Experiments and Structures

First of all we want to (re-)emphasize that an initialization of the *Location* to *Home* and the *Height* to *0*, combined with a somewhat larger *Time* domain, typically represents the Bounded Model Checking setting. We also repeat that, given this setting, the *Distance* to *Target* and *Home* and the *ClosestInspectionLocation* are inherently known to IDP and are not to be initialized. Most of the general structures and experiments follow this BMC setting, with as exception the ones defined in section 8.
In IDP syntax, "//" represents a line comment, and in the structures below, this will indicate the irrelevance of certain (initialization) lines.
Further a small clarification is needed regarding the output of the verification tasks. A verification that is stated yielding **True**, only holds if also *Model(s) exist:* states **True**. This extra check is chosen not to be done automatically within the *Procedure* in IDP to obtain a maximum of insight. It is clear that if no models exist at all, no verifications can be checked. This will however only be the case in section 3.1.

1 Example World

A very basic example world, consisting of two Inspection Locations and relatively small distances.

Structure

```
structure World: V {
    Time = {0..35}

    Weight = 0

    Height = {0..5}
    GroundHeight = {0}
    InspectionHeight = {1..2}
    FlyHeight = {3..4}
    RestrictedHeight = {5}
```

```
    Power = {0..100}
    Battery_Init = 100
    PowerUsage = {0..5}
    MoveToPower = 3
    TakePicturePower = 2
    NoOpPower = 1
    LiftPower = 5
    LowerPower = 2
    StaticLowPower = 25
    CriticalPower = 5

    Distance = {0..4}
    Inspection = {"Insp1"; "Insp2"}
    DistanceBetween =
        {"Home", "Home" -> 0; "Home", "Insp1" -> 2; "Home", "Insp2" -> 3;
        "Insp1", "Insp1" -> 0; "Insp1", "Home" -> 2; "Insp1","Insp2" -> 2;
        "Insp2", "Insp2"-> 0; "Insp2","Insp1"-> 2; "Insp2","Home" -> 3}

    Location_Init = Home
    Height_Init = 0
    //DistanceToTarget_Init =
    //DistanceToHome_Init =
    DistanceToRA_Init = 2
    Picture_Taken_Init = {}
    //ClosestInspectionLocationToDo_Init =

    DetourAssumption = 3
}
```

Output

As indicated by the initializations in the Structure, the properties are verified in a traditional BMC fashion.

```
Running verifications with the classical time theory.
Running verifications with the realistic battery.
Running verifications with the dynamic low power bound.
==========
Model(s) exist: true
Duration: 5 seconds.
==========
World assumptions hold: true
Duration: 4 seconds.
==========
Policy debug (pic height) correct: true
Duration: 6 seconds.
==========
Policy debug (pic location) correct: true
Duration: 7 seconds.
==========
Policy debug (move to TravelSpace) correct: true
Duration: 5 seconds.
==========
Policy debug (landing only at home) correct: true
Duration: 16 seconds.
==========
Goal property can hold: true
Duration: 8 seconds.
==========
Goal property always holds: true
Duration: 12 seconds.
==========
Restricted height property holds: true
Duration: 7 seconds.
==========
The drone will always reach Home (using battery, world,... from structure) 1: true
Duration: 11 seconds.
==========
The drone will always reach Home (using battery, world,... from structure) 2: true
Duration: 14 seconds.
==========
The drone will always reach Home (using battery, world,... from structure) 3: true
Duration: 14 seconds.
==========
```

```
The drone will always reach Home (using battery, world,... from structure) 4: true
Duration: 20 seconds.
=============================================
The theories regarding reaching home are equal: true
Duration: 8 seconds.
=============================================
Restricted area is never entered: true
Duration: 9 seconds.
=============================================
Verifications completed after 146 seconds.
```

2 Reaching Home

The Structure and output of the experiment regarding four different implementations of the *Reaching Home* verification.

Structure

```
structure World: V {
    Time = {0..40}

    Weight = 1

    Height = {0..5}
    GroundHeight = {0}
    InspectionHeight = {1..2}
    FlyHeight = {3..4}
    RestrictedHeight = {5}

    Power = {0..100}
    Battery_Init = 100
    PowerUsage = {0..5}
    MoveToPower = 3
    TakePicturePower = 2
    NoOpPower = 1
    LiftPower = 5
    LowerPower = 2
    StaticLowPower = 25
    CriticalPower = 5

    Distance = {0..4}
    Inspection = {"Insp1"; "Insp2"}
    DistanceBetween =
        {"Home", "Home" -> 0; "Home", "Insp1" -> 2; "Home", "Insp2" -> 3;
        "Insp1", "Insp1" -> 0; "Insp1", "Home" -> 2; "Insp1","Insp2" -> 2;
        "Insp2", "Insp2"-> 0; "Insp2","Insp1"-> 2; "Insp2","Home" -> 3}

    Location_Init = Home
    Height_Init = 0
    //DistanceToTarget_Init =
    //DistanceToHome_Init =
    DistanceToRA_Init = 2
    Picture_Taken_Init = {}
    //ClosestInspectionLocationToDo_Init =

    DetourAssumption = 0
}
```

Output

Relevant here, are the four outputs and execution times regarding *The drone will always reach Home...*

```
Running verifications with the classical time theory.
Running verifications with the realistic battery.
Running verifications with the dynamic low power bound.
═══════════════════════════════════════════════
Model(s) exist: true
Duration: 14 seconds.
═══════════════════════════════════════════════
World assumptions hold: true
Duration: 14 seconds.
═══════════════════════════════════════════════
Policy debug (pic height) correct: true
Duration: 17 seconds.
═══════════════════════════════════════════════
Policy debug (pic location) correct: true
Duration: 17 seconds.
═══════════════════════════════════════════════
Policy debug (move to TravelSpace) correct: true
Duration: 17 seconds.
═══════════════════════════════════════════════
Policy debug (landing only at home) correct: true
Duration: 51 seconds.
═══════════════════════════════════════════════
Goal property can hold: true
Duration: 16 seconds.
═══════════════════════════════════════════════
Goal property always holds: false
Duration: 21 seconds.
═══════════════════════════════════════════════
Restricted height property holds: true
Duration: 20 seconds.
═══════════════════════════════════════════════
The drone will always reach Home (using battery, world,... from structure) 1: true
Duration: 75 seconds.
═══════════════════════════════════════════════
The drone will always reach Home (using battery, world,... from structure) 2: true
Duration: 83 seconds.
═══════════════════════════════════════════════
The drone will always reach Home (using battery, world,... from structure) 3: true
Duration: 72 seconds.
═══════════════════════════════════════════════
The drone will always reach Home (using battery, world,... from structure) 4: true
Duration: 58 seconds.
═══════════════════════════════════════════════
The theories regarding reaching home are equal: true
Duration: 19 seconds.
═══════════════════════════════════════════════
Restricted area is never entered: true
Duration: 19 seconds.
═══════════════════════════════════════════════
Verifications completed after 513 seconds.
```

3 World Assumptions

In a Structure, information relevant for the World Assumptions is captured by the values in *DistanceBetween*.

3.1 True Violation

Structure

```
structure World: V {
    Time = {0..35}

    Weight = 0
```

```
    Height = {0..5}
    GroundHeight = {0}
    InspectionHeight = {1..2}
    FlyHeight = {3..4}
    RestrictedHeight = {5}

    Power = {0..50}
    Battery_Init = 50
    PowerUsage = {0..5}
    MoveToPower = 3
    TakePicturePower = 2
    NoOpPower = 1
    LiftPower = 5
    LowerPower = 2
    StaticLowPower = 25
    CriticalPower = 5

    Distance = {0..4}
    Inspection = {"Insp1"; "Insp2"}
    DistanceBetween =
        {"Home", "Home" -> 0; "Home", "Insp1" -> 2; "Home", "Insp2" -> 2;
        "Insp1", "Insp1" -> 0; "Insp1", "Home" -> 2; "Insp1","Insp2" -> 3;
        "Insp2", "Insp2"-> 0; "Insp2","Insp1"-> 3; "Insp2","Home" -> 2}

    Location_Init = Home
    Height_Init = 0
    //DistanceToTarget_Init =
    //DistanceToHome_Init =
    DistanceToRA_Init = 2
    Picture_Taken_Init = {}
    //ClosestInspectionLocationToDo_Init =

    DetourAssumption = 2
}
```

Note: turn InvariantCheckingAssumptions off in the model! (These try to enforce the static variant of the World Assumption.)

