Test script of Strategy, which provides a Strategy to configure Context

```
1  package test.auto;
2
3  // All the roles and operations are named corresponding to the GoF textbook.
4
5  // StrCtxRole: Context role of Strategy
6  // StrRole: Strategy role
7  // CtxInterOper: ContextInterface operation of Context role
8  public class Strategy_<name: {role: StrCtxRole}>_<name: {role: StrRole}>_<seq:>
9  {
10     public static void main(String[] args) throws java.lang.Exception {
11         {role: StrRole} strategy = new [concrete: {role: StrRole}](:);
12         {role: StrCtxRole} context = new {role: StrCtxRole}(:strategy);
13         context.{oper: CtxInterOper}(:);
14     }
15 }
```

Test script of Strategy, which uses a setter to configure Context with a Strategy

```
1  package test.auto;
2
3  // StrCtxRole: Context role of Strategy
4  // StrRole: Strategy role
5  // CtxInterOper: ContextInterface operation of Context role
6  public class Strategy_Setter_<name: {role: StrCtxRole}>_<name: {role: StrRole}>_<seq:>
7  {
8      public static void main(String[] args) throws java.lang.Exception {
9          {role: StrRole} strategy = new [concrete: {role: StrRole}](:);
10         {role: StrCtxRole} context = new {role: StrCtxRole}(:);
11         context.[setter: {role: StrRole}](:strategy);
12         context.{oper: CtxInterOper}(:);
13     }
14 }
```

Test script of State, which provides a State to configure Context

```
1  package test.auto;
2
3  // StaCtxRole: Context role of State
4  // StaRole: State role
5  // CtxReqOper: Request operation of Context
6  public class State_<name: {role: StaCtxRole}>_<name: {role: StaRole}>_<seq:>
7  {
8      public static void main(String[] args) throws java.lang.Exception {
9          {role: StaRole} state = new [concrete: {role: StaRole}](:);
10         {role: StaCtxRole} context = new {role: StaCtxRole}(:state);
11         context.{oper: CtxReqOper}(:);
12     }
13 }
```

Test script of State, which uses the default State of Context

```
1  package test.auto;
2
3  // StaCtxRole: Context role of State
4  // StaRole: State role
5  // CtxReqOper: Request operation of Context
6  public class State_DefaultState_<name: {role: StaCtxRole}>_<name: {role: StaRole}>_<seq:>
7  {
8      public static void main(String[] args) throws java.lang.Exception {
9          {role: StaCtxRole} context = new {role: StaCtxRole}(:);
10         context.{oper: CtxReqOper}(:);
11     }
12 }
13
```

Test script of Bridge

```
1  package test.auto;
2
3  // AbsRole: Abstraction role
4  // ImpRole: Implementor role
5  // AbsOper: Operation of Abstraction role
6  public class Bridge_<name: {role: AbsRole}>_<name: {role: ImpRole}>_<seq:>
7  {
8      public static void main(String[] args) throws java.lang.Exception {
9          {role: ImpRole} implementor = new [concrete: {role: ImpRole}](:);
10         {role: AbsRole} abstraction = new [concrete: {role: AbsRole}](:implementor);
11         abstraction.{oper: AbsOper}(:);
12     }
13 }
```

Test script of Command

```
1  package test.auto;
2
3  // ConcCmdRole: concrete Command role
4  // RecRole: Receiver role
5  // CmdExeOper: Execute operation of Command
6  public class Command_<name: {role: ConcCmdRole}>_<seq:>
7  {
8      public static void main(String[] args) throws java.lang.Exception {
9          {role: RecRole} receiver = new [concrete: {role: RecRole}](:);
10         {role: CmdRole} command = new {role: ConcCmdRole}(:receiver);
11         command.{oper: CmdExeOper}(:);
12     }
13 }
```

Test script of TemplateMethod

```
1  package test.auto;
2
3  // AbsClsRole: AbstractClass role
4  // TmpMtdOper: TemplateMethod operation of AbstractClass
5  public class TemplateMethod_<name: {role: AbsClsRole}>_<seq:>
6  {
7      public static void main(String[] args) throws java.lang.Exception {
8          {role: AbsClsRole} abstractClass = new [concrete: {role: AbsClsRole}](:);
9          abstractClass.{oper: TmpMtdOper}(:);
10     }
11 }
```

