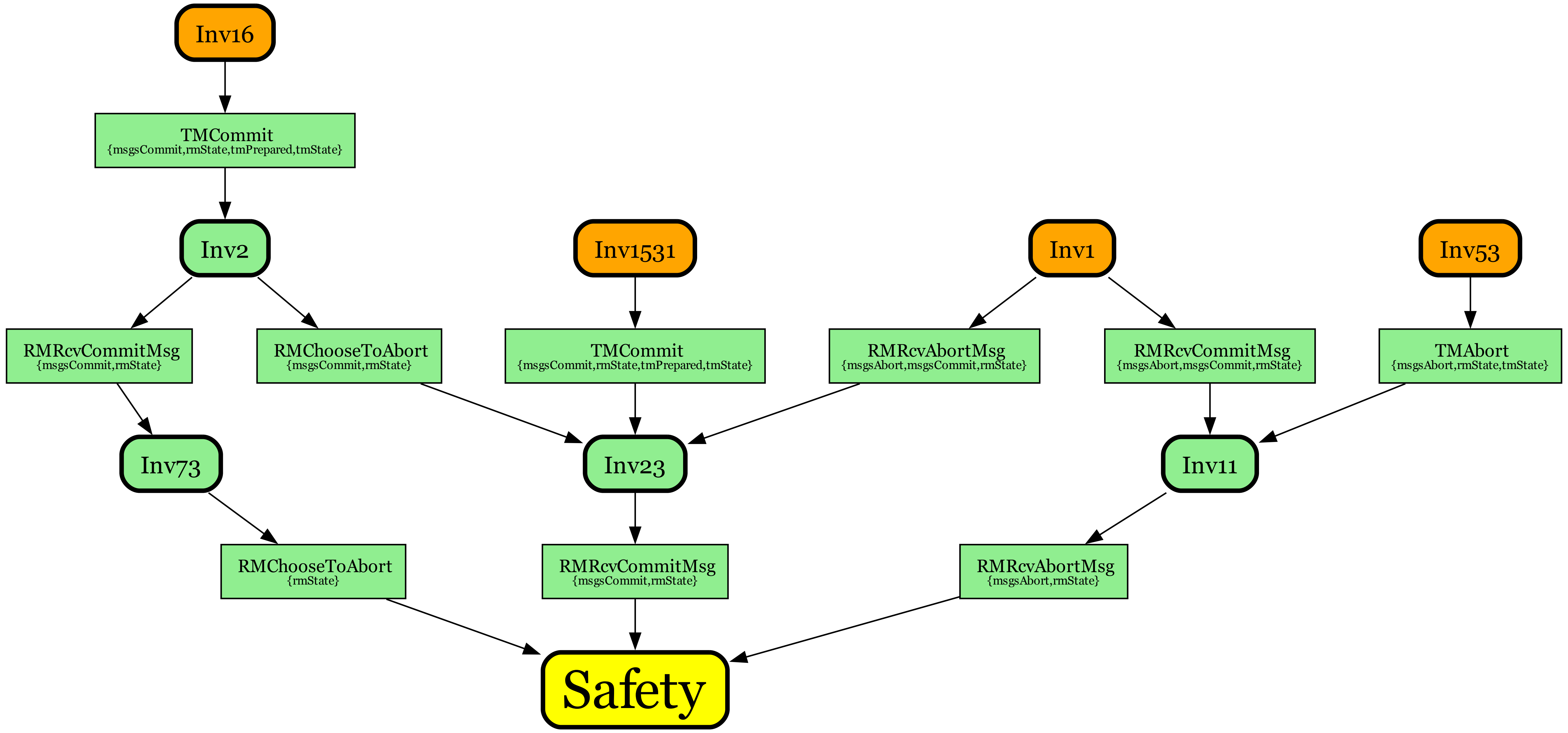


TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety

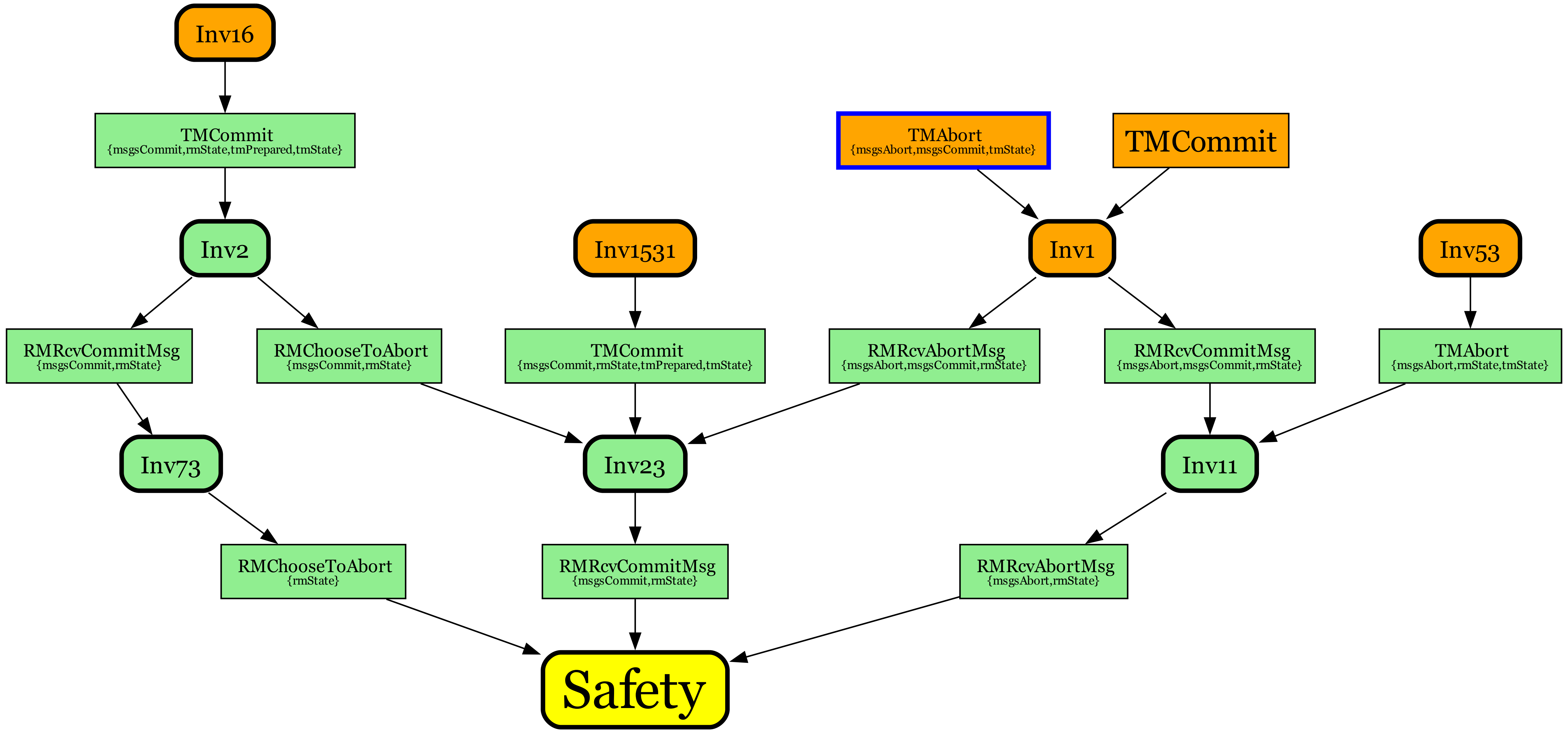

Inv16
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety



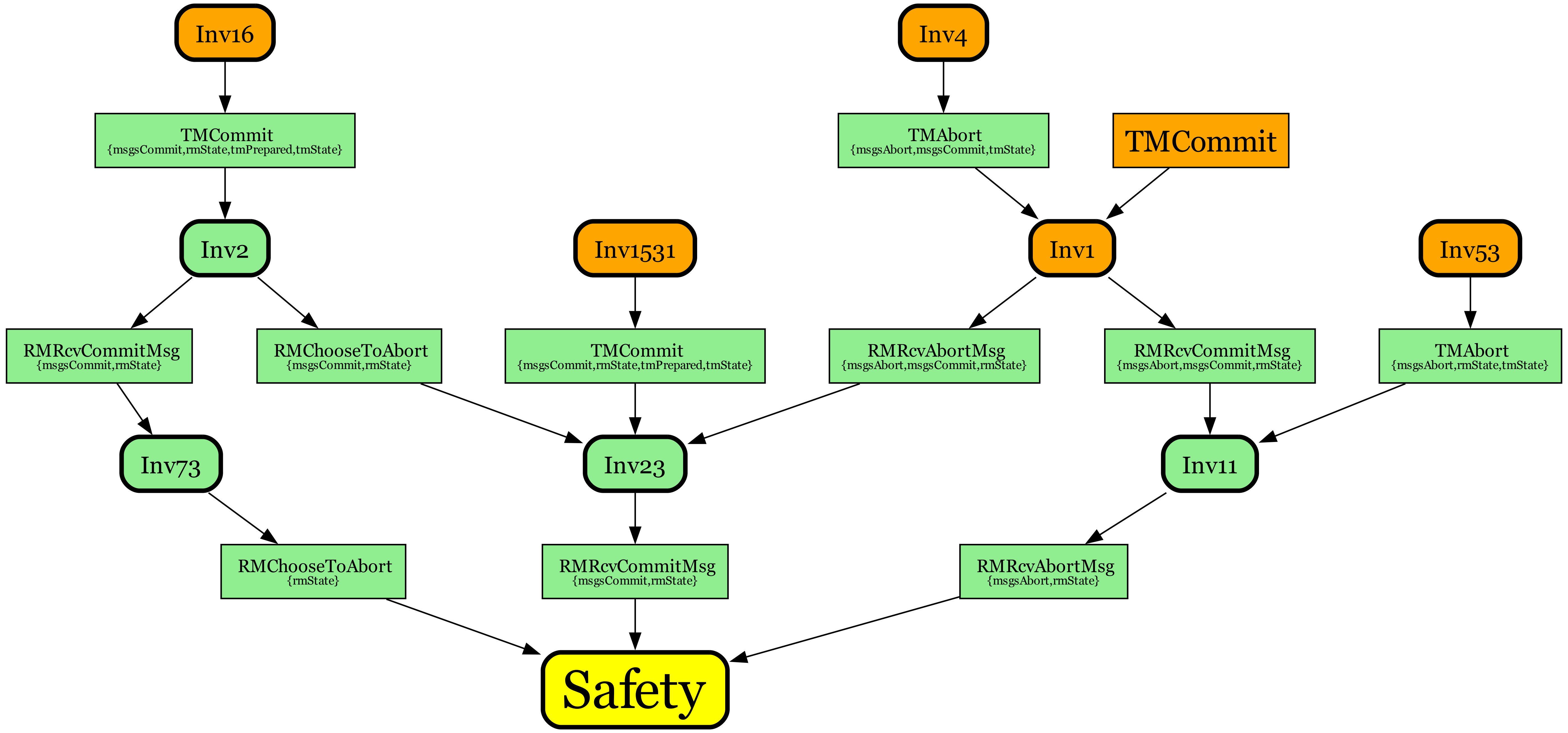
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety



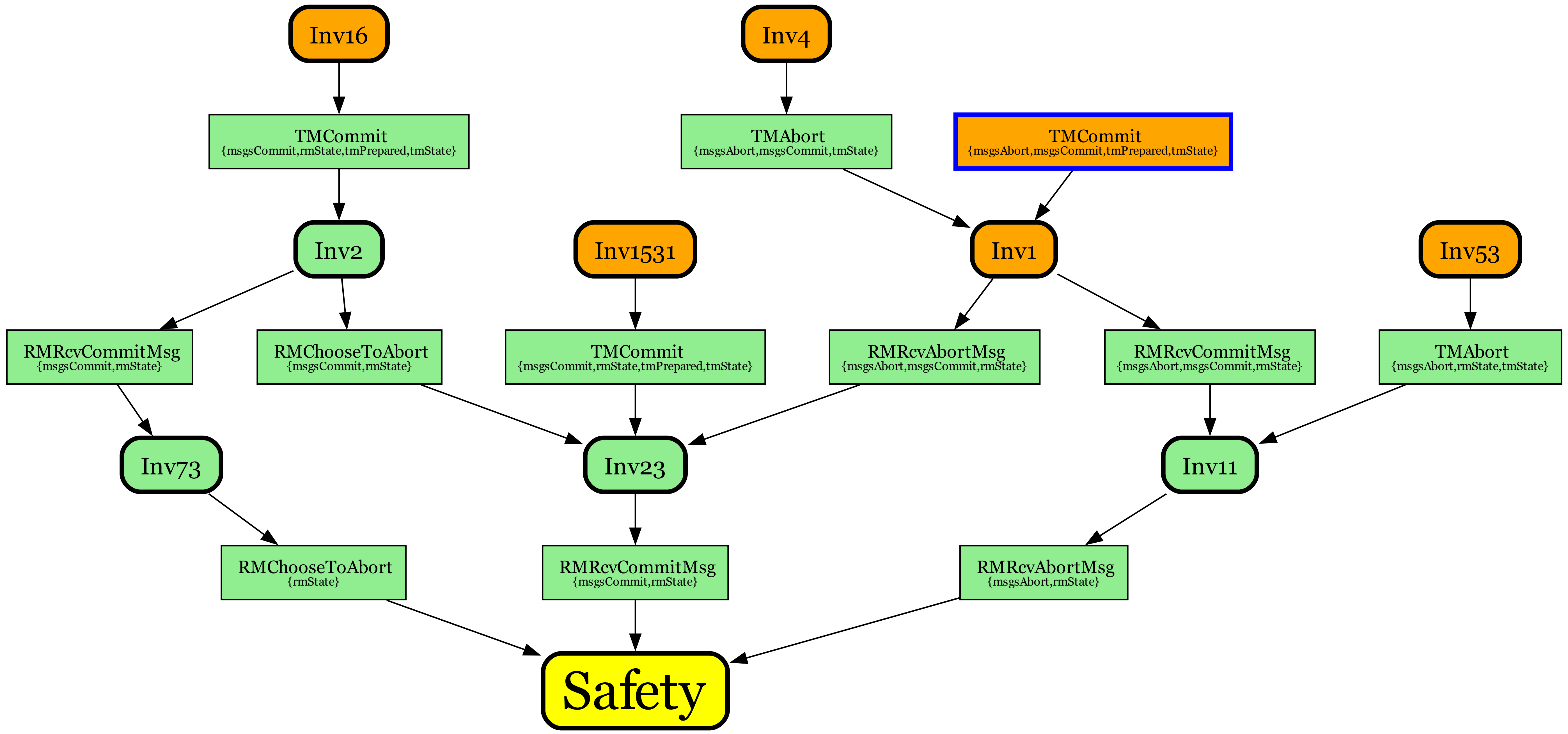
Inv16
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety



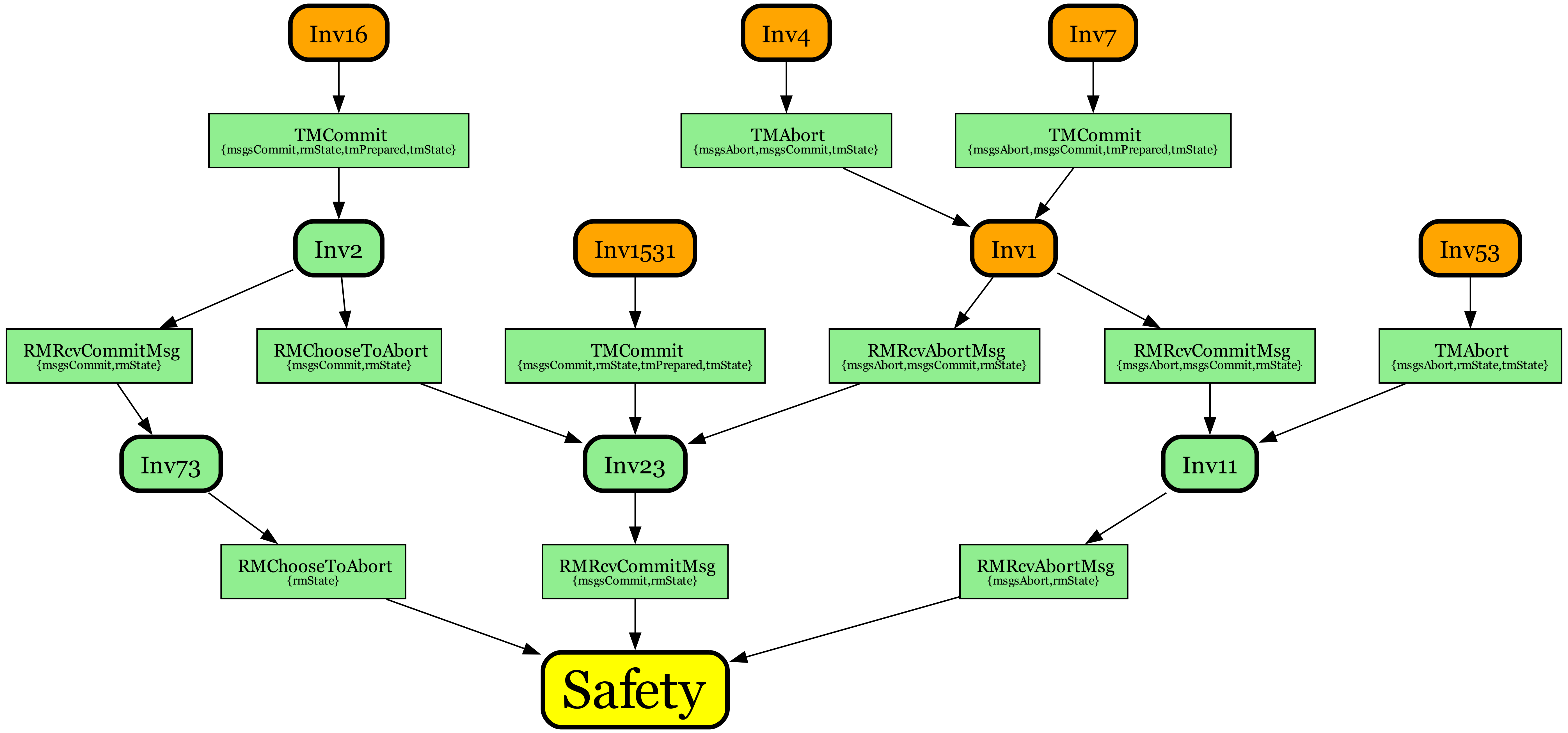
Inv16
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
Inv2
TMAbort
{msgsAbort,msgsCommit,tmState}
TMCommit
Inv1
Inv1531
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety



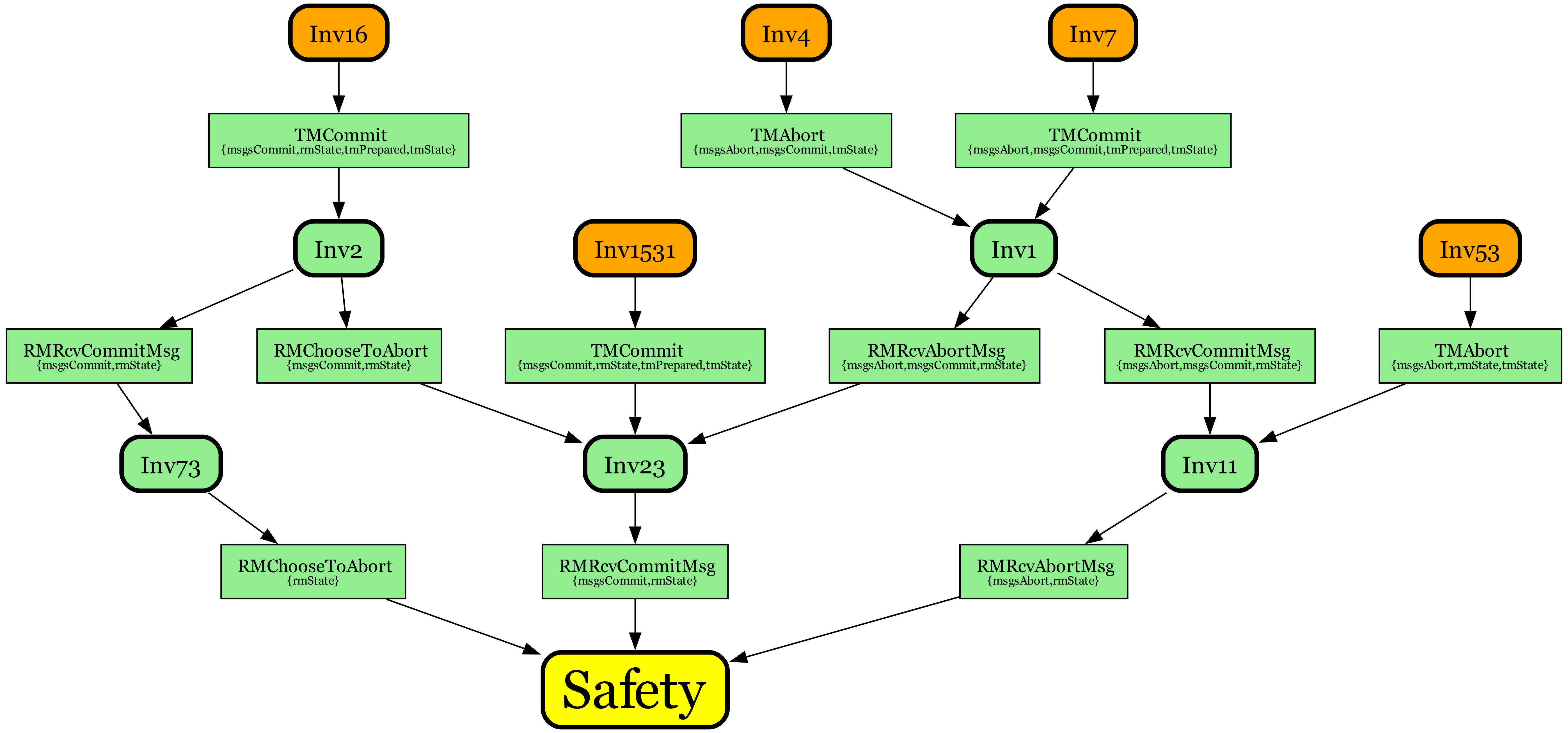
Inv16
Inv4
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
TMCommit
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety



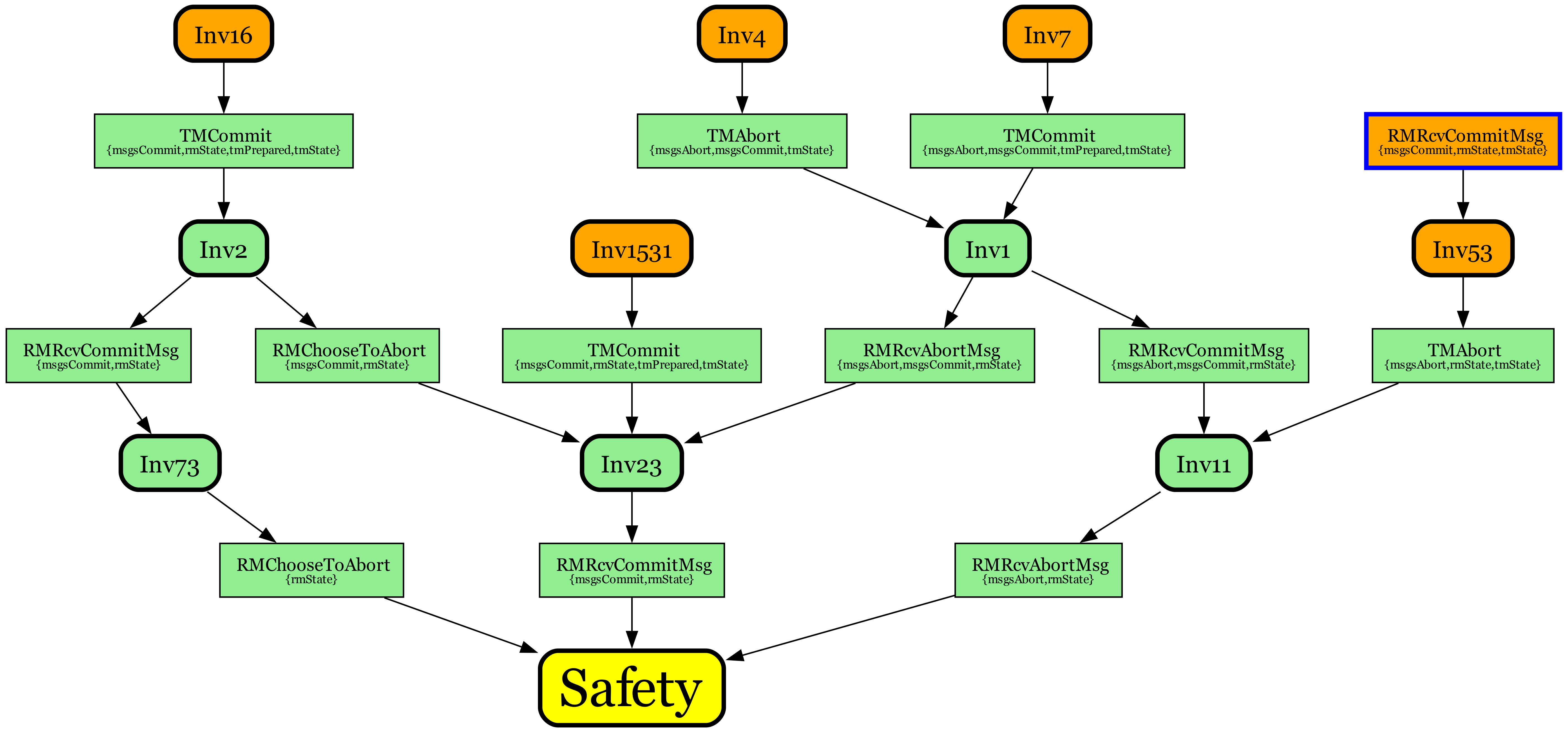
Inv16
Inv4
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety



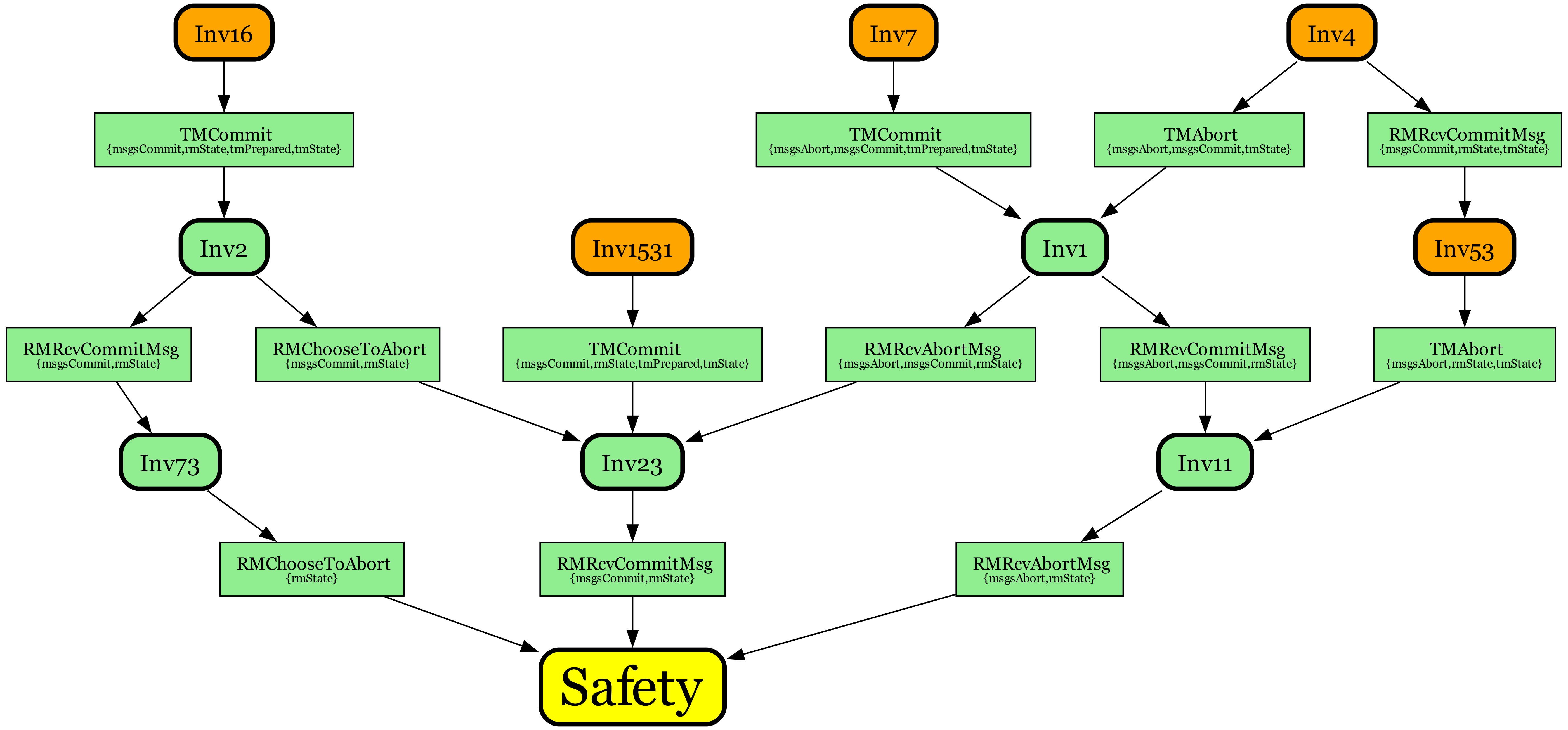
Inv16
Inv4
Inv7
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety



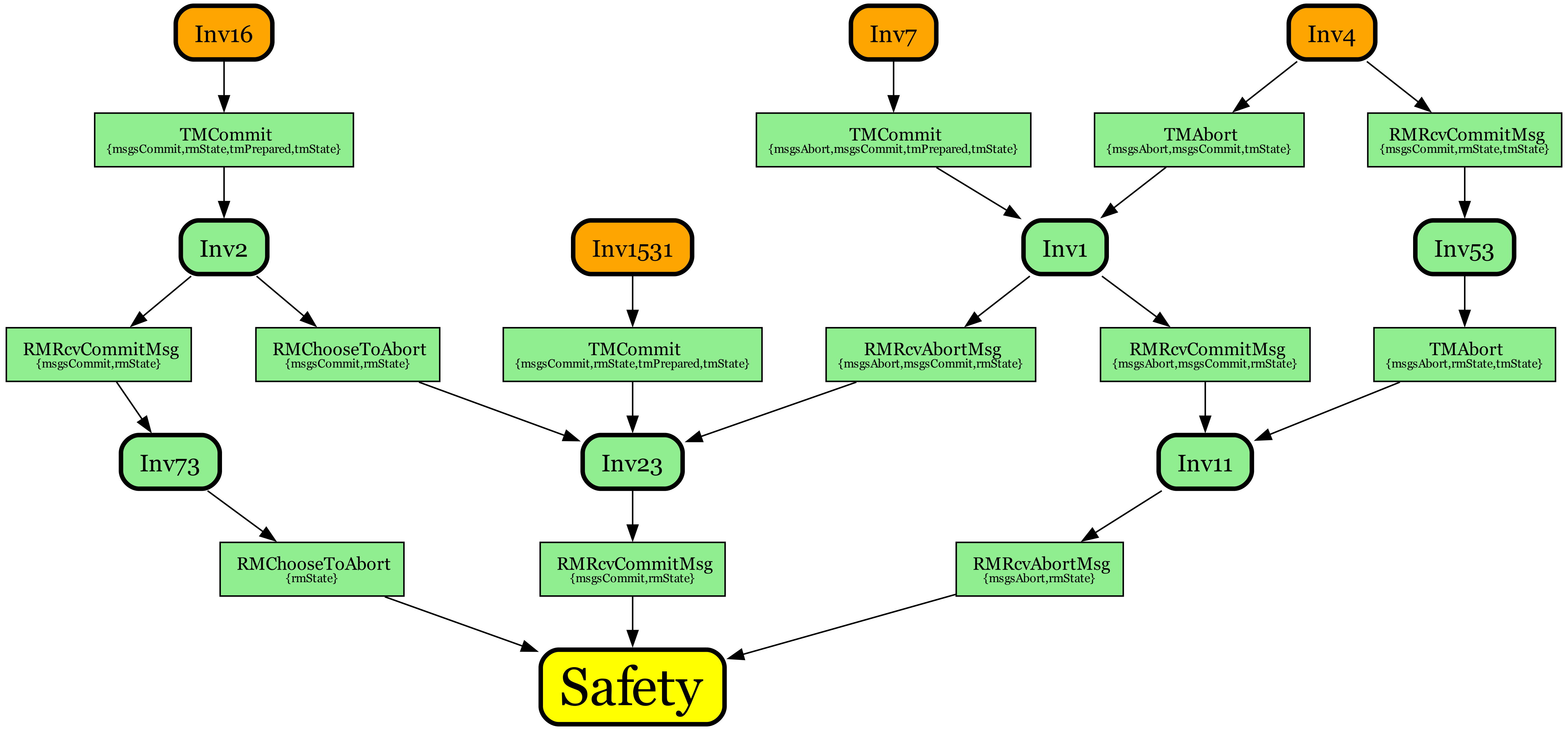
Inv16
Inv4
Inv7
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety

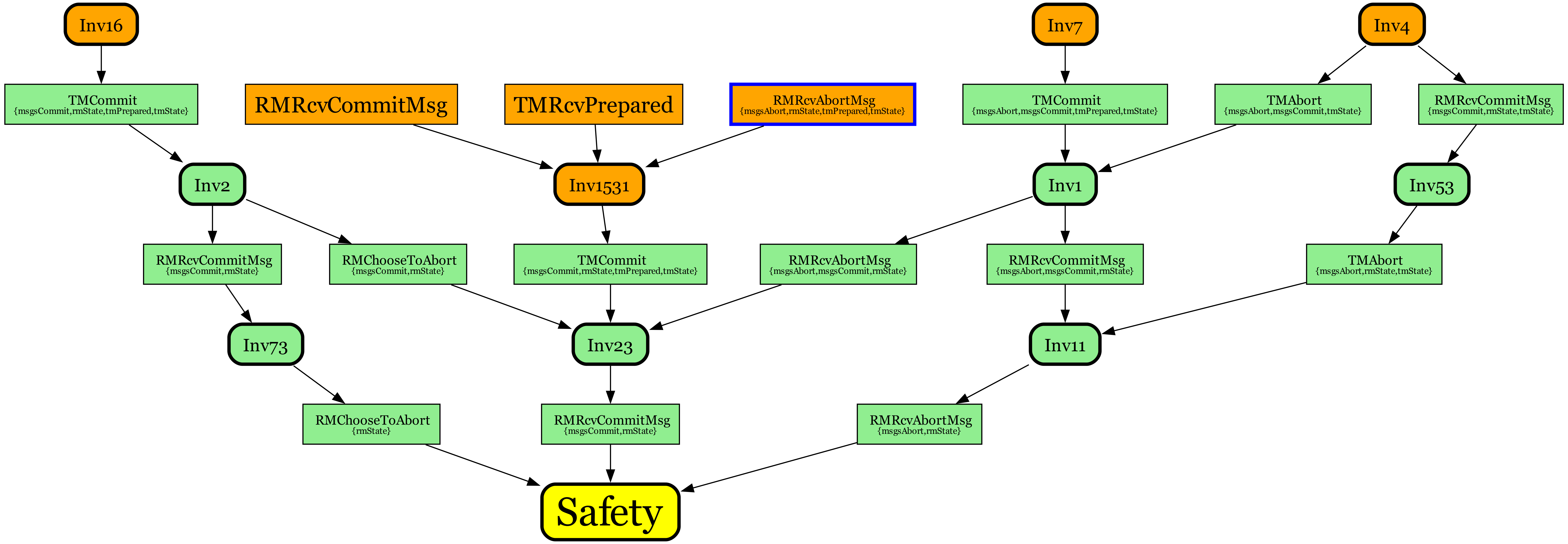


Inv16
Inv4
Inv7
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety

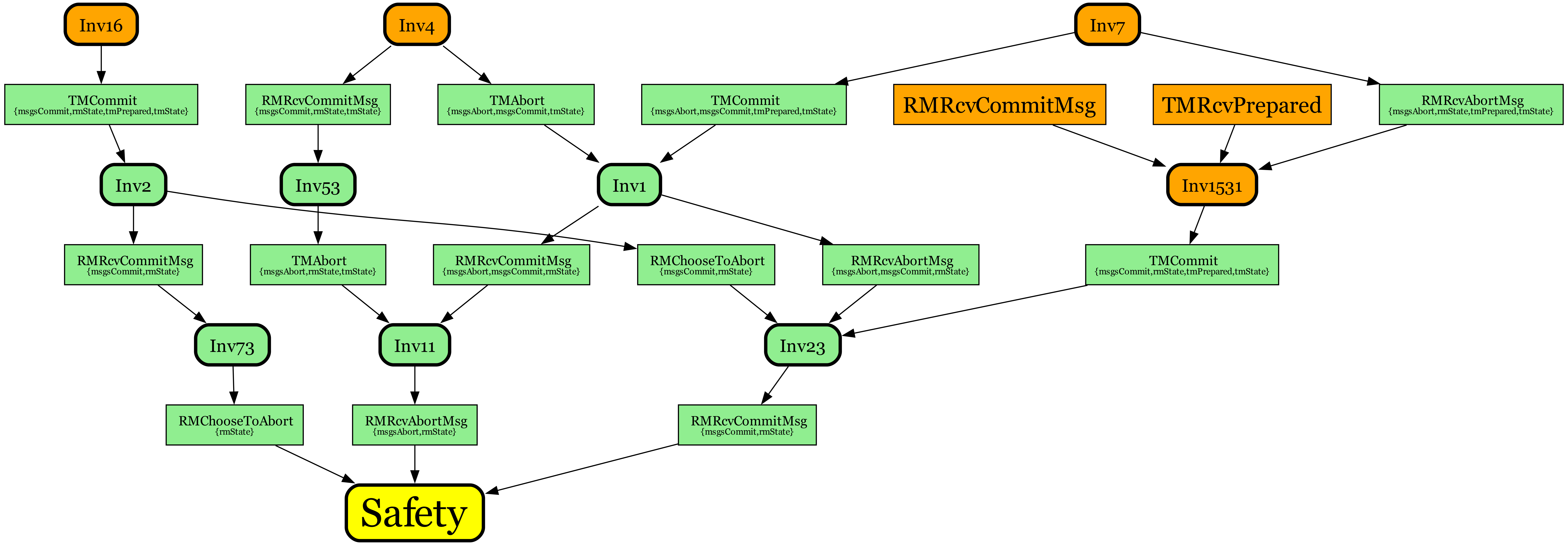


Inv16
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
Inv7
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
Inv4
TMAbort
{msgsAbort,msgsCommit,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety

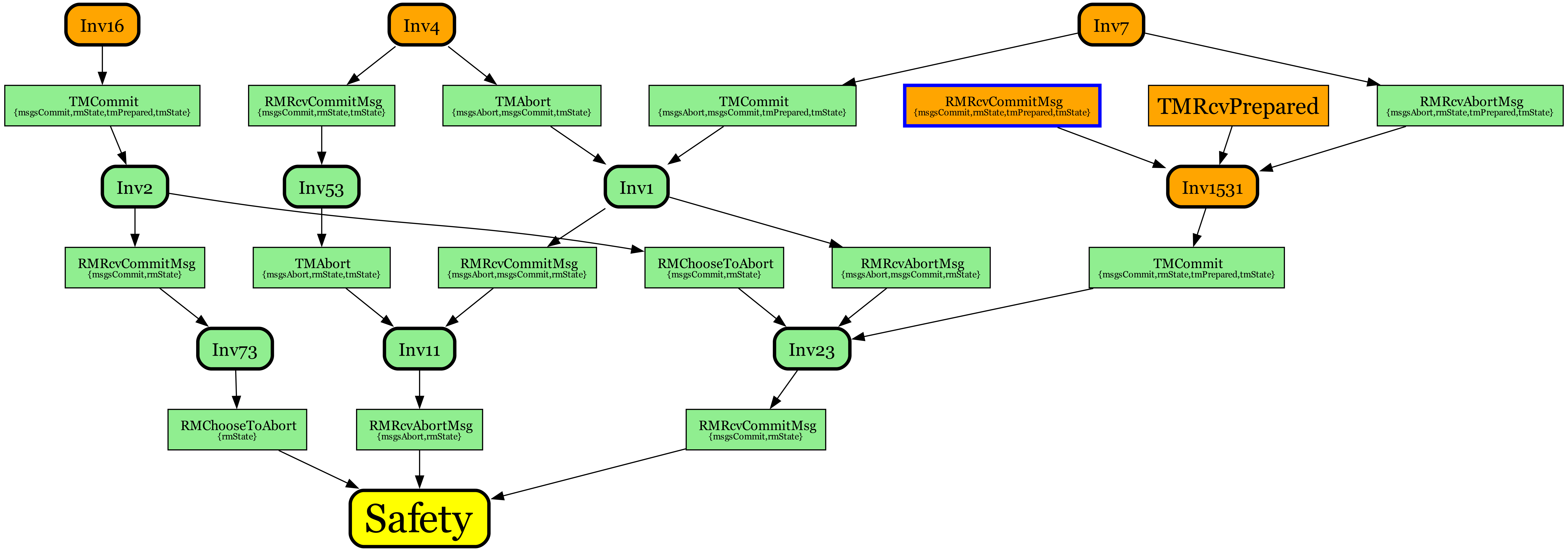

Inv16
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
Inv7
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
Inv4
TMAbort
{msgsAbort,msgsCommit,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety

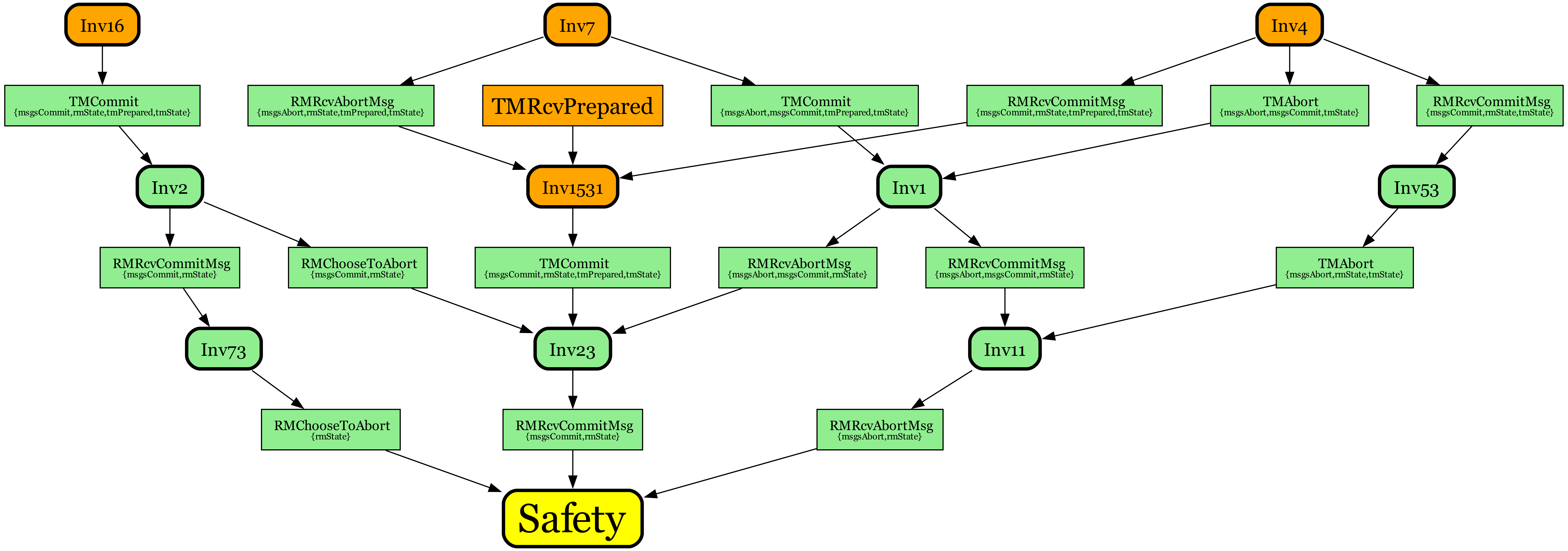


Inv16
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvCommitMsg
TMRcvPrepared
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
Inv7
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
Inv4
TMAbort
{msgsAbort,msgsCommit,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety

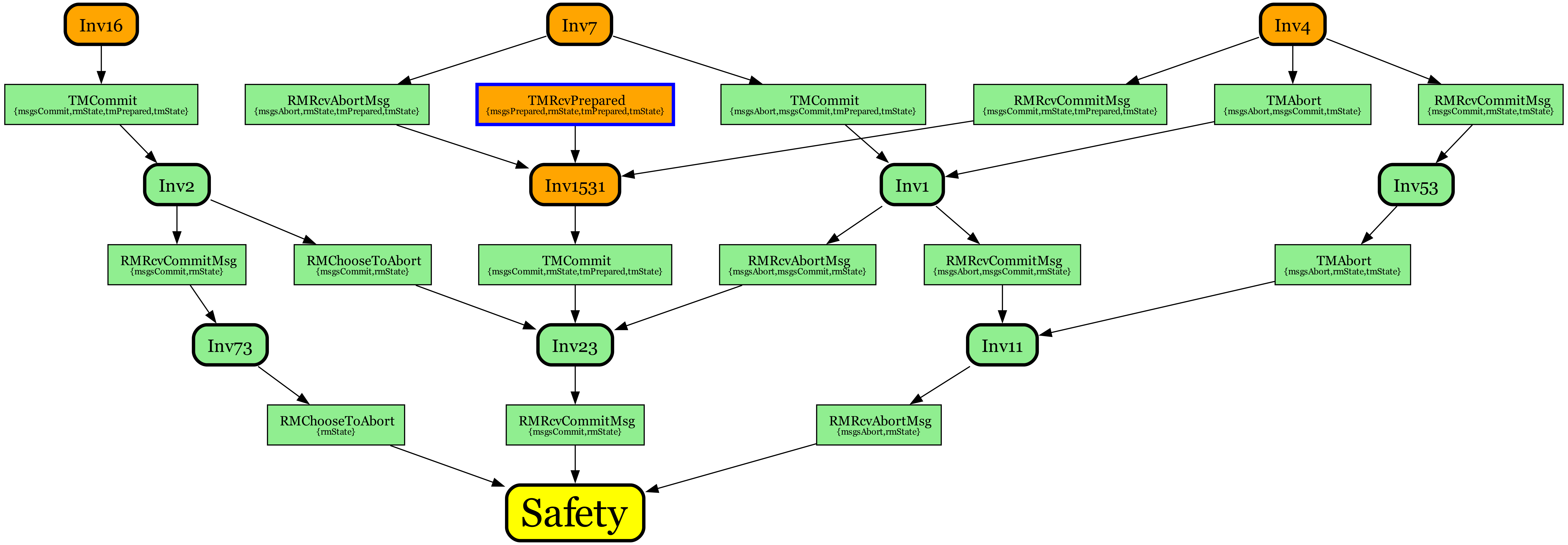


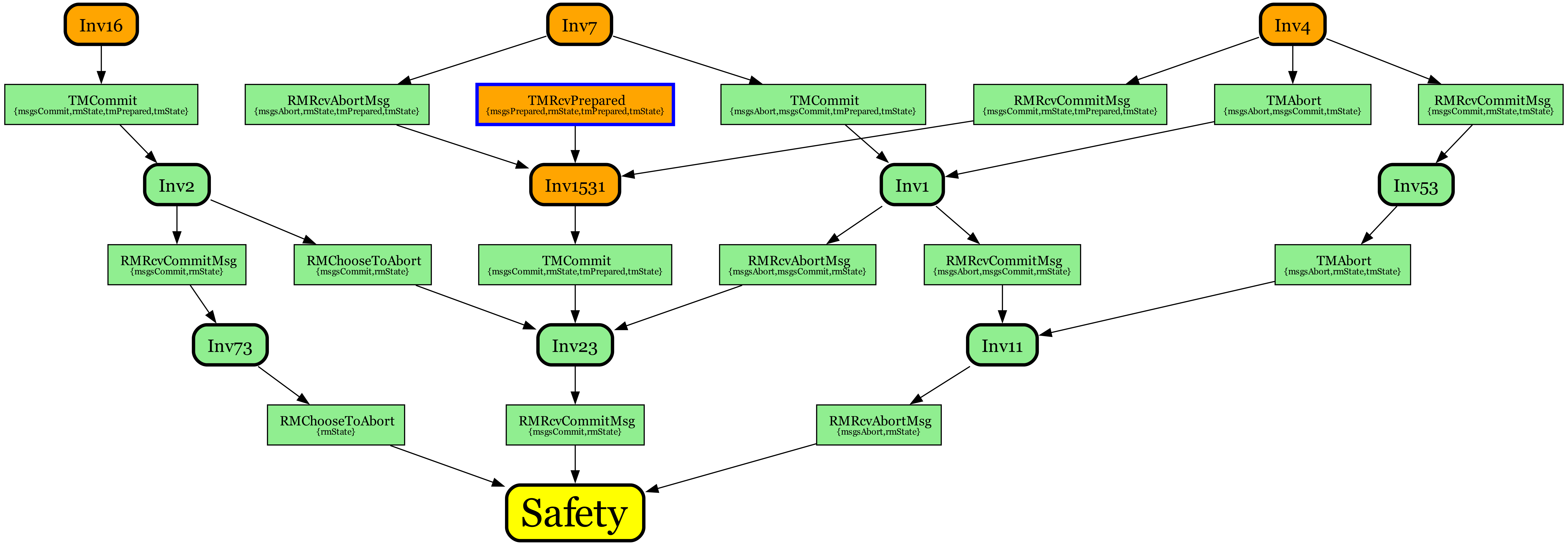
Inv16
Inv4
Inv7
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
RMRcvCommitMsg
TMRcvPrepared
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
Inv2
Inv53
Inv1
Inv1531
RMRcvCommitMsg
{msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
Inv73
Inv11
Inv23
RMChooseToAbort
{rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
Safety

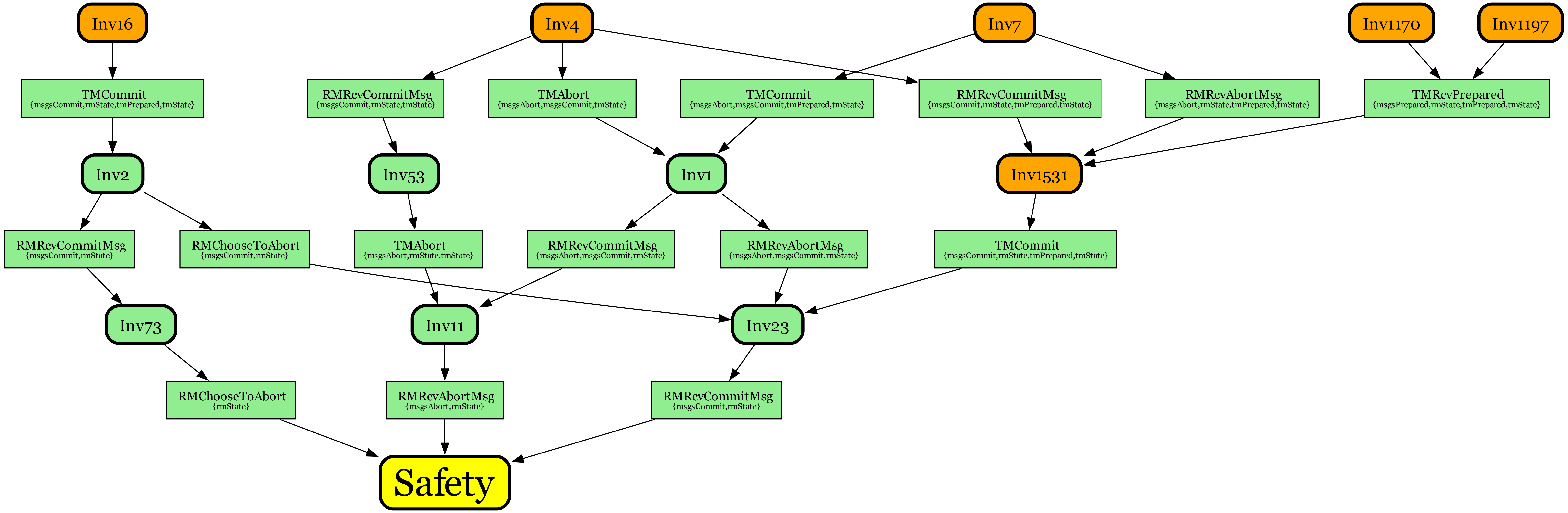

Inv16
Inv4
Inv7
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
TMRcvPrepared
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
Inv2
Inv53
Inv1
Inv1531
RMRcvCommitMsg
{msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
Inv73
Inv11
Inv23
RMChooseToAbort
{rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
Safety



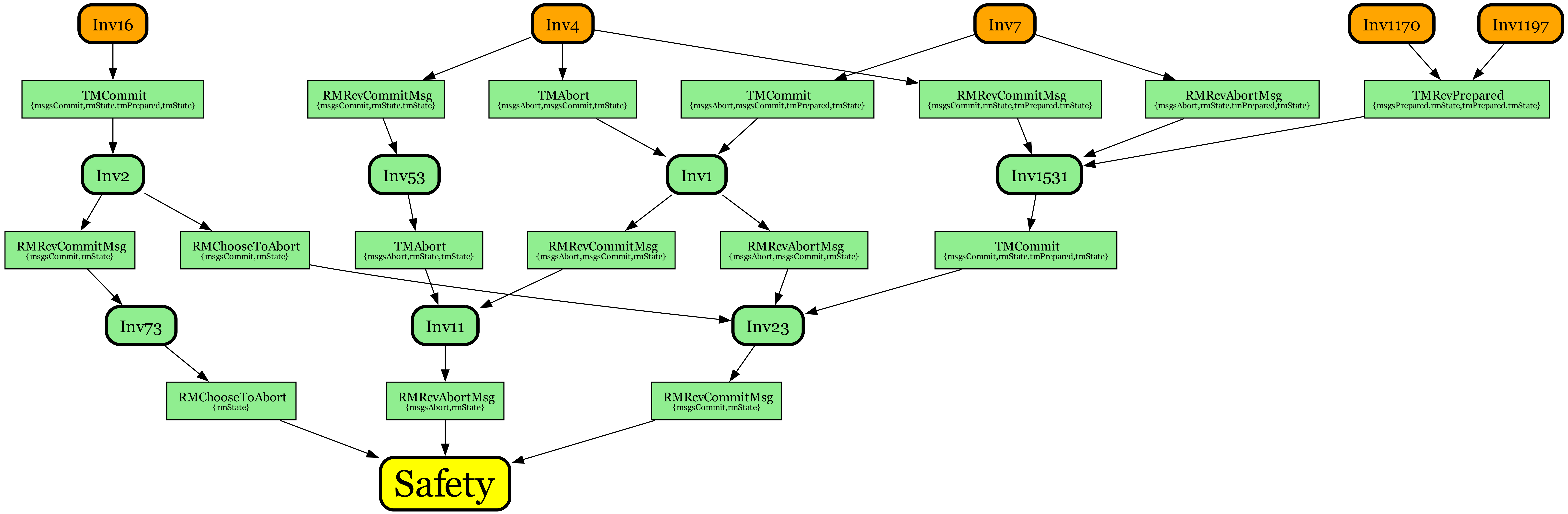
Inv16
Inv7
Inv4
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
TMRcvPrepared
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety


Inv16
Inv7
Inv4
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety

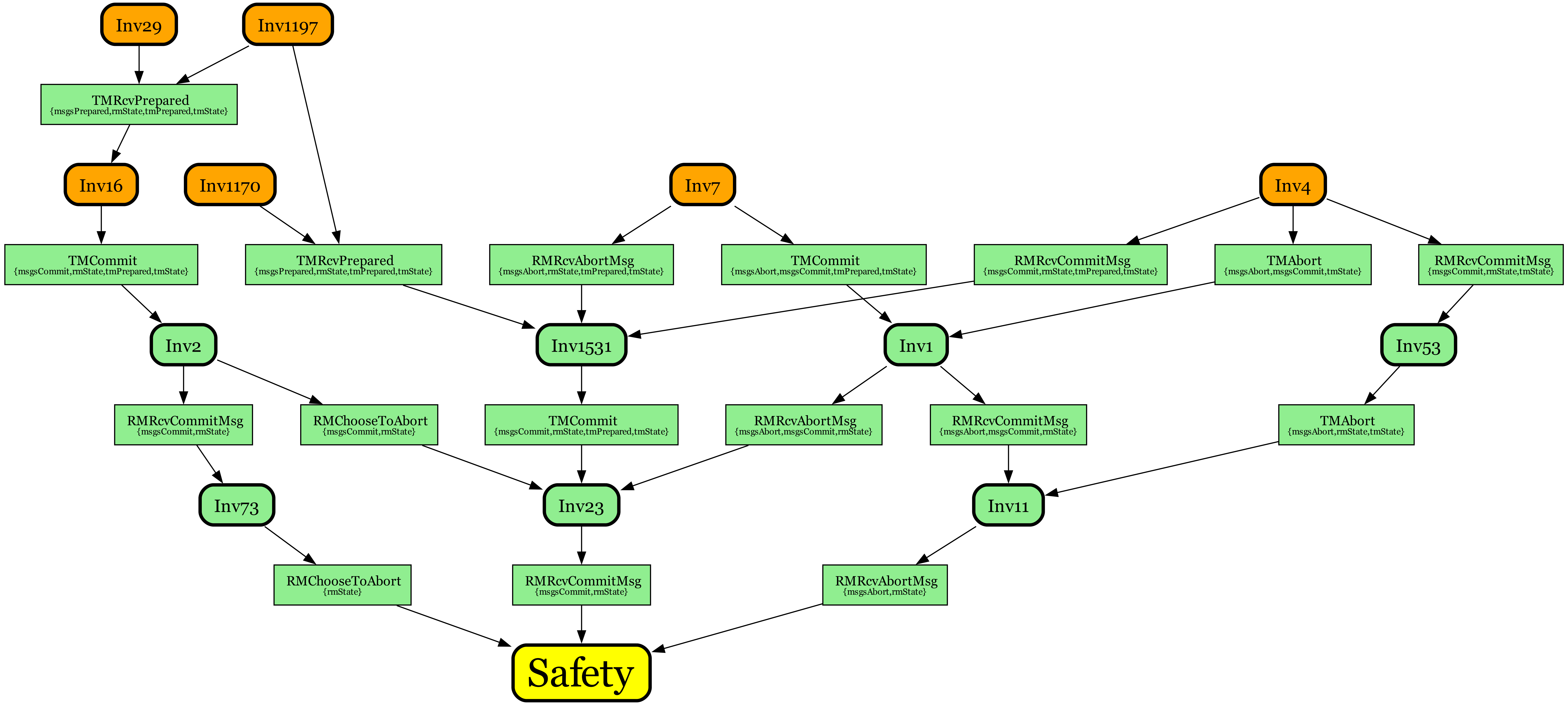

Inv16
Inv7
Inv4
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety

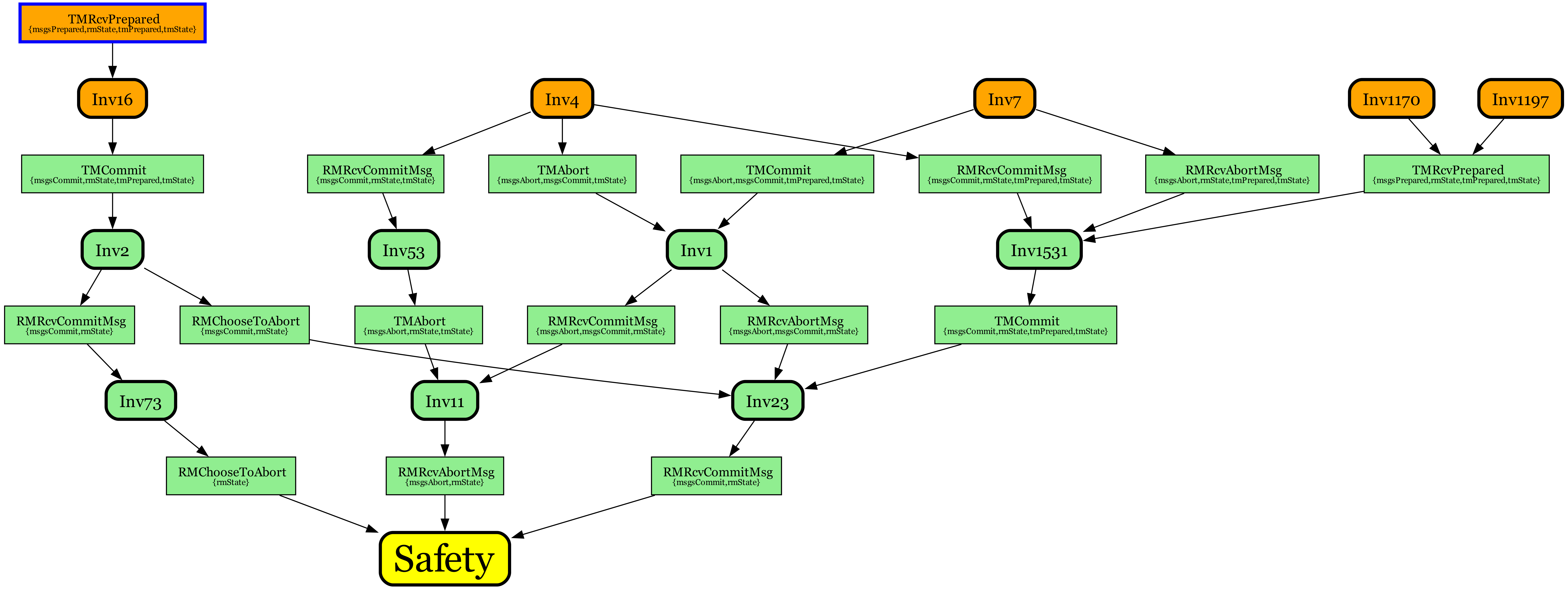


Inv16
Inv4
Inv7
Inv1170
Inv1197
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
Inv2
Inv53
Inv1
Inv1531
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
Inv73
Inv11
Inv23
RMChooseToAbort
{rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
Safety

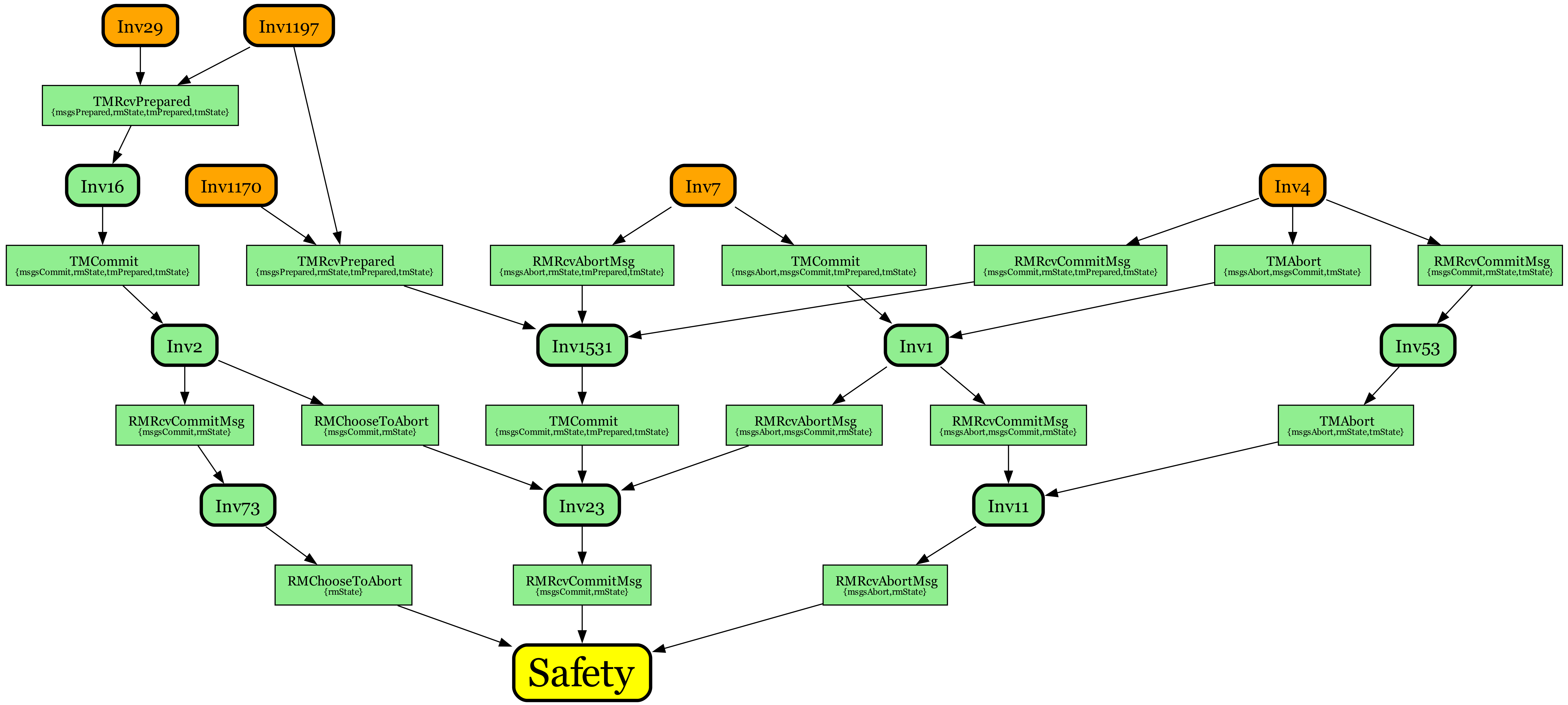


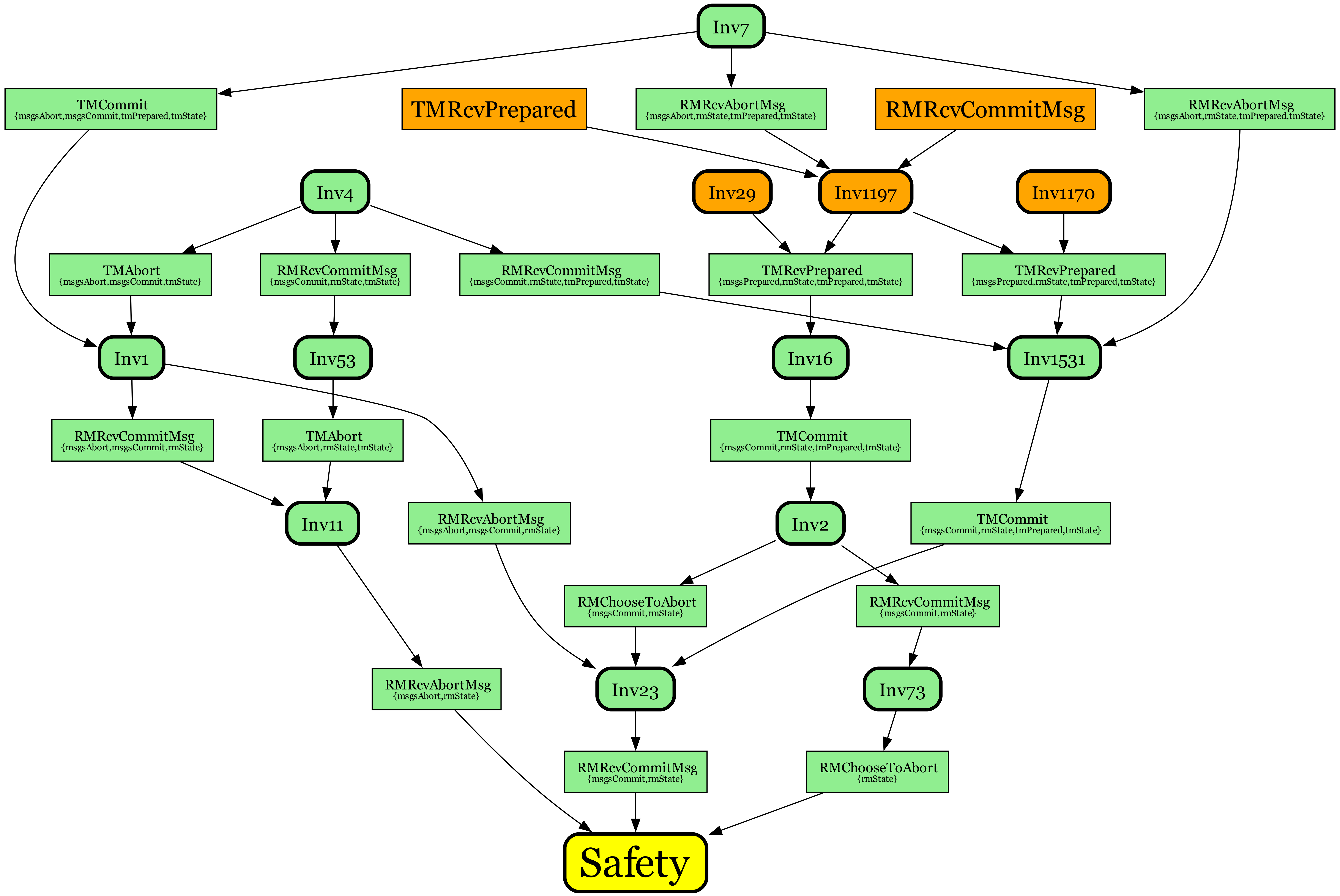
Inv16
Inv4
Inv7
Inv1170
Inv1197
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
Inv2
Inv53
Inv1
Inv1531
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
Inv73
Inv11
Inv23
RMChooseToAbort
{rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
Safety

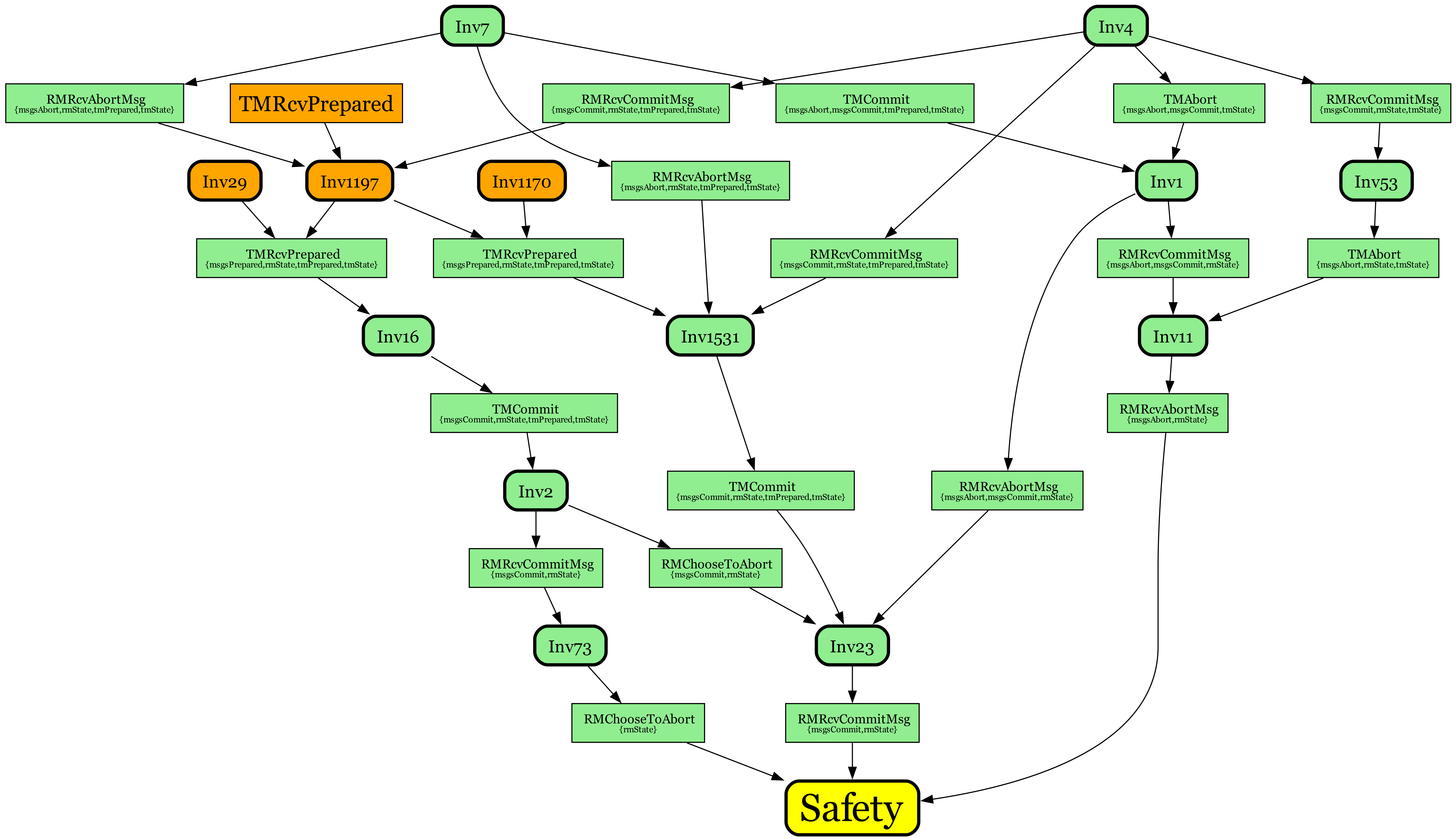

Inv29
Inv1197
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
Inv16
Inv1170
Inv7
Inv4
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety



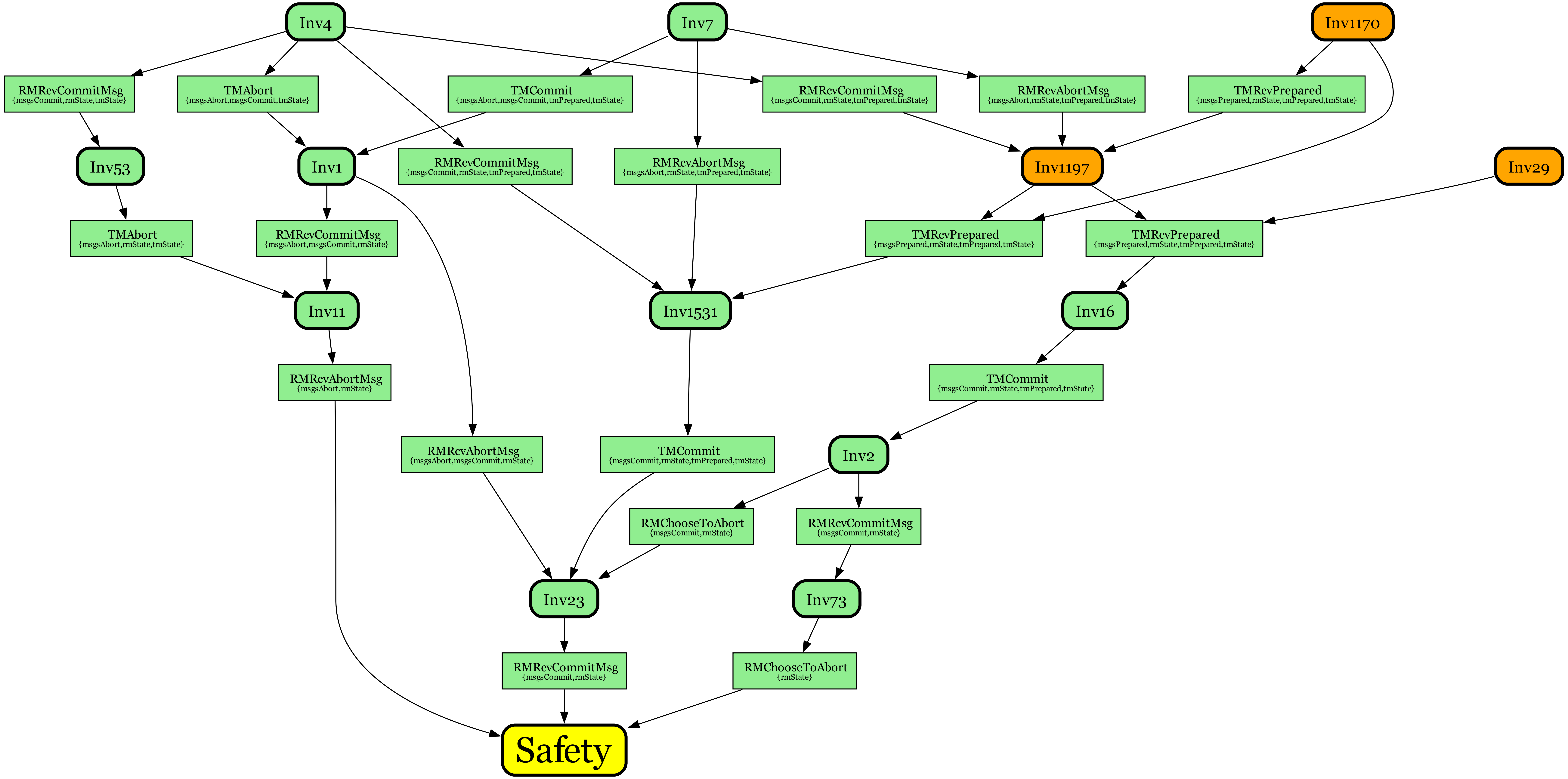
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
Inv16
Inv4
Inv7
Inv1170
Inv1197
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
Inv2
Inv53
Inv1
Inv1531
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
Inv73
Inv11
Inv23
RMChooseToAbort
{rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
Safety

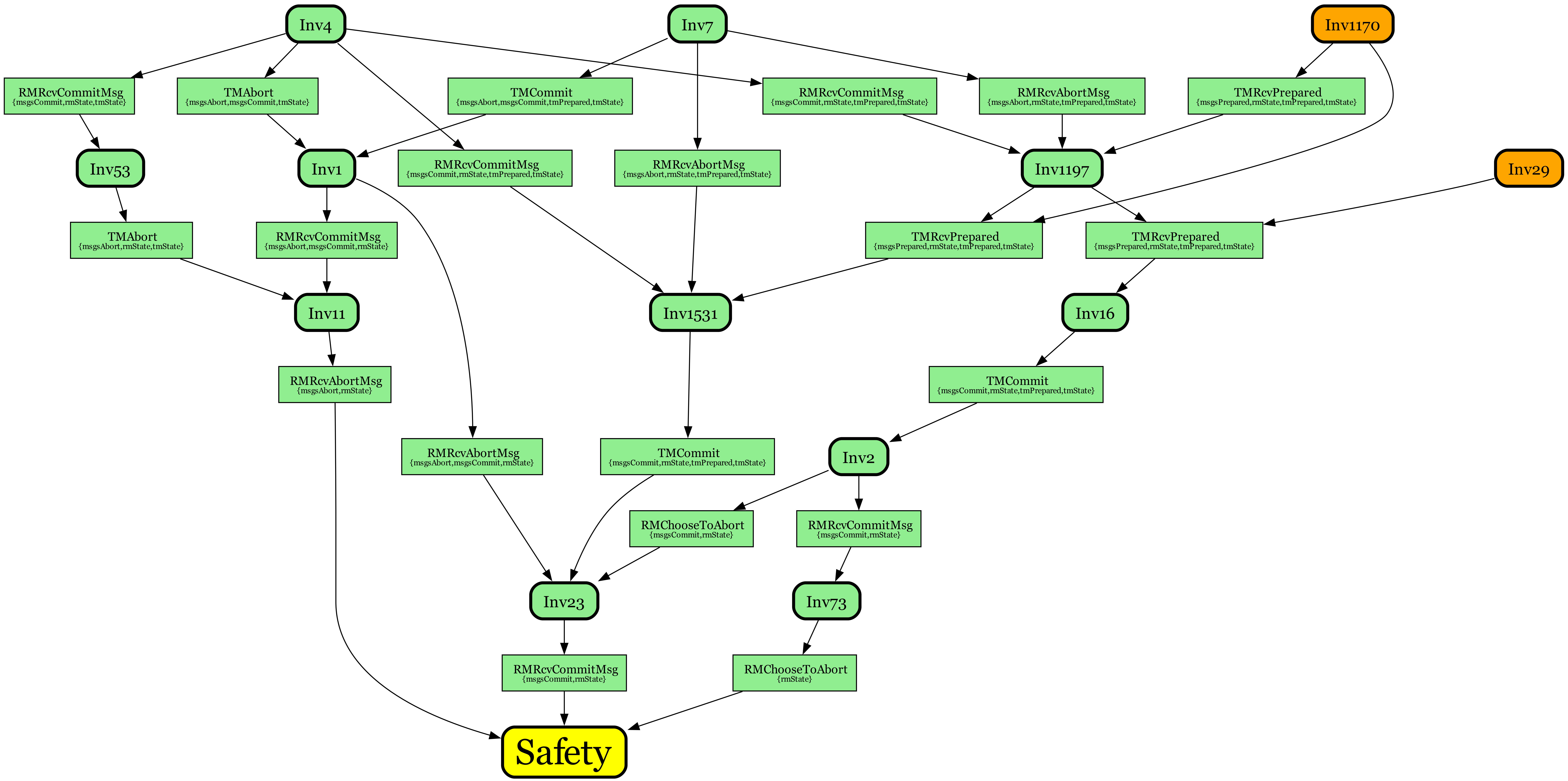

Inv29
Inv1197
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
Inv16
Inv1170
Inv7
Inv4
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
Inv2
Inv1531
Inv1
Inv53
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv73
Inv23
Inv11
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Safety


Inv7
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
TMRcvPrepared
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
RMRcvCommitMsg
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
Inv4
Inv29
Inv1197
Inv1170
TMAbort
{msgsAbort,msgsCommit,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
Inv1
Inv53
Inv16
Inv1531
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
Inv11
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
Inv2
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMChooseToAbort
{msgsCommit,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Inv23
Inv73
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{rmState}
Safety

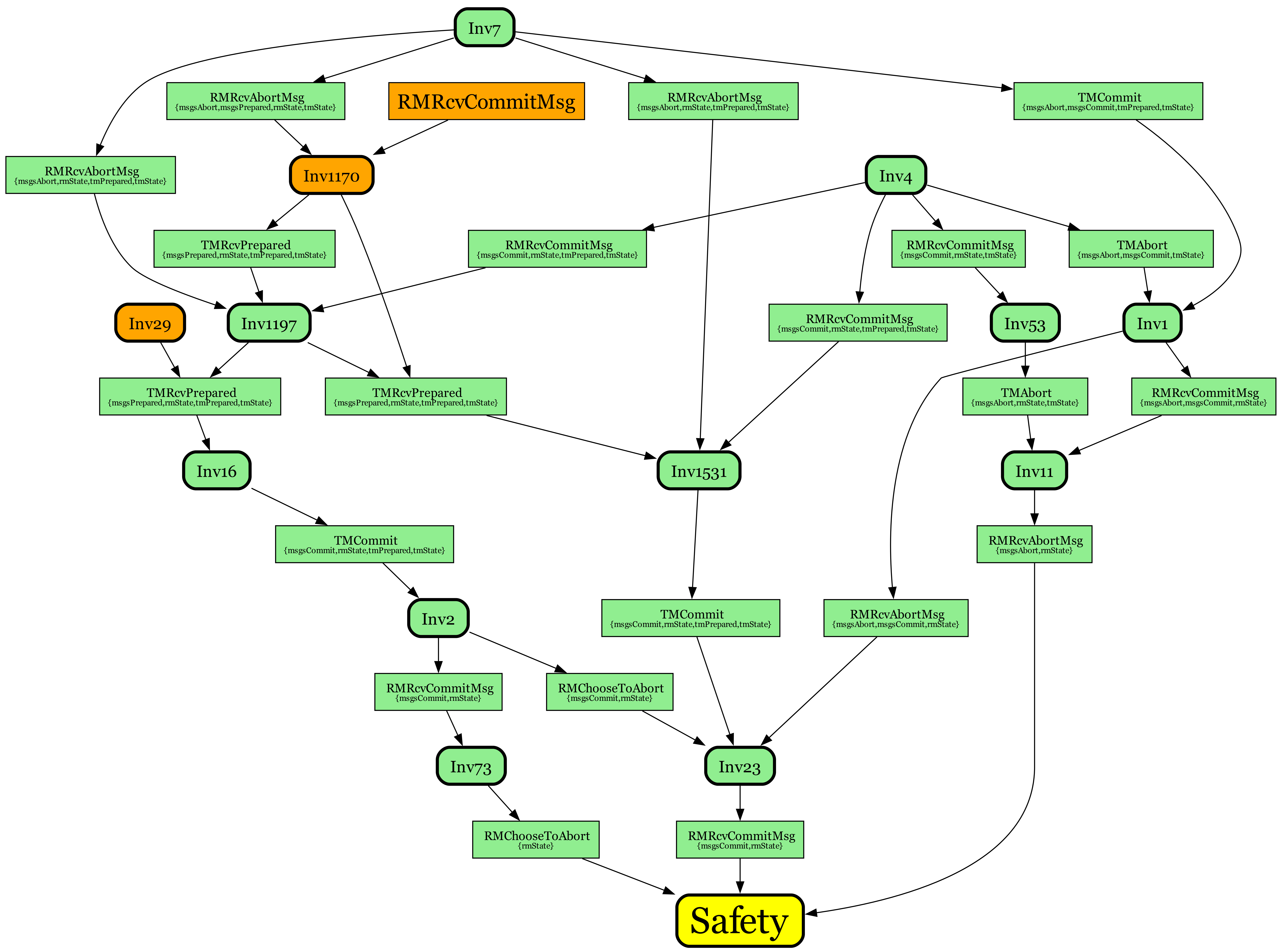


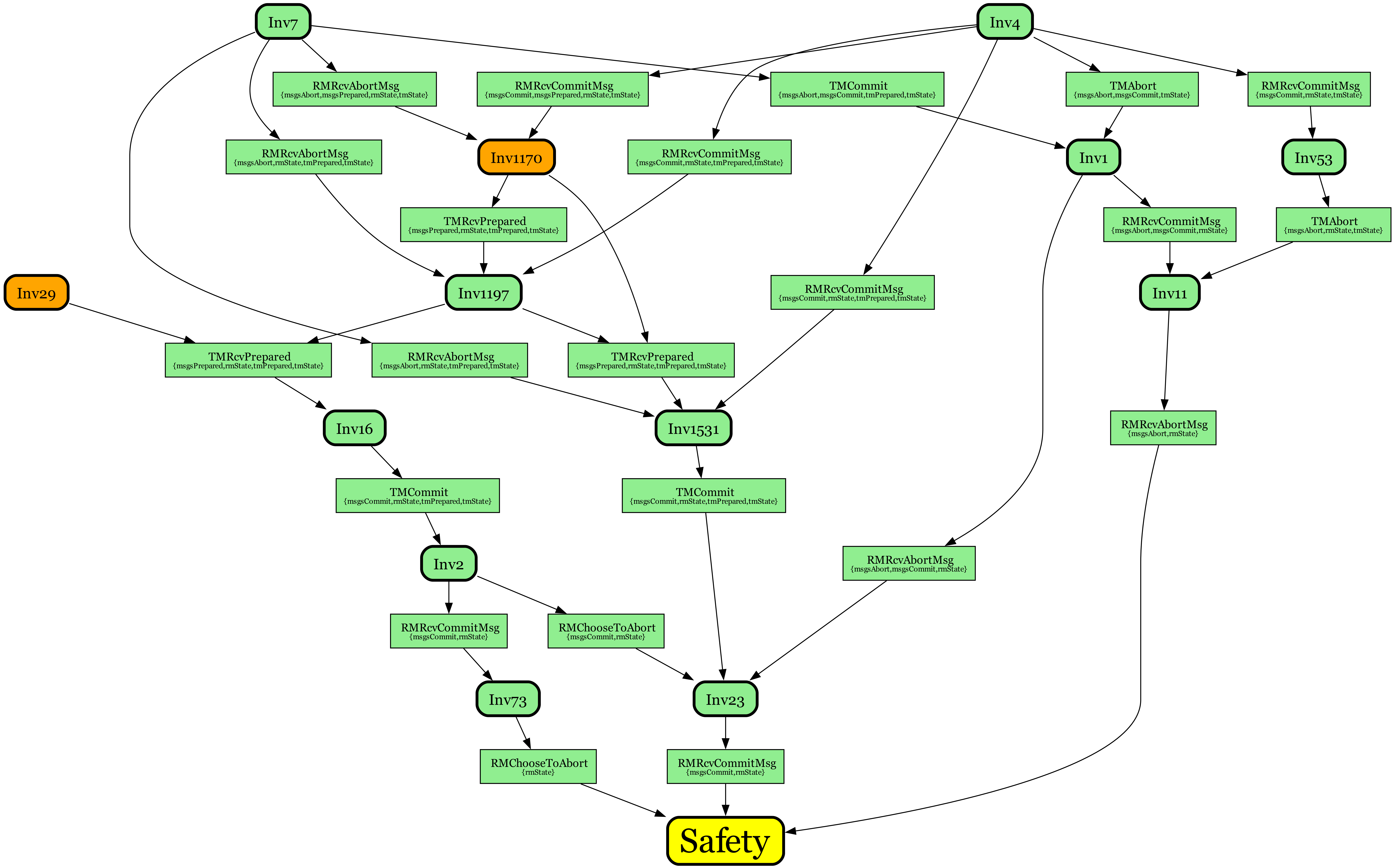
Inv7
Inv4
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
TMRcvPrepared
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
Inv29
Inv1197
Inv1170
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
Inv1
Inv53
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv16
Inv1531
Inv11
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Inv2
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
Inv73
Inv23
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
Safety


Inv4
Inv7
Inv1170
RMRcvCommitMsg {msgsCommit,rmState,tmState}
TMAbort {msgsAbort,msgsCommit,tmState}
TMCommit {msgsAbort,msgsCommit,tmPrepared,tmState}
RMRcvCommitMsg {msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg {msgsAbort,rmState,tmPrepared,tmState}
TMRcvPrepared {msgsPrepared,rmState,tmPrepared,tmState}
Inv53
Inv1
RMRcvCommitMsg {msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg {msgsAbort,rmState,tmPrepared,tmState}
Inv1197
Inv29
TMAbort {msgsAbort,rmState,tmState}
RMRcvCommitMsg {msgsAbort,msgsCommit,rmState}
TMRcvPrepared {msgsPrepared,rmState,tmPrepared,tmState}
TMRcvPrepared {msgsPrepared,rmState,tmPrepared,tmState}
Inv11
Inv1531
Inv16
RMRcvAbortMsg {msgsAbort,rmState}
TMCommit {msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg {msgsAbort,msgsCommit,rmState}
TMCommit {msgsCommit,rmState,tmPrepared,tmState}
Inv2
RMChooseToAbort {msgsCommit,rmState}
RMRcvCommitMsg {msgsCommit,rmState}
Inv23
Inv73
RMRcvCommitMsg {msgsCommit,rmState}
RMChooseToAbort {rmState}
Safety