Output

```
Running verifications with the classical time theory.
Running verifications with the realistic battery.
Running verifications with the dynamic low power bound.
==========
Model(s) exist: false
Duration: 4 seconds.
==========
World assumptions hold: true
Duration: 4 seconds.
==========
Policy debug (pic height) correct: true
Duration: 4 seconds.
==========
Policy debug (pic location) correct: true
Duration: 4 seconds.
==========
Policy debug (move to TravelSpace) correct: true
Duration: 5 seconds.
==========
Policy debug (landing only at home) correct: true
Duration: 5 seconds.
==========
Goal property can hold: false
Duration: 5 seconds.
==========
Goal property always holds: true
Duration: 5 seconds.
==========
Restricted height property holds: true
Duration: 5 seconds.
==========
```

```
The drone will always reach Home (using battery, world,... from structure) 1: true
Duration: 5 seconds.
==========
The drone will always reach Home (using battery, world,... from structure) 2: true
Duration: 5 seconds.
==========
The drone will always reach Home (using battery, world,... from structure) 3: true
Duration: 5 seconds.
==========
The drone will always reach Home (using battery, world,... from structure) 4: true
Duration: 5 seconds.
==========
The theories regarding reaching home are equal: true
Duration: 7 seconds.
==========
Restricted area is never entered: true
Duration: 5 seconds.
==========
Verifications completed after 73 seconds.
```

3.2 False Violation

Structure

```
structure World: V {
    Time = {0..50}

    Weight = 0

    Height = {0..5}
    GroundHeight = {0}
    InspectionHeight = {1..2}
    FlyHeight = {3..4}
    RestrictedHeight = {5}

    Power = {0..100}
    Battery_Init = 100
    PowerUsage = {0..5}
    MoveToPower = 3
    TakePicturePower = 2
    NoOpPower = 1
    LiftPower = 5
    LowerPower = 2
    StaticLowPower = 25
    CriticalPower = 5

    Distance = {0..5}
    Inspection = {"Insp1"; "Insp2"; "Insp3"}
    DistanceBetween =
        {"Home", "Home" -> 0; "Home", "Insp1" -> 2; "Home", "Insp2" -> 3;
        "Home", "Insp3" -> 4; "Insp1", "Insp1" -> 0; "Insp1", "Home" -> 2;
        "Insp1", "Insp2" -> 2; "Insp1", "Insp3" -> 4; "Insp2", "Insp2" -> 0;
        "Insp2", "Home" -> 3; "Insp2", "Insp1" -> 2; "Insp2", "Insp3" -> 2;
        "Insp3", "Insp3" -> 0; "Insp3", "Home" -> 4; "Insp3", "Insp1" -> 4;
        "Insp3", "Insp2" -> 2}

    Location_Init = Home
    Height_Init = 0
    //DistanceToTarget_Init =
    //DistanceToHome_Init =
    DistanceToRA_Init = 2
    Picture_Taken_Init = {}
    //ClosestInspectionLocationToDo_Init =

    DetourAssumption = 2
}
```

Note: turn InvariantCheckingAssumptions off in the model! (These try to enforce the static variant of the World Assumption.)

Output

```
Running verifications with the classical time theory.
Running verifications with the realistic battery.
Running verifications with the dynamic low power bound.
================================================
Model(s) exist: true
Duration: 14 seconds.
================================================
World assumptions hold: false
Duration: 15 seconds.
================================================
Policy debug (pic height) correct: true
Duration: 16 seconds.
================================================
Policy debug (pic location) correct: true
Duration: 14 seconds.
================================================
Policy debug (move to TravelSpace) correct: true
Duration: 12 seconds.
================================================
Policy debug (landing only at home) correct: true
Duration: 29 seconds.
================================================
Goal property can hold: true
Duration: 22 seconds.
================================================
Goal property always holds: true
Duration: 18 seconds.
================================================
Restricted height property holds: true
Duration: 15 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 1: true
Duration: 29 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 2: true
Duration: 41 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 3: true
Duration: 35 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 4: true
Duration: 25 seconds.
================================================
The theories regarding reaching home are equal: true
Duration: 21 seconds.
================================================
Restricted area is never entered: true
Duration: 29 seconds.
================================================
Verifications completed after 335 seconds.
```

4 Low Battery Bound

The Structure and outputs for different experiments regarding the Low Battery bound.

Structure

```
structure World: V {
    Time = {0..30}

    Weight = 1

    Height = {0..3}
    GroundHeight = {0}
    InspectionHeight = {1}
```

```
    FlyHeight = {2}
    RestrictedHeight = {3}

    Power = {0..75}      // 0..75 0..65 0..55 0..20
    Battery_Init = 75    // 75 65 55 20
    PowerUsage = {0..5}
    MoveToPower = 3
    TakePicturePower = 2
    NoOpPower = 1
    LiftPower = 5
    LowerPower = 2
    StaticLowPower = 45  // 10 45
    CriticalPower = 5

    Distance = {0..5}
    Inspection = {"Insp1"}

    DistanceBetween =
        {"Home", "Home" -> 0; "Home", "Insp1" -> 2;
        "Insp1", "Insp1" -> 0; "Insp1", "Home" -> 2}

    Location_Init = Home
    Height_Init = 0
    //DistanceToTarget_Init =
    //DistanceToHome_Init =
    DistanceToRA_Init = 2
    Picture_Taken_Init = {}
    //ClosestInspectionLocationToDo_Init =

    DetourAssumption = 0
}
```

Output: Dynamic Bound, Battery Init 75

```
Running verifications with the classical time theory.
Running verifications with the realistic battery.
Running verifications with the dynamic low power bound.
================================================
Model(s) exist: true
Duration: 10 seconds.
================================================
World assumptions hold: true
Duration: 10 seconds.
================================================
Policy debug (pic height) correct: true
Duration: 12 seconds.
================================================
Policy debug (pic location) correct: true
Duration: 12 seconds.
================================================
Policy debug (move to TravelSpace) correct: true
Duration: 13 seconds.
================================================
Policy debug (landing only at home) correct: true
Duration: 18 seconds.
================================================
Goal property can hold: true
Duration: 14 seconds.
================================================
Goal property always holds: true
Duration: 14 seconds.
================================================
Restricted height property holds: true
Duration: 13 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 1: true
Duration: 15 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 2: true
Duration: 19 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 3: true
Duration: 17 seconds.
================================================
```

```
The drone will always reach Home (using battery, world,... from structure) 4: true
Duration: 18 seconds.
=============================================
The theories regarding reaching home are equal: true
Duration: 16 seconds.
=============================================
Restricted area is never entered: true
Duration: 14 seconds.
=============================================
Verifications completed after 215 seconds.
```

