APPENDIX

This appendix presents the complete ProMoca models of the example cases that we considered in our paper "ProMoca: Modeling and Analysis of Agent Behaviors in Commitment Protocols" that appears in Journal of Artificial Intelligence Research in 2016 (vol. 57, pages 465–508). Note that there are differences between the code in this appendix and the code segments that are presented in the paper, which are adjusted for better readability and presentation. The source files can also be accessed from `http://mas.cmpe.boun.edu.tr/akin/promoca`

The ProMoca model for the aerospace aftercare case for four months:

```
globals {

  variable contract : {'none', 'valid', 'invalid', 'completed'} = 'none';
  variable period : {'none', '1', '2', '3'} = 'none';
  variable engine_fee : {'pending', 'paid', 'not_paid'} = 'pending';
  variable engine_delivery : {'pending', 'done', 'not_done'} = 'pending';
  variable approved_supplier : {'1', '2'} = '1';
  variable order_1_* : {'pending', 'made', 'not_made'} = 'pending';
  variable order_2_* : {'pending', 'made', 'not_made'} = 'pending';
  variable parts_delivery_* : {'pending', 'on_time', 'late', 'failed'} = 'pending';
  variable service_fee_* : {'pending', 'paid', 'not_paid'} = 'pending';
  variable engine_report_* : {'pending', 'provided', 'not_provided'} = 'pending';
  variable engine_service_* : {'pending', 'on_time', 'late'} = 'pending';
  variable engine_refund : {'pending', 'paid', 'not_paid'} = 'pending';
  variable service_penalty_* : {'pending', 'paid', 'not_paid'} = 'pending';
  variable failure_penalty_* : {'pending', 'paid', 'not_paid'} = 'pending';
  variable engine_repair_* : {'pending', 'done', 'failed'} = 'pending';
  variable maintenance_period_* : {'pending', 'ended'} = 'pending';
  variable engine_status : {'inoperative', 'operational'} = 'operational';

}

protocol {

 commitment(
  'c1', achievement, 1,
  'manufacturer', 'operator',
  engine_fee == 'paid',
  engine_fee == 'not_paid',
  engine_delivery == 'done',
  engine_delivery == 'not_done',
 ) {'aerospace_agency'}['c2'];

 commitment(
  'c2', achievement, 1,
  'manufacturer', 'operator',
  c1_status == 'violated',
  c1_status == 'fulfilled' or c1_status == 'released' or c1_status == 'expired',
  engine_refund == 'paid',
  engine_refund == 'not_paid'
 ) {'aerospace_agency'};

 commitment(
  'c3_*', achievement, 3,
  'manufacturer', 'operator',
  service_fee_* == 'paid' and engine_report_* == 'provided',
  service_fee_* == 'not_paid' or engine_report_* == 'not_provided',
  engine_service_* == 'on_time',
  engine_service_* == 'late',
 ) {'aerospace_agency'}['c4_*'];

 commitment(
  'c4_*', achievement, 3,
  'manufacturer', 'operator',
  c3_*_status == 'violated',
  c3_*_status == 'fulfilled' or c3_*_status == 'released' or c3_*_status == 'expired',
  service_penalty_* == 'paid',
  service_penalty_* == 'not_paid'
 ) {'aerospace_agency'};

 commitment(
  'c5_*', maintenance, 3,
  'manufacturer', 'operator',
  service_fee_* == 'paid' and engine_report_* == 'provided',
```

```
  service_fee_* == 'not_done' or engine_report_* == 'not_provided',
  engine_status_* == 'operational',
  maintenance_period_* == 'ended',
 ) {'aerospace_agency'}['c6_*'];

 commitment(
  'c6_*', achievement, 3,
  'manufacturer', 'operator',
  c5_status_* == 'violated',
  c5_status_* == 'fulfilled' or c5_status_* == 'released' or c5_status_* == 'expired',