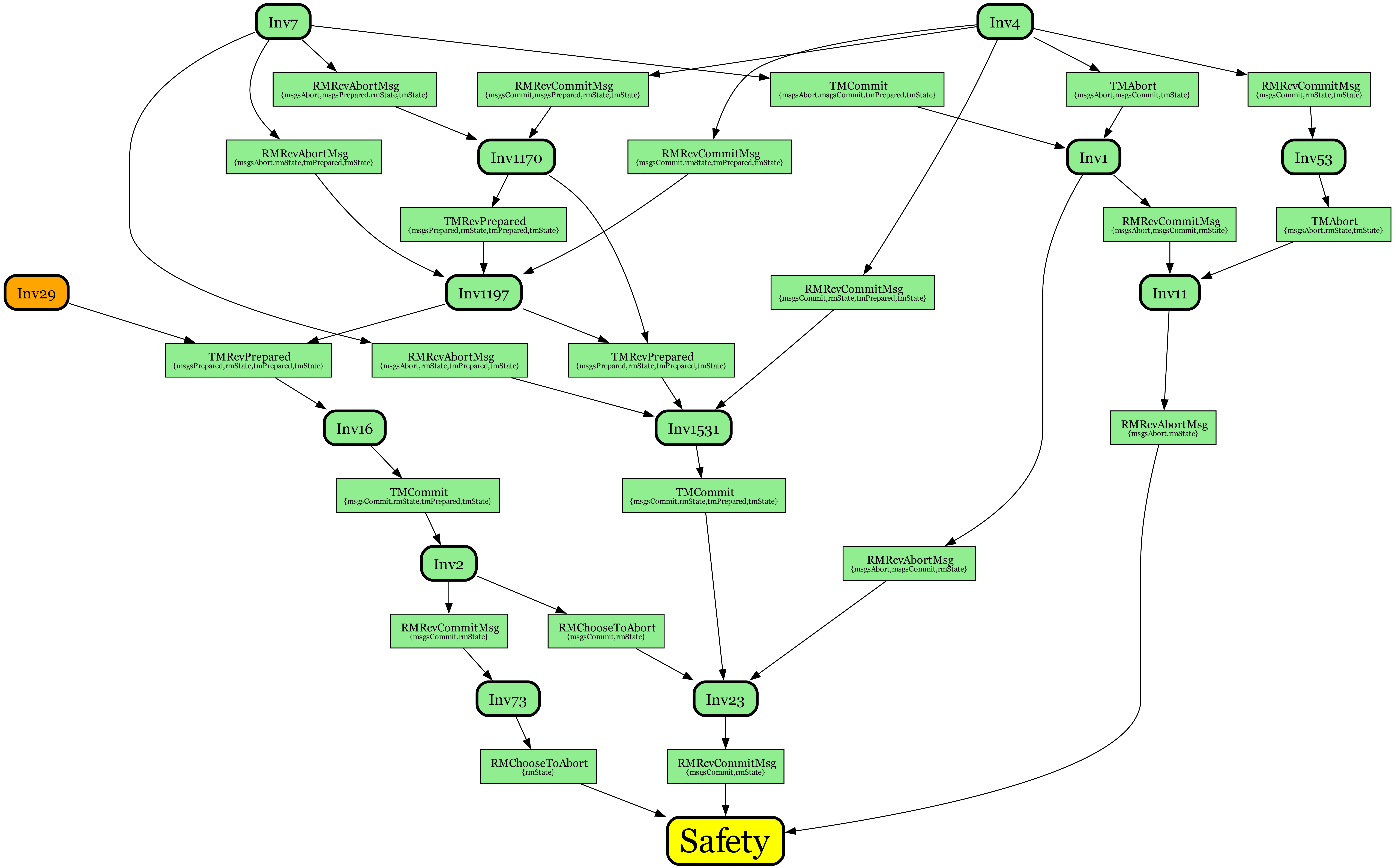


Inv4
Inv7
Inv1170
RMRcvCommitMsg {msgsCommit,rmState,tmState}
TMAbort {msgsAbort,msgsCommit,tmState}
TMCommit {msgsAbort,msgsCommit,tmPrepared,tmState}
RMRcvCommitMsg {msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg {msgsAbort,rmState,tmPrepared,tmState}
TMRcvPrepared {msgsPrepared,rmState,tmPrepared,tmState}
Inv53
Inv1
RMRcvCommitMsg {msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg {msgsAbort,rmState,tmPrepared,tmState}
Inv1197
Inv29
TMAbort {msgsAbort,rmState,tmState}
RMRcvCommitMsg {msgsAbort,msgsCommit,rmState}
TMRcvPrepared {msgsPrepared,rmState,tmPrepared,tmState}
TMRcvPrepared {msgsPrepared,rmState,tmPrepared,tmState}
Inv11
Inv1531
Inv16
RMRcvAbortMsg {msgsAbort,rmState}
TMCommit {msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg {msgsAbort,msgsCommit,rmState}
TMCommit {msgsCommit,rmState,tmPrepared,tmState}
Inv2
RMChooseToAbort {msgsCommit,rmState}
RMRcvCommitMsg {msgsCommit,rmState}
Inv23
Inv73
RMRcvCommitMsg {msgsCommit,rmState}
RMChooseToAbort {rmState}
Safety

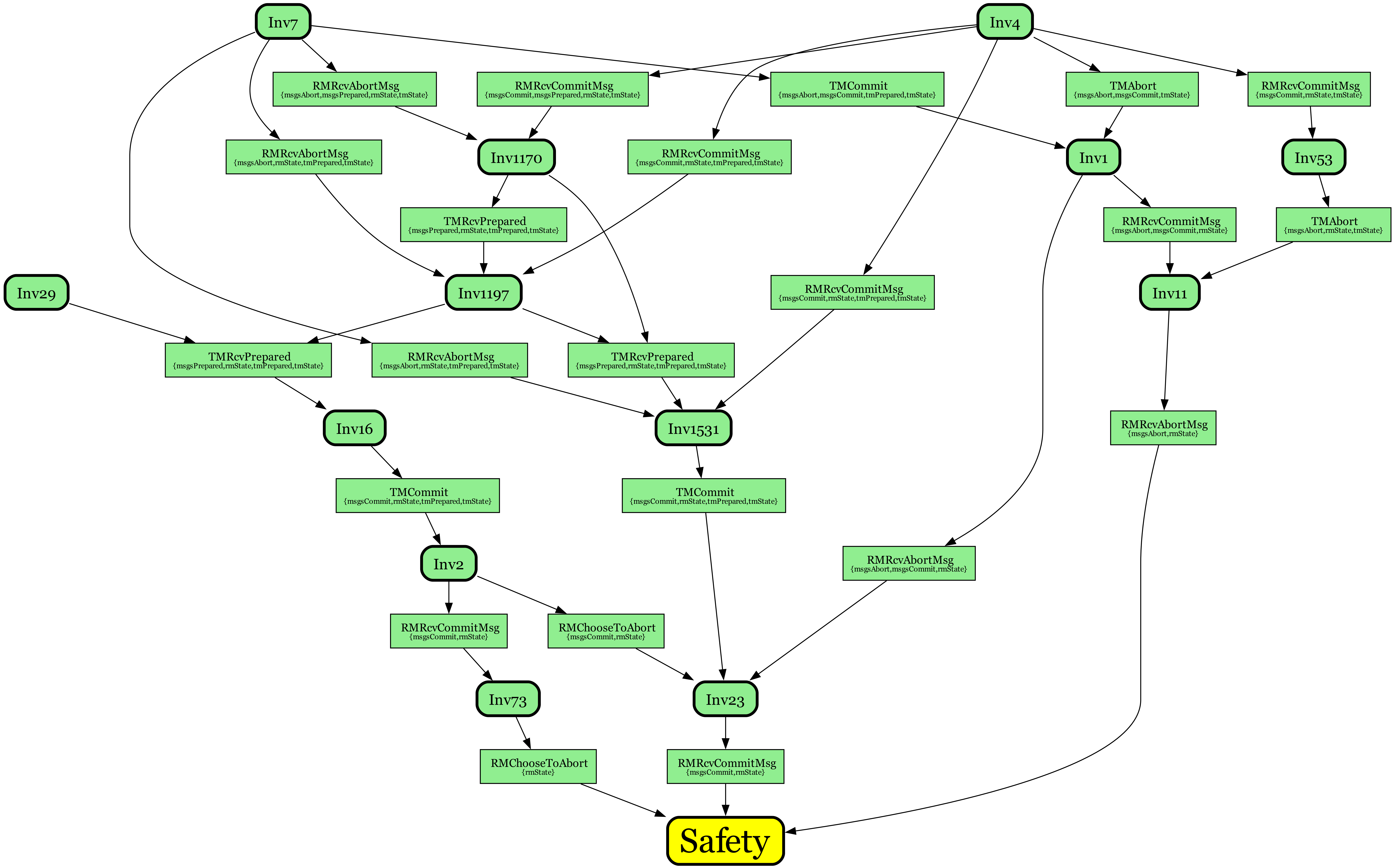

Inv7
RMRcvAbortMsg
{msgsAbort,msgsPrepared,rmState,tmState}
RMRcvCommitMsg
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
Inv1170
Inv4
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
Inv29
Inv1197
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
Inv53
Inv1
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
TMAbort
{msgsAbort,rmState,tmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
Inv16
Inv1531
Inv11
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,rmState}
Inv2
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
Inv73
Inv23
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
Safety



Inv7
Inv4
RMRcvAbortMsg {msgsAbort,msgsPrepared,rmState,tmState}
RMRcvCommitMsg {msgsCommit,msgsPrepared,rmState,tmState}
TMCommit {msgsAbort,msgsCommit,tmPrepared,tmState}
TMAbort {msgsAbort,msgsCommit,tmState}
RMRcvCommitMsg {msgsCommit,rmState,tmState}
RMRcvAbortMsg {msgsAbort,rmState,tmPrepared,tmState}
Inv1170
RMRcvCommitMsg {msgsCommit,rmState,tmPrepared,tmState}
Inv1
Inv53
TMRcvPrepared {msgsPrepared,rmState,tmPrepared,tmState}
RMRcvCommitMsg {msgsAbort,msgsCommit,rmState}
TMAbort {msgsAbort,rmState,tmState}
Inv29
Inv1197
RMRcvCommitMsg {msgsCommit,rmState,tmPrepared,tmState}
Inv11
TMRcvPrepared {msgsPrepared,rmState,tmPrepared,tmState}
RMRcvAbortMsg {msgsAbort,rmState,tmPrepared,tmState}
TMRcvPrepared {msgsPrepared,rmState,tmPrepared,tmState}
Inv16
Inv1531
RMRcvAbortMsg {msgsAbort,rmState}
TMCommit {msgsCommit,rmState,tmPrepared,tmState}
TMCommit {msgsCommit,rmState,tmPrepared,tmState}
Inv2
RMRcvAbortMsg {msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg {msgsCommit,rmState}
RMChooseToAbort {msgsCommit,rmState}
Inv73
Inv23
RMChooseToAbort {rmState}
RMRcvCommitMsg {msgsCommit,rmState}
Safety



Inv7
Inv4
RMRcvAbortMsg {msgsAbort,msgsPrepared,rmState,tmState}
RMRcvCommitMsg {msgsCommit,msgsPrepared,rmState,tmState}
TMCommit {msgsAbort,msgsCommit,tmPrepared,tmState}
TMAbort {msgsAbort,msgsCommit,tmState}
RMRcvCommitMsg {msgsCommit,rmState,tmState}
RMRcvAbortMsg {msgsAbort,rmState,tmPrepared,tmState}
Inv1170
RMRcvCommitMsg {msgsCommit,rmState,tmPrepared,tmState}
Inv1
Inv53
TMRcvPrepared {msgsPrepared,rmState,tmPrepared,tmState}
RMRcvCommitMsg {msgsAbort,msgsCommit,rmState}
TMAbort {msgsAbort,rmState,tmState}
Inv29
Inv1197
RMRcvCommitMsg {msgsCommit,rmState,tmPrepared,tmState}
Inv11
TMRcvPrepared {msgsPrepared,rmState,tmPrepared,tmState}
RMRcvAbortMsg {msgsAbort,rmState,tmPrepared,tmState}
TMRcvPrepared {msgsPrepared,rmState,tmPrepared,tmState}
Inv16
Inv1531
RMRcvAbortMsg {msgsAbort,rmState}
TMCommit {msgsCommit,rmState,tmPrepared,tmState}
TMCommit {msgsCommit,rmState,tmPrepared,tmState}
Inv2
RMRcvAbortMsg {msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg {msgsCommit,rmState}
RMChooseToAbort {msgsCommit,rmState}
Inv73
Inv23
RMChooseToAbort {rmState}
RMRcvCommitMsg {msgsCommit,rmState}
Safety


Inv7
Inv4
RMRcvAbortMsg
{msgsAbort,msgsPrepared,rmState,tmState}
RMRcvCommitMsg
{msgsCommit,msgsPrepared,rmState,tmState}
TMCommit
{msgsAbort,msgsCommit,tmPrepared,tmState}
TMAbort
{msgsAbort,msgsCommit,tmState}
RMRcvCommitMsg
{msgsCommit,rmState,tmState}
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
Inv1170
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
Inv1
Inv53
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
RMRcvCommitMsg
{msgsAbort,msgsCommit,rmState}
TMAbort
{msgsAbort,rmState,tmState}
Inv29
Inv1197
RMRcvCommitMsg
{msgsCommit,rmState,tmPrepared,tmState}
Inv11
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
RMRcvAbortMsg
{msgsAbort,rmState,tmPrepared,tmState}
TMRcvPrepared
{msgsPrepared,rmState,tmPrepared,tmState}
Inv16
Inv1531
RMRcvAbortMsg
{msgsAbort,rmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
TMCommit
{msgsCommit,rmState,tmPrepared,tmState}
Inv2
RMRcvAbortMsg
{msgsAbort,msgsCommit,rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
RMChooseToAbort
{msgsCommit,rmState}
Inv73
Inv23
RMChooseToAbort
{rmState}
RMRcvCommitMsg
{msgsCommit,rmState}
Safety

Inductive Proof Graph Synthesis

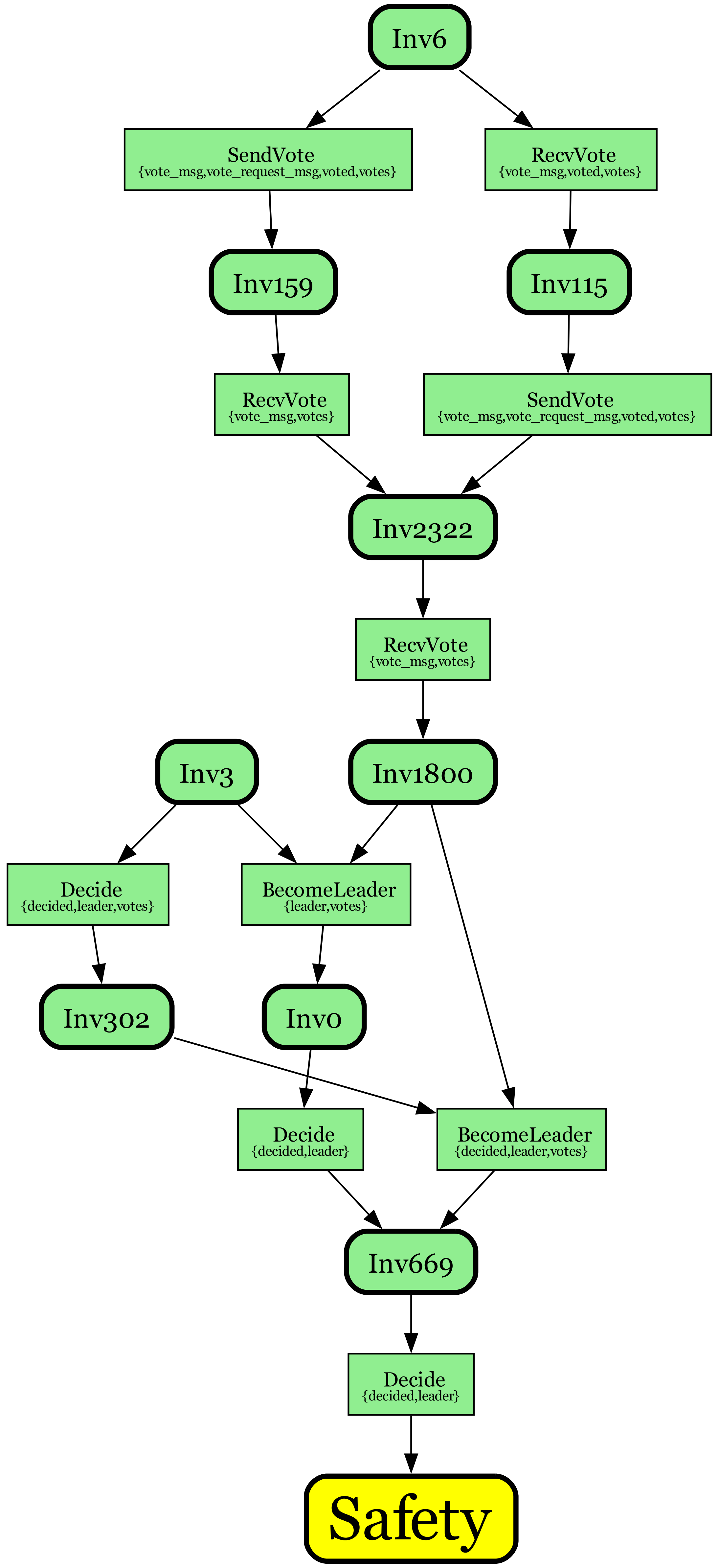
Implemented technique for synthesizing proof graphs into a tool, ***scimitar***, which accepts TLA+ protocol specifications.

Uses the TLC model checker and TLA+ proof system (TLAPS) for internal verification routines.

Syntax-guided technique for synthesizing lemmas. Guided by counterexamples to induction (CTIs) at each local inductive proof obligation.

github.com/will62794/scimitar

SimpleConsensus



Simplified leader-based consensus algorithm.

10 lemma nodes.

