TACKLING STATE SPACE EXPLOSION IN TLA^+ VISUALIZATIONS

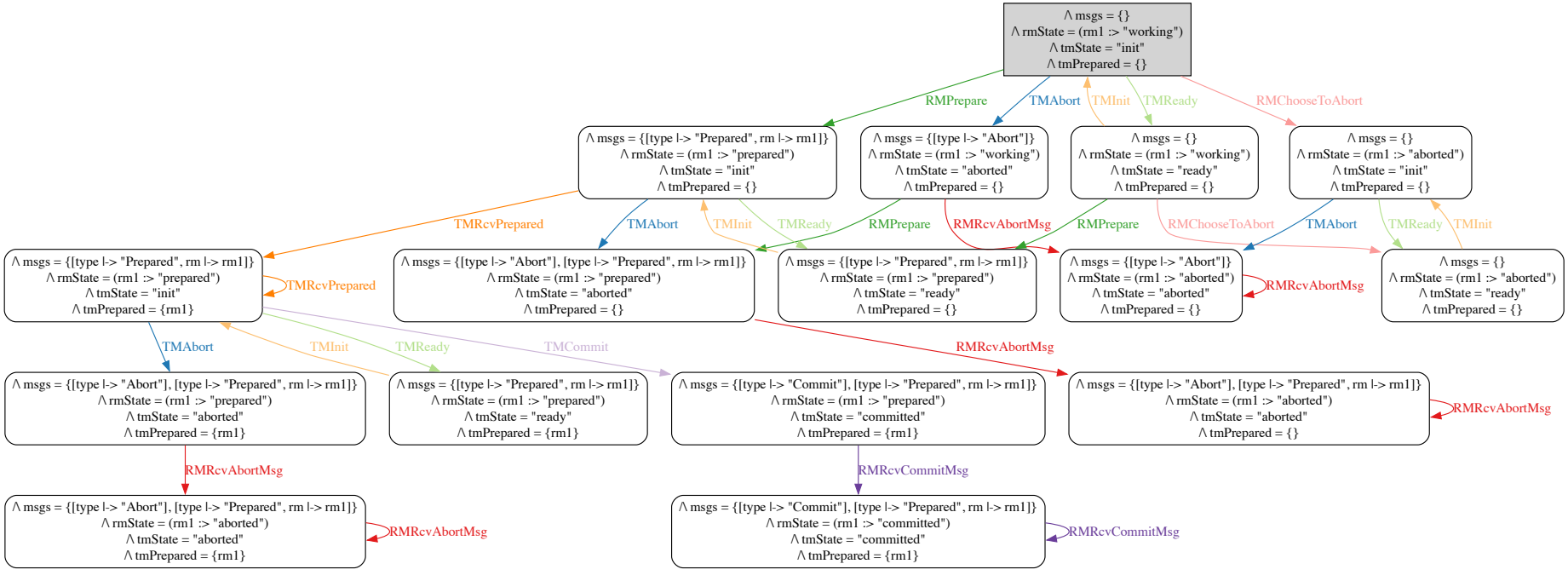
Daniel Stachnik
hi@dast.xyz

Tom Beckmann
tom.beckmann@hpi.de

Robert Hirschfeld
robert.hirschfeld@hpi.de

Hasso Plattner Institute
University of Potsdam

STATE DIAGRAMS ARE NICE...



State space of the Two-Phase-Commit Protocol with one resource manager (30 distinct states).

STATE DIAGRAMS ARE NICE...

...BUT THEY SUFFER FROM STATE EXPLOSION

State space of the Two-Phase-Commit Protocol
with three resource managers (1370 distinct states).

What can we do about it?

TLA^+'s niche are distributed systems.

TLA^+'s niche are distributed systems.

Distributed System = Actors + Messages

TLA^+'s niche are distributed systems.

Distributed System = Actors + Messages

Who are the actors in a TLA^+ spec?

$TMAbort \triangleq$

The TM spontaneously aborts the transaction.

$\land\ tmState = \text{“init”}$
$\land\ tmState' = \text{“aborted”}$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{“Abort”}]\}$
$\land$ UNCHANGED $\langle rmState,\ tmPrepared \rangle$

$RMPrepare(rm) \triangleq$

Resource manager $\$rm\$$ prepares.

$\land\ rmState[rm] = \text{“working”}$
$\land\ rmState' = [rmState \text{ EXCEPT } ![rm] = \text{“prepared”}]$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{“Prepared”},\ rm \mapsto rm]\}$
$\land$ UNCHANGED $\langle tmState,\ tmPrepared \rangle$

Who are the actors in a TLA^+ spec?

$TMAbort \triangleq$

The TM spontaneously aborts the transaction.

$\land\ tmState = \text{"init"}$
$\land\ tmState' = \text{"aborted"}$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{"Abort"}]\}$
$\land$ UNCHANGED $\langle rmState,\ tmPrepared\rangle$

$RMPrepare(rm) \triangleq$

Resource manager rm prepares.

$\land\ rmState[rm] = \text{"working"}$
$\land\ rmState' = [rmState \text{ EXCEPT } ![rm] = \text{"prepared"}]$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{"Prepared"},\ rm \mapsto rm]\}$
$\land$ UNCHANGED $\langle tmState,\ tmPrepared\rangle$

Mathematical notation doesn't give away the actors

Who are the actors in a TLA^+ spec?

$TMAbort \triangleq$

The TM spontaneously aborts the transaction.

$\land\ tmState = \text{"init"}$
$\land\ tmState' = \text{"aborted"}$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{"Abort"}]\}$
$\land$ UNCHANGED $\langle rmState,\ tmPrepared \rangle$

$RMPrepare(rm) \triangleq$

Resource manager $\$rm\$$ prepares.

$\land\ rmState[rm] = \text{"working"}$
$\land\ rmState' = [rmState \text{ EXCEPT } ![rm] = \text{"prepared"}]$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{"Prepared"},\ rm \mapsto rm]\}$
$\land$ UNCHANGED $\langle tmState,\ tmPrepared \rangle$

But conceptually we model them

Who are the actors in a TLA^+ spec?

$TMAbort \triangleq$

The TM spontaneously aborts the transaction.

$\land\ tmState = \text{"init"}$
$\land\ tmState' = \text{"aborted"}$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{"Abort"}]\}$
$\land$ UNCHANGED $\langle rmState,\ tmPrepared \rangle$

$RMPrepare(rm) \triangleq$

Resource manager rm prepares.

