Testname (Groovy)	Model used	Usage	Classes Covered	Notes/ TODOs
algMultProcedure	***generated***	generates a model "Multiplication" and animates it	ModelModifier, AlgorithmGenerationOptions, AlgorithmTranslator	
algorithmMultExample	***generated***	generates the model "Multiplication" from an algorithm	ModelModifier, AlgorithmGenerationOptions, AlgorithmTranslator	
animationCommands	***scheduler.mch***	uses animationCommands on a model	CheckMaxOperationsReachedStatusCommand, CheckTimeoutStatusCommand, GetEnabledOperationsCommand, GetOperationsByPredicateCommand, IStateSpaceModifier	
animationSmokeTest	***scheduler.mch***	creates 10 random Traces for a model		
anyOperationTest	***scheduler.mch***	anyOperation and anyEvent are executed properly	Trace	
assertionPropagation	***generated***	generates a model of a "binary search" algorithm	ModelModifier, AlgorithmGenerationOptions, AlgorithmTranslator	
asynchronusEval	***scheduler.mch***	checks if formulas can be registered and evaluated later on	EvaluationErrorResult	
bowlOfCherriesTest	***generated***	generate and animate the "bowlOfCherries" Model	ModelModifier, AlgorithmGenerationOptions, AlgorithmTranslator	
branchTrace	***scheduler.mch***	perform animations on a model, go back and have different step results in a correct trace	Trace, getTransitionList	
carsOnABridge	***generated***	generate cars on a bridge model		Can't fix
cbcChecking	***createErrors.bcm***	test constraint based dreadlock and invariant checking	CBCDeadlockChecker	
cbcTest	***scheduler.mch***	test if CBC solving correctly	CbcSolveCommand	TODO: another example and translation test missing
checkStdLib	***none***	check if standard library is present		
correctIncludes	***M1.mch***	check if machines are prefixed and represented correctly (and checks if subscription works correctly)	Dependency Graph	
counterLoadBumBumTest	***machine.bum.bum***	ensures that .bum.bum file can be loaded and doesn't result in empty machine	.	
CoverageTest	***scheduler.mch***	ComputeCoverageCommand returns expected result	ComputeCoverageCommand	

cspAssertions	***Deterministic1.csp***		CSPModel, CSPAssertionCommand	example for checking CSP assertion
enableMatrixTest	***scheduler.mch***	checks if info of GetEnableMatrixCommand is set correctly while executed	GetEnableMatrixCommand	
enumerationWarning	***scheduler.mch***	check if warnings are handled	EnumerationWarning	
euclidAlgorithm	***generated***	generates the model "Multiplication" from an algorithm	ModelModifier, AlgorithmGenerationOptions, AlgorithmTranslator	
evalStore	***scheduler.mch***	eval store MAY(?) be working correctly	EvalstoreCreateByStateCommand, EvalstoreEvalCommand (redundant), EvalstoreResult	maybe give it a better name... and change description
evaluateFormulasTest	***scheduler.mch***	check if registered formula is automatically evaluated in every state and can be found in the cache later on		there's a ticket for that so keep in mind -> command to unregister formulas PROBCORE-931
evaluationOfFormulasTest	***scheduler.mch***	check if evaluating of FORMULAS work	ClassicalB, TranslatedEvalResult, Atom	
executeUntil2	***Simple.mch***	for scheduler it is possible to randomly animate until a condition is met, check whether result trace is as expected	ExecuteUtilCommand	TODO: command to slow to make a new ExecuteUtilCommand and execute it
factorialAlgorithm	***generated***	generate and animate the "fatorialAlgorithm" Model	ModelModifier, AlgorithmGenerationOptions, AlgorithmTranslator	
filterByPredicate	***scheduler.mch***	test if method getStatesFromPredicate works correctly		
filterStates	***scheduler.mch***	test if command FilterStatesFromPredicateCommand works correctly	FilterStatesFromPredicateCommand	
findTrace	***scheduler.mch***	test if GetShortestTraceCommand works correctly	GetShortestTraceCommand	
formulaExpansion	***scheduler.mch***	test if ExpandFormulaCommand works correctly	ExpandFormulaCommand	
getStateRep	***scheduler.mch***	test if you can get a state representation from prolog	State, getStateRep	
getTransitionDiagrammTest	***scheduler.mch***	check if GetTransitionDiagrammCommand works correctly	GetTransitionDiagrammCommand	
getVersion	***scheduler.mch***	check if GetVersionCommand works correctly	GetVersionCommand	
levelsLoadBccTest	***levels.bcc***	check if a .bcc file can be loaded and doesn't result in empty machine	api.*_load	

