Appendix of the paper "Understanding Synthesized Reactive Systems Through Invariants"

The main paper was published at FM 2024, available at `https://doi.org/10.1007/978-3-031-71162-6_9`

Rüdiger Ehlers

Clausthal University of Technology, Clausthal-Zellerfeld, Germany

A Full specification of the GUI glue code synthesis example

We provide the full specifications of the GUI glue code synthesis case study here. They are in the input format used by the `slugs` tool, for which details are available in the author-archived version of the slugs tool paper [14]. In a nutshell, the specification file lines are split into input proposition definitions (`[INPUT]`), output proposition definitions (`[OUTPUT]`), initialization assumptions (`[ENV_INIT]`), initialization guarantees (`[SYS_INIT]`), safety assumptions (`[ENV_TRANS]`), safety guarantees (`[SYS_TRANS]`), liveness assumptions (`[ENV_LIVENESS]`), and liveness guarantees (`[SYS_LIVENESS]`). Proposition values *after* transitions are referred to by adding a dash (`'`) to the proposition name. The usual Boolean connectives for disjunction (`|`), conjunction (`&`), negation (`!`), and implication (`->`) can also be used, as well as braces. Comment lines start with a dash symbol (`#`).

Specification lines below that are too long are wrapped in the following.

For the BDDs showing the computed invariants in the following, blue thick edges with arrow heads define *then* successor edges, while red thinner lines without arrow head denote *else* successor edges. The BDDs shown in the appendix are automatically drawn.

A.1 Step 1

The starting specification is the following:

```
[INPUT]
clickPointSelection
forward
backward

[OUTPUT]
showPointSelectionPage
pointEdit0
pointEdit1
pointEdit2
pointEdit3
allPointsSelected
updatePointSelectionWindow
forwardButtonEnabled
```

```
[ENV_LIVENESS]

# We always eventually proceed
! showPointSelectionPage | clickPointSelection'

[ENV_INIT]
! forward
! backward
! clickPointSelection

[SYS_INIT]
showPointSelectionPage
pointEdit0
! pointEdit1
! pointEdit2
! pointEdit3
! forwardButtonEnabled
! allPointsSelected
! forwardButtonEnabled

[SYS_TRANS]
showPointSelectionPage & !backward' & !forward' -> showPointSelectionPage'

# PointEditing
pointEdit0 -> (!pointEdit1 & !pointEdit2 & !pointEdit3)
pointEdit1 -> (!pointEdit2 & !pointEdit3)
pointEdit2 -> (!pointEdit3)
pointEdit0 & clickPointSelection' -> pointEdit1' & updatePointSelectionWindow'
pointEdit1 & clickPointSelection' -> pointEdit2' & updatePointSelectionWindow'
pointEdit2 & clickPointSelection' -> pointEdit3' & updatePointSelectionWindow'
pointEdit3 & clickPointSelection' -> pointEdit0' & updatePointSelectionWindow'
pointEdit0 & !clickPointSelection' -> pointEdit0' & updatePointSelectionWindow'
pointEdit1 & !clickPointSelection' -> pointEdit1' & updatePointSelectionWindow'
pointEdit2 & !clickPointSelection' -> pointEdit2' & updatePointSelectionWindow'
pointEdit3 & !clickPointSelection' -> pointEdit3' & updatePointSelectionWindow'
allPointsSelected' <-> allPointsSelected | pointEdit3 & clickPointSelection'
```

Our approach computes a single BDD for this case, which is shown in Figure 1. With some exercise in BDD reading (and after getting used to that edges to the 0 sink not shown), it becomes easy to see that this invariant encodes that at most one of the `pointEdit` bits may be true at any point in time.