Output: Dynamic Bound, Battery Init 65

```
Running verifications with the classical time theory.
Running verifications with the realistic battery.
Running verifications with the dynamic low power bound.
=============================================
Model(s) exist: true
Duration: 10 seconds.
=============================================
World assumptions hold: true
Duration: 10 seconds.
=============================================
Policy debug (pic height) correct: true
Duration: 11 seconds.
=============================================
Policy debug (pic location) correct: true
Duration: 12 seconds.
=============================================
Policy debug (move to TravelSpace) correct: true
Duration: 14 seconds.
=============================================
Policy debug (landing only at home) correct: true
Duration: 15 seconds.
=============================================
Goal property can hold: true
Duration: 15 seconds.
=============================================
Goal property always holds: false
Duration: 12 seconds.
=============================================
Restricted height property holds: true
Duration: 12 seconds.
=============================================
The drone will always reach Home (using battery, world,... from structure) 1: true
Duration: 19 seconds.
=============================================
The drone will always reach Home (using battery, world,... from structure) 2: true
Duration: 17 seconds.
=============================================
The drone will always reach Home (using battery, world,... from structure) 3: true
Duration: 16 seconds.
=============================================
The drone will always reach Home (using battery, world,... from structure) 4: true
Duration: 15 seconds.
=============================================
The theories regarding reaching home are equal: true
Duration: 13 seconds.
=============================================
Restricted area is never entered: true
Duration: 15 seconds.
=============================================
Verifications completed after 206 seconds.
```

Output: Dynamic Bound, Battery Init 20

```
Running verifications with the classical time theory.
Running verifications with the realistic battery.
Running verifications with the dynamic low power bound.
=============================================
Model(s) exist: true
Duration: 10 seconds.
=============================================
```

```
World assumptions hold: true
Duration: 9 seconds.
=============================================
Policy debug (pic height) correct: true
Duration: 11 seconds.
=============================================
Policy debug (pic location) correct: true
Duration: 11 seconds.
=============================================
Policy debug (move to TravelSpace) correct: true
Duration: 13 seconds.
=============================================
Policy debug (landing only at home) correct: true
Duration: 13 seconds.
=============================================
Goal property can hold: false
Duration: 12 seconds.
=============================================
Goal property always holds: false
Duration: 11 seconds.
=============================================
Restricted height property holds: true
Duration: 13 seconds.
=============================================
The drone will always reach Home (using battery, world,... from structure) 1: true
Duration: 13 seconds.
=============================================
The drone will always reach Home (using battery, world,... from structure) 2: true
Duration: 12 seconds.
=============================================
The drone will always reach Home (using battery, world,... from structure) 3: true
Duration: 11 seconds.
=============================================
The drone will always reach Home (using battery, world,... from structure) 4: true
Duration: 12 seconds.
=============================================
The theories regarding reaching home are equal: true
Duration: 14 seconds.
=============================================
Restricted area is never entered: true
Duration: 11 seconds.
=============================================
Verifications completed after 176 seconds.
```

Output: Static Bound 10, Battery Init 65

```
Running verifications with the classical time theory.
Running verifications with the realistic battery.
Running verifications with the static low power bound.
=============================================
Model(s) exist: true
Duration: 9 seconds.
=============================================
World assumptions hold: true
Duration: 10 seconds.
=============================================
Policy debug (pic height) correct: true
Duration: 13 seconds.
=============================================
Policy debug (pic location) correct: true
Duration: 11 seconds.
=============================================
Policy debug (move to TravelSpace) correct: true
Duration: 11 seconds.
=============================================
Policy debug (landing only at home) correct: true
Duration: 13 seconds.
=============================================
Goal property can hold: true
Duration: 13 seconds.
=============================================
Goal property always holds: true
Duration: 12 seconds.
=============================================
```

```
Restricted height property holds: true
Duration: 15 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 1: true
Duration: 14 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 2: true
Duration: 14 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 3: true
Duration: 15 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 4: true
Duration: 14 seconds.
================================================
The theories regarding reaching home are equal: true
Duration: 14 seconds.
================================================
Restricted area is never entered: true
Duration: 14 seconds.
================================================
Verifications completed after 192 seconds.
```

Output: Static Bound 10, Battery Init 55

```
Running verifications with the classical time theory.
Running verifications with the realistic battery.
Running verifications with the static low power bound.
================================================
Model(s) exist: true
Duration: 9 seconds.
================================================
World assumptions hold: true
Duration: 10 seconds.
================================================
Policy debug (pic height) correct: true
Duration: 12 seconds.
================================================
Policy debug (pic location) correct: true
Duration: 12 seconds.
================================================
Policy debug (move to TravelSpace) correct: true
Duration: 11 seconds.
================================================
Policy debug (landing only at home) correct: false
Duration: 12 seconds.
================================================
Goal property can hold: true
Duration: 12 seconds.
================================================
Goal property always holds: false
Duration: 11 seconds.
================================================
Restricted height property holds: true
Duration: 13 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 1: false
Duration: 13 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 2: false
Duration: 12 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 3: false
Duration: 12 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 4: false
Duration: 14 seconds.
================================================
The theories regarding reaching home are equal: true
Duration: 14 seconds.
================================================
Restricted area is never entered: true
Duration: 12 seconds.
================================================
Verifications completed after 179 seconds.
```

Output: Static Bound 45, Battery Init 55

```
Running verifications with the classical time theory.
Running verifications with the realistic battery.
Running verifications with the static low power bound.
=============================================
Model(s) exist: true
Duration: 9 seconds.
=============================================
World assumptions hold: true
Duration: 12 seconds.
=============================================
Policy debug (pic height) correct: true
Duration: 10 seconds.
=============================================
Policy debug (pic location) correct: true
Duration: 13 seconds.
=============================================
Policy debug (move to TravelSpace) correct: true
Duration: 13 seconds.
=============================================
Policy debug (landing only at home) correct: true
Duration: 11 seconds.
=============================================
Goal property can hold: false
Duration: 12 seconds.
=============================================
Goal property always holds: false
Duration: 14 seconds.
=============================================
Restricted height property holds: true
Duration: 12 seconds.
=============================================
The drone will always reach Home (using battery, world,... from structure) 1: true
Duration: 12 seconds.
=============================================
The drone will always reach Home (using battery, world,... from structure) 2: true
Duration: 16 seconds.
=============================================
The drone will always reach Home (using battery, world,... from structure) 3: true
Duration: 11 seconds.
=============================================
The drone will always reach Home (using battery, world,... from structure) 4: true
Duration: 12 seconds.
=============================================
The theories regarding reaching home are equal: true
Duration: 13 seconds.
=============================================
Restricted area is never entered: true
Duration: 13 seconds.
=============================================
Verifications completed after 183 seconds.
```

5 Aggregate Function versus Inductive Definition

5.1 Smaller World

Structure

```
structure World: V {
    Time = {0..35}

    Weight = 0

    Height = {0..5}
    GroundHeight = {0}
    InspectionHeight = {1..2}
```

```
    FlyHeight = {3..4}
    RestrictedHeight = {5}

    Power = {0..50}
    Battery_Init = 50
    PowerUsage = {0..5}
    MoveToPower = 3
    TakePicturePower = 2
    NoOpPower = 1
    LiftPower = 5
    LowerPower = 2
    StaticLowPower = 20
    CriticalPower = 5

    Distance = {0..3}
    Inspection = {"Insp1"}
    DistanceBetween =
        {"Home", "Home" -> 0; "Insp1", "Insp1" -> 0;
        "Home", "Insp1" -> 2; "Insp1", "Home" -> 2}

    Location_Init = Home
    Height_Init = 0
    //DistanceToTarget_Init =
    //DistanceToHome_Init =
    DistanceToRA_Init = 2
    Picture_Taken_Init = {}
    //ClosestInspectionLocationToDo_Init =

    DetourAssumption = 0
}
```