  failure_penalty_* == 'paid' and engine_status_* == 'operational',
  faliure_penalty_* == 'not_paid' or engine_repair == 'failed',
 ) {'aerospace_agency'};

 commitment(
  'c7_*', achievement, 3,
  'supplier_1', 'manufacturer',
  order_1_* == 'made',
  order_1_* == 'not_made',
  parts_delivery_* == 'on_time' or parts_delivery_* == 'late',
  parts_delivery_* == 'failed'
 ) {'aerospace_agency'};

 commitment(
  'c8_*', achievement, 3,
  'supplier_2', 'manufacturer',
  order_2_* == 'made',
  order_2_* == 'not_made',
  parts_delivery_* == 'on_time' or parts_delivery_* == 'late',
  parts_delivery_* == 'failed'
 ) {'aerospace_agency'};

}

agent['manufacturer'] {

 locals {}

 goals {
  pagoal(engine_fee == 'paid' and service_fee_1 == 'paid' and
      service_fee_2 == 'paid' and service_fee_3 == 'paid');
 }

 behavior {
  // commit to c1 and c2 to initiate the purchase contract
  [c1_status == 'null' and c2_status == 'null'] {
      commit{'c1'} -> commit{'c2'} -> cont
  } <>
  // if operator detaches c1, make the maintenance contract valid
  [c1_status == 'active'] {
      deliver_engine{engine_delivery == 'done'} ->
      create_contract{contract == 'valid'} ->
      next_period{period = 1} -> cont
  } <>
  // if operator does not pay, stop interacting
  [c1_status == 'expired'] {
      stop
  } <>

  // FIRST PERIOD
  // when maintenance contract is initiated and suppliers are (conditionally)
  // committed to provide the parts, commit to the first period of maintenance
  [contract == 'valid' and period == '1' and c3_1_status == 'null' and c4_1_status == 'null' and
   c5_1_status == 'null' and c6_1_status == 'null' and c7_1_status == 'conditional' and
   c8_1_status == 'conditional'] {
      commit{'c3_1'} -> commit{'c4_1'} -> commit{'c5_1'} -> commit{'c6_1'} -> cont
  } <>
  // if the operator detaches c3_1 and c5_1
  [c3_1_status == 'active' and c5_1_status == 'active'] {
      // if the first supplier is approved, order the parts
      [approved_supplier == '1' and c7_1_status == 'conditional'] {
          make_order_1_1{order_1_1 = 'made'} -> release{'c8_1'} -> cont
      } <>
      // if the second supplier is approved, order the parts
      [approved_supplier == '2' and c8_1_status == 'conditional'] {
          make_order_2_1{order_2_1 = 'made'} -> release{'c7_1'} -> cont
      } <>
      // if parts are delivered on time, probability of on time service is 0.99
      [parts_delivery_1 == 'on_time'] {
          (0.99) on_time_service{engine_service_1 == 'on_time'} -> cont
          (0.01) late_service{engine_service_1 == 'late'} -> cont
```

```
    } <>
    // if parts are delivered late, probability of on time service is 0.75
    [parts_delivery_1 == 'late'] {
        (0.75) on_time_service{engine_service_1 == 'on_time'} -> cont
        (0.25) late_service{engine_service_1 == 'late'} -> cont
    }
} <>
// if failed to complete service on time, pay the penalty
[c4_1_status == 'active'] {
    pay_service_penalty{service_penalty_1 = 'paid'} -> cont
} <>
// if engine failure occurs
[c6_1_status == 'active'] {
    pay_failure_penalty{failure_penalty_1 = 'paid'} ->
    // there is a small probability of failing to repair the inoperative engine,
    // which leads to contract termination
    (0.99) repair_engine{engine_repair_1 == 'done'} -> cont
    (0.01) fail_repair_engine{engine_repair_1 == 'failed'} -> stop
} <>