Behavior specification of Strategy

```
# {0}: placeholder for a Strategy instance
# ?ctx: Context role
# ?str: Strategy role

{0} ptn:containsRole ?ctx; ptn:containsRole ?str.
?ctx a ptn:StrCtxRole.
?str a ptn:StrRole.

# instantiate Strategy
?concStr java:isEA ?str.
?mcStrInit jbh:methodCalled ?strInit; jbh:objTag ?strObj.
?concStr java:hasMethod ?strInit.
?strInit jbh:isInit true.

# instantiate Context
?mcCtxInit jbh:methodCalled ?ctxInit; jbh:objTag ?ctxObj.
?ctx java:hasMethod ?ctxInit; java:hasField ?strFld.
?ctxInit jbh:isInit true.
?strFld java:fieldTypeIs ?str.

filter exists {
  { # provide a Strategy to configure Context
    ?mcStrInit t:before ?mcCtxInit.
    ?mcCtxInit t:contains [jbh:visitedField ?strFld; jbh:ownerObjTag ?ctxObj;
        jbh:objTag ?strObj].
    bind(?mcCtxInit as ?mcSet)
  } union
  { # or use a setter to configure Context with a Strategy
    ?ctx java:hasMethod ?setter.
    ?setter java:hasModifier "public"; java:hasParamType ?str.
    ?mcSet jbh:methodCalled ?setter; jbh:objTag ?ctxObj;
        t:contains [jbh:visitedField ?strFld; jbh:ownerObjTag ?ctxObj; jbh:objTag ?strObj].
    { ?mcCtxInit t:before ?mcSet } union
    { ?mcCtxInit t:contains ?mcSet }
  }

```

```
  filter exists {
    # the Request operation is called
    ?ctx java:hasMethod ?reqOper.
    ?reqOper java:hasModifier "public".
    ?mcReq jbh:methodCalled ?reqOper; jbh:objTag ?ctxObj.

    ?mcSet t:before ?mcReq.

    # the Request operation calls the Handle operation
    ?mcReq t:contains ?mcHandle.
    ?mcHandle jbh:methodCalled [^java:hasMethod ?concStr]; jbh:methodCaller [^java:hasMethod ?ctx];
        jbh:paramObjTag ?pTag; jbh:objTag ?strObj.
    filter(?pTag>0)

    filter not exists {
      # change the state: field modification
      ?fm jbh:visitedField ?strFld; jbh:ownerObjTag ?ctxObj; jbh:fieldVisitMethod ?vm.
      ?mcSet t:before ?fm.
        filter(?vm != ?setter)
    }
  }
} # end filter exists
```

Behavior specification of State

```
# {0}: placeholder for a State instance
# ?ctx: Context role
# ?sta: State role

{0} ptn:containsRole ?ctx; ptn:containsRole ?sta.
?ctx a ptn:StaCtxRole.
?sta a ptn:StaRole.

# instantiate State
?concSta java:isEA ?sta. # ?concSta: concrete state
?enCStaInit jbh:methodCalled ?cStaInit; jbh:objTag ?sTag.
?concSta java:hasMethod ?cStaInit.
?cStaInit jbh:isInit true.

# instantiate Context
?enCtxInit jbh:methodCalled ?ctxInit; jbh:objTag ?cTag.
?ctx java:hasMethod ?ctxInit; java:hasField ?sFld.
?ctxInit jbh:isInit true.
?sFld java:fieldTypeIs ?sta.

filter exists {
  {
    { # construct a Context with a State
      ?enCStaInit t:before ?enCtxInit.
      ?enCtxInit t:contains [jbh:visitedField ?sFld; jbh:ownerObjTag ?cTag; jbh:objTag ?sTag].
    } union
    { # construct a Context with a default State
      ?enCtxInit t:contains ?enCStaInit
    }

    filter exists {
      # the Request operation is called
      ?ctx java:hasMethod ?_m.
      ?_m java:hasModifier "public".
      ?_r jbh:methodCalled ?_m; jbh:objTag ?cTag.

```

```
      ?enCtxInit t:before ?_r.
      ?enCStaInit t:before ?_r.

      # the Request operation calls the Handle operation
      ?_r t:contains [jbh:methodCalled [^java:hasMethod ?concSta]; jbh:objTag ?sTag].

      # change the state: field modification
      ?fm jbh:visitedField ?sFld; jbh:ownerObjTag ?cTag.
      ?_r t:contains ?fm
    }
  } union
  { # a variant in which State is lazily instantiated until a Request is delivered
    # the Request operation is called
    ?ctx java:hasMethod ?_m.
    ?_m java:hasModifier "public".
    ?_r jbh:methodCalled ?_m; jbh:objTag ?cTag.
    ?enCtxInit t:before ?_r.

    # instantiate State
    ?_r t:contains ?enCStaInit.

    # field modification
    ?fm jbh:visitedField ?sFld; jbh:ownerObjTag ?cTag; jbh:objTag ?sTag.
    ?_r t:contains ?fm.
    ?enCStaInit t:before ?fm.

    # the Request operation calls the Handle operation
    ?_r t:contains [jbh:methodCalled [^java:hasMethod ?concSta]; jbh:objTag ?sTag; ^t:before ?fm].
  }
} # end filter exists
```