5.2 Larger World

Structure

```
structure World: V {
    Time = {0..100}

    Weight = 0

    Height = {0..8}
    GroundHeight = {0}
    InspectionHeight = {1..3}
    FlyHeight = {4..7}
    RestrictedHeight = {8}

    Power = {0..150}
    Battery_Init = 150
    PowerUsage = {0..5}
    MoveToPower = 3
    TakePicturePower = 2
    NoOpPower = 1
    LiftPower = 5
    LowerPower = 2
    StaticLowPower = 20
    CriticalPower = 5

    Distance = {0..15}
    Inspection = {"Insp1"; "Insp2"; "Insp3"; "Insp4"}
    DistanceBetween =
        {"Home", "Home" -> 0; "Home", "Insp1" -> 2; "Home", "Insp2" -> 7;
        "Home", "Insp3" -> 9; "Home", "Insp4" -> 11;
        "Insp1", "Insp1" -> 0; "Insp1", "Home" -> 2; "Insp1", "Insp2" -> 3;
        "Insp1", "Insp3" -> 4; "Insp1", "Insp4" -> 5;
        "Insp2", "Insp2" -> 0; "Insp2", "Home" -> 7; "Insp2", "Insp1" -> 3;
        "Insp2", "Insp3" -> 8; "Insp2", "Insp4" -> 10;
        "Insp3", "Insp3" -> 0; "Insp3", "Home" -> 9; "Insp3", "Insp1" -> 4;
        "Insp3", "Insp2" -> 8; "Insp3", "Insp4" -> 14;
        "Insp4", "Insp4" -> 0; "Insp4", "Home" -> 11; "Insp4", "Insp1" -> 5;
        "Insp4", "Insp2" -> 10; "Insp4", "Insp3" -> 14}

    Location_Init = Home
    Height_Init = 0
    //DistanceToTarget_Init =
```

```
    //DistanceToHome_Init =
    DistanceToRA_Init = 5
    Picture_Taken_Init = {}
    //ClosestInspectionLocationToDo_Init =

    DetourAssumption = 3
}
```

6 Back Loop Experiments

6.1 Infinite Take-Off Loop

Without setting a limit on the Non-Determinism regarding failure of actions, a scenario exists where the drone will never successfully execute its first action.

Structure

```
structure World: V {
    Time = {0..30}

    Weight = 0

    Height = {0..5}
    GroundHeight = {0}
    InspectionHeight = {1..2}
    FlyHeight = {3..4}
    RestrictedHeight = {5}

    Power = {0..40}
    Battery_Init = 40
    PowerUsage = {0..5}
    MoveToPower = 3
    TakePicturePower = 2
    NoOpPower = 1
    LiftPower = 5
    LowerPower = 2
    StaticLowPower = 25
    CriticalPower = 5

    Distance = {0..5}
    Inspection = {"Insp1"}
    DistanceBetween =
        {"Home", "Home" -> 0; "Home", "Insp1" -> 2;
        "Insp1", "Insp1" -> 0; "Insp1", "Home" -> 2}

    Location_Init = Home
    Height_Init = 0
    //DistanceToTarget_Init =
    //DistanceToHome_Init =
    DistanceToRA_Init = 2
    Picture_Taken_Init = {}
    //ClosestInspectionLocationToDo_Init =

    DetourAssumption = 1
}
```

Note: remove the bound on non-determinism from the model. (I.e. weight = 0 is not used, but is interpreted as infinity instead!)

Output Model

Relevant to denote here is that:

- The back loop time theory is used.
- The dummy (infinite) battery is used.
- Curr_Location and Curr_Height never get changed.
- The Plan (thus the planning policy) does keep trying to lift the drone.

```
Running verifications with the back loop time theory.
Running verifications with the dummy battery.
Running verifications with the static low power bound.
Model 1
=======
structure  : V {
  Distance = { 0..5 }
  FlyHeight = { 3..4 }
  GroundHeight = { 0..0 }
  Height = { 0..5 }
  Inspection = { Insp1 }
  InspectionHeight = { 1..2 }
  Power = { 0..40 }
  PowerUsage = { 0..5 }
  RestrictedHeight = { 5..5 }
  Time = { 0..30 }
  AllPicturesTaken = {  }
  AtCriticalPower = {  }
  AtFlyHeight = {  }
  AtGroundHeight = { 0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15;
                     16; 17; 18; 19; 20; 21; 22; 23; 24; 25; 26; 27; 28; 29; 30 }
  AtHome = { 0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15;
             16; 17; 18; 19; 20; 21; 22; 23; 24; 25; 26; 27; 28; 29; 30 }
  AtInspection = {  }
  AtInspectionHeight = {  }
  AtInspectionToDo = {  }
  AtLowPower = {  }
  AtNormalPower = { 0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15;
                    16; 17; 18; 19; 20; 21; 22; 23; 24; 25; 26; 27; 28; 29; 30 }
  AtRestrictedHeight = {  }
  AtTravelSpace = {  }
  CN_DistanceToHome = {  }
  CN_DistanceToRA = {  }
  CN_DistanceToTarget = {  }
  CN_Height = {  }
  CN_Location = {  }
  C_DistanceToHome = {  }
  C_DistanceToRA = {  }
  C_DistanceToTarget = {  }
  C_Height = {  }
  C_Location = {  }
  C_Picture_Taken = {  }
  CloseToRA = {  }
  NonDetDecreaseDistRA = {  }
  NonDetIncreaseDistRA = {  }
  NonDetMove = {  }
  Picture_Taken = {  }
  Picture_Taken_Init = {  }
  BackLoopTime = 2