// end of period, continue to the next one
[(c3_1_status == 'fulfilled' or c3_1_status == 'compensated' or c3_1_status == 'expired') and
 (c5_1_status == 'fulfilled' or c5_1_status == 'compensated' or c5_1_status == 'expired')] {
    next_period{period = '2'} -> cont
} <>
// SECOND PERIOD, same as above using corresponding variables for the second period
[contract == 'valid' and period == '2' and c3_2_status == 'null' and c4_2_status == 'null' and
 c5_2_status == 'null' and c6_2_status == 'null' and c7_2_status == 'conditional' and
 c8_2_status == 'conditional'] {
    commit{'c3_2'} -> commit{'c4_2'} -> commit{'c5_2'} -> commit{'c6_2'} -> cont
} <>
[c3_2_status == 'active' and c5_2_status == 'active'] {
    [approved_supplier == '1' and c7_2_status == 'conditional'] {
        make_order_1_2{order_1_2 = 'made'} -> release{'c8_2'} -> cont
    } <>
    [approved_supplier == '2' and c8_2_status == 'conditional'] {
        make_order_2_2{order_2_2 = 'made'} -> release{'c7_2'} -> cont
    } <>
    [parts_delivery_2 == 'on_time'] {
        (0.99) on_time_service{engine_service_2 == 'on_time'} -> cont
        (0.01) late_service{engine_service_2 == 'late'} -> cont
    } <>
    [parts_delivery_2 == 'late'] {
        (0.75) on_time_service{engine_service_2 == 'on_time'} -> cont
        (0.25) late_service{engine_service_2 == 'late'} -> cont
    }
} <>
[c4_2_status == 'active'] {
    pay_service_penalty{service_penalty_2 = 'paid'} -> cont
} <>
[c6_2_status == 'active'] {
    pay_failure_penalty{failure_penalty_2 = 'paid'} ->
    (0.99) repair_engine{engine_repair_2 == 'done'} -> cont
    (0.01) fail_repair_engine{engine_repair_2 == 'failed'} -> stop
} <>
[(c3_2_status == 'fulfilled' or c3_2_status == 'compensated' orc3_2_status == 'expired') and
 (c5_2_status == 'fulfilled' or c5_2_status == 'compensated' or c5_2_status == 'expired')] {
    next_period{period = '3'} -> cont
} <>
// THIRD PERIOD, same as above using corresponding variables for the third period
[contract == 'valid' and period == '3' and c3_3_status == 'null' and c4_3_status == 'null' and
 c5_3_status == 'null' and c6_3_status == 'null' and c7_3_status == 'conditional' and
 c8_3_status == 'conditional'] {
    commit{'c3_3'} -> commit{'c4_3'} -> commit{'c5_3'} -> commit{'c6_3'} -> cont
} <>
[c3_3_status == 'active' and c5_3_status == 'active'] {
    [approved_supplier == '1' and c7_3_status == 'conditional'] {
        make_order_1_3{order_1_3 = 'made'} -> release{'c8_3'} -> cont
    } <>
    [approved_supplier == '2' and c8_3_status == 'conditional'] {
        make_order_2_3{order_2_3 = 'made'} -> release{'c7_3'} -> cont
    } <>
    [parts_delivery_3 == 'on_time'] {
        (0.99) on_time_service{engine_service_3 == 'on_time'} -> cont
        (0.01) late_service{engine_service_3 == 'late'} -> cont
    } <>
    [parts_delivery_3 == 'late'] {
        (0.75) on_time_service{engine_service_3 == 'on_time'} -> cont
        (0.25) late_service{engine_service_3 == 'late'} -> cont
    }
} <>
[c4_3_status == 'active'] {
```

```
      pay_service_penalty{service_penalty_3 = 'paid'} -> cont
 } <>
 [c6_3_status == 'active'] {
      pay_failure_penalty{failure_penalty_3 = 'paid'} ->
      (0.99) repair_engine{engine_repair_3 == 'done'} -> cont
      (0.01) fail_repair_engine{engine_repair_3 == 'failed'} -> stop
 } <>
 [(c3_3_status == 'fulfilled' or c3_3_status == 'compensated' or c3_3_status == 'expired') and
