```
module tests_model1_Amend

open model1


                                        /*****************************************************
*******
                                        *       CAN BE ENABLED/DISABLED:
                                        *

        ************************************************************/
                                        fact CAN_BE_ENABLED_DISABLED {
                                        // disable_auxiliary_atoms
                                        // enable_movedOnce
                                        // whole_session
                                         session_prefix
                                        // session_suffix
                                        // reach_all_motions
                                        //dynamic_mode_only
                                                no ErrorFlag
                                        }

                                        /****************************************
                                        *       SPECIFIC AMEND TESTING AND CO    *
                                        ****************************************/
                                pred SPECIFIC_AMEND_TESTING_AND_CO {}
                                run SPECIFIC_AMEND_TESTING_AND_CO

//------SHOULD-SUCCEED-----//
pred SHOULD_SUCCEED {}

 pred some_amend_applying_to_itself {
        some amend : Amend | some amend.isAppliedTo & Amend
 }
 run some_amend_applying_to_itself for 5
 expect 1

pred move_amend{

        some before, after :SessionState, motion : Amend |
                        move[before, after, motion]

 }
 run move_amend for 5 //, exactly 1 Member
 expect 1

pred show_Amend {

        some Amend
```

```
 }
 run show_Amend for 3 but  3 Motion, 3 SessionState, 3 Business
 expect 1

pred show_Amend_not_yieldingTo_anything {

        some a :Amend | no a.yieldsTo

}
 run show_Amend_not_yieldingTo_anything for 3 but  3 Motion, 3 SessionState, 3 Business
 expect 1


pred Amend_applying_to_the_root_and_yieldingTo_nothing {
        some a : Amend | some (a.isAppliedTo & RootMotion) and no a.yieldsTo
}
run Amend_applying_to_the_root_and_yieldingTo_nothing for 3 but 7 SessionState//, exactly 1
Member
expect 1

pred move_Amend {

        some before, after :SessionState, motion : Amend|
                        move[before, after, motion]
}

run move_Amend for 5
 expect 1

 pred disposeOf_Amend {

        some before, after :SessionState, motion : Amend|
                        disposeOf[before, after, motion]

 }
 run disposeOf_Amend for 3 but 7 SessionState//, exactly 1 Member
 expect 1

 pred adopt_Amend {

        some before, after :SessionState, motion : Amend|
                        adopt[before, after, motion]

 }
 run adopt_Amend for 3 but 5 SessionState//, exactly 1 Member
 expect 1
```

```
//------SHOULD-FAIL-----------//
pred SHOULD_FAIL {}



 pred some_Amend_yielding_to_another_not_applied_to_it {

        some amend : Amend |
                some another : (amend.yieldsTo & Amend) |
                        amend not in another.isAppliedTo
 }
 run some_Amend_yielding_to_another_not_applied_to_it for 5
 expect 0


/************************************************
                                                * AUXILIARY ATOMS AND AMEND TESTING
*

************************************************ */
                                                pred
AMEND_AND_AUXILIARY_ATOMS_TESTING{}
                                                run
AMEND_AND_AUXILIARY_ATOMS_TESTING

//------SHOULD-SUCCEED-----------//
pred SHOULD_SUCCEED {}

pred show_Aux {
        some aux : Aux | some aux.contexts
}
run show_Aux for 6
expect 1

pred a_world_with_disabled_auxiliary_atoms {
        disable_auxiliary_atoms
}
run a_world_with_disabled_auxiliary_atoms for 5
expect 1

pred a_world_with_enabled_auxiliary_atoms { //TODO
        not disable_auxiliary_atoms
}
run a_world_with_enabled_auxiliary_atoms for 7
expect 1

assert no_atom_has_auxiliaries_when_disabled {
        disable_auxiliary_atoms => no sessionStateElements.allAux
```

```
}
check no_atom_has_auxiliaries_when_disabled for 5
expect 0

pred show_Amend_when_auxiliaries_enabled {

	enable_auxiliary_atoms and some Amend
}
run show_Amend_when_auxiliaries_enabled for 5
expect 1

pred show_Amend_when_auxiliaries_disabled {

	disable_auxiliary_atoms and some a : Amend| hasTargets[a]
}
run show_Amend_when_auxiliaries_disabled for 5
expect 1

pred some_business_pointing_to_an_auxiliary {
	enable_auxiliary_atoms
	some b : Business | some b.allAux
}
run some_business_pointing_to_an_auxiliary for 5
expect 1

pred some_rootMotion_pointing_to_an_auxiliary {
	enable_auxiliary_atoms
	some r : RootMotion | some r.allAux
}
run some_rootMotion_pointing_to_an_auxiliary for 5
 expect 1

pred some_secondaryMotion_pointing_to_an_auxiliary {
	enable_auxiliary_atoms
	some sm : SecondaryMotion | some sm.allAux
}
run some_secondaryMotion_pointing_to_an_auxiliary for 5
 expect 1

pred a_transparent_amend {
	some elem : sessionStateElements, a : Amend |
		elem.yieldsTo = a and elem.allAux.yieldsTo = a.yieldsTo.allAux
}
run a_transparent_amend for 5  expect 1

pred show_division_of_the_question_when_auxiliaries_enabled {
	enable_auxiliary_atoms
	some DivisionOfQuestion
}
```

```
run  show_division_of_the_question_when_auxiliaries_enabled for 6  expect 1

pred show_createBlank_when_auxiliaries_enabled  {
        enable_auxiliary_atoms
        some CreateBlank
}
run show_createBlank_when_auxiliaries_enabled  for 6 //, exactly 1 Member
 expect 1

pred show_consideration_by_paragraph_when_auxiliaries_enabled {
        enable_auxiliary_atoms
        some ConsiderationByParagraph
}
run show_consideration_by_paragraph_when_auxiliaries_enabled  for 6
expect 1

pred Amend_applying_to_itself_when_auxiliaries_enabled {
        enable_auxiliary_atoms
        some amend : Amend | some amend.isAppliedTo & Amend
}
 run Amend_applying_to_itself_when_auxiliaries_enabled for 10 but 2 SessionState, 3 Business
 expect 1

pred Target_Amend_applying_to_Target_Amend_when_auxiliaries_enabled {
        enable_auxiliary_atoms
        some a1, a2 : Amend | a1.isAppliedTo = a2
                and hasTargets[a1] and hasTargets[a2]
                //and isMain[a1] and isMain[a2]
}
 run Target_Amend_applying_to_Target_Amend_when_auxiliaries_enabled for 12 but 2 SessionState,
3 Business
 expect 1

//------SHOULD-FAIL-----------//
pred SHOULD_FAIL {}

assert sessionStateElements_have_one_auxiliary_when_enabled {
        enable_auxiliary_atoms => all elem : sessionStateElements | one elem.allAux
}
check sessionStateElements_have_one_auxiliary_when_enabled for 6
expect 1

pred Amend_contained_in_its_own_context {
        some a : Amend | some Aux.contexts.a.a
}
run Amend_contained_in_its_own_context for 6
expect 0

pred an_element_pointing_to_itself_in_some_amend_context {
```

```
	some a : Amend, m : Motion, b : Business | m in m.(a.(Aux.contexts)) or b in b.(a.