  Battery = { 0->40; 1->40; 2->40; 3->40; 4->40; 5->40; 6->40; 7->40; 8->40;
              9->40; 10->40; 11->40; 12->40; 13->40; 14->40; 15->40; 16->40;
              17->40; 18->40; 19->40; 20->40; 21->40; 22->40; 23->40; 24->40;
              25->40; 26->40; 27->40; 28->40; 29->40; 30->40 }
  Battery_Init = 40
```

```
  ClosestInspectionLocationToDo =
    { 0->Insp1; 1->Insp1; 2->Insp1; 3->Insp1;
      4->Insp1; 5->Insp1; 6->Insp1; 7->Insp1; 8->Insp1; 9->Insp1; 10->Insp1;
      11->Insp1; 12->Insp1; 13->Insp1; 14->Insp1; 15->Insp1; 16->Insp1; 17->Insp1;
      18->Insp1; 19->Insp1; 20->Insp1; 21->Insp1; 22->Insp1; 23->Insp1; 24->Insp1;
      25->Insp1; 26->Insp1; 27->Insp1; 28->Insp1; 29->Insp1; 30->Insp1 }
  ClosestInspectionLocationToDo_Init = Insp1
  CriticalPower = 5
  Curr_DistanceToHome = { 0->0; 1->0; 2->0; 3->0; 4->0; 5->0; 6->0; 7->0; 8->0;
                          9->0; 10->0; 11->0; 12->0; 13->0; 14->0; 15->0; 16->0;
                          17->0; 18->0; 19->0; 20->0; 21->0; 22->0; 23->0; 24->0;
                          25->0; 26->0; 27->0; 28->0; 29->0; 30->0 }
  Curr_DistanceToRA = { 0->2; 1->2; 2->2; 3->2; 4->2; 5->2; 6->2; 7->2; 8->2;
                        9->2; 10->2; 11->2; 12->2; 13->2; 14->2; 15->2; 16->2;
                        17->2; 18->2; 19->2; 20->2; 21->2; 22->2; 23->2; 24->2;
                        25->2; 26->2; 27->2; 28->2; 29->2; 30->2 }
  Curr_DistanceToTarget = { 0->2; 1->2; 2->2; 3->2; 4->2; 5->2; 6->2; 7->2; 8->2;
                            9->2; 10->2; 11->2; 12->2; 13->2; 14->2; 15->2; 16->2;
                            17->2; 18->2; 19->2; 20->2; 21->2; 22->2; 23->2; 24->2;
                            25->2; 26->2; 27->2; 28->2; 29->2; 30->2 }
  Curr_Height = { 0->0; 1->0; 2->0; 3->0; 4->0; 5->0; 6->0; 7->0; 8->0; 9->0;
                  10->0; 11->0; 12->0; 13->0; 14->0; 15->0; 16->0; 17->0; 18->0;
                  19->0; 20->0; 21->0; 22->0; 23->0; 24->0; 25->0; 26->0; 27->0;
                  28->0; 29->0; 30->0 }
  Curr_Location = { 0->Home; 1->Home; 2->Home; 3->Home; 4->Home; 5->Home; 6->Home;
                    7->Home; 8->Home; 9->Home; 10->Home; 11->Home; 12->Home;
                    13->Home; 14->Home; 15->Home; 16->Home; 17->Home; 18->Home;
                    19->Home; 20->Home; 21->Home; 22->Home; 23->Home; 24->Home;
                    25->Home; 26->Home; 27->Home; 28->Home; 29->Home; 30->Home }
  DetourAssumption = 1
  DistanceBetween = { Home,Home->0; Home,Insp1->2; Insp1,Home->2; Insp1,Insp1->0 }
  DistanceToHome_Init = 0
  DistanceToRA_Init = 2
  DistanceToTarget_Init = 2
  Height_Init = 0
  LiftPower = 5
  Location_Init = Home
  LowPower = { 0->25; 1->25; 2->25; 3->25; 4->25; 5->25; 6->25; 7->25; 8->25;
               9->25; 10->25; 11->25; 12->25; 13->25; 14->25; 15->25; 16->25;
               17->25; 18->25; 19->25; 20->25; 21->25; 22->25; 23->25; 24->25;
               25->25; 26->25; 27->25; 28->25; 29->25; 30->25 }
  LowerPower = 2
  MoveToPower = 3
  Next = { 0->1; 1->2; 2->3; 3->4; 4->5; 5->6; 6->7; 7->8; 8->9; 9->10; 10->11;
           11->12; 12->13; 13->14; 14->15; 15->16; 16->17; 17->18; 18->19; 19->20;
           20->21; 21->22; 22->23; 23->24; 24->25; 25->26; 26->27; 27->28; 28->29;
           29->30; 30->2 }
  NoOpPower = 1
  Plan = { 0->Lift; 1->Lift; 2->Lift; 3->Lift; 4->Lift; 5->Lift; 6->Lift; 7->Lift;
           8->Lift; 9->Lift; 10->Lift; 11->Lift; 12->Lift; 13->Lift; 14->Lift;
           15->Lift; 16->Lift; 17->Lift; 18->Lift; 19->Lift; 20->Lift; 21->Lift;
           22->Lift; 23->Lift; 24->Lift; 25->Lift; 26->Lift; 27->Lift; 28->Lift;
           29->Lift; 30->Lift }
  PowerUsageAt = { 0->0; 1->0; 2->0; 3->0; 4->0; 5->0; 6->0; 7->0; 8->0; 9->0;
                   10->0; 11->0; 12->0; 13->0; 14->0; 15->0; 16->0; 17->0; 18->0;
                   19->0; 20->0; 21->0; 22->0; 23->0; 24->0; 25->0; 26->0; 27->0;
                   28->0; 29->0; 30->0 }
  Start = 0
  StaticLowPower = 25
  TakePicturePower = 2
  Target = { 0->Insp1; 1->Insp1; 2->Insp1; 3->Insp1; 4->Insp1; 5->Insp1; 6->Insp1;
             7->Insp1; 8->Insp1; 9->Insp1; 10->Insp1; 11->Insp1; 12->Insp1;
             13->Insp1; 14->Insp1; 15->Insp1; 16->Insp1; 17->Insp1; 18->Insp1;
             19->Insp1; 20->Insp1; 21->Insp1; 22->Insp1; 23->Insp1; 24->Insp1;
             25->Insp1; 26->Insp1; 27->Insp1; 28->Insp1; 29->Insp1; 30->Insp1 }
  Weight = 0
}
```