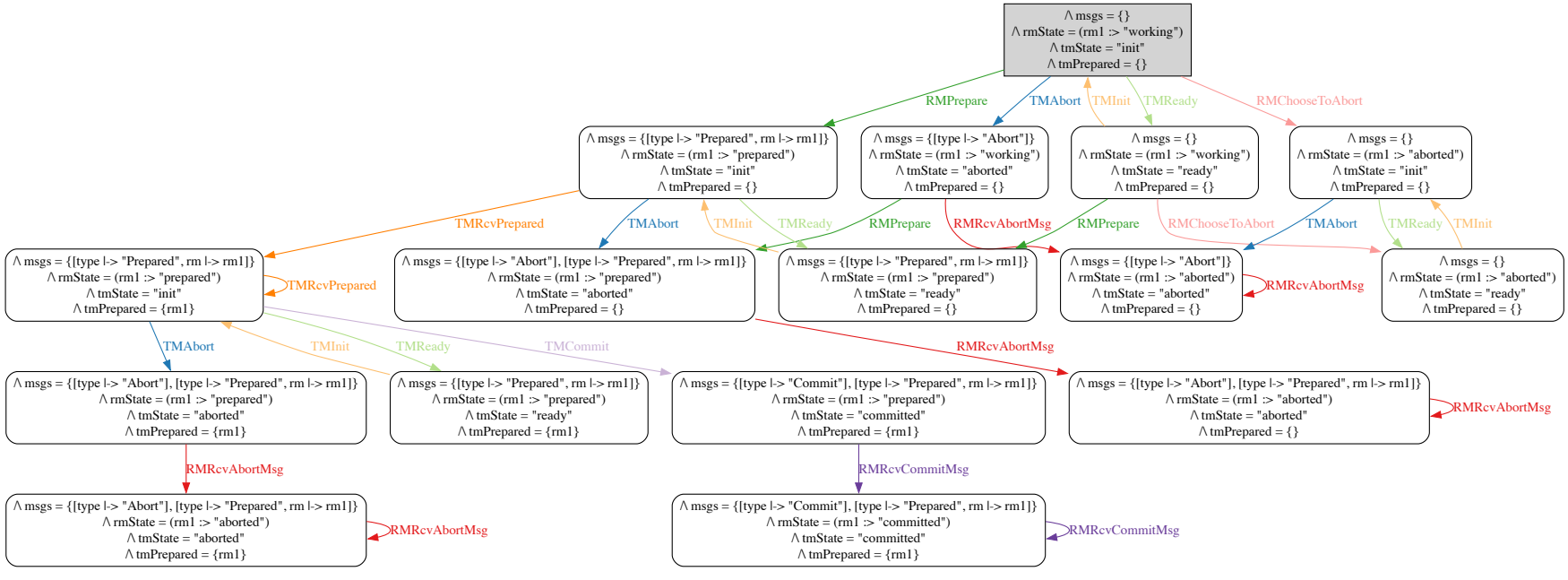
$\land\ rmState[rm] = \text{"working"}$
$\land\ rmState' = [rmState \text{ EXCEPT } ![rm] = \text{"prepared"}]$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{"Prepared"},\ rm \mapsto rm]\}$
$\land$ UNCHANGED $\langle tmState,\ tmPrepared \rangle$

Our actors are:

- Transaction Manager TM
- Resource Managers $RM1, RM2..RMn$

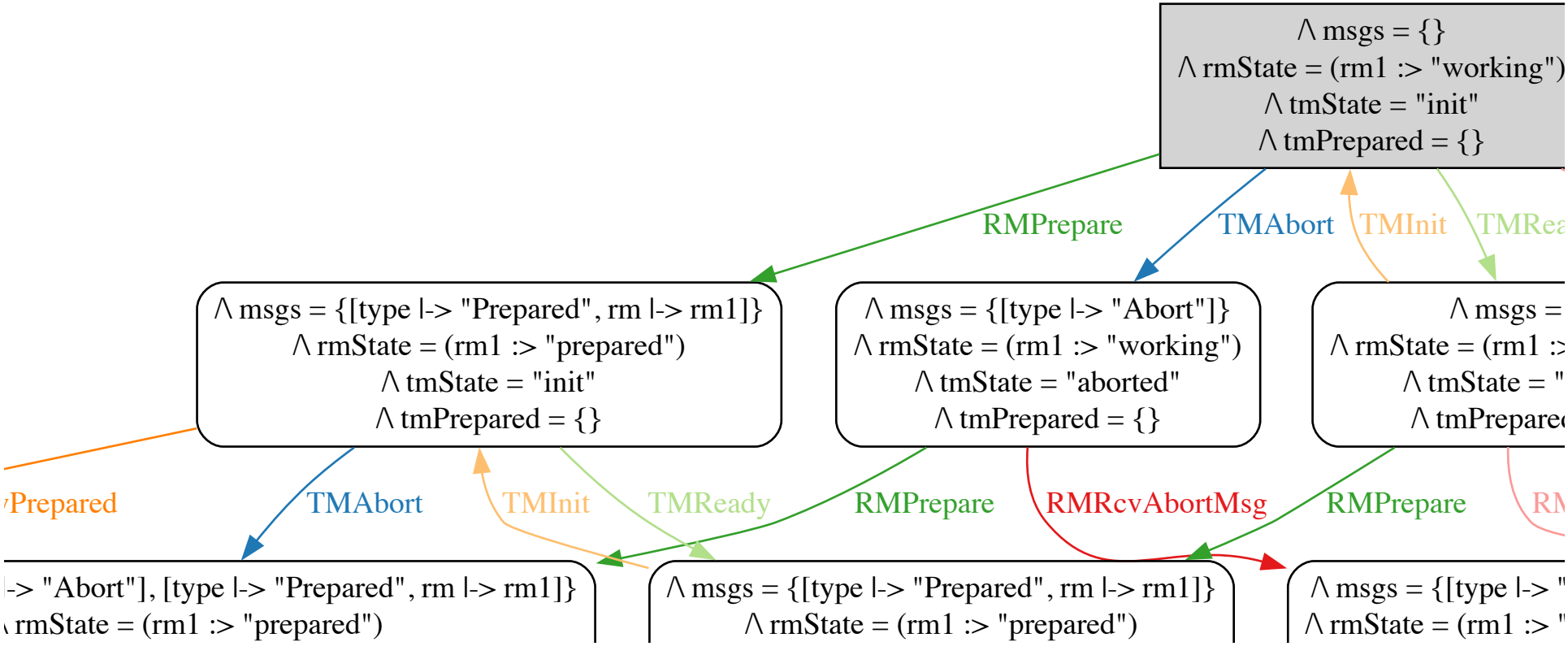
Who are the actors in a TLA^+ spec?

COMING BACK TO OUR STATE DIAGRAM



State space of the Two-Phase-Commit Protocol with one resource manager (30 distinct states).