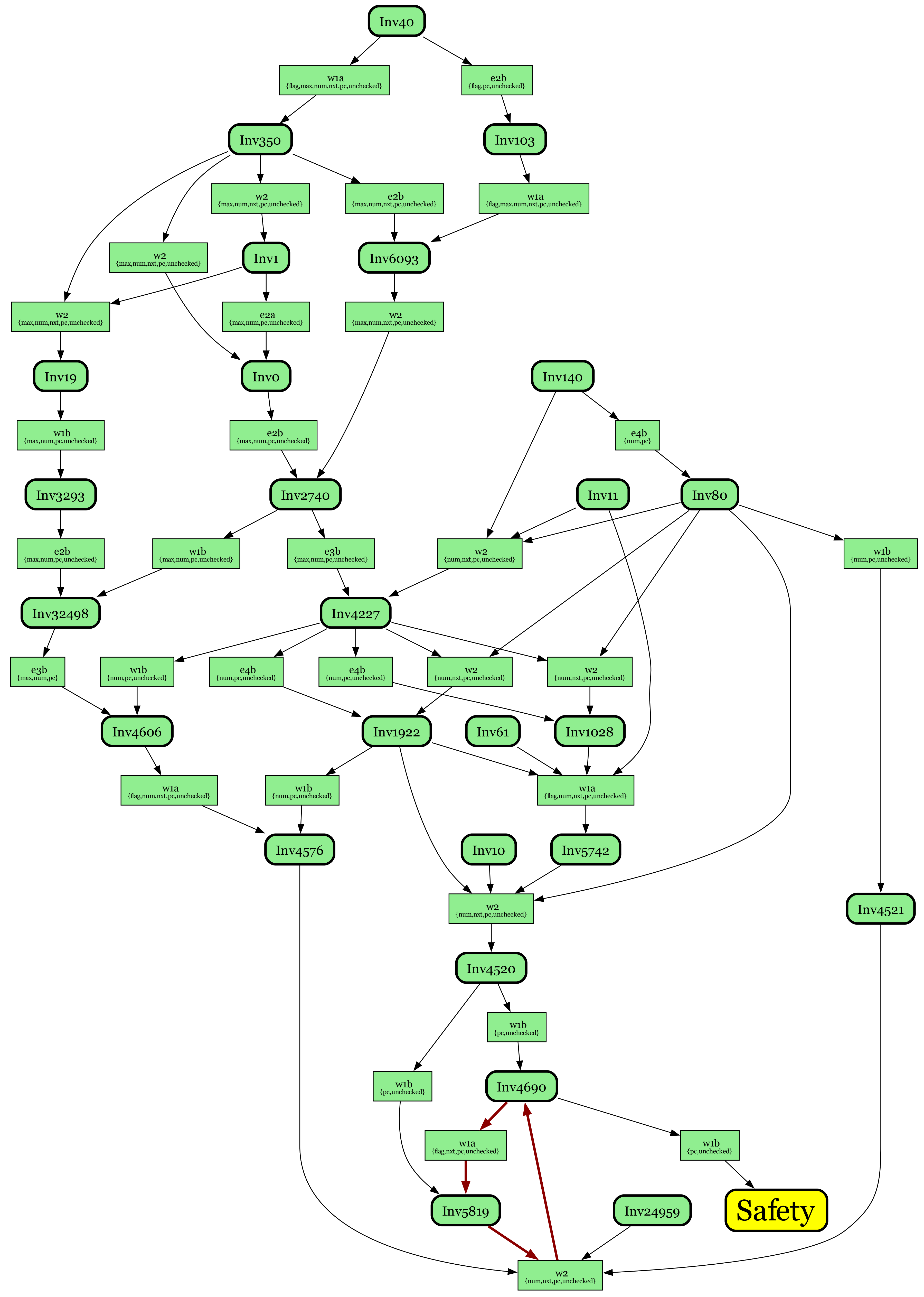
6 state variables, 5 actions.

Synthesized in ~5 minutes.

VARIABLE
vote_request_msg,
voted,
vote_msg,
votes,
leader,
decided

$$Safety \triangleq \forall n_1, n_2 \in Node, v_1, v_2 \in Value : (v_1 \in decided[n_1] \land v_2 \in decided[n_2]) \Rightarrow (v_1 = v_2)$$

Bakery



Lamport's concurrent Bakery algorithm for mutual exclusion.

27 lemma nodes

6 state variables, 14 actions.

Synthesized in ~ 4.5 hours

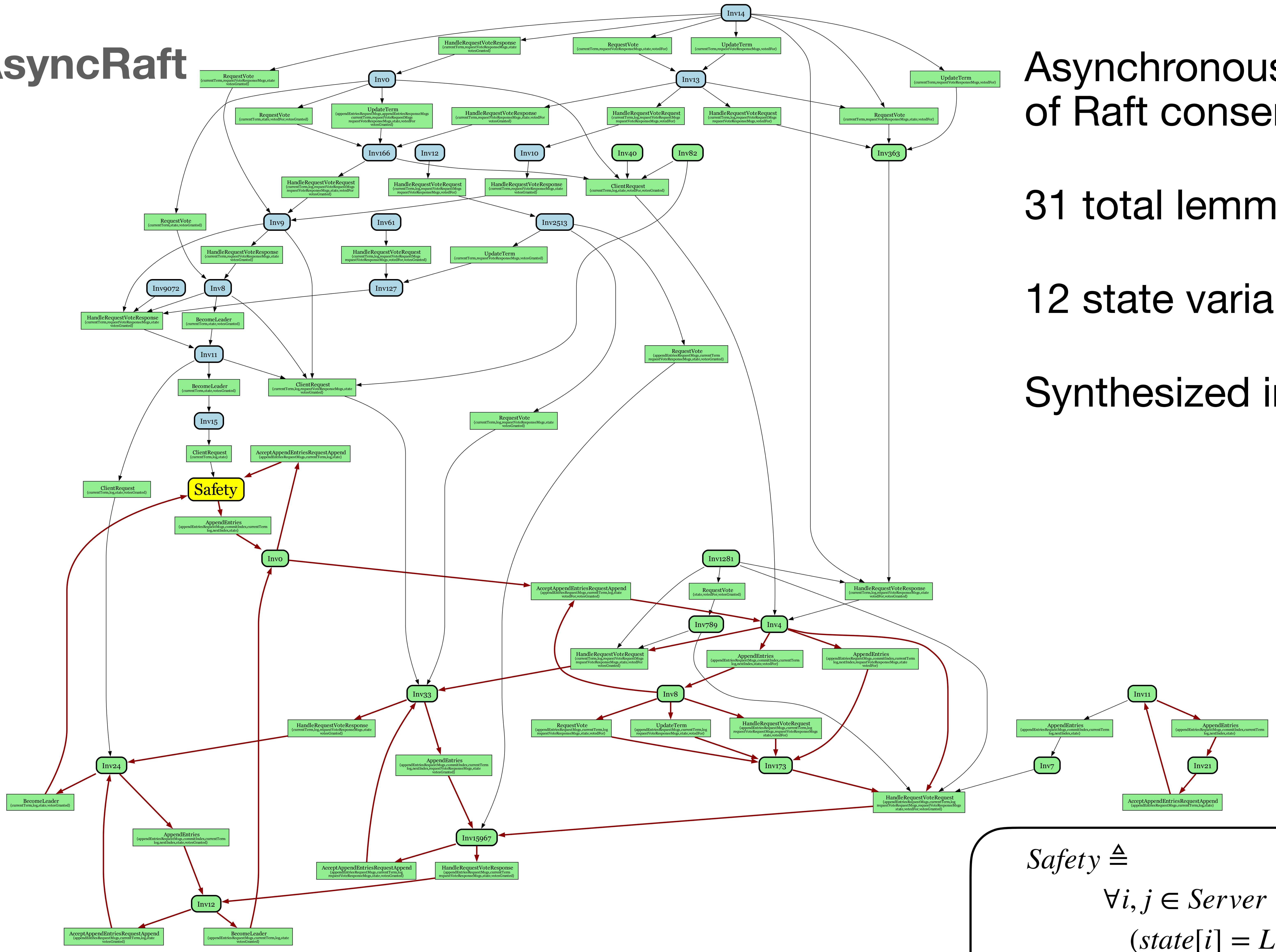
VARIABLES
num,
flag,
pc,
unchecked,
max,
nxt

Mutual exclusion property.

$$Safety \triangleq$$

$$\forall i, j \in Procs : (i \neq j) \Rightarrow \neg(pc[i] = cs \land pc[j] = cs)$$

AsyncRaft



Asynchronous specification of Raft consensus.

31 total lemma nodes.

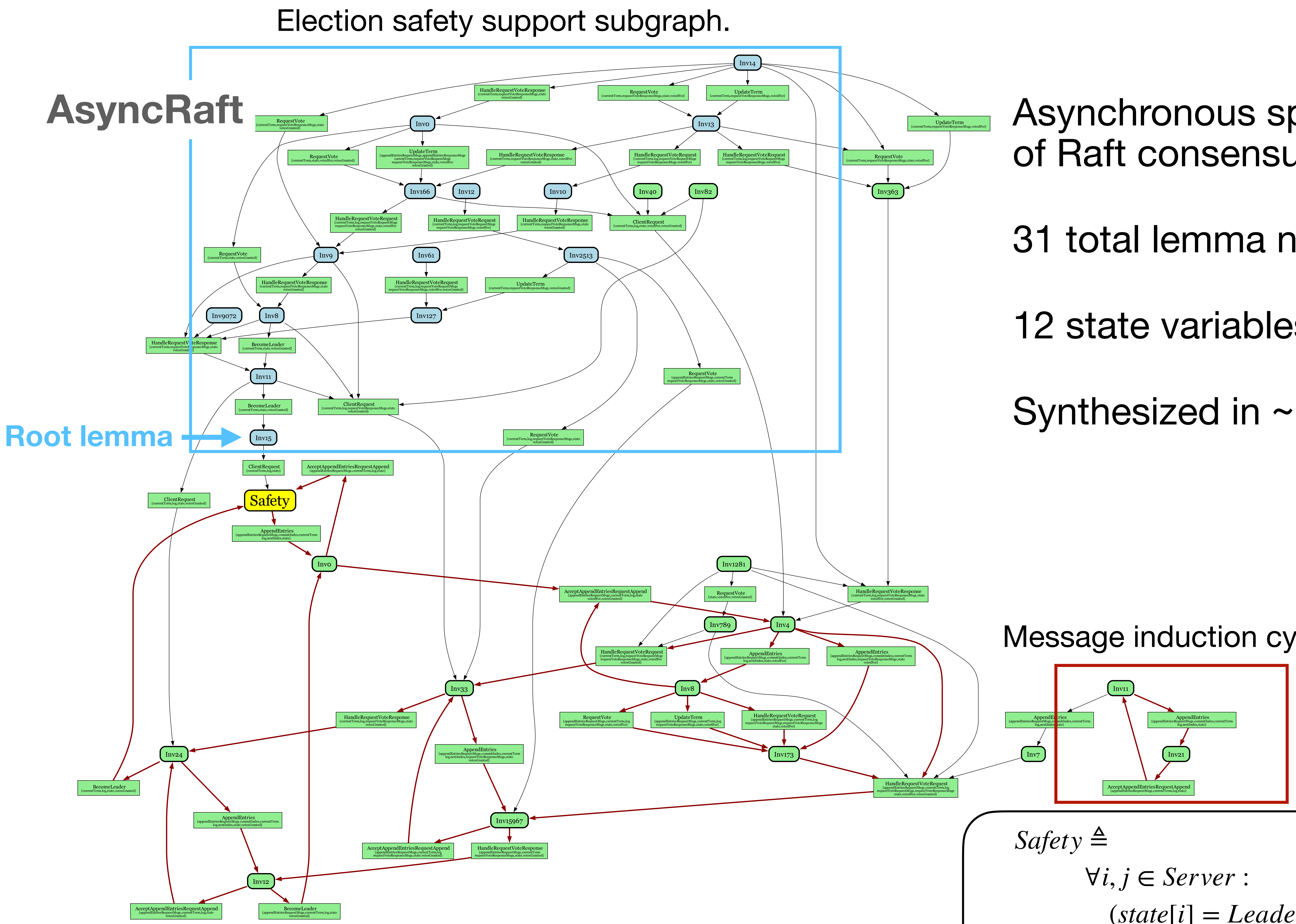
12 state variables, 9 actions.

Synthesized in ~14 hours.

VARIABLES
requestVoteRequestMsgs
requestVoteResponseMsgs
appendEntriesRequestMsgs
appendEntriesResponseMsgs
currentTerm
state
votedFor
log
commitIndex
votesGranted
nextIndex
matchIndex

$$
\begin{aligned}
&Safety \triangleq \\
&\quad \forall i, j \in Server : \\
&\qquad (state[i] = Leader) \Rightarrow \\
&\qquad \nexists k \in DOMAIN\ log[j] : \\
&\qquad\quad \wedge (log[j][k] = currentTerm[i]) \\
&\qquad\quad \wedge \nexists ind \in DOMAIN\ log[i] : (ind = k \wedge log[i][k] = log[j][k]))
\end{aligned}
$$

AsyncRaft



Asynchronous specification of Raft consensus.

31 total lemma nodes.

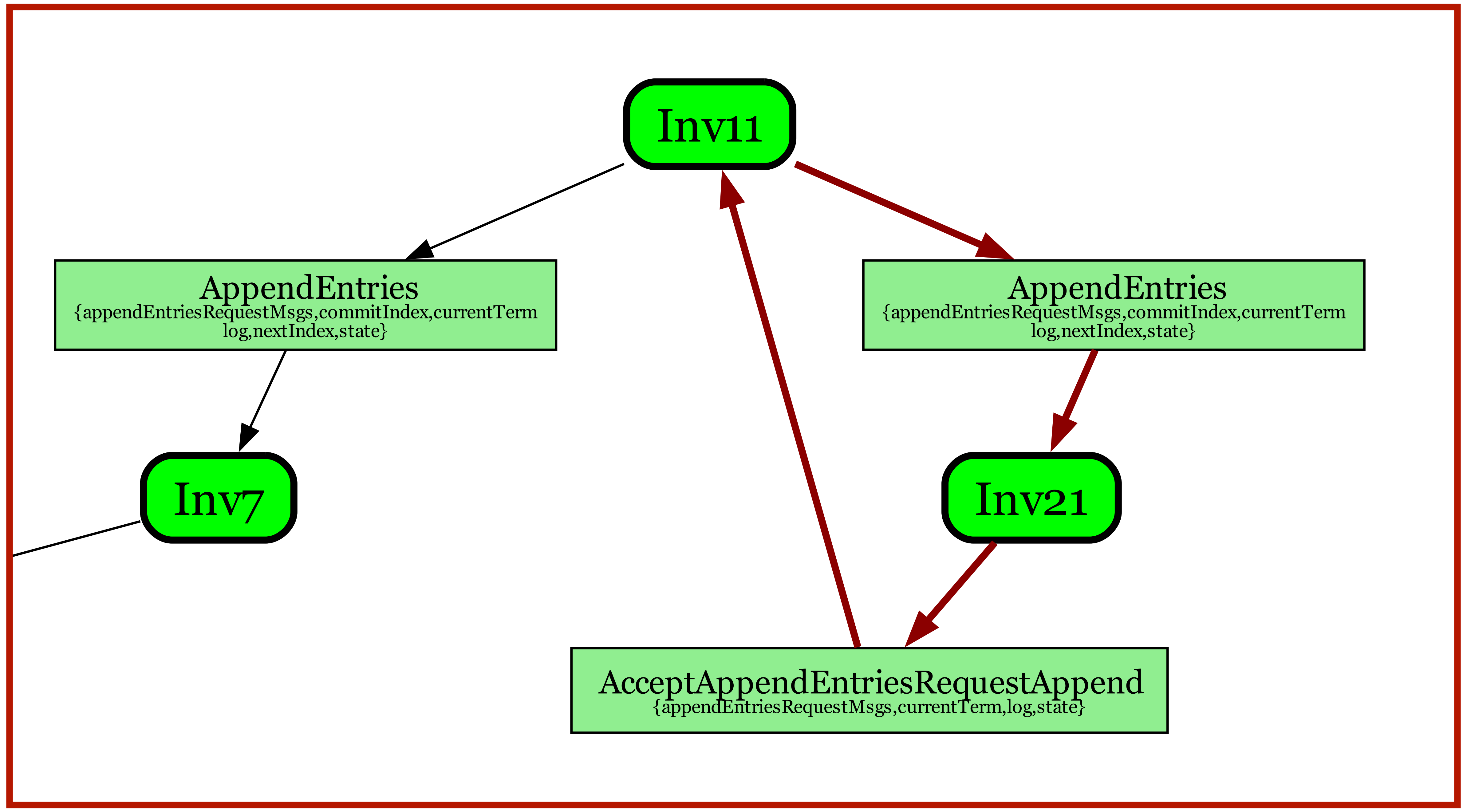
12 state variables, 9 actions.

Synthesized in ~14 hours.

VARIABLES
- *requestVoteRequestMsgs*
- *requestVoteResponseMsgs*
- *appendEntriesRequestMsgs*
- *appendEntriesResponseMsgs*
- *currentTerm*
- *state*
- *votedFor*
- *log*
- *commitIndex*
- *votesGranted*
- *nextIndex*
- *matchIndex*

$$
\begin{aligned}
&Safety \triangleq \\
&\quad \forall i, j \in Server : \\
&\quad\quad (state[i] = Leader) \Rightarrow \\
&\quad\quad \nexists k \in \text{DOMAIN } log[j] : \\
&\quad\quad\quad \land (log[j][k] = currentTerm[i]) \\
&\quad\quad\quad \land \nexists ind \in \text{DOMAIN } log[i] : (ind = k \land log[i][k] = log[j][k]))
\end{aligned}
$$

AsyncRaft - *Induction Cycle Motifs*



Induction cycles manifest in message passing distributed protocols.