6.2 Infinite Loop after Landing

It is intuitive that, using the Back Loop time theory and realistic battery, the back loop time step has to occur after the drone's position and battery level are no longer changing.

Structure

```
structure World: V {
    Time = {0..20}

    Weight = 0

    Height = {0..5}
    GroundHeight = {0}
    InspectionHeight = {1..2}
    FlyHeight = {3..4}
    RestrictedHeight = {5}

    Power = {0..40}
    Battery_Init = 40
    PowerUsage = {0..5}
    MoveToPower = 3
    TakePicturePower = 2
    NoOpPower = 1
    LiftPower = 5
    LowerPower = 2
    StaticLowPower = 25
    CriticalPower = 5

    Distance = {0..5}
    Inspection = {"Insp1"}
    DistanceBetween =
        {"Home", "Home" -> 0; "Home", "Insp1" -> 2;
        "Insp1", "Insp1" -> 0; "Insp1", "Home" -> 2}

    Location_Init = Home
    Height_Init = 0
    //DistanceToTarget_Init =
    //DistanceToHome_Init =
    DistanceToRA_Init = 2
    Picture_Taken_Init = {}
    //ClosestInspectionLocationToDo_Init =

    DetourAssumption = 1
}
```

Output Model

Relevant to denote here is that:

- The back loop time theory is used.
- The realistic battery is used.
- After time step 17, Curr_Location, Curr_Height,... and **Battery** do not get changed anymore.
- The back loop time step is chosen (by IDP) at 17. (In fact, 18, 19 and 20 would have been possible too.)

```
Running verifications with the back loop time theory.
Running verifications with the realistic battery.
Running verifications with the dynamic low power bound.
Model 1
=======
structure  : V {
  Distance = { 0..5 }
  FlyHeight = { 3..4 }
  GroundHeight = { 0..0 }
  Height = { 0..5 }
  Inspection = { Insp1 }
  InspectionHeight = { 1..2 }
  Power = { 0..40 }
  PowerUsage = { 0..5 }
  RestrictedHeight = { 5..5 }
  Time = { 0..20 }
  AllPicturesTaken = {  }
  AtCriticalPower = { 12; 13; 14; 15; 16; 17; 18; 19; 20 }
  AtFlyHeight = { 3; 4; 5; 6; 7 }
  AtGroundHeight = { 0; 10; 11; 12; 13; 14; 15; 16; 17; 18; 19; 20 }
  AtHome = { 0; 1; 2; 3; 7; 8; 9; 10; 11; 12; 13; 14; 15; 16; 17; 18; 19; 20 }
  AtInspection = {  }
  AtInspectionHeight = { 1; 2; 8; 9 }
  AtInspectionToDo = {  }
  AtLowPower = { 4; 5; 6; 7; 8; 9; 10; 11 }
  AtNormalPower = { 0; 1; 2; 3 }
  AtRestrictedHeight = {  }
  AtTravelSpace = { 4; 5; 6 }
  CN_DistanceToHome = { 3,0; 3,2; 3,3; 3,4; 3,5; 4,0; 4,1; 4,3;
                        4,4; 4,5; 5,0; 5,2; 5,3; 5,4; 5,5 }
  CN_DistanceToRA = { 3,0; 3,2; 3,3; 3,4; 3,5; 4,0; 4,1; 4,3; 4,4; 4,5 }
  CN_DistanceToTarget = { 3,0; 3,2; 3,3; 3,4; 3,5; 4,0; 4,1; 4,3;
                          4,4; 4,5; 5,0; 5,2; 5,3; 5,4; 5,5 }
  CN_Height = { 0,0; 0,2; 0,3; 0,4; 0,5; 1,0; 1,1; 1,3; 1,4; 1,5; 2,0; 2,1; 2,2;
                2,4; 2,5; 7,0; 7,1; 7,3; 7,4; 7,5; 8,0; 8,2; 8,3; 8,4; 8,5; 9,1;
                9,2; 9,3; 9,4; 9,5 }
  CN_Location = { 3,Home; 3,Insp1; 6,Insp1; 6,TravelSpace }
  C_DistanceToHome = { 3,1; 4,2; 5,1 }
  C_DistanceToRA = { 3,1; 4,2 }
  C_DistanceToTarget = { 3,1; 4,2; 5,1 }
  C_Height = { 0,1; 1,2; 2,3; 7,2; 8,1; 9,0 }
  C_Location = { 3,TravelSpace; 6,Home }
  C_Picture_Taken = {  }
  CloseToRA = { 4 }
  NonDetDecreaseDistRA = { 3 }
  NonDetIncreaseDistRA = {  }
  NonDetMove = { 0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15;
                 16; 17; 18; 19; 20 }
  Picture_Taken = {  }
  Picture_Taken_Init = {  }
  BackLoopTime = 17
  Battery = { 0->40; 1->35; 2->30; 3->25; 4->22; 5->19; 6->16; 7->13;
              8->11; 9->9; 10->7; 11->6; 12->5; 13->4; 14->3; 15->2;
              16->1; 17->0; 18->0; 19->0; 20->0 }
  Battery_Init = 40
  ClosestInspectionLocationToDo =
    { 0->Insp1; 1->Insp1; 2->Insp1; 3->Insp1; 4->Insp1; 5->Insp1; 6->Insp1;
      7->Insp1; 8->Insp1; 9->Insp1; 10->Insp1; 11->Insp1; 12->Insp1; 13->Insp1;
      14->Insp1; 15->Insp1; 16->Insp1; 17->Insp1; 18->Insp1; 19->Insp1; 20->Insp1 }
  ClosestInspectionLocationToDo_Init = Insp1
  CriticalPower = 5
  Curr_DistanceToHome = { 0->0; 1->0; 2->0; 3->0; 4->1; 5->2; 6->1; 7->0;
                          8->0; 9->0; 10->0; 11->0; 12->0; 13->0; 14->0;
                          15->0; 16->0; 17->0; 18->0; 19->0; 20->0 }
  Curr_DistanceToRA = { 0->2; 1->2; 2->2; 3->2; 4->1; 5->2; 6->2; 7->2;
                        8->2; 9->2; 10->2; 11->2; 12->2; 13->2; 14->2;
                        15->2; 16->2; 17->2; 18->2; 19->2; 20->2 }
  Curr_DistanceToTarget = { 0->2; 1->2; 2->2; 3->2; 4->1; 5->2; 6->1; 7->0;
                            8->0; 9->0; 10->0; 11->0; 12->0; 13->0; 14->0;
                            15->0; 16->0; 17->0; 18->0; 19->0; 20->0 }
  Curr_Height = { 0->0; 1->1; 2->2; 3->3; 4->3; 5->3; 6->3; 7->3; 8->2;
                  9->1; 10->0; 11->0; 12->0; 13->0; 14->0; 15->0; 16->0;
                  17->0; 18->0; 19->0; 20->0 }
  Curr_Location = { 0->Home; 1->Home; 2->Home; 3->Home; 4->TravelSpace;
                    5->TravelSpace; 6->TravelSpace; 7->Home; 8->Home; 9->Home;
                    10->Home; 11->Home; 12->Home; 13->Home; 14->Home; 15->Home;
                    16->Home; 17->Home; 18->Home; 19->Home; 20->Home }
```

```
  DetourAssumption = 1
  DistanceBetween = { Home,Home->0; Home,Insp1->2; Insp1,Home->2; Insp1,Insp1->0 }
  DistanceToHome_Init = 0
  DistanceToRA_Init = 2
  DistanceToTarget_Init = 2
  Height_Init = 0
  LiftPower = 5
  Location_Init = Home
  LowPower = { 0->14; 1->16; 2->18; 3->20; 4->23; 5->26; 6->23; 7->20; 8->18;
               9->16; 10->14; 11->14; 12->14; 13->14; 14->14; 15->14; 16->14;
               17->14; 18->14; 19->14; 20->14 }
  LowerPower = 2
  MoveToPower = 3
  Next = { 0->1; 1->2; 2->3; 3->4; 4->5; 5->6; 6->7; 7->8; 8->9; 9->10;
           10->11; 11->12; 12->13; 13->14; 14->15; 15->16; 16->17; 17->18;
           18->19; 19->20; 20->17 }
  NoOpPower = 1
  Plan = { 0->Lift; 1->Lift; 2->Lift; 3->MoveTowardsTarget; 4->MoveAwayFromRA;
           5->MoveTowardsTarget; 6->MoveTowardsTarget; 7->Lower; 8->Lower;
           9->Lower; 10->NoOp; 11->NoOp; 12->NoOp; 13->NoOp; 14->NoOp;
           15->NoOp; 16->NoOp; 17->NoOp; 18->NoOp; 19->NoOp; 20->NoOp }
  PowerUsageAt = { 0->5; 1->5; 2->5; 3->3; 4->3; 5->3; 6->3; 7->2; 8->2;
                   9->2; 10->1; 11->1; 12->1; 13->1; 14->1; 15->1; 16->1;
                   17->0; 18->0; 19->0; 20->0 }
  Start = 0
  StaticLowPower = 25
  TakePicturePower = 2
  Target = { 0->Insp1; 1->Insp1; 2->Insp1; 3->Insp1; 4->Home; 5->Home;
             6->Home; 7->Home; 8->Home; 9->Home; 10->Home; 11->Home;
             12->Home; 13->Home; 14->Home; 15->Home; 16->Home; 17->Home;
             18->Home; 19->Home; 20->Home }
  Weight = 0
}
```

7 Comparing Power Usage (Subtype)

Structure

```
structure World: V {
    Time = {0..30}

    Weight = 1

    Height = {0..5}
    GroundHeight = {0}
    InspectionHeight = {1..2}
    FlyHeight = {3..4}
    RestrictedHeight = {5}

    Power = {0..40}
    Battery_Init = 40
    PowerUsage = {0..5}
    MoveToPower = 3
    TakePicturePower = 2
    NoOpPower = 1
    LiftPower = 5
    LowerPower = 2
    StaticLowPower = 5
    CriticalPower = 5

    Distance = {0..5}
    Inspection = {"Insp1"}
    DistanceBetween =
        {"Home", "Home" -> 0; "Home", "Insp1" -> 2;
        "Insp1", "Insp1" -> 0; "Insp1", "Home" -> 2}

    Location_Init = Home
    Height_Init = 0
    //DistanceToTarget_Init =
    //DistanceToHome_Init =
```

```
    DistanceToRA_Init = 2
    Picture_Taken_Init = {}
    //ClosestInspectionLocationToDo_Init =

    DetourAssumption = 1
}
```

8 Initializations

For the structures in this section, the *Init* lines (near the bottom of the structure) are most relevant. For Invariant Checking a very small Time domain is used.
Side-note: For Invariant Checking, not capable of checking any liveness aspects, the DetourAssumption is not relevant. The same holds for any verification of liveness properties in this setting.

8.1 Full Manual Initialization for Invariant Checking

The drone is initialized close to the Restricted Area. The verification makes sure that it is impossible that the drone ends up going into the Restricted Area.