This invariant can be (manually) encoded as additional safety guarantees as follows:

```
[SYS_TRANS]
pointEdit0' -> (!pointEdit1' & !pointEdit2' & !pointEdit3')
pointEdit1' -> (!pointEdit2' & !pointEdit3')
pointEdit2' -> (!pointEdit3')
```

A.2 Step 2

The following specification parts are added:

```
[OUTPUT]
showSliderPage

[SYS_INIT]
! showSliderPage

[SYS_TRANS]
```

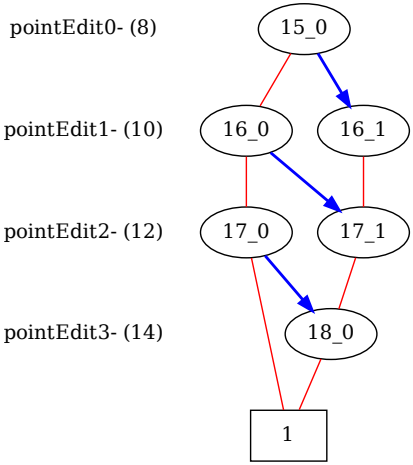

Fig. 1. Computed Invariant BDD

.

```
showPointSelectionPage & forward' -> showSliderPage'
showSliderPage & backward' -> showPointSelectionPage'

# Only one page visible
showPointSelectionPage' -> (!showSliderPage')
showSliderPage' -> (!showPointSelectionPage')
```

A.3 Step 3

The following specification parts are added:

```
[INPUT]
sliderMove
runningPreviewThread

[OUTPUT]
startPreviewThread
needToStartPreviewThread

[ENV_TRANS]
sliderMove' -> showSliderPage
startPreviewThread -> runningPreviewThread'
!runningPreviewThread & !startPreviewThread -> !runningPreviewThread'

[ENV_LIVENESS]
! runningPreviewThread | startPreviewThread

[ENV_INIT]
! sliderMove
! runningPreviewThread

[SYS_INIT]
```

```
! needToStartPreviewThread
! startPreviewThread

[SYS_TRANS]
showSliderPage -> (forwardButtonEnabled <-> (!needToStartPreviewThread &&
    !runningPreviewThread))

# Preview Thread starting
runningPreviewThread' -> !startPreviewThread'
needToStartPreviewThread' <-> ((needToStartPreviewThread & !startPreviewThread') |
    (showPointSelectionPage & forward')) | sliderMove'

[SYS_LIVENESS]
! needToStartPreviewThread | startPreviewThread
```

Two invariants are computed after the addition, for which versions optimized by removing unnecessary variables are shown in Figure 2 and Figure 3.

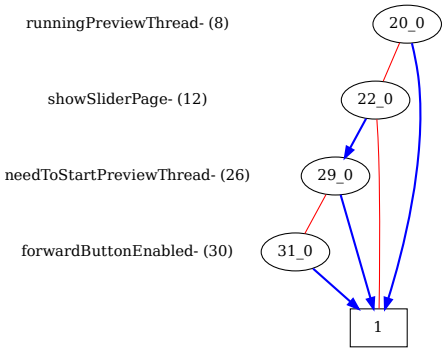

Fig. 2. Computed Invariant BDD

.

A.4 Step 4

The following specification parts are added:

```
[INPUT]
runningPreciseComputationThread

[OUTPUT]
showPleaseWaitPage
showResolutionSelectionPage
startPreciseComputationThread

[ENV_TRANS]
# The following assumptions is now needed to avoid
# unrealizability, but wasn't needed before.
! forward' | ! backward'
```