(Aux.contexts))
}
run an_element_pointing_to_itself_in_some_amend_context for 5
expect 0

pred some_sessionState_element_that_is_an_auxiliary_atom {

	some elem : sessionStateElements | isAuxiliary[elem]

}
run some_sessionState_element_that_is_an_auxiliary_atom for 5 //, exactly 1 Member
expect 0

pred some_cycle_in_the_auxiliary_atoms_relation {

	some elem : Motion+Business | elem in elem.^allAux

}
run some_cycle_in_the_auxiliary_atoms_relation for 5 //, exactly 1 Member
expect 0

                                        /*****************************
                                        * CREATEBLANK TESTING *
                                        *****************************/
                                        pred CREATEBLANK_TESTING {}
                                        run CREATEBLANK_TESTING

pred show_createBlank {
	some CreateBlank
}
run show_createBlank for 5 //, exactly 1 Member
 expect 1

pred move_create_blank{

	some before, after :SessionState, motion : CreateBlank |
			move[before, after, motion]

 }
 run move_create_blank for 5 //, exactly 1 Member
 expect 1

                                        /*****************************************
                                        * CONSIDERBYPARAGRAPH TESTING *
                                        *****************************************/
                                        pred CONSIDERBYPARAGRAPH_TESTING {}
                                        run CONSIDERBYPARAGRAPH_TESTING
```

```
pred show_consideration_by_paragraph {
	some ConsiderationByParagraph
}
run show_consideration_by_paragraph for 5
expect 1

pred move_considerbyparagraph{

	some before, after :SessionState, motion : ConsiderationByParagraph |
			move[before, after, motion]

}
run move_considerbyparagraph for 5 //, exactly 1 Member
expect 1

/*****************************************
* DIVISIONOFQUESTION TESTING *
*****************************************/
pred DIVISIONOFQUESTION_TESTING {}
run DIVISIONOFQUESTION_TESTING

//------SHOULD-SUCCEED-----------//

pred show_divisionOfQuestion {

	some DivisionOfQuestion

}
run show_divisionOfQuestion for 6
expect 1

pred show_multiple_divisionOfQuestion {

	not lone DivisionOfQuestion and disable_auxiliary_atoms

}
run show_multiple_divisionOfQuestion for 6
expect 1

pred move_divisionOfQuestion {

	some before, after :SessionState, motion : DivisionOfQuestion|
			move[before, after, motion]

}
run move_divisionOfQuestion for 5 //, exactly 1 Member
expect 1

pred disposeOf_divisionOfQuestion {
```

```
	some before, after :SessionState, motion : DivisionOfQuestion|
		disposeOf[before, after, motion]

}
run disposeOf_divisionOfQuestion for 5 //, exactly 1 Member
expect 1

pred adopt_divisionOfQuestion {

	some before, after :SessionState, motion : DivisionOfQuestion|
		adopt[before, after, motion]

}
run adopt_divisionOfQuestion for 5 //, exactly 1 Member
expect 1

pred DivisionOfQuestion_tpoing_some_Amend {
		some doq : DivisionOfQuestion |
		some a : Amend |
			a in doq.takesOver

}run DivisionOfQuestion_tpoing_some_Amend for 5 but 1 SessionState
expect 1

pred adopt_divisionOfQuestion_with_separatedQuestions_that_yieldTo {
		some before, after :SessionState, motion : DivisionOfQuestion |
			adopt[before, after, motion] and
			some (motion.separatedQuestions-motion.isAppliedTo).^yieldsTo
}
run adopt_divisionOfQuestion_with_separatedQuestions_that_yieldTo for 5 //, exactly 1 Member
expect 1

pred adopt_divisionOfQuestion_with_separatedQuestions_that_yieldTo_other_divisionOfQuestion {
		some before, after :SessionState, motion : DivisionOfQuestion |
			adopt[before, after, motion] and
			some ((motion.separatedQuestions-
(motion.isAppliedTo)).^yieldsTo)&DivisionOfQuestion
}
run adopt_divisionOfQuestion_with_separatedQuestions_that_yieldTo_other_divisionOfQuestion for
7
expect 1

pred newly_separatedQuestion_yielding_to_divisionOfQuestion {
	some doq : DivisionOfQuestion | some ((doq.separatedQuestions-
(doq.isAppliedTo)).yieldsTo)&DivisionOfQuestion
}
run  newly_separatedQuestion_yielding_to_divisionOfQuestion for 6 but 4 SessionState expect 1
```

```
pred
adopt_divisionOfQuestion_with_separatedQuestions_that_yield_directly_To_other_divisionOfQuestio
n {
                some before, after :SessionState, motion : DivisionOfQuestion |
                        adopt[before, after, motion] and
                        some ((motion.separatedQuestions-
(motion.isAppliedTo)).yieldsTo)&DivisionOfQuestion
 }
 run
adopt_divisionOfQuestion_with_separatedQuestions_that_yield_directly_To_other_divisionOfQuestio
n for 7
 expect 1

//------SHOULD-FAIL-----------//

pred divisionOfQuestion_on_top_of_divisionOfquestion {
        some disj doq1, doq2 : DivisionOfQuestion |
                doq2 in doq1.^yieldsTo
}
run  divisionOfQuestion_on_top_of_divisionOfquestion for 5 but 1 SessionState
expect 0

pred division_of_question_with_overlappling_separated_questions {
        some disj dq1, dq2 : DivisionOfQuestion | some
(dq1.separatedQuestions&dq2.separatedQuestions)
}
run division_of_question_with_overlappling_separated_questions for 7
expect 0

                                        /*****************************************************
*********
                                         *       AMEND AND DIVISION OF THE QUESTION
TESTING    *

********************************************************** */
                                        pred
AMEND_AND_DIVISION_OF_THE_QUESTION_TESTING  {}
                                        run
AMEND_AND_DIVISION_OF_THE_QUESTION_TESTING

 pred DivisionOfQuestion_yielding_to_Amend_of_the_Root_and_applingto_the_Root {
        some motion : DivisionOfQuestion |
                (some amend : Amend |
