# Strongly Connected Components

* Only grouping of two in two dies
* Can group all together if single SCC
* Can be the case that does not group anything (DCG)
* Decision if min Size of SCC -> some might not have property
* Totally partioning if minCompsize in {0,1}
* If not totally: For overview maybe better to group all non-matching states
* Images are made with minCycleSize >= 3

# Strongly Connected Bottom Components

* Currently no models found with BSCC

# Identical Outgoing Actions

* The set of outgoing actions can be empty!? (Randomized two process wait) -> only then relevant if clusternNonMatchingStates -> otherwise truly partitioning
* Very strong compression due to normally very few actions -> normally all combinations exist -> not a lot of knowledge gained but probabilities and further zoom

# Distance Cluster (Forward)

* Very good overview about Structure and Progression
* Very good compression (15 States with granularity 3 for RDP)
* From an intentional perspective makes little sense to use to use clusterNonMatchingStates

# Collapse Dual Direction Cluster

* Overall low effect: No effect in RTPW, RDP and DCG and only groupings of two in the rest
* No obvious Motivation for usage
* Because of little impact no real difference with regard to clusterNonMatchingStates

# Atomic Propositions Identical

* Strong compression WHEN USING TOTAL GROUPING (in examples, but normally few APs – I think) -> total Grouping recommended
* Good overview or starting point for Exploration

# Property Cluster

* Not yet evaluated

# Has Cycle Cluster

* Finds Cycle
* More suitable if clusterNonMatchingStates
* Single or only as an example for composotion

# Cycle Cluster (Hiding vs Showing)

###### Two dies\_0:

* Action: same, mincyclesize >= 3, clusternms = true, Exact-> overview of cycles
* Without clusternms -> little effect BUT hides information
* No findings with same action

###### TPME\_1:

* Almost no matching with mode Exact
* Same Actions: Two Cycles found!
* JoinSet motivation

###### RDP\_2:

* One SCC -> does not finish (runtimeError)

###### RTPW\_3:

* With preprocessing of strongly connected components about 2 to 5 minutes without ram settings (Exact, 3, actions enabled)
* 27 + 97 SCC with

DCG\_4:

* No cycles with mincyclesize >= 2, clusternms = true, Exact

# Parameters Cluster

* Strong compression when using without values -> shows modules?
* Works with values

# Amount of outgoing action

* Very good overview of the structure -> good starting point for exploring
* Notable: states with no outgoing actions! And very often states with one outgoing action
* Good for which decisions to make since there are none if = 1
* Not tested with action restriction: Set of considered actions is restricted to given ones
  + Might have single action -> 0 if not of required
  + Set size is reduced to number of actions that are amongst the given ones  
    -> set size <= set size given actions

# Initstates

* Only DCG has more than one initstate but also only four
* Hiding probably wont be of huge relevance, maybe finding them
* In general the property cluster can be used

Usecases

Paramters: look into a module (begin to understand the structure)  
Model does not work (?):

* check if there are states that should not be contained
* If available use distance to find guilty predecessor

If no actions are used outactionsident can be used to determine modules:

* Die1 are states from which on die1 takes an action
* Die2 are states from which on die2 takes an action
* Die1,die2 are states from which on both can take actions (shared state)

Use has outaction

Specific Ideas Usecases

1. Path to badstate (not corrupt but one the system should better not land in)
   * Property Cluster
   * Hide states that can not reach that states *ReachabilityView*
   * Hide states with single action: Might corrupt path?
2. Understand a given MDP (maybe two differing on SCC)
   * Modules (Parameters)
   * Modules + init
   * Strongly connected components
   * OutActionsIdent (to see which Actions)
   * *Show modules as states (not possible I think)*
3. Bugfix an MDP (Bad state) (like (2,2) in Two Process Mutual Exclusion)
   * Search for one or more bad states
   * Check if they are reachable (Paramters DNF)
   * Find notes by that they are reachable
4. Bugfix an MDP (find) cycle
   * See if MDP has Cycle
5. Performance (on scalable MDP e.g. Dining Cyptographers)

Views with little use

* All ActionViews! -> hasAction maybe for bugfixing?
* DoubleDirViews

Performance Testing (Scalability)

* Dining Cryptographers