  (c5_3_status == 'fulfilled' or c5_3_status == 'compensated' or c5_3_status == 'expired')] {
      next_period{period = 'none'} -> cont
 } <>
 // conclude the maintenance contract
 [contract == 'valid' and next_period == 'none'] {
      conclude_contract{contract = 'completed'} -> stop
 }

}

beliefs {
 // manufacturer's beliefs about the operator's beliefs
 ['operator'] {
   // the operator detaches condition c1 with 0.99 probability
   [c1_status == 'conditional'] {
       (0.99) pay_engine_fee{engine_fee = 'paid'} -> cont
       (0.01) not_pay_engine_fee{engine_fee = 'not_paid'} -> stop
   } <>
   // for all periods the operator detaches commitments c3_* and c5_* with
   // probability 0.95, but in rare cases the operator might fail to pay for
   // the service and/or provide the required engine report
   // FIRST PERIOD
   [c3_1_status == 'conditional' and c5_1_status == 'conditional'] {
       (0.95) pay_service_fee{service_fee_1 = 'paid'} ->
              provide_engine_report{engine_report_1 = 'provided'} -> cont
       (0.02) not_pay_service_fee{service_fee_1 = 'not_paid'} ->
              provide_engine_report{engine_report_1 = 'provided'} -> cont
       (0.02) pay_service_fee{service_fee_1 = 'paid'} ->
              not_provide_engine_report{engine_report_1 = 'not_provided'} -> cont
       (0.01) not_pay_service_fee{service_fee_1 = 'not-paid'} ->
              not_provide_engine_report{engine_report_1 = 'not_provided'} -> cont
   } <>
   // SECOND PERIOD
   [c3_2_status == 'conditional' and c5_2_status == 'conditional'] {
       (0.95) pay_service_fee{service_fee_2 = 'paid'} ->
              provide_engine_report{engine_report_2 = 'provided'} -> cont
       (0.02) not_pay_service_fee{service_fee_2 = 'not_paid'} ->
              provide_engine_report{engine_report_2 = 'provided'} -> cont
       (0.02) pay_service_fee{service_fee_2 = 'paid'} ->
              not_provide_engine_report{engine_report_2 = 'not_provided'} -> cont
       (0.01) not_pay_service_fee{service_fee_2 = 'not-paid'} ->
               not_provide_engine_report{engine_report_2 = 'not_provided'} -> cont
   } <>
   // THIRD PERIOD
   [c3_1_status == 'conditional' and c5_1_status == 'conditional'] {
       (0.95) pay_service_fee{service_fee_1 = 'paid'} ->
              provide_engine_report{engine_report_2 = 'provided'} -> cont
       (0.02) not_pay_service_fee{service_fee_1 = 'not_paid'} ->
              provide_engine_report{engine_report_2 = 'provided'} -> stop
       (0.02) pay_service_fee{service_fee_1 = 'paid'} ->
              not_provide_engine_report{engine_report_2 = 'not_provided'} -> stop
       (0.01) not_pay_service_fee{service_fee_1 = 'not-paid'} ->
              not_provide_engine_report{engine_report_2 = 'not_provided'} -> stop
   } <>
   // if the engine fails and the situation is not compensated by repairing
   // the engine, operator terminates the contract
   [c6_1_status == 'violated' or c6_2_status == 'violated' or c6_3_status == 'violated'] {
       terminate_contract{contract = 'invalid'} -> stop
   } <>
   // stop when the contract is completed
   [contract == 'completed')] {
       stop
   }
 };

 // manufacturer's beliefs about the first supplier
 ['supplier_1'] {
   [period == '1' and c7_1_status == 'null'] {
       commit('c7_1') -> cont
   } <>
   [c7_1_status == 'active'] {