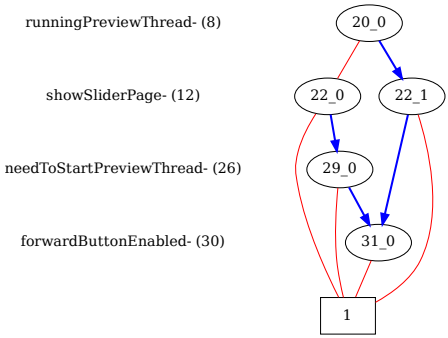

Fig. 3. Computed Invariant BDD

.

```
showResolutionSelectionPage -> !forward'

startPreciseComputationThread -> runningPreciseComputationThread'
!runningPreciseComputationThread & !startPreciseComputationThread ->
   !runningPreciseComputationThread'

[ENV_LIVENESS]
! runningPreciseComputationThread | startPreciseComputationThread

[ENV_INIT]
! runningPreciseComputationThread

[SYS_INIT]
! startPreciseComputationThread

[SYS_TRANS]
showSliderPage & forward' -> showPleaseWaitPage'

showPleaseWaitPage & forward' -> showResolutionSelectionPage'
showPleaseWaitPage & backward' -> showSliderPage'
showResolutionSelectionPage & backward' ->showSliderPage'

showSliderPage & forward' <-> startPreciseComputationThread'
showPleaseWaitPage -> !forwardButtonEnabled
showPleaseWaitPage -> (showPleaseWaitPage' | backward' | showResolutionSelectionPage')
showPleaseWaitPage -> (showPleaseWaitPage' <-> !(runningPreciseComputationThread &
    !runningPreciseComputationThread' | backward'))
showResolutionSelectionPage -> !runningPreciseComputationThread
```

This is the addition from which onwards the winning states and the reachable-and-winning states in the game start to become bigger – here, we have 26 and 161 BDD nodes, respectively.

The computed invariant BDD is however smaller and shown in Figure 4. To facilitate its interpretation, a script (`dotBDDToBoolLabCommands.py`) that is part of the `slugs` distribution is used to translate it to a sequence of commands for reconstructing the BDD in the *Boolean Function Lab* that is available for use in

a web browser at `https://www.ruediger-ehlers.de/boollab/`. By reodering the variable declarations, one can experiment with which order leads to the most readable BDD. The result is shown in Figure 1d.

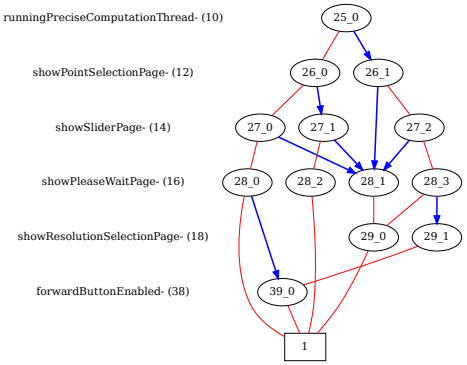

Fig. 4. Computed Invariant BDD

.

We condense what can be seen in the BDD as the following invariants:

```
# Reachability of what is shown
showPointSelectionPage' -> !showSliderPage' & !showPleaseWaitPage' & !showResolutionSelectionPage'
showSliderPage' -> !showPointSelectionPage' & !showPleaseWaitPage' & !showResolutionSelectionPage'
showPleaseWaitPage' -> !showPointSelectionPage'  & !showSliderPage' & !showResolutionSelectionPage'
showResolutionSelectionPage' -> !showPointSelectionPage'  & !showSliderPage' & !showPleaseWaitPage'
showPointSelectionPage' | showSliderPage' | showPleaseWaitPage' | showResolutionSelectionPage'

# Invariants
showResolutionSelectionPage' -> ! runningPreciseComputationThread'
showPleaseWaitPage' -> !forwardButtonEnabled'
```

A.5 Step 5

The following specification parts are added:

```
[INPUT]
runningRescalingThread

[OUTPUT]
startRescalingThread
updatePreview

[ENV_TRANS]
startRescalingThread -> runningRescalingThread'
!runningRescalingThread & !startRescalingThread -> !runningRescalingThread'

[ENV_LIVENESS]
! runningRescalingThread | startRescalingThread

[ENV_INIT]
```