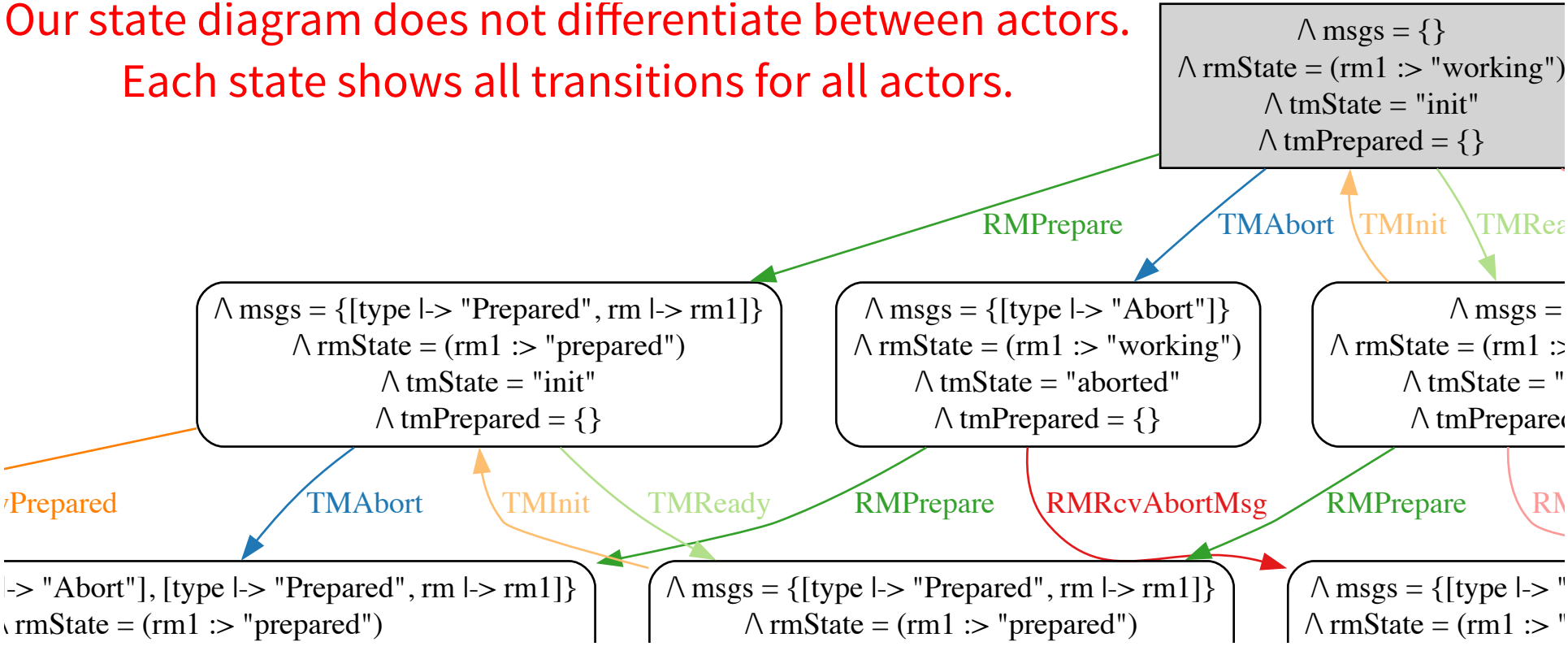
COMING BACK TO OUR STATE DIAGRAM



State space of the Two-Phase-Commit Protocol with one resource manager (30 distinct states).

COMING BACK TO OUR STATE DIAGRAM

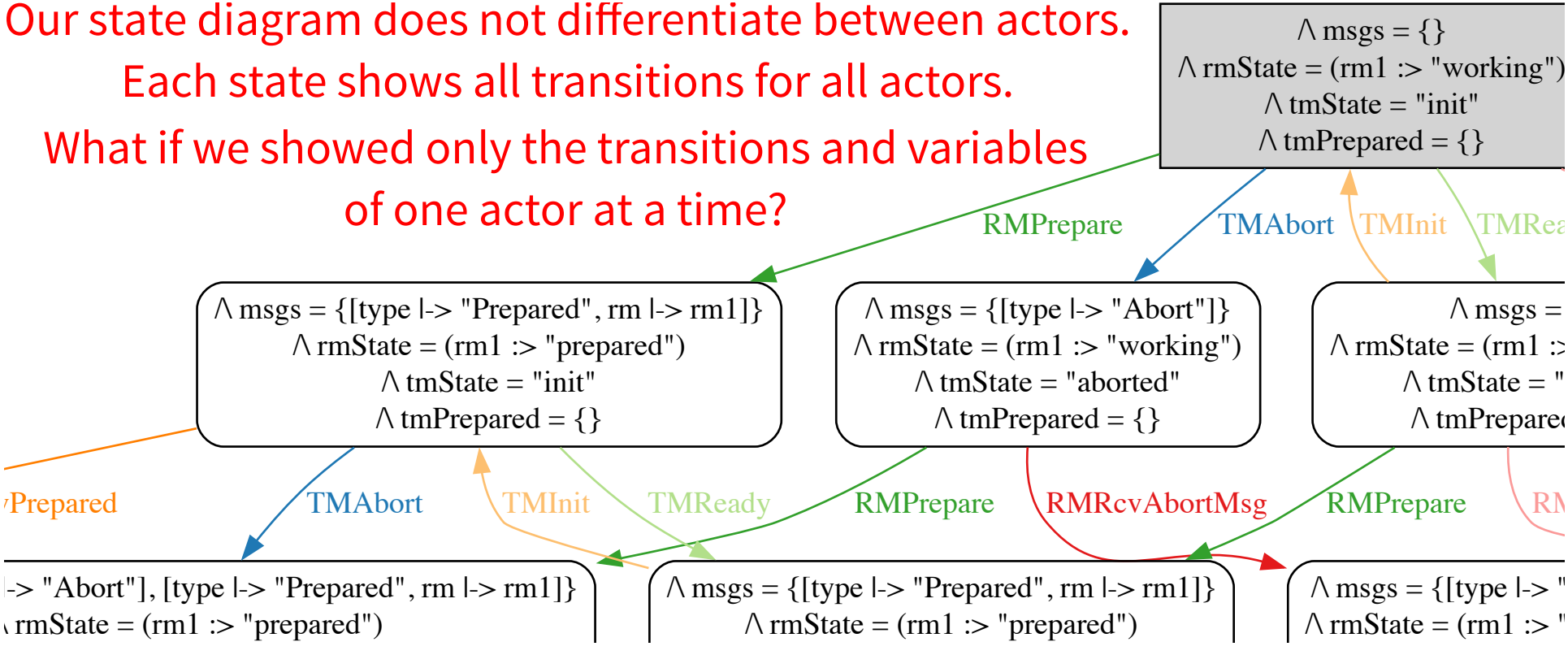
Our state diagram does not differentiate between actors.
Each state shows all transitions for all actors.



State space of the Two-Phase-Commit Protocol
with one resource manager (30 distinct states).

COMING BACK TO OUR STATE DIAGRAM

Our state diagram does not differentiate between actors.
Each state shows all transitions for all actors.
What if we showed only the transitions and variables of one actor at a time?