Invariants that must hold in *local* state and also *network* state.

$$Inv11 \triangleq \forall i \in Server : \forall k \in \text{DOMAIN } log[i] : log[i][k] \leq currentTerm[i]$$

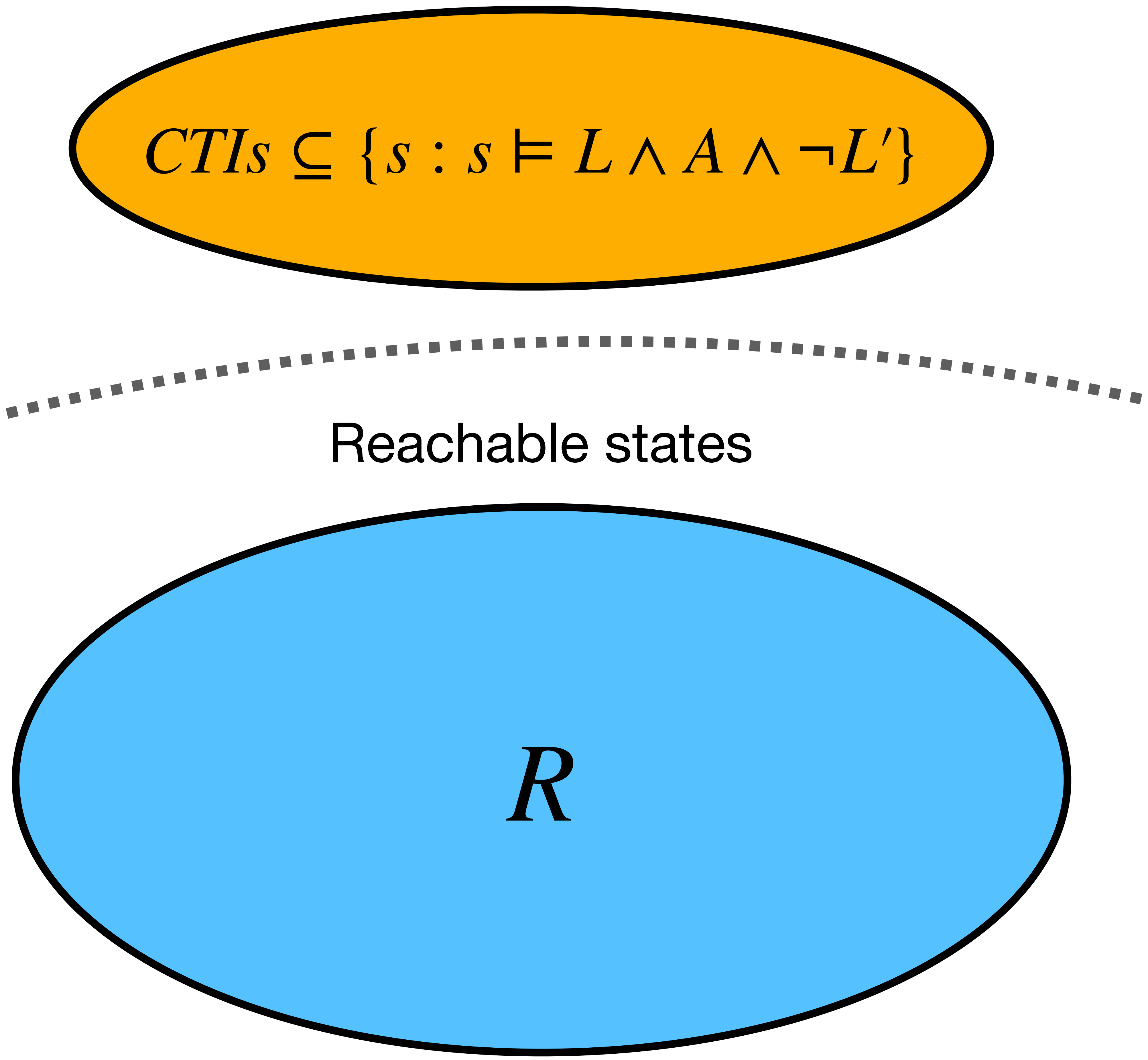
Local invariant

$$Inv21 \triangleq \forall m \in appendEntriesRequestMsgs : (m.mentries \neq \langle\rangle \Rightarrow m.mentries[1] \leq m.mterm)$$

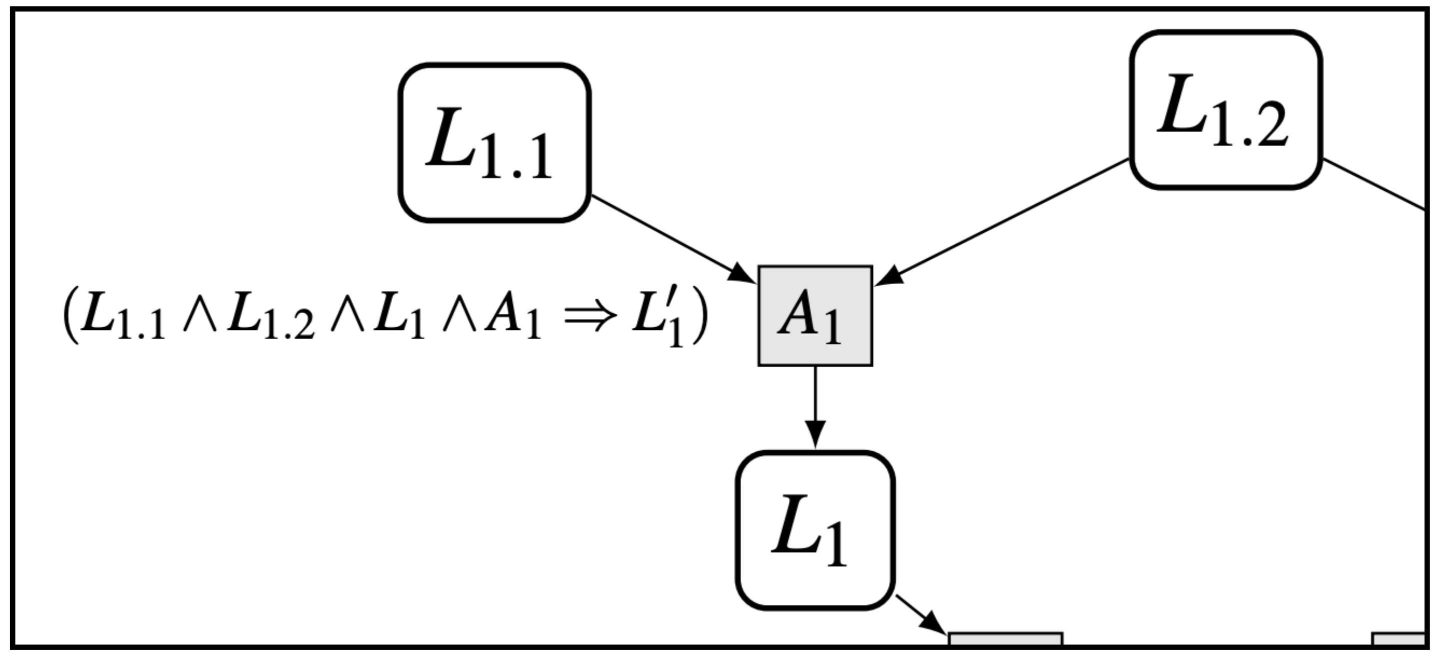
Network invariant

Slicing Optimizations

Lemma synthesis routine at node (L, A) can be viewed as learning a predicate to separate reachable states R and local CTIs.

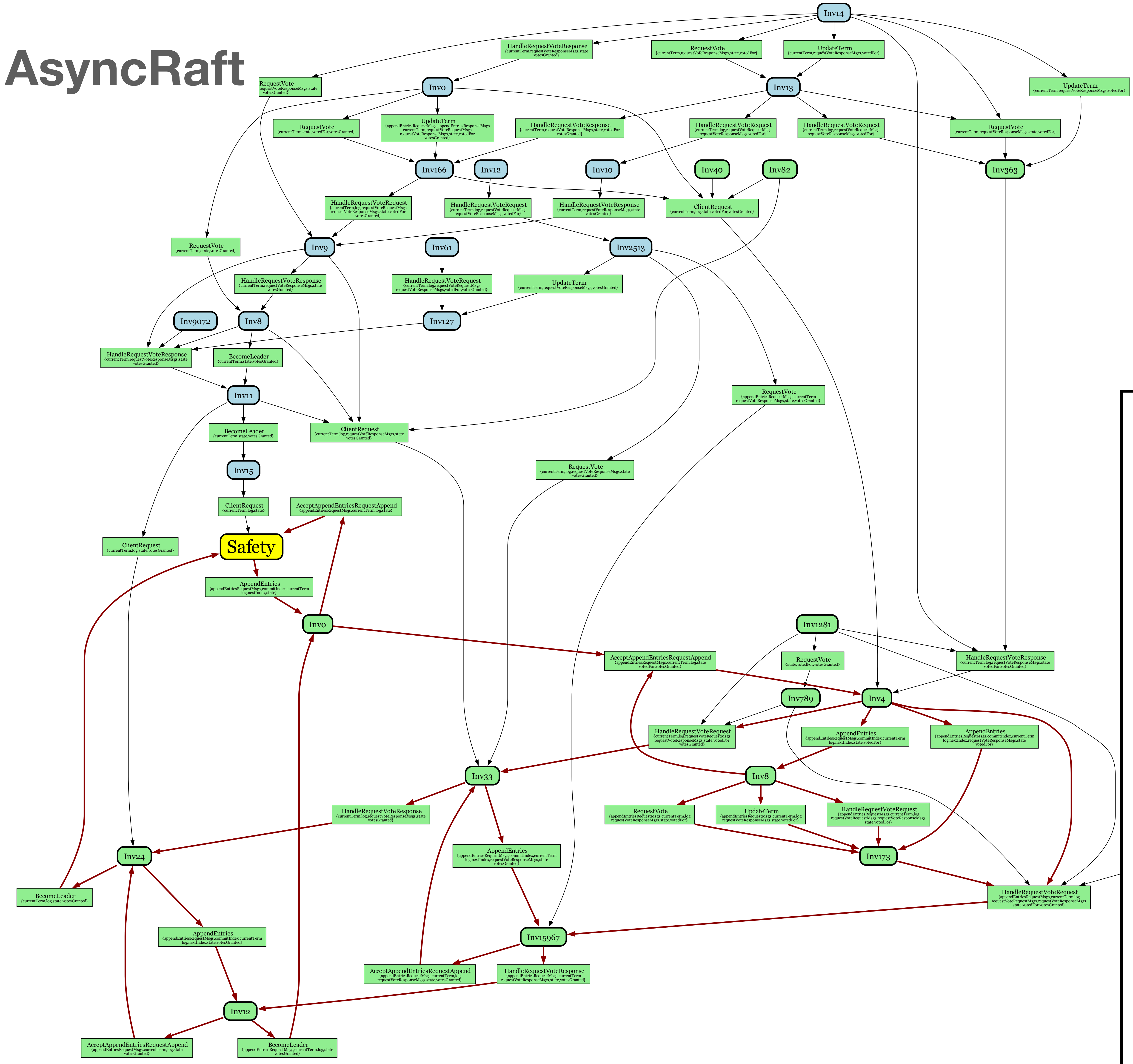


Support lemmas (separator)



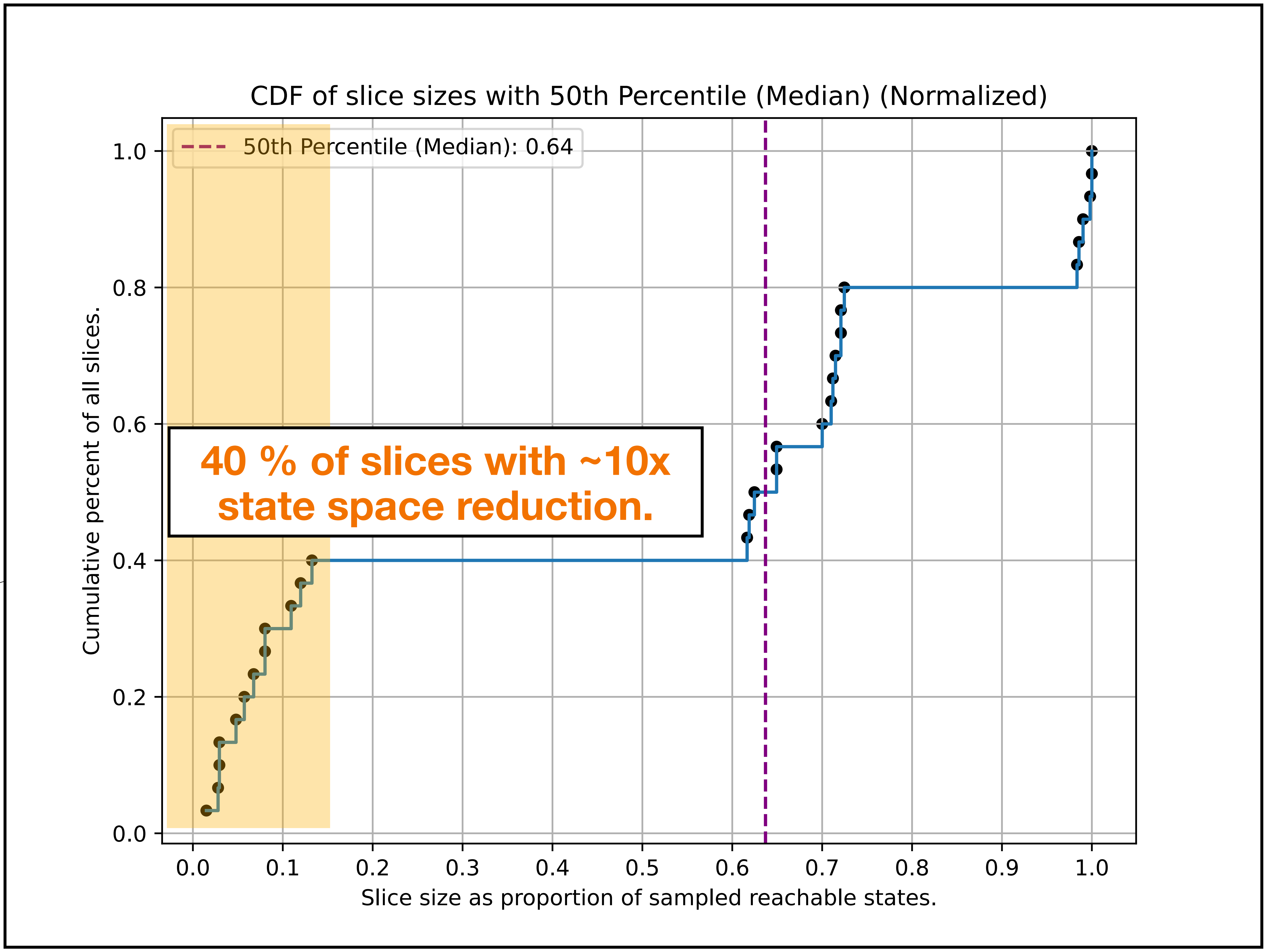
Cost typically dominated by $|R|$ but can be cached upfront for many local synthesis tasks.

AsyncRaft

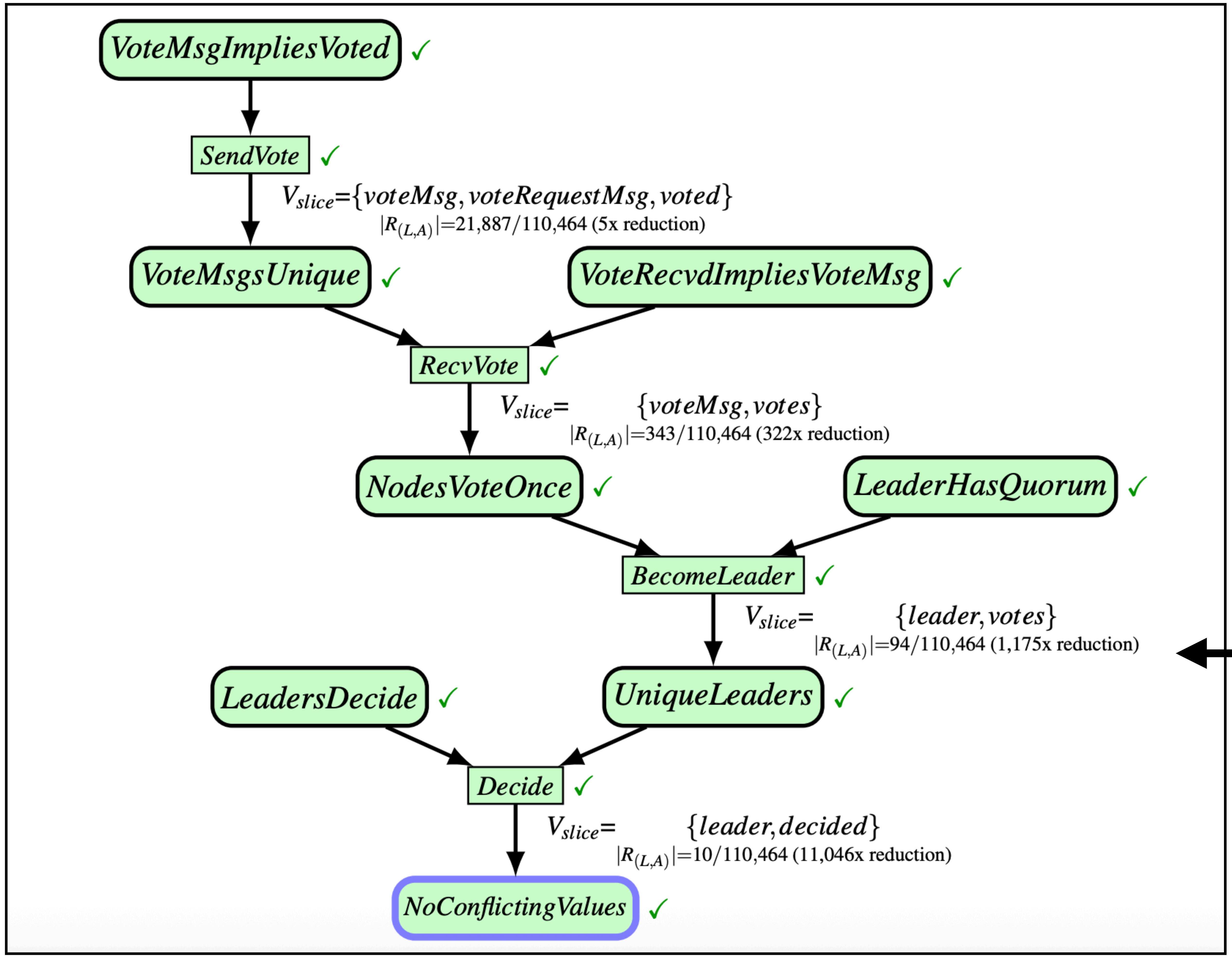


Grammar and **state** slicing optimizations both enabled by graph structure.

Project down to variables in slice, accelerate syntax-guided synthesis.



SimpleConsensus



For finite protocol model with 3 servers, **110,464 reachable states R.**

Some variable slices with over 1000x reduction in R. (e.g. slice projection to 2/6 total state variables).

Conclusions & Future Work

Inductive proof decomposition: a compositional, interpretable approach to inductive invariant development.

More exploration of empirical structure of these graphs.

Other compositional features for synthesis acceleration

- Quantifier structures
- Local grammar tuning, etc.

Tool: https://github.com/will62794/scimitar

Paper draft: https://will62794.github.io