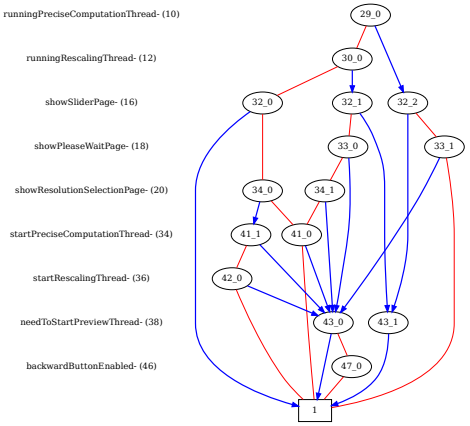

Fig. 5. Computed Invariant BDD

.

```
! runningRescalingThread

[SYS_INIT]
! startRescalingThread
! updatePreview

[SYS_TRANS]
# RescalingThread
(showResolutionSelectionPage' & !showResolutionSelectionPage) <-> startRescalingThread'

# Don't start the additional thread if it is still running
runningRescalingThread' -> !startRescalingThread'
runningPreviewThread' -> !startPreviewThread'
runningPreciseComputationThread -> !startPreciseComputationThread'
```

A.6 Step 6

The following specification parts are added:

```
[OUTPUT]
backwardButtonEnabled

[ENV_TRANS]
backward' -> backwardButtonEnabled
```

The winning states and reachable-and-winning states after this addition have 44 and 255 BDD nodes, respectively. The computed optimized invariant is again substantially smaller and shown in Figure 5. We condense what can be seen in the BDD as the following invariants, added as safety guarantees to the system:

```
forward' & !needToStartPreviewThread' -> !backwardButtonEnabled'
(runningPreciseComputationThread' | runningRescalingThread') & showSliderPage' ->
    needToStartPreviewThread'
# The following invariant was strengthened by hand.
showPleaseWaitPage' -> !backwardButtonEnabled'
# Normal ones again
showResolutionSelectionPage' & (startRescalingThread' | runningRescalingThread') &
    !needToStartPreviewThread' -> !backwardButtonEnabled'
```

B Full Generalized Buffer Benchmark

In this part of the appendix, we show all of the computed invariants and the specification for the generalized buffer case study.

The invariants below are for the modified search algorithm (with W instead of B for line 17 in the algorithm and when including initial positions that are not in W for the computation of B). However, the first three invariants below the same as those computed by the original algorithm. Only afterwards, the invariants differ.

In the step in which three invariants were identified, their conjunction yields a BDD with 531 nodes, which shows that the decomposition is really needed to analyze the invariant(s).

B.1 Step 1

The starting specification is the following:

```
[INPUT]
StoB_REQ0
StoB_REQ1
StoB_REQ2
RtoB_ACK0
RtoB_ACK1
FULL
EMPTY

[OUTPUT]
BtoS_ACK0
BtoS_ACK1
BtoS_ACK2
BtoR_REQ0
BtoR_REQ1
stateG7_0
stateG7_1
ENQ
DEQ
stateG12
SLC0
SLC1

[ENV_INIT]
! StoB_REQ0
! StoB_REQ1
! StoB_REQ2
! RtoB_ACK0
! RtoB_ACK1
! FULL
EMPTY
```

```
[ENV_TRANS]


[ENV_LIVENESS]

[SYS_INIT]
!BtoS_ACK0
! BtoS_ACK1
! BtoS_ACK2
! BtoR_REQ0
! BtoR_REQ1
! stateG7_0
stateG7_1
!ENQ
!DEQ
! stateG12
! SLC0
! SLC1

[SYS_TRANS]

((! StoB_REQ0 & (StoB_REQ0')) -> (! BtoS_ACK0'))
((! BtoS_ACK0 & ! StoB_REQ0) -> (! BtoS_ACK0'))
((! StoB_REQ1 & (StoB_REQ1')) -> (! BtoS_ACK1'))
((! BtoS_ACK1 & ! StoB_REQ1) -> (! BtoS_ACK1'))
((! StoB_REQ2 & (StoB_REQ2')) -> (! BtoS_ACK2'))
((! BtoS_ACK2 & ! StoB_REQ2) -> (! BtoS_ACK2'))


[SYS_LIVENESS]

(StoB_REQ0 <-> BtoS_ACK0)
(StoB_REQ1 <-> BtoS_ACK1)
(StoB_REQ2 <-> BtoS_ACK2)
```

The specification is realizable, and there are 0 mixed monotone/antitone invariants computed.