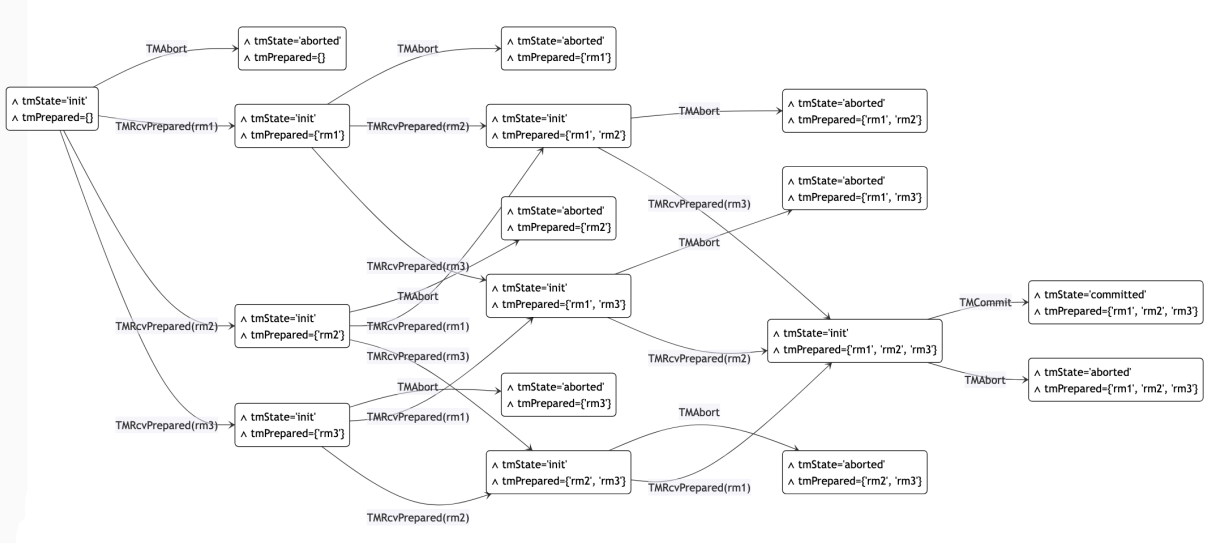
State space of the Two-Phase-Commit Protocol with one resource manager (30 distinct states).

WE WOULD GO FROM THIS

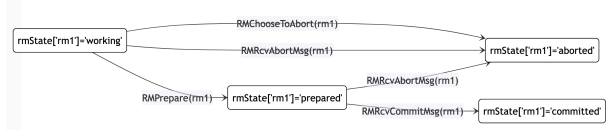
State space of the Two-Phase-Commit Protocol
with three resource managers (1370 distinct states).

TO THIS

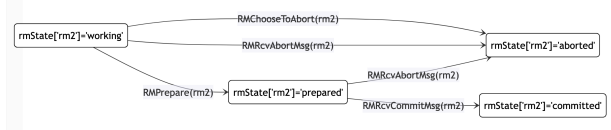
TM



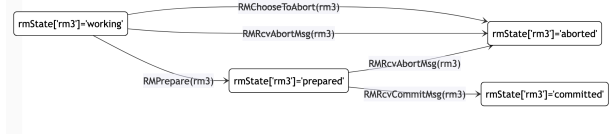
$RM\ 1$



$RM\ 2$

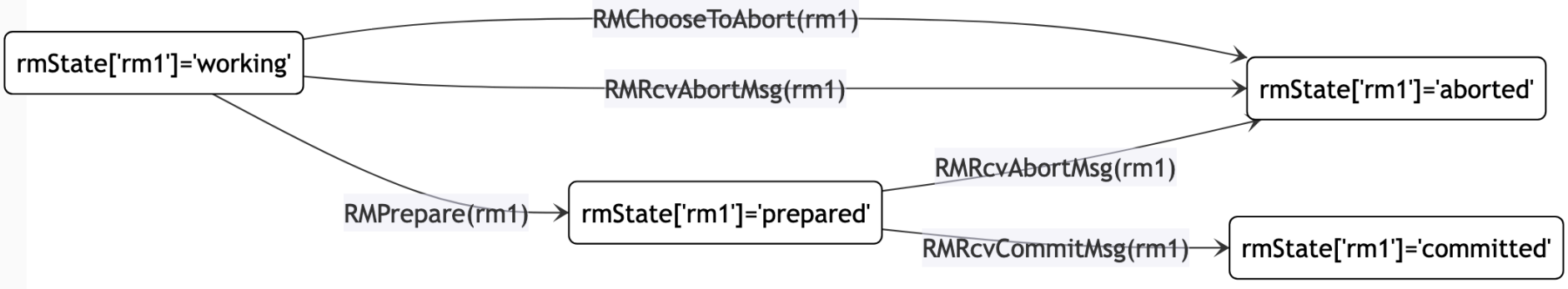


$RM\ 3$



Individual actor's state diagrams of the Two-Phase-Commit Protocol with three resource managers.

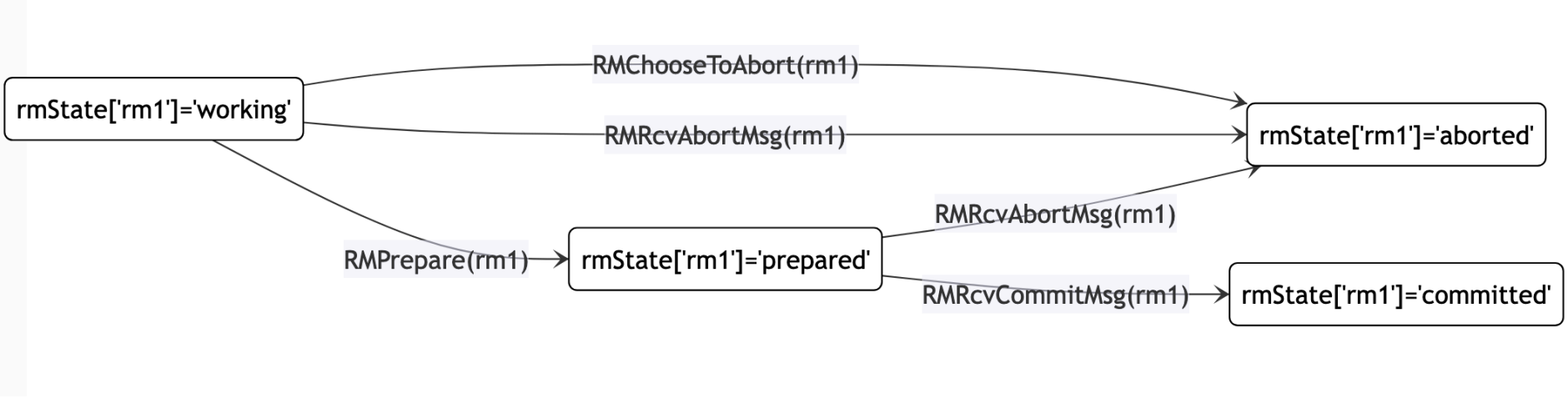
$RM\ 1$ IN DETAIL



The state space of $RM\ 1$ consists of all possible values of $rmState[rm1]$.

All actions that change that variable are shown above.