                        amend in motion.yieldsTo
                        and some (amend.isAppliedTo & RootMotion))
                and some (motion.isAppliedTo & RootMotion)
 }

 pred DivisionOfQuesstion_yielding_to_amend_of_Root_and_applying_to_some_secondary {
```

```
        some motion : DivisionOfQuestion |
                (some amend : Amend |
                        amend in motion.yieldsTo
                        and some (amend.isAppliedTo & RootMotion))
                and some (motion.isAppliedTo & SecondaryMotion)

}

pred DivisionOfQuestion_yielding_to_some_amend_of_the_root_applying_to_this_amend {
                some motion : DivisionOfQuestion |
                (some amend : Amend |
                        amend in motion.yieldsTo
                        and some (amend.isAppliedTo & RootMotion)
                        and amend in motion.isAppliedTo)
}

pred DivisionOfQuestion_tpoing_some_Amend_of_root_and_applying_to_root {
                some motion : DivisionOfQuestion |
                (some amend : Amend |
                        amend in motion.takesOver
                        and some (amend.isAppliedTo & RootMotion))
                and some (motion.isAppliedTo & RootMotion)
}

pred DivisionOfQuestion_tpoing_some_Amend_of_root_and_applying_to_a_secondary {
                some motion : DivisionOfQuestion |
                (some amend : Amend |
                        amend in motion.takesOver
                        and some (amend.isAppliedTo & RootMotion))
                and some (motion.isAppliedTo & (SecondaryMotion-Amend) )
}

pred DivisionOfQuestion_tpoing_some_Amend_of_root_and_applying_to_this_amend {
                some motion : DivisionOfQuestion |
                (some amend : Amend |
                        amend in motion.takesOver
                        and some (amend.isAppliedTo & RootMotion)
                        and amend in motion.isAppliedTo)
}

pred DivisionOfQuestion_indirectly_above_some_amend_of_the_root_and_applying_to_the_root {
                some motion : DivisionOfQuestion |
                (some amend : Amend |
                        amend in (motion.takesOver.^takesOver)
                        and some (amend.isAppliedTo & RootMotion))
                and some (motion.isAppliedTo & RootMotion)
}

pred DivisionOfQuestion_indirectly_above_some_amend_of_the_root_and_apply_to_a_Secondary {
```

```
		some motion : DivisionOfQuestion |
		(some amend : Amend |
			amend in (motion.takesOver.^takesOver)
			and some (amend.isAppliedTo & RootMotion)
		and some (motion.isAppliedTo & SecondaryMotion-amend)) // check this syntax
}

pred DivisionOfQuestion_indirectly_above_some_amend_of_the_root_and_applying_to_this_amend
{
		some motion : DivisionOfQuestion |
		(some amend : Amend |
			amend in (motion.takesOver.^takesOver)
			and some (amend.isAppliedTo & RootMotion)
			and amend in motion.isAppliedTo)
}

pred DivisionOfQuestion_indirectly_below_some_amend_of_the_root_and_applying_to_the_root {
		some motion : DivisionOfQuestion |
		(some amend : Amend |
			amend in (motion.yieldsTo.^yieldsTo)
			and some (amend.isAppliedTo & RootMotion))
		and some (motion.isAppliedTo & RootMotion)
}

pred DivisionOfQuestion_indirectly_below_some_amend_of_the_root_and_applying_to_a_secondary
{
		some motion : DivisionOfQuestion |
		(some amend : Amend |
			amend in (motion.yieldsTo.^yieldsTo)
			and some (amend.isAppliedTo & RootMotion))
		and some (motion.isAppliedTo & SecondaryMotion)
}

pred DivisionOfQuestion_indirectly_below_some_amend_of_the_root_and_applying_to_this_amend
{
		some motion : DivisionOfQuestion |
		(some amend : Amend |
			amend in (motion.yieldsTo.^yieldsTo)
			and some (amend.isAppliedTo & RootMotion)
			and amend in motion.isAppliedTo)
}

/* *** */


pred DivisionOfQuestion_yielding_to_Amend_of_a_Secondary_and_applingto_the_Root {
	some motion : DivisionOfQuestion |
		(some amend : Amend |
			amend in motion.yieldsTo
```

```
                    and some (amend.isAppliedTo & SecondaryMotion))
            and some (motion.isAppliedTo & RootMotion)
}

pred DivisionOfQuesstion_yielding_to_amend_of_a_Secondary_and_applying_to_this_secondary {
    some motion : DivisionOfQuestion |
        (some amend : Amend |
            amend in motion.yieldsTo
            and (some sm : SecondaryMotion | sm in amend.isAppliedTo and sm in
motion.isAppliedTo)
        )
}

pred DivisionOfQuesstion_yielding_to_amend_of_a_Secondary_and_applying_to_another_secondary
{
    some motion : DivisionOfQuestion |
        (some amend : Amend |
            amend in motion.yieldsTo
            and some(amend.isAppliedTo & SecondaryMotion)
            and (some sm : SecondaryMotion | sm not in amend.isAppliedTo and sm in
motion.isAppliedTo)
        )
}

pred DivisionOfQuestion_yielding_to_some_amend_of_a_Secondary_applying_to_this_amend {
  some motion : DivisionOfQuestion |
        (some amend : Amend |
            amend in motion.yieldsTo
            and some(amend.isAppliedTo & SecondaryMotion)
            and amend in motion.isAppliedTo
        )
}

pred DivisionOfQuestion_tpoing_some_Amend_of_a_Secondary_and_applying_to_root {
    some motion : DivisionOfQuestion |
        (some amend : Amend |
            amend in motion.takesOver
            and some(amend.isAppliedTo & SecondaryMotion))
        and some (motion.isAppliedTo & RootMotion)
}

pred DivisionOfQuestion_tpoing_some_Amend_of_a_Secondary_and_applying_to_this_secondary {
    some motion : DivisionOfQuestion |
        (some amend : Amend |
            amend in motion.takesOver
            and (some sm : SecondaryMotion | sm in amend.isAppliedTo and sm in
motion.isAppliedTo))

}
```

```
 pred
DivisionOfQuestion_tpoing_some_Amend_of_a_Secondary_and_applying_to_another_secondary {