B.2 Step 2

For this step, the specification was extended as follows:

```
[SYS_TRANS]
((BtoS_ACK0 & StoB_REQ0) -> (BtoS_ACK0'))
((BtoS_ACK1 & StoB_REQ1) -> (BtoS_ACK1'))
((BtoS_ACK2 & StoB_REQ2) -> (BtoS_ACK2'))
```

The specification is realizable, and there are 0 mixed monotone/antitone invariants computed.

B.3 Step 3

For this step, the specification was extended as follows:

```
[ENV_TRANS]
((StoB_REQ0 & !BtoS_ACK0) -> (StoB_REQ0'))
(BtoS_ACK0 -> (! StoB_REQ0'))
((StoB_REQ1 & ! BtoS_ACK1) -> (StoB_REQ1'))
(BtoS_ACK1 -> (! StoB_REQ1'))
((StoB_REQ2 & ! BtoS_ACK2) -> (StoB_REQ2'))
```

```
(BtoS_ACK2 -> (! StoB_REQ2'))
[SYS_TRANS]
((!BtoS_ACK0) | (! BtoS_ACK1))
(!BtoS_ACK0) | (! BtoS_ACK2)
((! BtoS_ACK1) | (! BtoS_ACK2))
```

The specification is realizable, and there is one mixed monotone/antitone invariants computed. The computed invariant is shown in and Fig. 6.

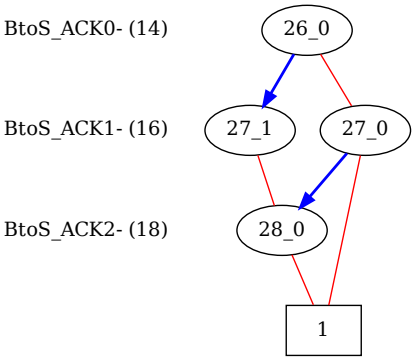

Fig. 6. Computed Invariant BDD

To account for the new invariant, the specification was extended as follows:

```
[SYS_TRANS]
(! BtoS_ACK0' | ! BtoS_ACK1')
(! BtoS_ACK0' | ! BtoS_ACK2')
(! BtoS_ACK1' | ! BtoS_ACK2')
```

B.4 Step 4

For this step, the specification was extended as follows:

```
[SYS_TRANS]
((BtoR_REQ0 & ! RtoB_ACK0) -> (BtoR_REQ0'))
((BtoR_REQ1 & ! RtoB_ACK1) -> (BtoR_REQ1'))
((! BtoR_REQ0) | (! BtoR_REQ1))
((BtoR_REQ0 & BtoR_REQ1) -> FALSE)
((! stateG7_1 & ! BtoR_REQ0 & BtoR_REQ1) -> (stateG7_1' & ! stateG7_0'))
((stateG7_1 & BtoR_REQ0 & ! BtoR_REQ1) -> (! stateG7_1' & ! stateG7_0'))
((! stateG7_1 & ! BtoR_REQ0 & ! BtoR_REQ1) -> (! stateG7_1' & stateG7_0'))
((stateG7_1 & ! BtoR_REQ0 & ! BtoR_REQ1) -> (stateG7_1' & stateG7_0'))
((! stateG7_1 & ! stateG7_0 & BtoR_REQ0 & ! BtoR_REQ1) -> (! stateG7_1' & ! stateG7_0'))
((stateG7_1 & ! stateG7_0 & ! BtoR_REQ0 & BtoR_REQ1) -> (stateG7_1' & ! stateG7_0'))
((! stateG7_1 & stateG7_0 & BtoR_REQ0) -> FALSE)
((stateG7_1 & stateG7_0 & BtoR_REQ1) -> FALSE)
```