Structure

```
structure World: V {
    Time = {0..1}

    Weight = 1

    Height = {0..10}
    GroundHeight = {0}
    InspectionHeight = {1..3}
    FlyHeight = {4..9}
    RestrictedHeight = {10}

    Power = {0..200}
    Battery_Init = 175
    PowerUsage = {0..5}
    MoveToPower = 3
    TakePicturePower = 2
    NoOpPower = 1
    LiftPower = 5
    LowerPower = 2
    StaticLowPower = 5
    CriticalPower = 5

    Distance = {0..7}
    Inspection = {"Insp1"; "Insp2"}
    DistanceBetween =
        {"Home", "Home" -> 0; "Home", "Insp1" -> 4; "Home", "Insp2" -> 6;
        "Insp1", "Insp1" -> 0; "Insp1", "Home" -> 4; "Insp1","Insp2" -> 5;
        "Insp2", "Insp2"-> 0; "Insp2","Insp1"-> 5; "Insp2","Home" -> 6}

    Location_Init = TravelSpace
    Height_Init = 6
    DistanceToTarget_Init = 3
    DistanceToHome_Init = 7
    DistanceToRA_Init = 1
    Picture_Taken_Init = {Insp1}
```

```
    ClosestInspectionLocationToDo_Init = Insp2

    //DetourAssumption =
}
```

Output

The verification *Restricted area is never entered* is relevant here.

```
Running verifications with the classical time theory.
Running verifications with the realistic battery.
Running verifications with the dynamic low power bound.
================================================
Model(s) exist: true
Duration: 21 seconds.
================================================
World assumptions hold: true
Duration: 19 seconds.
================================================
Policy debug (pic height) correct: true
Duration: 19 seconds.
================================================
Policy debug (pic location) correct: true
Duration: 21 seconds.
================================================
Policy debug (move to TravelSpace) correct: true
Duration: 20 seconds.
================================================
Policy debug (landing only at home) correct: true
Duration: 21 seconds.
================================================
Goal property can hold: false
Duration: 21 seconds.
================================================
Goal property always holds: false
Duration: 21 seconds.
================================================
Restricted height property holds: true
Duration: 21 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 1: false
Duration: 22 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 2: false
Duration: 20 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 3: false
Duration: 22 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 4: false
Duration: 22 seconds.
================================================
The theories regarding reaching home are equal: true
Duration: 23 seconds.
================================================
Restricted area is never entered: true
Duration: 23 seconds.
================================================
Verifications completed after 316 seconds.
```

8.2 IDP Initialization for Invariant Checking

Note that here, even the initialization of DistanceBetween is handed off to IDP.
Since the complete Initialization is now done by IDP, we are not bound to one specific scenario anymore, hence all safety properties are now relevant to be tested.

Structure

```
structure World: V {
    Time = {0..1}

    Weight = 1

    Height = {0..5}
    GroundHeight = {0}
    InspectionHeight = {1..2}
    FlyHeight = {3..4}
    RestrictedHeight = {5}

    Power = {0..250}
    Battery_Init = 250
    PowerUsage = {0..5}
    MoveToPower = 3
    TakePicturePower = 2
    NoOpPower = 1
    LiftPower = 5
    LowerPower = 2
    CriticalPower = 5

    Distance = {0..10}
    Inspection = {"Insp1"; "Insp2"}
    //DistanceBetween =

    //Location_Init =
    //Height_Init =
    //DistanceToTarget_Init =
    //DistanceToHome_Init =
    //DistanceToRA_Init =
    //Picture_Taken_Init =
    //ClosestInspectionLocationToDo_Init =

    //DetourAssumption =
}
```

Output

```
Running verifications with the classical time theory.
Running verifications with the realistic battery.
Running verifications with the dynamic low power bound.
================================================
Model(s) exist: true
Duration: 27 seconds.
================================================
World assumptions hold: true
Duration: 25 seconds.
================================================
Policy debug (pic height) correct: true
Duration: 28 seconds.
================================================
Policy debug (pic location) correct: true
Duration: 27 seconds.
================================================
Policy debug (move to TravelSpace) correct: true
Duration: 26 seconds.
================================================
Policy debug (landing only at home) correct: true
Duration: 28 seconds.
================================================
Goal property can hold: true
Duration: 29 seconds.
================================================
Goal property always holds: false
Duration: 29 seconds.
================================================
Restricted height property holds: true
Duration: 27 seconds.
================================================
The drone will always reach Home (using battery, world,... from structure) 1: false
Duration: 29 seconds.
================================================
```

```
The drone will always reach Home (using battery, world,... from structure) 2: false
Duration: 36 seconds.
=============================================
The drone will always reach Home (using battery, world,... from structure) 3: false
Duration: 36 seconds.
=============================================
The drone will always reach Home (using battery, world,... from structure) 4: false
Duration: 27 seconds.
=============================================
The theories regarding reaching home are equal: true
Duration: 28 seconds.
=============================================
Restricted area is never entered: true
Duration: 29 seconds.
=============================================
Verifications completed after 431 seconds.
```

8.3 IDP Initialization for Bounded Model Checking

This structure is equal to the one of the example world 1, minus the initializations. As this is a BMC setting (opposed to an IC approach), a larger Time domain is used.
Relevant to denote here is that IDP chooses the:

- Location_Init to be Insp2.
- Height_Init to be 1.

This position in the world would never be visited using the current Policy in a complete BMC setting.

Further, remark that:

- the drone starts with Battery level:
  Battery(0) = 30.
- the initial expected Power required to reach Home is:
  LowPower(0) = 29.
- the drone (nevertheless 29 ¡ 30) does not succeed in returning Home:
  Curr_Location(20) = TravelSpace.

Structure

```
structure World: V {
    Time = {0..20}

    Weight = 0

    Height = {0..5}
    GroundHeight = {0}
    InspectionHeight = {1..2}
    FlyHeight = {3..4}
    RestrictedHeight = {5}
```

```
    Power = {0..60}
    Battery_Init =30
    PowerUsage = {0..5}
    MoveToPower = 3
    TakePicturePower = 2
    NoOpPower = 1
    LiftPower = 5
    LowerPower = 2
    StaticLowPower = 25
    CriticalPower = 5

    Distance = {0..4}
    Inspection = {"Insp1"; "Insp2"}
    DistanceBetween =
        {"Home", "Home" -> 0; "Home", "Insp1" -> 2; "Home", "Insp2" -> 3;
        "Insp1", "Insp1" -> 0; "Insp1", "Home" -> 2; "Insp1","Insp2" -> 2;
        "Insp2", "Insp2"-> 0; "Insp2","Insp1"-> 2; "Insp2","Home" -> 3}

    //Location_Init =
    //Height_Init =
    //DistanceToTarget_Init =
    //DistanceToHome_Init =
    //DistanceToRA_Init =
    //Picture_Taken_Init =
    //ClosestInspectionLocationToDo_Init =

    DetourAssumption = 3
}
```