$RM\ 1$ IN DETAIL



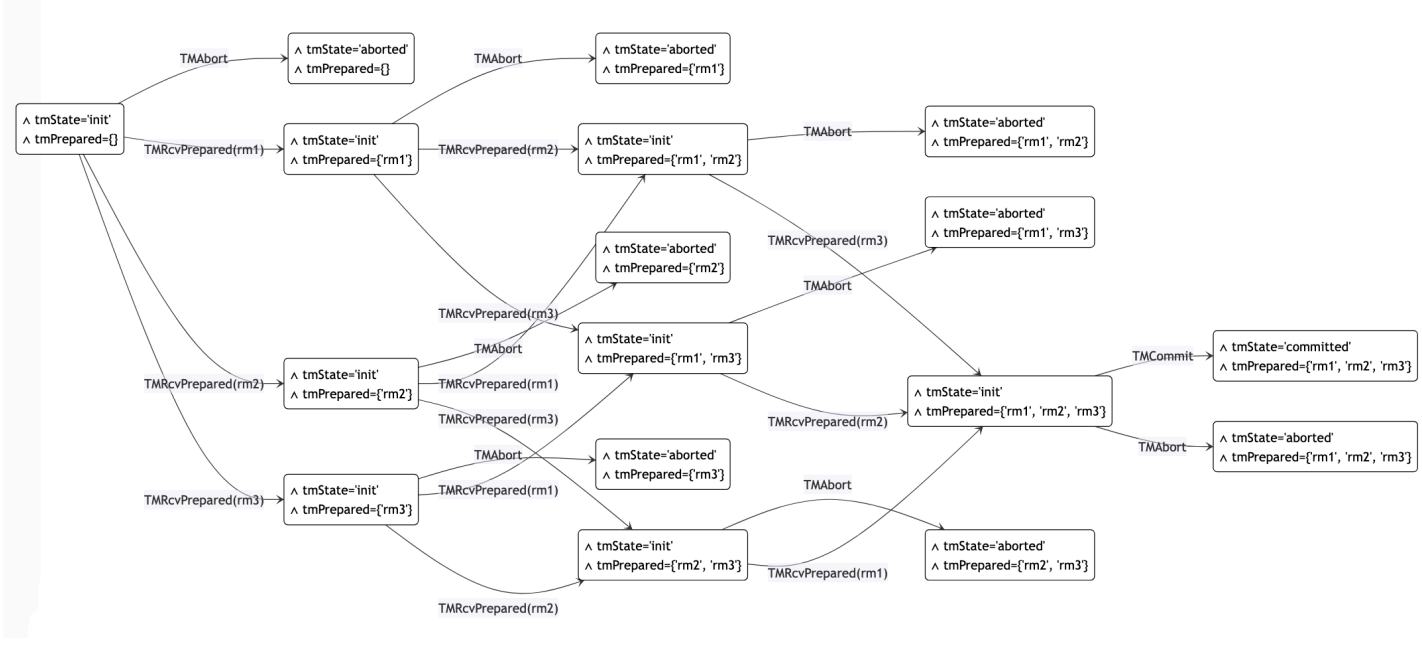
The state space of $RM\ 1$ consists of all possible values of $rmState[rm1]$.

All actions that change that variable are shown above.

Limitations:

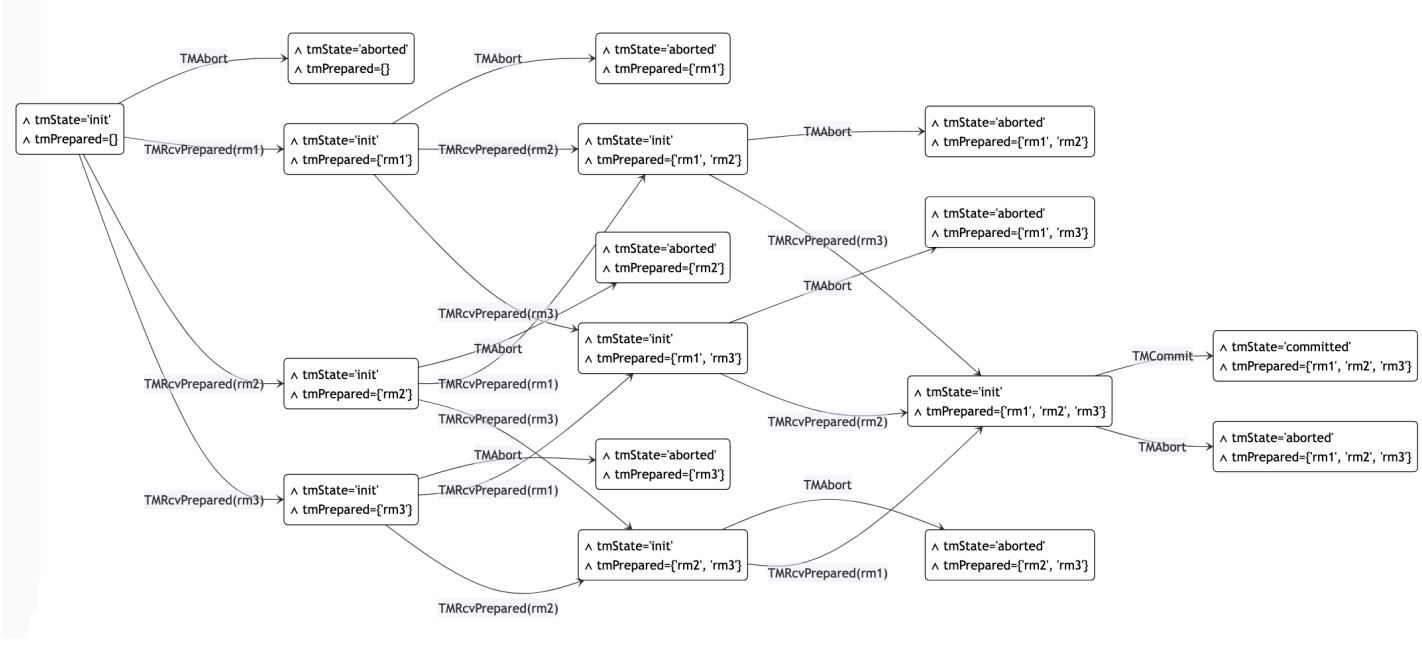
- Actions like $RMPrepare$ have side effects. They change the $msgs$ variable.
- Transitions can depend on a condition, e.g., $RMRcvAbortMsg$ depends on prior actions of the TM.

TM IN DETAIL



TM is more complicated, but still fits on a slide.