       (0.95) deliver{parts_delivery_1 = 'on_time'} -> cont
       (0.05) not_deliver{parts_delivery_1 = 'late'} -> cont
```

```
    } <>
    [period == '2' and c7_2_status == 'null'] {
        commit('c7_2') -> cont
    } <>
    [c7_2_status == 'active'] {
        (0.9) deliver{parts_delivery_2 = 'on_time'} -> cont
        (0.1) not_deliver{parts_delivery_2 = 'late'} -> cont
    } <>
    [period == '3' and c7_3_status == 'null'] {
        commit('c7_3') -> cont
    } <>
    [c7_3_status == 'active'] {
        (0.9) deliver{parts_delivery_3 = 'on_time'} -> stop
        (0.1) not_deliver{parts_delivery_3 = 'late'} -> stop
    }
  };

  // manufacturer's beliefs about the second supplier
  ['supplier_8'] {
    [period == '1' and c8_1_status == 'null'] {
        commit('c8_1') -> cont
    } <>
    [c8_1_status == 'active'] {
        (0.9) deliver{parts_delivery_1 = 'on_time'} -> cont
        (0.1) not_deliver{parts_delivery_1 = 'late'} -> cont
    } <>
    [period == '2' and c8_2_status == 'null'] {
        commit('c8_2') -> cont
    } <>
    [c8_2_status == 'active'] {
        (0.85) deliver{parts_delivery_2 = 'on_time'} -> cont
        (0.15) not_deliver{parts_delivery_2 = 'late'} -> cont
    } <>
    [period == '3' and c8_3_status == 'null'] {
        commit('c8_3') -> cont
    } <>
    [c8_3_status == 'active'] {
        (0.95) deliver{parts_delivery_3 = 'on_time'} -> stop
        (0.05) not_deliver{parts_delivery_3 = 'late'} -> stop
    }
  };

  // manufacturer's beliefs about engine failures
  ['engine'] {
    // PERIOD 1, engine does not fail no matter it is serviced on time or late
    [c5_1_status == 'active'] {
        no_failure{maintenance_period_1 = 'ended'} -> cont
    } <>
    // PERIOD 2, engine might fail if it is serviced late
    [c5_2_status == 'active'] {
        [engine_service_2 == 'on_time'] {
            no_failure{maintenance_period_2 = 'ended'} -> cont
        } <>
        [engine_service_2 == 'late'] {
            (0.99) no_failure{maintenance_period_2 = 'ended'} -> cont
            (0.01) failure{engine_status = 'inoperative'} -> cont
        }
    } <>
    // PERIOD 3, engine might fail with a higher probability than the second period
    // if it is serviced late in the third period
    [c5_3_status == 'active'] {
        [engine_service_3 == 'on_time'] {
            no_failure{maintenance_period_3 = 'ended'} -> stop
        } <>
        [engine_service_3 == 'late'] {
            (0.97) no_failure{maintenance_period_3 = 'ended'} -> stop
            (0.03) failure{engine_status_3 = 'inoperative'} -> stop
        }
    }
  };

}
```

The ProMoca model for the NetBill case with two simultaneous customers:

```
// PROMOCA model of the Netbill protocol

// Global variables and their domains
globals {
 variable offer_1 : {'none', 'made', 'accepted', 'expired'} = 'none'
 variable offer_2 : {'none', 'made', 'accepted', 'expired'} = 'none'
 variable goods_1 : {'pending', 'delivered', 'not_delivered'} = 'pending';
 variable goods_2 : {'pending', 'delivered', 'not_delivered'} = 'pending';
 variable fee_1 : {'pending', 'paid', 'not_paid'} = 'pending';
 variable fee_2 : {'pending', 'paid', 'not_paid'} = 'pending';
 variable receipt_1 : {'pending', 'sent', 'not_sent'} = 'pending';
 variable receipt_2 : {'pending', 'sent', 'not_sent'} = 'pending';
}

// Commitment protocol
protocol {

 commitment(
  'c1_1', achievement, 1, 'customer_1', 'merchant',