        some motion : DivisionOfQuestion |
                (some amend : Amend |
                        amend in motion.takesOver
                        and some(amend.isAppliedTo & SecondaryMotion)
                        and (some sm : SecondaryMotion | sm not in (amend.isAppliedTo+ amend) and
sm in motion.isAppliedTo)
                )
 }

 pred DivisionOfQuestion_tpoing_some_Amend_of_a_Secondary_and_applying_to_this_amend {
   some motion : DivisionOfQuestion |
                (some amend : Amend |
                        amend in motion.takesOver
                        and some(amend.isAppliedTo & SecondaryMotion)
                        and amend in motion.isAppliedTo
                )
 }

 pred DivisionOfQuestion_indirectly_above_some_amend_of_a_Secondary_and_applying_to_the_root
{
        some motion : DivisionOfQuestion |
                (some amend : Amend |
                        amend in (motion.takesOver.^takesOver)
                        and some (amend.isAppliedTo & SecondaryMotion))
                and some (motion.isAppliedTo & RootMotion)
 }

 pred
DivisionOfQuestion_indirectly_above_some_amend_of_a_Secondary_and_apply_to_this_Secondary {
                some motion : DivisionOfQuestion |
                (some amend : Amend |
                        amend in (motion.takesOver.^takesOver)
                        and (some sm : SecondaryMotion | sm in amend.isAppliedTo and sm in
motion.isAppliedTo))
 }

 pred
DivisionOfQuestion_indirectly_above_some_amend_of_a_Secondary_and_apply_to_another_Seconda
ry {
   some motion : DivisionOfQuestion |
                (some amend : Amend |
                        amend in motion.takesOver.^takesOver
                        and some(amend.isAppliedTo & SecondaryMotion)
                        and (some sm : SecondaryMotion | sm not in amend.isAppliedTo and sm in
motion.isAppliedTo)
                )
```

```
}

/**should fail*/
pred DivisionOfQuestion_indirectly_above_some_amend_of_a_Secondary_and_applying_to_this_amend {
	some motion : DivisionOfQuestion |
		(some amend : Amend |
			amend in (motion.takesOver.^takesOver)
			and some (amend.isAppliedTo & SecondaryMotion)
			and amend in motion.isAppliedTo)
}

pred DivisionOfQuestion_indirectly_below_some_amend_of_a_Secondary_and_applying_to_the_root
{
	some motion : DivisionOfQuestion |
		(some amend : Amend |
			amend in (motion.yieldsTo.^yieldsTo)
			and some (amend.isAppliedTo & SecondaryMotion))
		and some (motion.isAppliedTo & RootMotion)
}

pred DivisionOfQuestion_indirectly_below_some_amend_of_a_Secondary_and_applying_to_this_secondar
y {
	some motion : DivisionOfQuestion |
		(some amend : Amend |
			amend in (motion.yieldsTo.^yieldsTo)
			and (some sm : SecondaryMotion | sm in amend.isAppliedTo and sm in
motion.isAppliedTo))
}

pred DivisionOfQuestion_indirectly_below_some_amend_of_a_Secondary_and_applying_to_another_seco
ndary {
   some motion : DivisionOfQuestion |
		(some amend : Amend |
			amend in motion.yieldsTo.^yieldsTo
			and some(amend.isAppliedTo & SecondaryMotion)
			and (some sm : SecondaryMotion | sm not in amend.isAppliedTo and sm in
motion.isAppliedTo)
		)
}

pred DivisionOfQuestion_indirectly_below_some_amend_of_a_Secondary_and_applying_to_this_amend {
  some motion : DivisionOfQuestion |
		(some amend : Amend |
			amend in (motion.yieldsTo.^yieldsTo)
			and some (amend.isAppliedTo & SecondaryMotion)
```

```
                    and amend in motion.isAppliedTo)
}

 run DivisionOfQuestion_yielding_to_Amend_of_the_Root_and_applingto_the_Root  for 6 but 1
SessionState expect 0
 run DivisionOfQuesstion_yielding_to_amend_of_Root_and_applying_to_some_secondary  for 6 but 1
SessionState expect 0
 run DivisionOfQuestion_yielding_to_some_amend_of_the_root_applying_to_this_amend  for 6 but 1
SessionState expect 0
 run DivisionOfQuestion_tpoing_some_Amend_of_root_and_applying_to_root  for 6 but 1
SessionState expect 0
 run DivisionOfQuestion_tpoing_some_Amend_of_root_and_applying_to_a_secondary  for 6 but 1
SessionState expect 0
 run DivisionOfQuestion_tpoing_some_Amend_of_root_and_applying_to_this_amend  for 6 but 1
SessionState expect 0
 run DivisionOfQuestion_indirectly_above_some_amend_of_the_root_and_applying_to_the_root  for
6 but 1 SessionState expect 0
 run DivisionOfQuestion_indirectly_above_some_amend_of_the_root_and_apply_to_a_Secondary  for
6 but 1 SessionState expect 0
 run DivisionOfQuestion_indirectly_above_some_amend_of_the_root_and_applying_to_this_amend
for 6 but 1 SessionState expect 0
 run DivisionOfQuestion_indirectly_below_some_amend_of_the_root_and_applying_to_the_root  for
6 but 1 SessionState expect 0
 run DivisionOfQuestion_indirectly_below_some_amend_of_the_root_and_applying_to_a_secondary
for 6 but 1 SessionState expect 0
 run DivisionOfQuestion_indirectly_below_some_amend_of_the_root_and_applying_to_this_amend
for 6 but 1 SessionState expect 0
 run DivisionOfQuestion_yielding_to_Amend_of_a_Secondary_and_applingto_the_Root  for 6 but 1
SessionState expect 0
 run DivisionOfQuesstion_yielding_to_amend_of_a_Secondary_and_applying_to_this_secondary  for
6 but 1 SessionState expect 0
 run DivisionOfQuesstion_yielding_to_amend_of_a_Secondary_and_applying_to_another_secondary
for 6 but 1 SessionState expect 0
 run DivisionOfQuestion_yielding_to_some_amend_of_a_Secondary_applying_to_this_amend  for 6
but 1 SessionState expect 0
 run DivisionOfQuestion_tpoing_some_Amend_of_a_Secondary_and_applying_to_root  for 6 but 1
SessionState expect 0
 run DivisionOfQuestion_tpoing_some_Amend_of_a_Secondary_and_applying_to_this_secondary  for
6 but 1 SessionState expect 0
 run DivisionOfQuestion_tpoing_some_Amend_of_a_Secondary_and_applying_to_another_secondary
for 6 but 1 SessionState expect 0
