Nach der Definition eines Modells ist nachzuweisen:

1. $[\![COMPINV_{AirlineSystem}]\!]_{\beta,\sigma_{init},\sigma_{init}} = true.$
2. z.B.
   $\mathcal{T}_{\text{AirlineSystem}} \models$ `context` $Airline :: deploy(p : Person, f : Flight)$

   `pre` : $p <> null \text{ and } f <> null \text{ and}$
   $self.flights \rightarrow includes(f) \text{ and}$
   $p.flights \rightarrow excludes(f) \text{ and}$
   $(employees \rightarrow includes(p) \text{ or}$
   $partners \rightarrow exists(pa \mid pa.employees \rightarrow includes(p))) \text{ and}$
   $COMPINV_{AirlineSystem} \text{ and } CLASSINV_{\Delta}$

   `post` : $p.flights = p.flights@pre \rightarrow including(f) \text{ and}$
   $COMPINV_{AirlineSystem} \text{ and } CLASSINV_{\Delta}$

wobei

$COMPINV_{AirlineSystem} =$

$invEmployeeFlight \text{ and } invExistsPilotSteward \text{ and } invPartners \text{ and}$
$invOneAirlineForPerson \text{ and } invAtLeastTwo \text{ and } invBidirect \text{ and } invPersonAbstract$

$CLASSINV_{\Delta} =$

$Person.allInstances() \rightarrow forAll(self \mid self.salary >= 0 \text{ and } self.salary <= 20000) \text{ and}$
$Airline.allInstances() \rightarrow forAll(self \mid self.partners \rightarrow excludes(self)) \text{ and}$
$Flight.allInstances() \rightarrow forAll(self \mid self.airline <> null)$

Vorgehen:

Wir müssen nur Zustände betrachten, in denen die Vorbedingung gilt. Wir "'denken"' uns also einen Zustand σ^-, in dem die Vorbedingung gilt.

Jetzt müssen wir zeigen:

1. Es gibt im Transitionssystem keine Transitionen $\sigma^- \overset{deploy(...)}{\longrightarrow} \perp$
2. Für alle Transitionen $\sigma^- \overset{deploy(...)}{\longrightarrow} \sigma$ des Transitionssystems erfüllt σ die Nachbedingung.

Punkt 1 ist schnell gezeigt: Die einzige "'verbotene"' Transition $\sigma^- \overset{deploy()}{\longrightarrow} \perp$ fordert, dass der erste Paramer (für die Person) `null` ist. Dies wird jedoch in der Vorbedingung ausgeschlossen.

Für Punkt 2 verwenden wir unser Wissen über die Konstruktion von σ. In der Transition entsteht σ durch Hinzufügen von einem Flug (`vf`) zur Liste `vp.flights` der Person. Alles andere bleibt gleich.

Wir müssen also nur diejenigen Prädikate überprüfen, die sich auf die Liste `Person.flights` beziehen.

- Explizite Nachbedingung: `p.flights = p.flights@pre->including(f)` entspricht genau der Konstruktion der Transition.
- Komponenteninvariante `invEmployeeFlight`

```
context AirlineSystem
  inv  invEmployeeFlight:
       Person.allInstances() -> forAll( p |
         p.flights -> forAll( f |
           (f.airline.employees -> includes(p)) or
           (f.airline.partners -> exists( pa |
              pa.employees -> includes(p)
            ))
         )
       )
```

  `invEmployeeFlight` gilt in σ, weil in der Vorbedingung entsprechendes für den an die Methode `deploy` übergebenen Flug gefordert wurde.

- Komponenteninvariante `invExistsPilotSteward`

```
context AirlineSystem
  inv  invExistsPilotSteward:
       Flight.allInstances() -> forAll( f |
         Pilot.allInstances() -> exists( p |
           p.flights -> includes(f)
         ) and
         Steward.allInstances() -> exists( s |
           s.flights -> includes(f)
         )
       )
```

  `invExistsPilotSteward` galt laut Vorbedingung in σ^-. Da zur Liste der Flüge ja ein Element hinzugefügt wurde, kann in σ für keinen Flug eine zugeordnete Person weggefallen sein.

- Komponenteninvariante `invAtleastTwo`

```
context AirlineSystem
  inv  invAtleastTwo:
       Flight.allInstances() -> forAll( f |
         Person.allInstances() -> select( p |
   p.flights -> includes(f)
         ) -> size() >= 2
       )
```

  `invAtleastTwo` galt laut Vorbedingung in σ^-. Nach der Ausführung von `deploy` ist für jeden Flug f die Menge aller Personen, die diesem Flug zugeteilt sind, entweder gleich geblieben oder um eins gewachsen. Da nur `->size() >= 2` gefordert ist, muss `invAtleastTwo` auch in σ gelten. Laut Komponentenspezifikation können (obwohl es eigentlich unwirtschaftlich wäre) auch mehr als 2 Piloten demselben Flug zugeteilt werden.