  goods_1 == 'delivered', goods_1 == 'not_delivered',
  fee_1 == 'paid', fee_1 == 'not_paid'
 ) {'intermediation_server'};

 commitment(
  'c2_1', achievement, 1, 'merchant', 'customer_1',
  c1_1_status == 'conditional', offer_1 == 'expired',
  goods_1 == 'delivered', goods_1 == 'not_delivered',
 ) {'intermediation_server'};

 commitment(
  'c3_1', achievement, 1, 'merchant', 'customer_1',
  fee_1 == 'paid', fee_1 == 'not_paid',
  receipt_1 == 'sent', receipt_1 == 'not_sent',
 ) {'intermediation_server'};

 commitment(
  'c1_2', achievement, 1, 'customer_2', 'merchant',
  goods_2 == 'delivered', goods_2 == 'not_delivered',
  fee_2 == 'paid', fee_2 == 'not_paid'
 ) {'intermediation_server'};

 commitment(
  'c2_2', achievement, 1, 'merchant', 'customer_2',
  c1_2_status == 'conditional', offer_2 == 'expired',
  goods_2 == 'delivered', goods_2 == 'not_delivered',
 ) {'intermediation_server'};

 commitment(
  'c3_2', achievement, 1, 'merchant', 'customer_2',
  fee_2 == 'paid', fee_2 == 'not_paid',
  receipt_2 == 'sent', receipt_2 == 'not_sent',
 ) {'intermediation_server'};

}

// Definition of the merchant
agent['merchant'] {

 // Local variables and their domains for the merchant
 locals {
  variable stock : {'not_available', 'available'} = 'available';
  variable transaction_1 : {'pending', 'done'} = 'pending';
  variable transaction_2 : {'pending', 'done'} = 'pending';
 }

 // Goals of the merchant
 goals {
  pagoal(fee_1 == 'paid');
  pagoal(fee_2 == 'paid');
  agoal(c2_1_status == 'active', goods_1 == 'delivered', goods_1 == 'not_delivered');
  agoal(c2_2_status == 'active', goods_2 == 'delivered', goods_2 == 'not_delivered');
  agoal(c3_1_status == 'active', receipt_1 == 'sent', receipt_1 == 'not_sent');
  agoal(c3_2_status == 'active', receipt_2 == 'sent', receipt_2 == 'not_sent');
 }

 // Behavior of the merchant
```

```
behavior {
 [stock == 'not_available'] {
     request_goods{goods = 'available'} -> cont
 } <>
 [offer_1 == 'none' and stock == 'available'] {
     commit{'c2_1'} -> commit{'c3_1'} -> make_offer{offer_1 = 'made'} -> cont
 } <>
 [offer_2 == 'none' and stock == 'available'] {
     commit{'c2_2'} -> commit{'c3_2'} -> make_offer{offer_2 = 'made'} -> cont
 } <>
 [c2_1_status == 'expired'] {
     end{transaction_1 = 'done'} -> cont
 } <>
 [c2_2_status == 'expired'] {
     end{transaction_2 = 'done'} -> cont
 } <>
 [c2_1_status == 'active' and stock == 'available'] {
     (0.95) make_delivery{goods_1 = 'delivered'} ->
            (stock_depleted{stock = 'not_available'} -> cont <> cont)
     (0.05) fail_delivery{goods_1 = 'not_delivered'} -> end{transaction_1 = 'done'} -> cont
 } <>
 [c2_2_status == 'active' and stock == 'available'] {
     (0.95) make_delivery{goods_2 = 'delivered'} ->
            (stock_depleted{stock = 'not_available'} -> cont <> cont)
     (0.05) fail_delivery{goods_2 = 'not_delivered'} -> end{transaction_2 = 'done'} -> cont
 } <>
 [c2_1_status == 'fulfilled' and c3_1_status == 'active'] {
     send_receipt{receipt_1 = 'sent'} -> end{transaction_1 = 'done'} -> cont
 } <>
 [c2_2_status == 'fulfilled' and c3_2_status == 'active'] {
     send_receipt{receipt_2 = 'sent'} -> end{transaction_2 = 'done'} -> cont
 } <>
 [transaction_1 == 'done' and transaction_2 == 'done'] {
     stop
 }

}

// Beliefs of the merchant
beliefs {
 // The merchant's beliefs about the first customer's behavior.