 run DivisionOfQuestion_tpoing_some_Amend_of_a_Secondary_and_applying_to_this_amend  for 6
but 1 SessionState expect 1
 run DivisionOfQuestion_indirectly_above_some_amend_of_a_Secondary_and_applying_to_the_root
for 6 but 1 SessionState expect 0
 run
DivisionOfQuestion_indirectly_above_some_amend_of_a_Secondary_and_apply_to_this_Secondary
for 6 but 1 SessionState expect 0
 run
```

```
DivisionOfQuestion_indirectly_above_some_amend_of_a_Secondary_and_apply_to_another_Seconda
ry  for 6 but 1 SessionState expect 1
 run
DivisionOfQuestion_indirectly_above_some_amend_of_a_Secondary_and_applying_to_this_amend
for 6 but 1 SessionState expect 0
 run DivisionOfQuestion_indirectly_below_some_amend_of_a_Secondary_and_applying_to_the_root
for 6 but 1 SessionState expect 1
 run
DivisionOfQuestion_indirectly_below_some_amend_of_a_Secondary_and_applying_to_this_secondar
y  for 6 but 1 SessionState expect 0
 run
DivisionOfQuestion_indirectly_below_some_amend_of_a_Secondary_and_applying_to_another_seco
ndary  for 6 but 1 SessionState expect 1 //Maybe a new rule
 run
DivisionOfQuestion_indirectly_below_some_amend_of_a_Secondary_and_applying_to_this_amend
for 6 but 1 SessionState expect 0

module tests_model1_application

/*******************
 * Chose model  *
 *******************/
open model1
//open model0


                                        /*********************************
                                        *CAN BE ENABLED/DISABLED *
                                        *********************************/
                                        fact CAN_BE_ENABLED_DISABLED {
                                        //disable_auxiliary_atoms
                                        // enable_movedOnce
                                        // whole_session
                                        session_prefix
                                        // session_suffix
                                        // dynamic_mode_only
                                        // reach_all_motions
                                        no ErrorFlag
                                        }


/************************************************
                APPLICATION STRUCTURE TESTING
************************************************/
pred APPLICATION_STRUCTURE_TESTING {}

/*****SHOULD SUCCEED******/
pred SHOULD_SUCCEED{}
```

```
pred some_secondary_applied_to_a_set {
	some m : Motion | not lone m.isAppliedTo
}
run some_secondary_applied_to_a_set for 7 but 1 SessionState, exactly 1 Business expect 1

assert a_secondary_applies_to_a_set_of_consecutive_motions {
	 no Motion or (some motion : Motion | (not lone motion.isAppliedTo) =>
		all m : motion.isAppliedTo | //TODO : use let aSet = motion.isAppliedTo |
			//it is the head
			(motion.isAppliedTo)in (m+m.(^takesOver)) or
			//or it is the tail
			(motion.isAppliedTo) in (m+m.(^yieldsTo)) or
			//or it is in the middle (with no gaps around it)
			m.(takesOver + yieldsTo) in motion.isAppliedTo)
}
check a_secondary_applies_to_a_set_of_consecutive_motions for 7 but 1 SessionState, exactly 1
Business
expect 0

pred some_sm_applied_to_one_or_more_parents_in_the_precedence_structure {

	some sm : SecondaryMotion | sm.isAppliedTo in sm.^takesOver

}
run some_sm_applied_to_one_or_more_parents_in_the_precedence_structure for 10 but  exactly 1
Business expect 1

pred motion_applyingTo_unique_non_immediately_pending_question {
	some sm : SecondaryMotion | one sm.isAppliedTo and sm.isAppliedTo not in sm.takesOver
}
run motion_applyingTo_unique_non_immediately_pending_question for 5 but 7 Int
expect 1

assert only_IncidentalMotion_apply_to_motions_in_a_different_precedence_branch {

	all sm : SecondaryMotion | (some sm.isAppliedTo and some (sm.isAppliedTo - sm.^takesOver))

			=> sm in IncidentalMotion
}
check only_IncidentalMotion_apply_to_motions_in_a_different_precedence_branch for 10 but  exactly
1 Business
expect 0

/*pred motion_applyingTo_unique_non_immediately_pending_Amend {
	some sm : SecondaryMotion | one sm.isAppliedTo and

	sm.isAppliedTo not in sm.takesOver and

	some sm.isAppliedTo&Amend
```

```
}
run motion_applyingTo_unique_non_immediately_pending_Amend for 5 but 7 Int
expect 1
*/

/*****SHOULD FAIL******/
pred SHOULD_FAIL{}

pred some_secondary_motion_applying_to_a_child_in_the_precedence_tree {

        some m : SecondaryMotion | some m.isAppliedTo and m.isAppliedTo in m.^yieldsTo

}run some_secondary_motion_applying_to_a_child_in_the_precedence_tree for 7 but exactly 1
Business, exactly 1 SessionState
expect 0

//for 10 solves for more than 5mn

module tests_model1_domain_elements

open model1

                                        /*********************************
                                        *CAN BE ENABLED/DISABLED *
                                        *********************************/
                                        fact CAN_BE_ENABLED_DISABLED {
                                        disable_auxiliary_atoms
                                        // enable_movedOnce
                                        // whole_session
                                        session_prefix
                                        // session_suffix
                                        // dynamic_mode_only
                                        // reach_all_motions
                                        no ErrorFlag
                                        }


/************************************
 *        DOMAIN ELEMENTS TESTING  *
 ************************************/

/*********************************************
 ***** MOTIONS AND ACTIONS TESTS *****
 *********************************************/
pred MOTIONS_AND_ACTIONS_TESTS{}
run MOTIONS_AND_ACTIONS_TESTS{}