Behavior specification of Bridge

```
# {0}: placeholder for a Bridge instance
# ?abs: Abstraction role
# ?imp: Implementor role

{0} ptn:containsRole ?abs; ptn:containsRole ?imp;
  ptn:containsOper ?absOper; ptn:containsOper ?impOper.

?abs a ptn:AbsRole.
?imp a ptn:ImpRole.
?absOper a ptn:AbsOper; java:localNameIs ?absOperLn; java:methodSig ?absOperSig.
?impOper a ptn:ImpOper; java:localNameIs ?impOperLn; java:methodSig ?impOperSig.

# ?refAbs: refined Abstraction
?refAbs java:isEA ?abs; java:hasMethod ?refAbsOper.
?refAbsOper java:localNameIs ?absOperLn; java:methodSig ?absOperSig.

# ?concImp: concrete Implementor
?concImp java:isA ?imp; java:hasMethod ?concImpOper.
?concImpOper java:localNameIs ?impOperLn; java:methodSig ?impOperSig.

# the Implementor field of Abstraction is modified
?fmImp jbh:visitedField [java:fieldTypeIs ?imp]; jbh:fieldObjTag ?fldObj.

# the operation method of Abstraction or refined Abstraction is called
?miAbsOper jbh:methodCalled ?refAbsOper.

# the operation method of concrete Implementor is called
?miImpOper jbh:methodCalled ?concImpOper; jbh:objTag ?fldObj.

?fmImp t:before ?miAbsOper.
?miAbsOper t:contains ?miImpOper
```

Behavior specification of Command

```
# {0}: placeholder for a Command instance
# ?cmd: Command role
# ?rec: Receiver role
# ?exe: execute method of Command

{0} ptn:containsRole ?cmd; ptn:containsRole ?rec;
  ptn:containsOper ?exe.

?cmd a ptn:CmdRole.
?rec a ptn:RecRole.
?exe a ptn:CmdExeOper; java:localNameIs ?exeLn; java:methodSig ?exeSig.

# ?concExe: a method of a concrete Command, which overrides ?exe
?concCmd java:isA ?cmd; java:hasMethod ?concExe.
?concExe java:localNameIs ?exeLn; java:methodSig ?exeSig.

# ?concAct: the action method of a concrete Receiver
?concRec java:isEA ?rec; java:hasMethod ?concAct.

# the execute method of the concrete Command is called
?miConcExe jbh:methodCalled ?concExe.

# the action method of the Receiver is called
?miConcAct jbh:methodCalled ?concAct.

?miConcExe t:contains ?miConcAct.
```

Behavior specification of TemplateMethod

```
# {0}: placeholder for a TemplateMethod instance
# ?absCls: AbstractClass role
# ?tmpMtdOper: template method operation
# ?primOper: primitive operation

{0} ptn:containsRole ?absCls;
  ptn:containsOper ?tmpMtdOper; ptn:containsOper ?primOper.

?absCls a ptn:AbsClsRole.
?tmpMtdOper a ptn:TmpMtdOper; java:localNameIs ?tmpMtdOperLn; java:methodSig ?tmpMtdOperSig.
?primOper a ptn:PrimOper; java:localNameIs ?primOperLn; java:methodSig ?primOperSig.

# ?concTmpMtd: a concrete method that overrides ?tmpMtdOper (can be itself)
[] java:isEA ?absCls; java:hasMethod ?concTmpMtd.
?concTmpMtd java:localNameIs ?tmpMtdOperLn; java:methodSig ?tmpMtdOperSig.

# ?concPrimOper: a concrete method that overrides ?primOper
[] java:isA ?absCls; java:hasMethod ?concPrimOper.
?concPrimOper java:localNameIs ?primOperLn; java:methodSig ?primOperSig.
filter(?concTmpMtd != ?primOperLn)

# the template method is called
?miTmpMtdOper jbh:methodCalled ?concTmpMtd.

# the primitive method is called
?miPrimOper jbh:methodCalled ?concPrimOper.

?miTmpMtdOper t:contains ?miPrimOper.
```