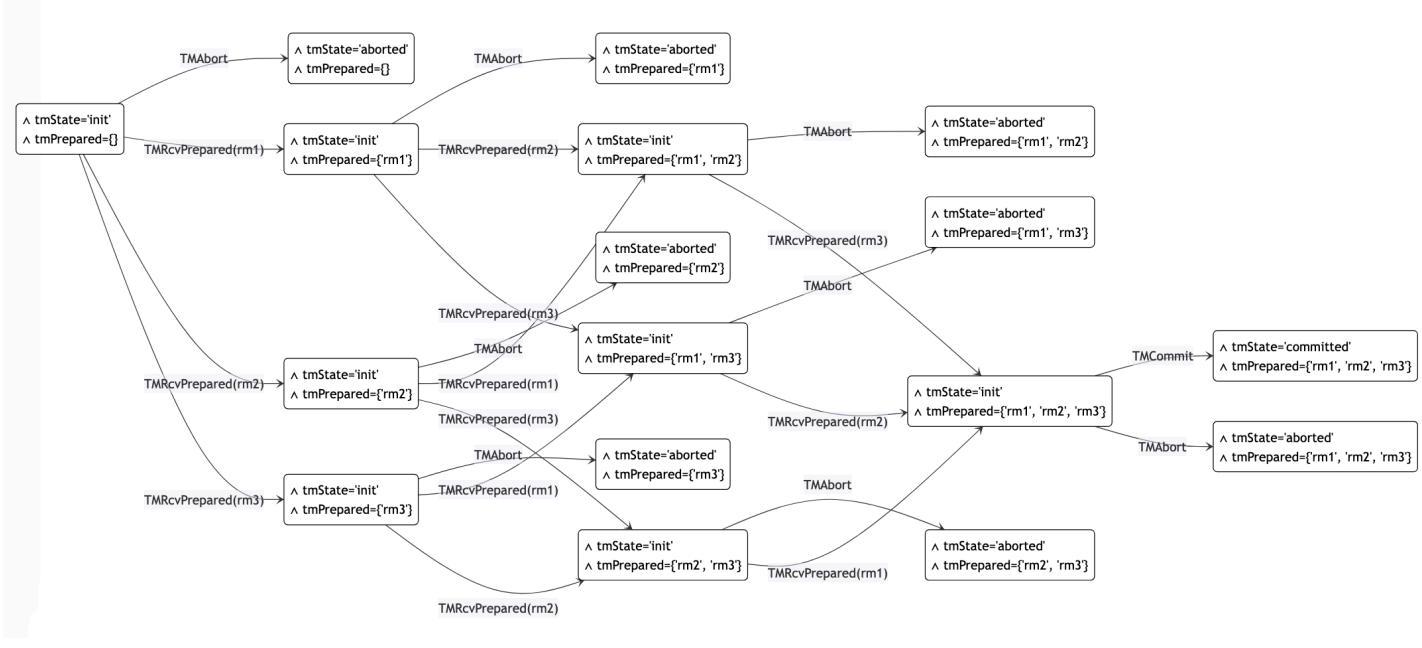
TM IN DETAIL



TM is more complicated, but still fits on a slide.

Note that $tmPrepared$ causes most of the state complexity.

TM IN DETAIL



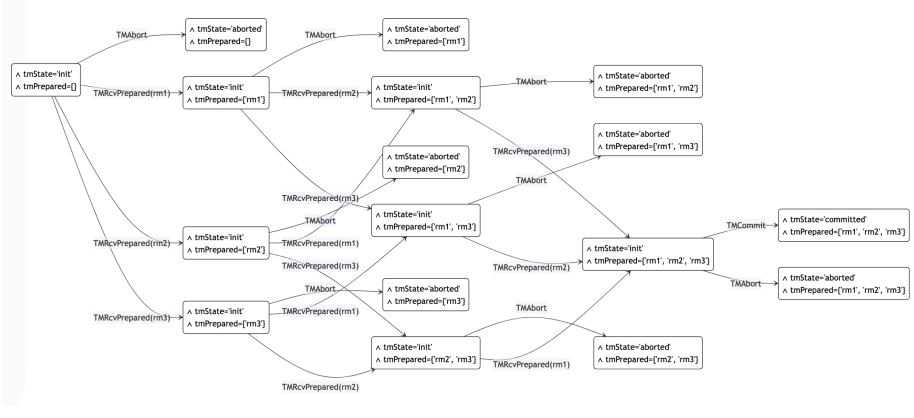
TM is more complicated, but still fits on a slide.

Note that $tmPrepared$ causes most of the state complexity.

What happens for more complex specs?

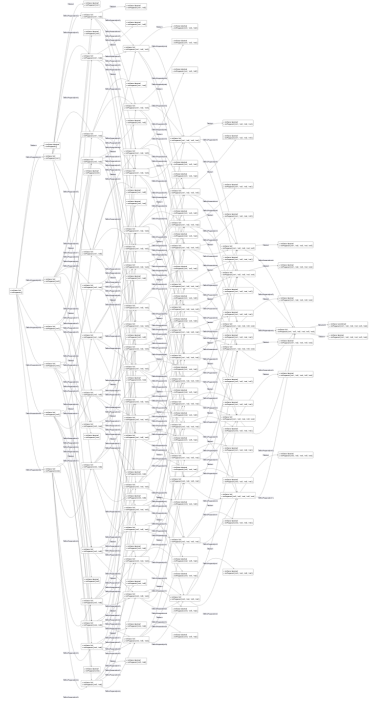
TM IN DETAIL

$3\ RMs$



- 288 distinct states in total
- 17 states in TM Actor

$6\ RMs$

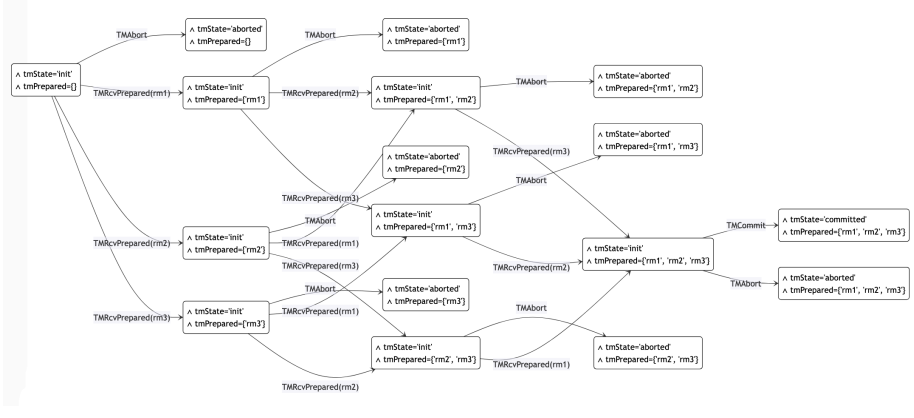


- 50816 distinct states in total
- 129 states in TM Actor

TM IN DETAIL

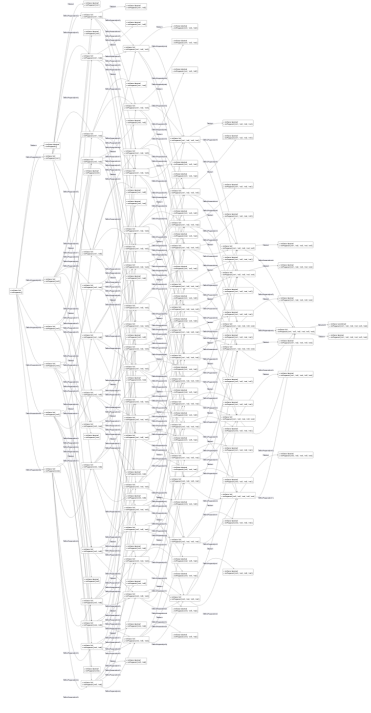
State explosion is inevitable.