                                                        /**********************
                                                        SHOULD-SUCCEED
```

```
                                                    **********************/
                                                pred SHOULD_SUCCEED {}

 /*************************
        SUBSIDIARY_MOTION
  *************************/
 pred SUBSIDIARY_MOTIONS {}

pred move_PostponeIndefinitely {
        some before, after : SessionState, motion : PostponeIndefinitely| move[before, after, motion]
}run move_PostponeIndefinitely for 5 expect 1

pred move_AmendDebatable {
        some before, after : SessionState, motion : AmendDebatable | move[before, after, motion]
}run move_AmendDebatable for 5 expect 1

pred move_AmendUndebatable{
        some before, after : SessionState, motion : AmendUndebatable | move[before, after, motion]
}run move_AmendUndebatable for 8 expect 1

pred move_Commit{
        some before, after : SessionState, motion : Commit| move[before, after, motion]
}run move_Commit for 5 expect 1


pred move_Postpone {
        some before, after : SessionState, motion : Postpone| move[before, after, motion]
}run move_Postpone for 5 expect 1


pred move_LimitExtendDebate{
        some before, after : SessionState, motion : LimitExtendDebate| move[before, after, motion]
}run move_LimitExtendDebate for 5 expect 1


pred move_PreviousQuestion {
        some before, after : SessionState, motion : PreviousQuestion | move[before, after, motion]
}run move_PreviousQuestion for 5 expect 1


pred move_LayOnTheTable{
        some before, after : SessionState, motion : LayOnTheTable| move[before, after, motion]
}run move_LayOnTheTable for 5 expect 1

pred move_LayOnTheTable_in_whole_session{
        some before, after : SessionState, motion : LayOnTheTable| move[before, after, motion]
        and whole_session
}run move_LayOnTheTable_in_whole_session for 5 expect 1
```

```
//***//

pred adopt_PostponeIndefinitely {
	some before, after : SessionState, motion : PostponeIndefinitely | adopt[before, after, motion]
}run adopt_PostponeIndefinitely for 7 expect 1

pred adopt_AmendDebatable{
	some before, after : SessionState, motion : AmendDebatable| adopt[before, after, motion]
}run adopt_AmendDebatable for 7 expect 1


pred adopt_AmendUndebatable{
	some before, after : SessionState, motion : AmendUndebatable| adopt[before, after, motion]
}run adopt_AmendUndebatable for 8 expect 1


pred adopt_Commit{
	some before, after : SessionState, motion : Commit| adopt[before, after, motion]
}run adopt_Commit for 7 expect 1

pred adopt_Postpone {
	some before, after : SessionState, motion : Postpone| adopt[before, after, motion]
}run adopt_Postpone for 5 expect 1


pred adopt_LimitExtendDebate{
	some before, after : SessionState, motion : LimitExtendDebate | adopt[before, after, motion]
}run adopt_LimitExtendDebate for 7 expect 1


pred adopt_PreviousQuestion {
	some before, after : SessionState, motion : PreviousQuestion | adopt[before, after, motion]
}run adopt_PreviousQuestion for 7 expect 1


pred adopt_LayOnTheTable{
	some before, after : SessionState, motion : LayOnTheTable | adopt[before, after, motion]
}run adopt_LayOnTheTable for 6 but 7 SessionState expect 1

pred adopt_LayOnTheTable_in_whole_session{
	some before, after : SessionState, motion : LayOnTheTable | adopt[before, after, motion]
	and whole_session
}run adopt_LayOnTheTable_in_whole_session for 7  expect 1
//NOTE : for 7 => big solving time when no instance found

//***//

pred disposeOf_PostponeIndefinitely {
	some before, after : SessionState, motion : PostponeIndefinitely | disposeOf[before, after,
```

```
motion]
}run disposeOf_PostponeIndefinitely for 5 expect 1


pred disposeOf_AmendDebatable {
	some before, after : SessionState, motion : AmendDebatable | disposeOf[before, after, motion]
}run disposeOf_AmendDebatable for 5 expect 1


pred disposeOf_AmendUndebatable{
	some before, after : SessionState, motion : AmendUndebatable | disposeOf[before, after,
motion]
}run disposeOf_AmendUndebatable for 8 expect 1


pred disposeOf_Commit{
	some before, after : SessionState, motion : Commit | disposeOf[before, after, motion]
}run disposeOf_Commit for 5 expect 1


pred disposeOf_Postpone {
	some before, after : SessionState, motion : Postpone| disposeOf[before, after, motion]
}run disposeOf_Postpone for 5 expect 1


pred disposeOf_LimitExtendDebate{
	some before, after : SessionState, motion : LimitExtendDebate | disposeOf[before, after,
motion]
}run disposeOf_LimitExtendDebate for 5 expect 1


pred disposeOf_PreviousQuestion {
	some before, after : SessionState, motion : PreviousQuestion | disposeOf[before, after, motion]
}run disposeOf_PreviousQuestion for 5 expect 1


pred disposeOf_LayOnTheTable{
	some before, after : SessionState, motion : LayOnTheTable | disposeOf[before, after, motion]
}run disposeOf_LayOnTheTable for 5 expect 1

//pred disposeOf_LaidOnTheTable_secondaryMotion{
//	some before, after : SessionState, motion : SecondaryMotion |
//		disposeOf[before, after, motion] and motion in before.laidOnTheTable
//}run disposeOf_LaidOnTheTable_secondaryMotion for 7 expect 1

//pred disposeOf_LaidOnTheTable_rootMotion{
//	some before, after : SessionState, motion : RootMotion |
//		disposeOf[before, after, motion] and motion in before.laidOnTheTable
//}run disposeOf_LaidOnTheTable_rootMotion for 6 expect 1
```

```
//***//

pred lose_PostponeIndefinitely {
	some before, after : SessionState, motion : PostponeIndefinitely | not adopt[before, after, motion] and disposeOf[before, after, motion]
}run lose_PostponeIndefinitely for 5 expect 1


pred lose_AmendDebatable {
	some before, after : SessionState, motion : AmendDebatable | not adopt[before, after, motion] and disposeOf[before, after, motion]