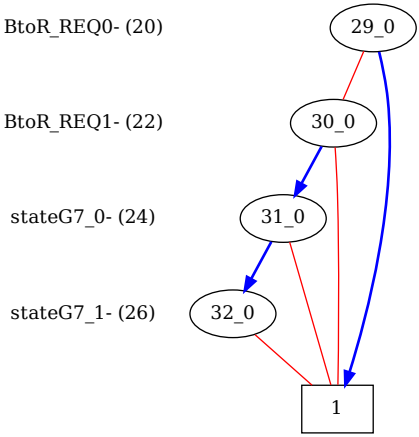

Fig. 7. Computed Invariant BDD

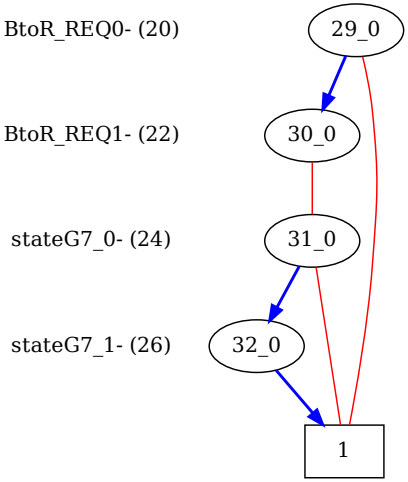

Fig. 8. Computed Invariant BDD

The specification is realizable, and there are 2 mixed monotone/antitone invariants computed. The computed invariants are shown in Fig. 7 and Fig. 8.

To account for the new invariants, the specification was extended as follows:

```
[SYS_TRANS]
BtoR_REQ0' | ! BtoR_REQ1' | ! stateG7_0' | ! stateG7_1'
! BtoR_REQ0' | ! BtoR_REQ1'
BtoR_REQ0' & stateG7_0' -> stateG7_1'
```

B.5 Step 5

For this step, the specification was extended as follows:

```
[SYS_TRANS]
(RtoB_ACK0 -> (! BtoR_REQ0'))
(RtoB_ACK1 -> (! BtoR_REQ1'))
((!BtoS_ACK0 & (BtoS_ACK0')) -> (ENQ'))
((!BtoS_ACK0 & (BtoS_ACK0'))  -> (! SLC0' & ! SLC1'))
((! BtoS_ACK1 & (BtoS_ACK1')) -> (ENQ'))
((! BtoS_ACK1 & (BtoS_ACK1')) <-> (SLC0' & ! SLC1'))
((! BtoS_ACK2 & (BtoS_ACK2')) -> (ENQ'))
((! BtoS_ACK2 & (BtoS_ACK2')) <-> (! SLC0' & SLC1'))
(((BtoS_ACK0 | (!BtoS_ACK0')) & (BtoS_ACK1 | (! BtoS_ACK1')) &
    (BtoS_ACK2 | (! BtoS_ACK2'))) -> (!ENQ'))
```

The specification is realizable, and there are 0 mixed monotone/antitone invariants computed.

B.6 Step 6

For this step, the specification was extended as follows:

```
[ENV_TRANS]
(! BtoR_REQ0 -> (! RtoB_ACK0'))
(! BtoR_REQ1 -> (! RtoB_ACK1'))
((BtoR_REQ0 & RtoB_ACK0) -> (RtoB_ACK0'))
((BtoR_REQ1 & RtoB_ACK1) -> (RtoB_ACK1'))
(ENQ & !DEQ) -> (! EMPTY')
((DEQ & !ENQ) -> (!FULL'))
((ENQ <-> DEQ) -> ((FULL <-> (FULL')) & ( EMPTY <-> ( EMPTY'))))
[ENV_LIVENESS]
(BtoR_REQ0 <-> RtoB_ACK0)
(BtoR_REQ1 <-> RtoB_ACK1)
[SYS_TRANS]
((RtoB_ACK0 & (! RtoB_ACK0')) -> (DEQ'))
((RtoB_ACK1 & (! RtoB_ACK1')) -> (DEQ'))
(((! RtoB_ACK0 | (RtoB_ACK0')) & (! RtoB_ACK1 |
    (RtoB_ACK1'))) -> (!DEQ'))
(FULL & !DEQ) -> !ENQ
( EMPTY -> !DEQ)
```

The specification is realizable, and there are 3 mixed monotone/antitone invariants computed. The computed invariants are shown in Fig. 9, Fig. 10 and Fig. 11.