 ['customer_1'] {
     [offer_1 == 'made'] {
         (0.6) commit{'c1_1'} -> accept{order_1 = 'accepted'} -> cont
         (0.4) reject{order_1 = 'expired'} -> release{'c3_1'} -> stop
     } <>
     [c1_1_status == 'active'] {
         (0.99) pay{fee_1 = 'paid'} -> cont
         (0.01) fail_pay{fee_1 = 'not_paid'} -> stop
     } <>
     [c1_1_status == 'expired'] {
         release{'c3_1'} -> stop
     }
     [receipt_1 == 'sent' or c1_1_status == 'expired'] {
         stop
     }
 };

 // The merchant's beliefs about the second customer's behavior.
 ['customer_2'] {
     [offer_2 == 'made'] {
         (0.9) commit{'c1_2'} -> accept{order_2 = 'accepted'} -> cont
         (0.1) reject{order_2 = 'expired'} -> release{'c3_2'} -> stop
     } <>
     [c1_2_status == 'active'] {
         (0.75) pay{fee_2 = 'paid'} -> cont
         (0.25) fail_pay{fee_2 = 'not_paid'} -> stop
     } <>
     [c1_2_status == 'expired'] {
         release{'c3_2'} -> stop
     }
     [receipt_2 == 'sent'] {
         stop
     }
 };

}
```

The ProMoca model for the AGFIL insurance case with two sequential claims:

```
// PROMOCA model of the AGFIL example

globals {

 variable policy_status : {'none', 'valid', 'expired'} = 'none';
 variable policy_fee : {'pending', 'paid', 'not_paid'} = 'pending';
 variable policy_creation : {'pending', 'done', 'failed'} = 'pending';
 variable claim_status_1 : {'pending', 'made', 'not_made'} = 'pending';
 variable claim_status_2 : {'pending', 'made', 'not_made'} = 'pending';
 variable claim_handling_1 : {'pending', 'done', 'failed'} = 'pending';
 variable claim_handling_2 : {'pending', 'done', 'failed'} = 'pending';
 variable claim_validity_1 : {'unassigned', 'valid', 'invalid'} = 'unassigned';
 variable claim_validity_2 : {'unassigned', 'valid', 'invalid'} = 'unassigned';
 variable validation_request_1 : {'pending', 'made', 'not_made'} = 'pending';
 variable validation_1 : {'pending', 'made', 'not_made'} = 'pending';
 variable payment_1 : {'pending', 'paid', 'not_paid'} = 'pending';
 variable payment_2 : {'pending', 'paid', 'not_paid'} = 'pending';

}

protocol {

 commitment(
  'c1', achievement, 1, 'agfil' 'customer',
  policy_fee == 'paid',
  policy_fee == 'not_paid',
  c2_status == 'conditional' and c3_status == 'conditional' and
  c4_status == 'conditional' and c5_status == 'conditional',
  policy_creation == 'failed'
 );

 commitment(
  'c2', achievement, 1, 'agfil', 'customer',
  claim_status_1 == 'made', policy_status == 'expired',
  claim_handling_1 == 'done', claim_handling_1 == 'failed'
 );

 commitment(
  'c3', achievement, 1, 'agfil', 'customer',
  claim_status_2 == 'made', policy_status == 'expired',
  claim_handling_2 == 'done', claim_handling_2 == 'failed'
 );

 commitment(
  'c4', achievement, 1, 'agfil', 'customer',
  claim_status_1 == 'made' and claim_validity_1 == 'valid',
  (claim_status_1 == 'made' and claim_validity_1 == 'invalid') or policy_status == 'expired',
  payment_1 == 'made', payment_1 == 'failed'
 );

 commitment(
  'c5', achievement, 1, 'agfil', 'customer',
  claim_status_2 == 'made' and claim_validity_2 == 'valid',
  (claim_status_2 == 'made' and claim_validity_2 == 'invalid') or policy_status == 'expired',
  payment_2 == 'made', payment_2 == 'failed'
 );

 commitment(
  'c6', achievement, 1, 'euro', 'agfil',
  claim_status_1 == 'made', policy_status == 'expired',
  claim_handling_1 == 'done', claim_handling_1 == 'failed'
 );