$3\ RMs$



- 288 distinct states in total
- 17 states in TM Actor

$6\ RMs$

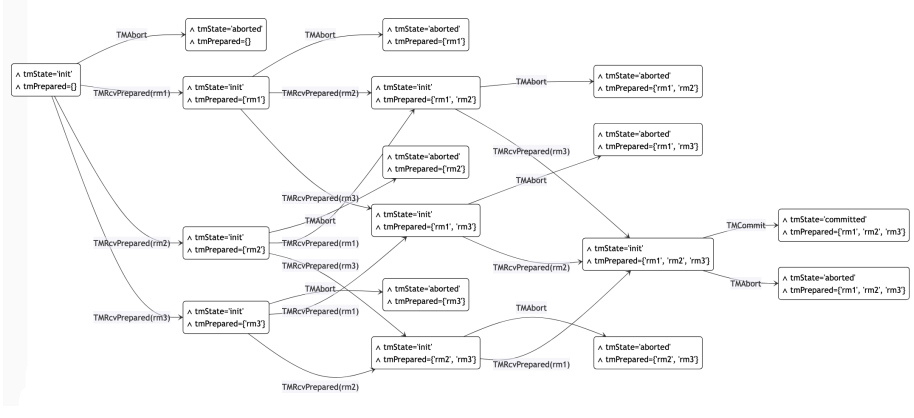


- 50816 distinct states in total
- 129 states in TM Actor

TM IN DETAIL

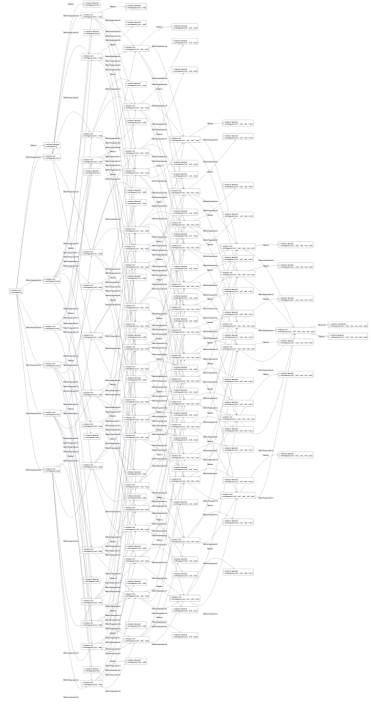
State explosion is inevitable.
There's one more thing we can do, though!

$3\ RMs$



- 288 distinct states in total
- 17 states in TM Actor

$6\ RMs$



- 50816 distinct states in total
- 129 states in TM Actor

PROJECTION DIAGRAMS

As it turns out, the actor metaphor is just one projection of many possible

Mastering the Visualization of Larger State Spaces with Projection Diagrams

Lukas Ladenberger(✉) and Michael Leuschel

Institut Für Informatik, Universität Düsseldorf, Düsseldorf, Germany
{ladenberger,leuschel}@cs.uni-duesseldorf.de

Abstract. State space visualization is a popular technique for supporting the analysis of formal models. It often allows users to get a global view of the system and to identify structural similarities, symmetries, and unanticipated properties. However, state spaces typically become very large, so human inspection of the visualization becomes difficult. To overcome this challenge, we present an approach which can considerably reduce the size of the state space by creating *projection diagrams*. Moreover, we present an approach to link a projection diagram with a domain specific visualization. The projection diagram construction can be initiated directly from user-selected graphical elements without the user having to write formulas or having to know the variables or internal structure of the model. This makes the projection diagram inspection and construction accessible to non-formal method experts. These techniques have been implemented within the ProB toolset, and we demonstrate their benefits and usefulness on several examples.

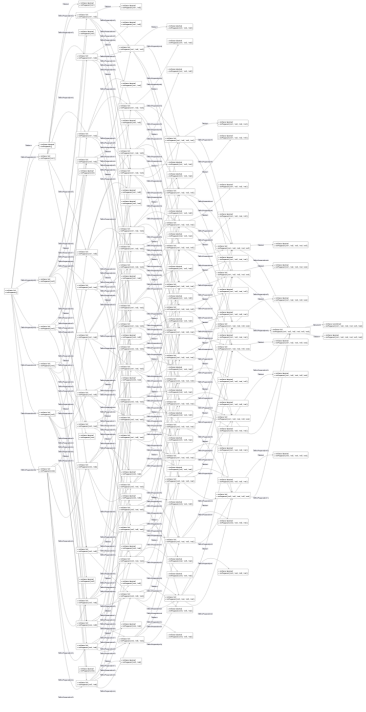
Keywords: Formal methods · B-Method · State space · Visualization · Human inspection · Domain specific visualization · Tool support

1 Introduction and Motivation

In state-based formal methods, such as the Classical-B method [2] and its successor Event-B [1], the system behaviour is modelled by *states* and *transitions*. A state is a particular configuration of variables, whereas transitions link two

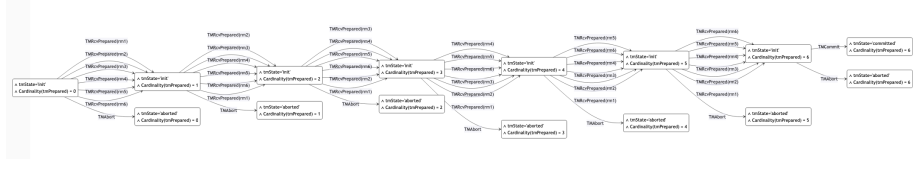
TM, PROJECTED

$6\ RMs$



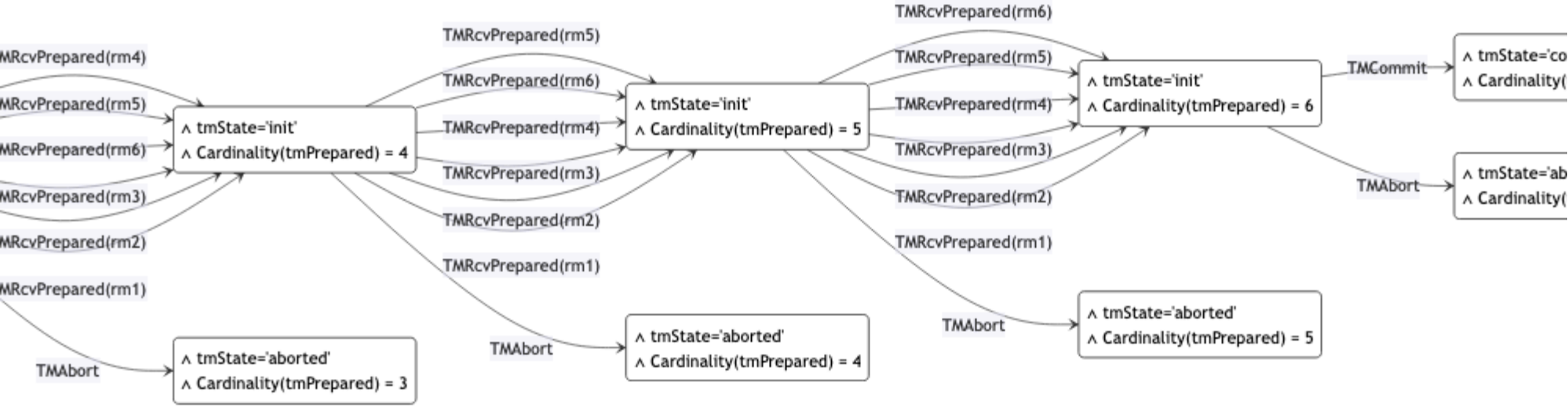
- 50816 distinct states in total
- 129 states in TM Actor