To account for the new invariants, the specification was extended as follows:

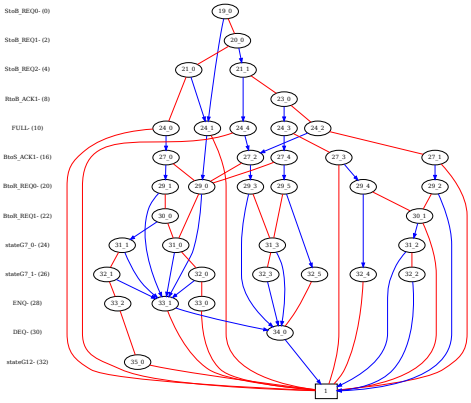

Fig. 9. Computed Invariant BDD

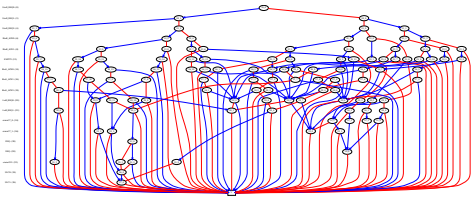

Fig. 10. Computed Invariant BDD

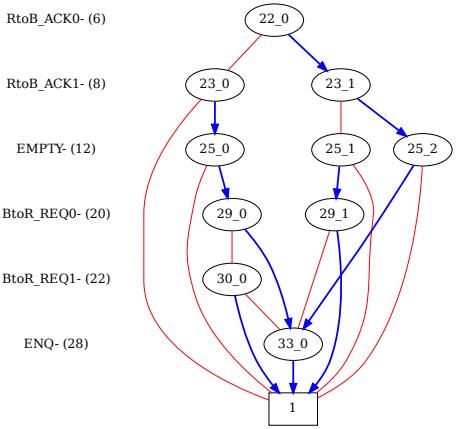

Fig. 11. Computed Invariant BDD

```
[SYS_TRANS]
(FULL' & !DEQ') -> !ENQ'
(EMPTY' -> !DEQ')

FULL' & ((StoB_REQ0' & BtoS_ACK0') | (StoB_REQ1' & BtoS_ACK1') | (StoB_REQ2' & BtoS_ACK2'))
    -> DEQ'
FULL' & ((StoB_REQ0' & BtoS_ACK0') | (StoB_REQ1' & BtoS_ACK1') | (StoB_REQ2' & BtoS_ACK2'))
    -> (BtoR_REQ0' | stateG7_0' | stateG7_1')
FULL' & ((StoB_REQ0' & BtoS_ACK0') | (StoB_REQ1' & BtoS_ACK1') | (StoB_REQ2' & BtoS_ACK2'))
    -> (BtoR_REQ1' | stateG7_0' | stateG7_1')

(StoB_REQ0' -> !BtoS_ACK0') & (StoB_REQ1' -> !BtoS_ACK1') & (StoB_REQ2' -> !BtoS_ACK2')
    & EMPTY' & !BtoR_REQ0' & !BtoR_REQ1' -> (RtoB_ACK1' -> stateG7_0')
(StoB_REQ0' -> !BtoS_ACK0') & (StoB_REQ1' -> !BtoS_ACK1') & (StoB_REQ2' -> !BtoS_ACK2')
    & EMPTY' & !BtoR_REQ0' & !BtoR_REQ1' -> (!RtoB_ACK1' -> (stateG7_0' | stateG7_1'))

RtoB_ACK0' & ! BtoR_REQ0' -> !(DEQ' & !ENQ')
EMPTY' -> (!RtoB_ACK0' | ENQ')
RtoB_ACK1' & ! BtoR_REQ1' -> !(DEQ' & !ENQ')
EMPTY' -> (!RtoB_ACK1' | ENQ')


(StoB_REQ0' & BtoS_ACK0' & ! StoB_REQ1' & ! StoB_REQ2') | (StoB_REQ1' & BtoS_ACK1' &
    ! StoB_REQ0' & ! StoB_REQ2') | (StoB_REQ2' & BtoS_ACK2' & ! StoB_REQ0' & ! StoB_REQ1')
    -> ((RtoB_ACK0' & !BtoR_REQ0' -> !BtoR_REQ1') & (BtoR_REQ0' & RtoB_ACK1' -> !stateG7_1'))

(!StoB_REQ0' & BtoS_ACK0' & !StoB_REQ1' & !StoB_REQ2') | (!StoB_REQ1' & BtoS_ACK1' &