Output Model, Counter Example for Reaching Home

```
Running verifications with the classical time theory.
Running verifications with the realistic battery.
Running verifications with the dynamic low power bound.
Model 1
=======
structure  : V {
  Distance = { 0..4 }
  FlyHeight = { 3..4 }
  GroundHeight = { 0..0 }
  Height = { 0..5 }
  Inspection = { Insp1; Insp2 }
  InspectionHeight = { 1..2 }
  Power = { 0..60 }
  PowerUsage = { 0..5 }
  RestrictedHeight = { 5..5 }
  Time = { 0..20 }
  AllPicturesTaken = {  }
  AtCriticalPower = { 7; 8; 9; 10; 11; 12; 13; 14; 15; 16; 17; 18; 19; 20 }
  AtFlyHeight = { 2; 3; 4; 5; 6; 7 }
  AtGroundHeight = { 10; 11; 12; 13; 14; 15; 16; 17; 18; 19; 20 }
  AtHome = {  }
  AtInspection = { 0; 1; 2 }
  AtInspectionHeight = { 0; 1; 8; 9 }
  AtInspectionToDo = {  }
  AtLowPower = { 1; 2; 3; 4; 5; 6 }
  AtNormalPower = { 0 }
  AtRestrictedHeight = {  }
  AtTravelSpace = { 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15; 16; 17; 18; 19; 20 }
  CN_DistanceToHome = { 2,0; 2,1; 2,3; 2,4; 3,0; 3,2; 3,3; 3,4; 4,0; 4,1; 4,3; 4,4;
                        5,0; 5,2; 5,3; 5,4; 6,0; 6,1; 6,3; 6,4 }
  CN_DistanceToRA = { 2,0; 2,1; 2,3; 2,4; 3,0; 3,2; 3,3; 3,4; 4,0; 4,1; 4,3; 4,4;
                      5,0; 5,2; 5,3; 5,4; 6,0; 6,1; 6,3; 6,4 }
  CN_DistanceToTarget = { 2,0; 2,1; 2,3; 2,4; 3,0; 3,2; 3,3; 3,4; 4,0; 4,1; 4,3;
                          4,4; 5,0; 5,2; 5,3; 5,4; 6,1; 6,2; 6,3; 6,4 }
  CN_Height = { 0,0; 0,1; 0,3; 0,4; 0,5; 1,0; 1,1; 1,2; 1,4; 1,5; 7,0; 7,1; 7,3;
                7,4; 7,5; 8,0; 8,2; 8,3; 8,4; 8,5; 9,1; 9,2; 9,3; 9,4; 9,5 }
  CN_Location = { 2,Home; 2,Insp1; 2,Insp2 }
  C_DistanceToHome = { 2,2; 3,1; 4,2; 5,1; 6,2 }
  C_DistanceToRA = { 2,2; 3,1; 4,2; 5,1; 6,2 }
  C_DistanceToTarget = { 2,2; 3,1; 4,2; 5,1; 6,0 }
  C_Height = { 0,2; 1,3; 7,2; 8,1; 9,0 }
  C_Location = { 2,TravelSpace }
  C_Picture_Taken = {  }
  CloseToRA = { 4; 6 }
```

```
NonDetDecreaseDistRA = { 2; 3; 4; 5; 6 }
NonDetIncreaseDistRA = {  }
NonDetMove = { 0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15; 16;
               17; 18; 19; 20 }
Picture_Taken = { 0,Home; 0,Insp2; 0,TravelSpace; 1,Home; 1,Insp2;
                  1,TravelSpace; 2,Home; 2,Insp2; 2,TravelSpace; 3,Home;
                  3,Insp2; 3,TravelSpace; 4,Home; 4,Insp2; 4,TravelSpace;
                  5,Home; 5,Insp2; 5,TravelSpace; 6,Home; 6,Insp2; 6,TravelSpace;
                  7,Home; 7,Insp2; 7,TravelSpace; 8,Home; 8,Insp2; 8,TravelSpace;
                  9,Home; 9,Insp2; 9,TravelSpace; 10,Home; 10,Insp2;
                  10,TravelSpace; 11,Home; 11,Insp2; 11,TravelSpace; 12,Home;
                  12,Insp2; 12,TravelSpace; 13,Home; 13,Insp2; 13,TravelSpace;
                  14,Home; 14,Insp2; 14,TravelSpace; 15,Home; 15,Insp2;
                  15,TravelSpace; 16,Home; 16,Insp2; 16,TravelSpace; 17,Home;
                  17,Insp2; 17,TravelSpace; 18,Home; 18,Insp2; 18,TravelSpace;
                  19,Home; 19,Insp2; 19,TravelSpace; 20,Home; 20,Insp2;
                  20,TravelSpace }
Picture_Taken_Init = { Home; Insp2; TravelSpace }
BackLoopTime = 0
Battery = { 0->30; 1->25; 2->20; 3->17; 4->14; 5->11; 6->8; 7->5; 8->3; 9->1;
            10->0; 11->0; 12->0; 13->0; 14->0; 15->0; 16->0; 17->0; 18->0;
            19->0; 20->0 }
Battery_Init = 30
ClosestInspectionLocationToDo =
  { 0->Insp1; 1->Insp1; 2->Insp1; 3->Insp1; 4->Insp1; 5->Insp1; 6->Insp1;
    7->Insp1; 8->Insp1; 9->Insp1; 10->Insp1; 11->Insp1; 12->Insp1; 13->Insp1;
    14->Insp1; 15->Insp1; 16->Insp1; 17->Insp1; 18->Insp1; 19->Insp1; 20->Insp1 }
ClosestInspectionLocationToDo_Init = Insp1
CriticalPower = 5
Curr_DistanceToHome = { 0->3; 1->3; 2->3; 3->2; 4->1; 5->2; 6->1; 7->2; 8->2;
                        9->2; 10->2; 11->2; 12->2; 13->2; 14->2; 15->2; 16->2;
                        17->2; 18->2; 19->2; 20->2 }
Curr_DistanceToRA = { 0->3; 1->3; 2->3; 3->2; 4->1; 5->2; 6->1; 7->2; 8->2;
                      9->2; 10->2; 11->2; 12->2; 13->2; 14->2; 15->2; 16->2;
                      17->2; 18->2; 19->2; 20->2 }
Curr_DistanceToTarget = { 0->2; 1->3; 2->3; 3->2; 4->1; 5->2; 6->1; 7->0;
                          8->0; 9->0; 10->0; 11->0; 12->0; 13->0; 14->0;
                          15->0; 16->0; 17->0; 18->0; 19->0; 20->0 }
Curr_Height = { 0->1; 1->2; 2->3; 3->3; 4->3; 5->3; 6->3; 7->3; 8->2; 9->1;
                10->0; 11->0; 12->0; 13->0; 14->0; 15->0; 16->0; 17->0;
                18->0; 19->0; 20->0 }
Curr_Location = { 0->Insp2; 1->Insp2; 2->Insp2; 3->TravelSpace;
                  4->TravelSpace; 5->TravelSpace; 6->TravelSpace;
                  7->TravelSpace; 8->TravelSpace; 9->TravelSpace;
                  10->TravelSpace; 11->TravelSpace; 12->TravelSpace;
                  13->TravelSpace; 14->TravelSpace; 18->TravelSpace;
                  19->TravelSpace; 20->TravelSpace }
DetourAssumption = 3
DistanceBetween = { Home,Home->0; Home,Insp1->2; Home,Insp2->3;
                    Insp1,Home->2; Insp1,Insp1->0; Insp1,Insp2->2;
                    Insp2,Home->3; Insp2,Insp1->2; Insp2,Insp2->0 }
DistanceToHome_Init = 3
DistanceToRA_Init = 3
DistanceToTarget_Init = 2
Height_Init = 1
LiftPower = 5
Location_Init = Insp2
LowPower = { 0->29; 1->31; 2->33; 3->30; 4->27; 5->30; 6->27; 7->30; 8->28;
             9->26; 10->24; 11->24; 12->24; 13->24; 14->24; 15->24; 16->24;
             17->24; 18->24; 19->24; 20->24 }
LowerPower = 2
MoveToPower = 3
Next = { 0->1; 1->2; 2->3; 3->4; 4->5; 5->6; 6->7; 7->8; 8->9; 9->10;
         10->11; 11->12; 12->13; 13->14; 14->15; 15->16; 16->17; 17->18;
         18->19; 19->20 }
NoOpPower = 1
Plan = { 0->Lift; 1->Lift; 2->MoveTowardsTarget; 3->MoveTowardsTarget;
         4->MoveAwayFromRA; 5->MoveTowardsTarget; 6->MoveAwayFromRA;
         7->Lower; 8->Lower; 9->Lower; 10->NoOp; 11->NoOp; 12->NoOp;
         13->NoOp; 14->NoOp; 15->NoOp; 16->NoOp; 17->NoOp; 18->NoOp;
         19->NoOp; 20->NoOp }
PowerUsageAt = { 0->5; 1->5; 2->3; 3->3; 4->3; 5->3; 6->3; 7->2;
                 8->2; 9->1; 10->0; 11->0; 12->0; 13->0; 14->0;
                 15->0; 16->0; 17->0; 18->0; 19->0; 20->0 }
Start = 0
StaticLowPower = 25
TakePicturePower = 2
```

```
  Target = { 0->Insp1; 1->Home; 2->Home; 3->Home; 4->Home; 5->Home; 6->Home;
             7->TravelSpace; 8->TravelSpace; 9->TravelSpace; 10->TravelSpace
             11->TravelSpace; 12->TravelSpace; 13->TravelSpace; 14->TravelSpace;
             15->TravelSpace; 16->TravelSpace; 17->TravelSpace; 18->TravelSpace;
             19->TravelSpace; 20->TravelSpace }
  Weight = 0
}
```