$6\ RMs$



- 50816 distinct states in total
- Project $tmPrepared$ on its cardinality
- $numProjectedStates = |RM| \cdot 2 + 1$

TM, PROJECTED



- 50816 distinct states in total
- Project $tmPrepared$ on its cardinality
- $numProjectedStates = |RM| \cdot 2 + 1$

```
function projectTM(state) {
  const numPrepared = state.tmPrepared.length;
  return `
  ∧ tmState='${state.tmState}'
  ∧ Cardinality(tmPrepared) = ${numPrepared}`
}
```

HOW DOES IT WORK?

Given:

I. State dump from TLC
(a directed graph with each node being a distinct state and the edges being actions)

II. Projections of state on actor state for each actor
(generally: projection of state on projected state)

Algorithm:

1. Initialize empty graph for each actor
2. For each (transition, projection) pair:
    2a. Project pre-state and post-state
    2b. If projections differ: Add transition and projected states to actor's graph
3. Return each graph per actor

Limitations:

- Requires additional input (I., II.)
- Does not show stuttering steps
- Scalability concerns

LIMITATIONS AND SOLUTIONS

Requires additional input

- Option 1: Provide projection functions in spec
  Introduce something like ALIAS for the whole state dump
  Nice: no extra language for defining projection functions
  But: no interactivity
- Option 2: Export to a machine-readable format that allows simple parsing of *variables*
  something like JSON export of whole state dump
  Nice: interactivity
  But: what language for projection functions? disk usage of state dumps?

Does not show stuttering steps

Scalability concerns

APPENDIX: PROJECTED DIAGRAMS WITH BUILT-IN FEATURES OF TLC #1

1. Add an ALIAS expression in your spec that projects on one actor
2. Run tlc to dump the state graph. It will apply the ALIAS expression to each state. (if not: either your tlc version is outdated or the ALIAS expression is incorrect)
3. Run this script to merge the redundant states and transitions of the dot file.

APPENDIX: PROJECTED DIAGRAMS WITH BUILT-IN FEATURES OF TLC #2

1. Add an ALIAS expression in your spec that projects on one actor. For example:

```
Alias == [
  tmState |-> tmState,
  sizeTmPrepared |-> Cardinality(tmPrepared)]
```

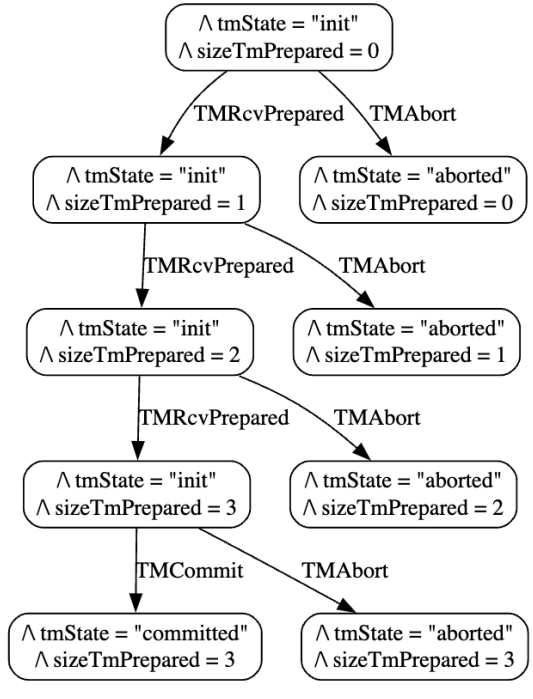
2. Run tlc to dump the state graph. For example:

```
tlc -dump dot,actionlabels graph.dot TwoPhase.tla
```

3. Run this script to merge the redundant states and transitions of the dot file.

APPENDIX: PROJECTED DIAGRAMS WITH BUILT-IN FEATURES OF TLC #3

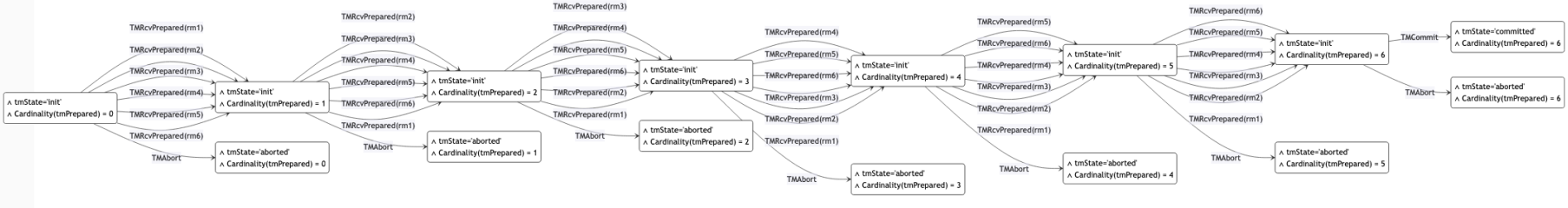
Open the graph-projected.dot file:

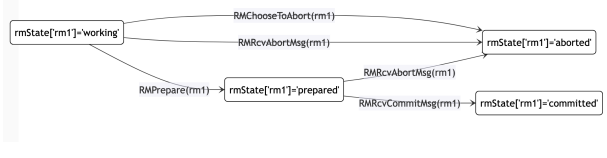


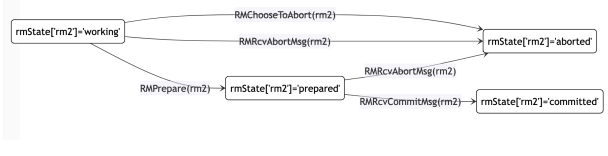
SUMMARY

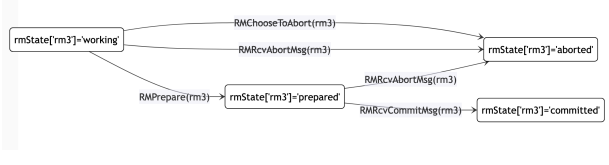
Using the visual actor metaphor and projection functions, we can turn state diagrams like this...

... into diagrams like this:









Tackling State Space Explosion in TLA^+ Visualizations

Thank you for your attention!

Daniel Stachnik
hi@dast.xyz

Tom Beckmann
tom.beckmann@hpi.de

Robert Hirschfeld
robert.hirschfeld@hpi.de