    !StoB_REQ0' & !StoB_REQ2') | (!StoB_REQ0' & !BtoS_ACK0' & !StoB_REQ1' & !BtoS_ACK1'
    & !StoB_REQ2') -> (!BtoR_REQ0' & RtoB_ACK0' & BtoR_REQ1' -> (!EMPTY' & stateG7_0' & !DEQ'))
(!StoB_REQ0' & BtoS_ACK0' & !StoB_REQ1' & !StoB_REQ2') | (!StoB_REQ1' & BtoS_ACK1' &
    !StoB_REQ0' & !StoB_REQ2') | (!StoB_REQ0' & !BtoS_ACK0' & !StoB_REQ1' & !BtoS_ACK1'
    & !StoB_REQ2') -> ((BtoR_REQ0' & !EMPTY') -> ((stateG7_0' | !stateG7_1')
    & (stateG7_0' & stateG7_1' -> !DEQ')))
(!StoB_REQ0' & BtoS_ACK0' & !StoB_REQ1' & !StoB_REQ2') | (!StoB_REQ1' & BtoS_ACK1'
    & !StoB_REQ0' & !StoB_REQ2') | (!StoB_REQ0' & !BtoS_ACK0' & !StoB_REQ1' & !BtoS_ACK1'
    & !StoB_REQ2') -> ((BtoR_REQ0' & EMPTY') -> !stateG7_1')
(!StoB_REQ0' & BtoS_ACK0' & !StoB_REQ1' & !StoB_REQ2') | (!StoB_REQ1' & BtoS_ACK1'
    & !StoB_REQ0' & !StoB_REQ2') | (!StoB_REQ0' & !BtoS_ACK0' & !StoB_REQ1' & !BtoS_ACK1'
    & !StoB_REQ2') -> ((!BtoR_REQ0' & !RtoB_ACK0' & BtoR_REQ1' & EMPTY') -> !stateG7_0')
(!StoB_REQ0' & BtoS_ACK0' & !StoB_REQ1' & !StoB_REQ2') | (!StoB_REQ1' & BtoS_ACK1'
    & !StoB_REQ0' & !StoB_REQ2') | (!StoB_REQ0' & !BtoS_ACK0' & !StoB_REQ1' & !BtoS_ACK1'
    & !StoB_REQ2') -> ((!BtoR_REQ0' & !RtoB_ACK0' & BtoR_REQ1' & !EMPTY') -> (!stateG7_0' | !DEQ'))
```

B.7 Step 7

For this step, the specification was extended as follows:

```
[SYS_TRANS]
((! stateG12 &  EMPTY) -> (! stateG12'))
((! stateG12 & DEQ  ) -> (! stateG12'))
((! stateG12 & ! EMPTY & !DEQ) -> (stateG12'))
((stateG12 & !DEQ  ) -> (stateG12'))
((stateG12 & DEQ  ) -> (! stateG12'))
[SYS_LIVENESS]
(! stateG12)
```

The specification is realizable, and there are 0 mixed monotone/antitone invariants computed.