}run lose_AmendDebatable for 5 expect 1


pred lose_AmendUndebatable{
	some before, after : SessionState, motion : AmendUndebatable | not adopt[before, after, motion] and disposeOf[before, after, motion]
}run lose_AmendUndebatable for 8 expect 1


pred lose_Commit{
	some before, after : SessionState, motion : Commit | not adopt[before, after, motion] and disposeOf[before, after, motion]
}run lose_Commit for 5 expect 1


pred lose_Postpone {
	some before, after : SessionState, motion : Postpone | not adopt[before, after, motion] and disposeOf[before, after, motion]
}run lose_Postpone for 5 expect 1


pred lose_LimitExtendDebate{
	some before, after : SessionState, motion : LimitExtendDebate | not adopt[before, after, motion] and disposeOf[before, after, motion]
}run lose_LimitExtendDebate for 5 expect 1


pred lose_PreviousQuestion {
	some before, after : SessionState, motion : PreviousQuestion | not adopt[before, after, motion] and disposeOf[before, after, motion]
}run lose_PreviousQuestion for 5 expect 1


pred lose_LayOnTheTable{
	some before, after : SessionState, motion : LayOnTheTable | not adopt[before, after, motion] and disposeOf[before, after, motion]
```

```
}run lose_LayOnTheTable for 5 expect 1

 /*************************
        PRIVILEGED-MOTIONS
  *************************/
pred PRIVILEGED_MOTIONS {}

pred move_CallForOrdersOfDay {
        some before, after : SessionState, motion : CallForOrdersOfDay | move[before, after, motion]
}run move_CallForOrdersOfDay for 5 expect 1

pred move_RaiseQuestionOfPrivilege{
        some before, after : SessionState, motion : RaiseQuestionOfPrivilege | move[before, after,
motion]
}run move_RaiseQuestionOfPrivilege for 5 expect 1


pred move_Recess{
        some before, after : SessionState, motion : Recess | move[before, after, motion]
}run move_Recess for 5 expect 1


pred move_Adjourn {
        some before, after : SessionState, motion : Adjourn | move[before, after, motion]
}run move_Adjourn for 5 expect 1


pred move_FixTimeAdjourn{
        some before, after : SessionState, motion : FixTimeAdjourn| move[before, after, motion]
}run move_FixTimeAdjourn for 5 expect 1

//***//


pred adopt_CallForOrdersOfDay {
        some before, after : SessionState, motion : CallForOrdersOfDay| adopt[before, after, motion]
}run adopt_CallForOrdersOfDay for 3 but /*exactly 2 RootMotion,*/ 5 SessionState expect 1

pred adopt_RaiseQuestionOfPrivilege{
        some before, after : SessionState, motion : RaiseQuestionOfPrivilege | adopt[before, after,
motion]
}run adopt_RaiseQuestionOfPrivilege for 7 expect 1

pred adopt_QuestionOfPrivilege{
        some before, after : SessionState, motion : RaiseQuestionOfPrivilege.questionOfPrivilege |
                adopt[before, after, motion]
}run adopt_QuestionOfPrivilege for 7 expect 1

pred adopt_Recess{
```

```
	some before, after : SessionState, motion : Recess | adopt[before, after, motion]
}run adopt_Recess for 7 expect 1


pred adopt_Adjourn {
	some before, after : SessionState, motion : Adjourn| adopt[before, after, motion]
}run adopt_Adjourn for 7 expect 1


pred adopt_FixTimeAdjourn{
	some before, after : SessionState, motion : FixTimeAdjourn | adopt[before, after, motion]
}run adopt_FixTimeAdjourn for 7 expect 1

//***//

pred disposeOf_CallForOrdersOfDay {
	some before, after : SessionState, motion : CallForOrdersOfDay | disposeOf[before, after,
motion]
}run disposeOf_CallForOrdersOfDay for 5 expect 1


pred disposeOf_RaiseQuestionOfPrivilege{
	some before, after : SessionState, motion : RaiseQuestionOfPrivilege | disposeOf[before, after,
motion]
}run disposeOf_RaiseQuestionOfPrivilege for 5 expect 1

pred disposeOf_QuestionOfPrivilege{
	some before, after : SessionState, motion : RaiseQuestionOfPrivilege.questionOfPrivilege |
 disposeOf[before, after, motion]
}run disposeOf_QuestionOfPrivilege for 5 expect 1


pred disposeOf_Recess{
	some before, after : SessionState, motion : Recess | disposeOf[before, after, motion]
}run disposeOf_Recess for 5 expect 1


pred disposeOf_Adjourn {
	some before, after : SessionState, motion : Adjourn | disposeOf[before, after, motion]
}run disposeOf_Adjourn for 5 expect 1


pred disposeOf_FixTimeAdjourn {
	some before, after : SessionState, motion : FixTimeAdjourn | disposeOf[before, after, motion]
}run disposeOf_FixTimeAdjourn for 5 expect 1

//***//

pred lose_CallForOrdersOfDay {
```

```
	some before, after : SessionState, motion : CallForOrdersOfDay | not adopt[before, after, motion] and disposeOf[before, after, motion]
}run lose_CallForOrdersOfDay for 5 expect 1


pred lose_RaiseQuestionOfPrivilege {
	some before, after : SessionState, motion : RaiseQuestionOfPrivilege | not adopt[before, after, motion] and disposeOf[before, after, motion]
}run lose_RaiseQuestionOfPrivilege for 5 expect 1


pred lose_Recess{
	some before, after : SessionState, motion : Recess| not adopt[before, after, motion] and disposeOf[before, after, motion]
}run lose_Recess for 5 expect 1


pred lose_Adjourn {
	some before, after : SessionState, motion : Adjourn| not adopt[before, after, motion] and disposeOf[before, after, motion]
}run lose_Adjourn for 5 expect 1


pred lose_FixTimeAdjourn{
	some before, after : SessionState, motion : FixTimeAdjourn| not adopt[before, after, motion] and disposeOf[before, after, motion]
}run lose_FixTimeAdjourn for 5 expect 1


pred move_Incidental {
	some before, after : SessionState, motion : IncidentalMotion | move[before, after, motion]
}run move_Incidental for 5 expect 1


 /*************************
	INCIDENTAL-MOTIONS
  *************************/
pred INCIDENTAL_MOTIONS{}

pred adopt_Incidental {
