

```

<package> formcharts

<class> ModelingElement
<attributes>
+ name: String;
<endclass>

<class> Page <specializes> ModelingElement
  <attributes>
    isInitial : Boolean;
  <operations>
    + findReachableRecursive( visited : Set(Page) , toVisit : Set(Page) ,
reached : Set(Page) ) : Set(Page);
<endclass>

<class> Action <specializes> ModelingElement
<endclass>

<associations>
+ ModelingElement.prev [1..*]    <-> + Page.nextPages [0..*]    ;
+ ModelingElement.prev [1..*]    <-> + Action.nextActions [0..*]  ;
<endpackage>

```

```

package formcharts

context Page
  inv onlyOneInitial:
    self.allInstances()->one( isInitial )

context ModelingElement
  inv noDuplicateNames :
    self.allInstances()->select( i |
      i.name = self.name and i <> self )->isEmpty()

context ModelingElement
  inv eitherPagesOrActionsFollow :
    self.nextActions->isEmpty() <> self.nextPages->isEmpty()

context Page
  inv noActionFollowedByAction :
    self.nextActions->isEmpty()

context Page
  def: directlyReachablePages() : Set(Page)
    = nextPages->union(self.nextActions.nextPages->asSet())

context Page
  def: reachablePages() : Set(Page)
    = let visited : Set(Page) = Set {} ,
      toVisit : Set(Page) = Set { self } ,
      reached : Set(Page) = Set {}
      in findReachableRecursive( visited, toVisit, reached )

context Page::findReachableRecursive( visited : Set(Page) ,
  toVisit : Set(Page) , reached : Set(Page) ) : Set(Page)
  body: if toVisit->isEmpty()
    then reached

```

```

else let chooseNext : Page = toVisit->any(true)
in
  let newVisited : Set(Page)
    = visited->including(chooseNext) ,
    newToVisit : Set(Page)
    = toVisit->excluding(chooseNext)->union(
      (
        chooseNext.directlyReachablePages()
        ->excluding(chooseNext) - visited ) - (reached)
      ) ,
    newReached : Set(Page)
    =
reached->union(chooseNext.directlyReachablePages()->union(toVisit)
in
  findReachableRecursive ( newVisited, newToVisit, newReached )
endif

context Page
def: isReachableFrom( p : Page ) : Boolean
  = p.reachablePages()->includes(self)

context Page
inv allPagesReachableFromInitialPage :
  let initialPage : Page
  = self.allInstances()->select( isInitial )->any(true)
  in
  self.isReachableFrom(initialPage)

endpackage

```