levelsLoadBucTest	***levels.buc***	check if a .buc file can be loaded and doesn't result in empty machine	api.*_load	
llParsingAlgorithm	***generated***	generate and animate the "LLParsing" Model	ModelModifier, AlgorithmGenerationOptions, AlgorithmTranslator	
loadAndSaveEventB	***lift0.bcm***	test if loading and storing an eventB example works		
loadClassicalBfromAst	***generated***	load generated B machine and test it	Bparser	
loadEventBFromZip	***Farmer.zip***	check if model can be loaded from zip file	ModelFactory	
loadEventBLift	***lift0.bcm***	check if the models are still accessible, when an EventB file is loaded	Dependency Graph.ERefType	
loadFromAst	***generated***	check if it's possible to load a model from an ast	BParser, ModelFactory	
loadFromString	***generated***	check if it's possible to load a model from a string	ModelFactory	
loadLift	***Lift.mch***	check if Lift.mch is loaded		
ltlTest	***scheduler.mch***	test if LTLFormulas can be checked	LTL, LTLCounterExample, LTLError, ModelChecker	
modelAPI	***EmptyMachine.bcm***	test if ModelAPI works correctly	ModelModifier, make	
modelAPI2	***EmptyMachine.bcm***	test if ModelAPI works correctly	ModelModifier, make	
modelCheck	***scheduler.mch***	test if Model Checking works correctly	ConsistencyChecker, ModelCheck	
modelCheckingProcedures	***generated***	create model of model checking algorithm		
modelConstruction	***generated***	test if it's possible to construct a model	ModelModifier, getModel	
nonExistentFilesCaught	***none***	test if FileNotFoundException is thrown if an invalid file is given	api.*_load	
opRepresentation	***MultipleExample.mch Deterministic1.csp***	check if Operations are expanded as expected, when calling getCurrentTransition		

predicateCommands	***scheduler.mch***	test if PrimePredicateCommand works correctly	PrimePredicateCommand	
preferencesTest	***scheduler.mch***	set Preferences and check if they work	GetCurrentPreferenceCommand	
proB344TypeError	***Life.mch Marriage.mch***	check if ProB-344 is still resolved		
prologASTTest	***scheduler.mch***	Check if PrologAST builds correctly	GetModelCheckingCommand, PrologAST	
russischeBauernmulti	***generated***	generate and animate the "Bauern2" Model	ModelModifier, AlgorithmGenerationOptions, AlgorithmTranslator	
schedulerLoadBcmTest	***Scheduler0.bcm***	check if a .bcm file can be loaded and doesn't result in empty machine, and check if some steps can be done	api.*_load	
schedulerLoadBumTest	***Scheduler0.bum***	check if a .bum file can be loaded and doesn't result in empty machine, and check if some steps can be done	api.*_load	
schedulerTests	***scheduler.mch***	test some attributes of the scheduler model		if this test doesn't work, most of the other tests wouldn't work either
searchForGoal	***scheduler.mch***	test if checking for goals works	ModelCheckingOptions, checkGoal	
SerializeTest	***scheduler.mch***	test if state can be serialized	SerializeStateCommand	
simpleMultiplication	***generated***	generate and animate "SimpleMult" model	ModelModifier, AlgorithmGenerationOptions, AlgorithmTranslator	
syncedTraces	***scheduler.mch lift0.bcm***	check if it's possible to sync traces with event and parameter combinations	SyncedTraces, Trace	
testCommand	***scheduler.mch***	test if it's possible to create a simple text command	SimpleTextCommand	
testPrettyTransitions	***lift0.bcm***	test if pretty print works	Trace, getPrettyRep	
testStub	***scheduler.mch***	RAW MODEL OF HOW A TEST SHOULD LOOK LIKE, WITH SCHEDULER AS MODEL...		maybe change the machine model here, would result in different machines in the next tests and rename to template
testUnitPluginActivation	***Empty.mch***	test if unit plugin can be activated and accessed successfully	GetPluginResultCommand	
testWDerrorsInEvaluation	***scheduler.mch***	test if evaluation errors can be caught and evaluated simply	EvaluationErrorResult	

theoryIntegration	***Mch.bcm***	test if it's possible to load and create and animate and Event-B model that uses the theory plugin	Theory, ModelModifier	
traceFromToNode	***scheduler.mch***	test if you can create a trace between two given nodes	FindTraceBetweenNodesCommand	
translateSubscribe	***scheduler.mch***	test if a formula object can be subscribed	ClassicalBModel, subscribe	
truncateValues	***MultipleExample.mch***	test if expanding and truncating formulas works	FormulaExpand	has no description
typeCheck	***scheduler.mch***	test if it's possible to type check a formula	ClassicalB, Formula, typecheck	
unsatCore	***scheduler.mch***	test if UnsatCoreCommand and MinimumUnsatCoreCommand work properly	UnsatCoreCommand, MinimumUnsatCoreCommand	dead code? and MinimumUnsatCoreCommand is not used
usedMachineVariableScoping	***Foo.mch***	test if variables are correctly prefixed	ClassicalBModel, variables, invariants	
useSeqsInAlgorithm	***generated***	test if it's possible to load and create and animate and Event-B model that uses the theory plugin	Theory, ModelModifier, AlgorithmTranslator	
valueTranslator	***none***	check if values are translated correctly	Tanslator, translate	intelliJ shows me a bug there
variantGeneration	***generated***	test if it's possible to generate "MyLoopProject" and check termination of loops	ModelModifier, AlgorithmGenerationOptions, AlgorithmTranslator	

Legend
grey= model, orange= TODO, green= ticket! yellow= Template