	some before, after : SessionState, motion : IncidentalMotion | adopt[before, after, motion]
}run adopt_Incidental for 8 expect 1


pred disposeOf_Incidental {
	some before, after : SessionState, motion : IncidentalMotion | disposeOf[before, after, motion]
}run disposeOf_Incidental for 5 expect 1
```

```
pred lose_Incidental {
        some before, after : SessionState, motion : IncidentalMotion | not adopt[before, after, motion]
and disposeOf[before, after, motion]
}run lose_Incidental for 5 expect 1

//***//
                                                                                    /*****************
                                                                                   SHOULD_FAIL

*****************/
                                                                                  pred SHOULD_SUCCEED
{}


 /*************************
        PRIVILEGED-MOTIONS
  *************************/
pred PRIVILEGED_MOTIONS {}

/************************************************
 ***** MISCELLANEOUS MOTION TESTS *****
************************************************/
pred MISCELLANEOUS_MOTION_TESTS{}
run MISCELLANEOUS_MOTION_TESTS{}


/**********************

        SHOULD_SUCCEED

**********************/
                                                                                  pred
SHOULD_SUCCEED{}

 /*************************
        SUBSIDIARY_MOTIONS
  *************************/
 pred SUBSIDIARY_MOTIONS {}

pred whole_session_with_non_empty_Table {
        whole_session// and
        some adopted.LayOnTheTable
}
run whole_session_with_non_empty_Table for 5
expect 1

pred show_the_PreviousQuestion {
        one pq : Motion| pq in PreviousQuestion
}
```

```
run show_the_PreviousQuestion for 5
expect 1

pred some_Amend_yielding_to_another_subsidiary_motion {
	some a : Amend | some (a.yieldsTo & SubsidiaryMotion)

}
run some_Amend_yielding_to_another_subsidiary_motion for 5 but exactly 1 Business
expect 1

pred moving_layOnTheTable_on_top_of_the_previousQuestion_and_at_least_2_secondaries {
	some lat : LayOnTheTable, pq : PreviousQuestion |
		pq in lat.takesOver and not lone (pq.isAppliedTo&SecondaryMotion)
}
run moving_layOnTheTable_on_top_of_the_previousQuestion_and_at_least_2_secondaries for 5 but
10 SessionState
expect 1

 /*************************
	PRIVILEGED-MOTIONS
  *************************/
pred PRIVILEGED_MOTIONS {}

pred show_CallForOrdersOfDay {
	some CallForOrdersOfDay
}
run  show_CallForOrdersOfDay for 5
expect 1

                                                                  /*****************
                                                                 SHOULD_FAIL

*****************/
                                                                        pred SHOULD_FAIL{}

 /*************************
	SUBSIDIARY_MOTIONS
  *************************/
pred some_Commit_Applied_to_non_consecutive_secondary_parent {
	some commit : Commit |
		no (commit.isAppliedTo & commit.takesOver) and
		no (commit.isAppliedTo & RootMotion)
}
run some_Commit_Applied_to_non_consecutive_secondary_parent for 7 but exactly 1 SessionState,
exactly 1 Business
expect 0

pred some_PreviousQuestion_Applied_to_non_consecutive_secondary_parent {
	some previousQuestion : PreviousQuestion |
```

```
                no (previousQuestion.isAppliedTo & previousQuestion.takesOver) and
                no (previousQuestion.isAppliedTo & RootMotion)
}
run some_PreviousQuestion_Applied_to_non_consecutive_secondary_parent for 7 but exactly 1
SessionState, exactly 1 Business
expect 0

 /*************************
        PRIVILEGED-MOTIONS
  *************************/

/*************************
 ***** BUSINESSES *****
*************************/
pred BUSINESSES {}
run BUSINESSES {}


/**********************

        SHOULD_SUCCEED

**********************/
                                                                pred
SHOULD_SUCCEED{}


                                                                    /******************
                                                                SHOULD_FAIL

******************/
                                                                pred SHOULD_FAIL{}

/***********************
 * SESSION STATES *
************************/
pred SESSION_STATES{}
run SESSION_STATES{}


/**********************

        SHOULD_SUCCEED

**********************/
                                                                pred
SHOULD_SUCCEED{}

assert all_sessionStates_have_a_next_but_the_last {
```

```
                some SessionState => (one ss : SessionState | no ss.next and
                        all ss : SessionState - ss | one ss.next )
}
check all_sessionStates_have_a_next_but_the_last for 5
expect 0

assert all_sessionStates_have_a_previous_but_the_first {
                some SessionState => (one ss : SessionState | no ss.prev and
                        all ss : SessionState - ss | one ss.prev )
}
check all_sessionStates_have_a_previous_but_the_first for 5
expect 0

assert
all_sessionStates_are_next_to_one_previous_sessionStates_and_previous_to_one_next_sessionStates {
        all ss1 : SessionState-firstSessionState | one next.ss1
        all ss2 : SessionState-lastSessionState | one prev.ss2
}
check
all_sessionStates_are_next_to_one_previous_sessionStates_and_previous_to_one_next_sessionStates
for 5
expect 0

                                                                /*****************
                                                               SHOULD_FAIL

*****************/
                                                                        pred SHOULD_FAIL{}

pred a_session_with_no_order_of_business {
        some ss : SessionState | no ss.orderOfBusiness
}
run a_session_with_no_order_of_business for 5  expect 0
```