 commitment(
  'c7', achievement, 1, 'euro', 'agfil',
  claim_status_2 == 'made', policy_status == 'expired',
  claim_handling_2 == 'done', claim_handling_2 == 'failed'
 );

 commitment(
  'c8', achievement, 1, 'consultant', 'agfil',
  validation_request_1 == 'made', policy_status == 'expired',
  validation_1 == 'made', validation_1 == 'failed'
 );

 commitment(
```

```
  'c9', achievement, 1, 'consultant', 'agfil',
  validation_request_2 == 'made', policy_status == 'expired',
  validation_2 == 'made', validation_2 == 'failed'
 );

}

agent['agfil'] {

 locals {}

 goals {
   agoal(claim_status_1 == 'made', claim_handling_1 == 'made', claim_handling_1 == 'failed');
   agoal(claim_status_2 == 'made', claim_handling_2 == 'made', claim_handling_2 == 'failed');
 }

 behavior {
    [c1_status == 'active'] {
        commit{'c2'} -> commit{'c3'} -> commit{'c4'} -> commit{'c5'} ->
        create_policy{policy_status = 'valid'; policy_creation = 'done' -> cont
    } <>
    [c4_status == 'active' or (c4_status == 'conditional' and validation_1 == 'failed')] {
        pay{payment_1 = 'made'} -> cont
    } <>
    [c5_status == 'active' or (c5_status == 'conditional' and validation_2 == 'failed')] {
        pay{payment_2 = 'made'} -> policy_expires{policy_status = 'expired'} -> cont
    } <>
    [c8_status == 'conditional'] {
        request_validation{validation_request_1 = 'made'} -> cont
    } <>
    [c9_status == 'conditional'] {
        request_validation{validation_request_2 = 'made'} -> cont
    } <>
    [policy_status == 'expired'] {
       stop
    }
 }

 beliefs {

  ['customer'] {
      [c1_status == 'conditional'] {
          (0.99) pay{policy_fee = 'paid'} -> cont
          (0.01) fail_pay{policy_fee = 'not_paid'; policy_status = 'expired'} -> stop
      } <>
      [policy_status = 'valid' and claim_status_1 == 'pending'] {
          (0.5) claim{claim_status_1 = 'made'} -> cont
          (0.5) no_more_claims{policy_status = 'expired'} -> stop
      } <>
      [policy_status = 'valid' and claim_status_1 == 'made' and claim_status_2 == 'pending'] {
          (0.25) claim{claim_status_2 = 'made'} -> cont
          (0.75) no_more_claims{policy_status = 'expired'} -> stop
      } <>
      [policy_status == 'expired'] {
          stop
      }
  };

  ['euro'] {
      [c6_status == 'active'] {
          (0.95) handle_claim{claim_handling_1 = 'done'} -> cont
          (0.05) unavailable{claim_handling_1 = 'failed'} -> cont
      } <>
      [c6_status == 'active'] {
          (0.95) handle_claim{claim_handling_2 = 'done'} -> cont
          (0.05) unavailable{claim_handling_2 = 'failed'} -> cont
      } <>
      [policy_status == 'expired'] {
          stop
      }
  };

  ['consultant'] {
      [c8_status == 'active'] {
          (0.99) validate{validation_1 = 'made'} ->
              (0.8) claim_valid{claim_validity_1 = 'valid'} -> cont
              (0.2) claim_invalid{claim_validity_1 = 'invalid'} -> cont
          (0.01) fail_to_validate{validation_1 = 'failed'} -> cont
      } <>
      [c9_status == 'active'] {
          (0.99) validate{validation_2 = 'made'} ->
```

```
                (0.8) claim_valid{claim_validity_2 = 'valid'} -> cont
                (0.2) claim_invalid{claim_validity_2 = 'invalid'} -> cont
            (0.01) fail_to_validate{validation_2 = 'failed'} -> cont
        } <>
        [policy_status == 'expired'] {
            stop
        }
    };

}
```