Mapping Safety Properties for Embedded Control Applications to Certifiably Correct Implementations

Andreas Stahlhofen, Dieter Zöbel

Institute of Software Technology, Research Group Real-Time Systems, University of Koblenz-Landau, Germany

astahlhofen@uni-koblenz.de
zoebel@uni-koblenz.de

May 12, 2014

Agenda

Motivation

Compactor Scenario

Reconsideration of the model

Case Study

Conclusion

Agenda

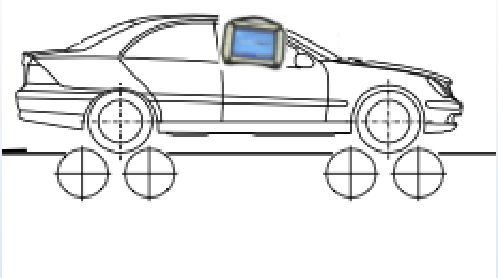
Motivation

Compactor Scenario

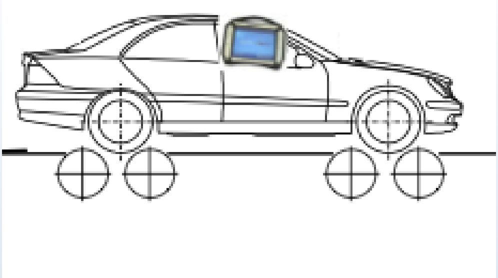
Reconsideration of the model

Case Study

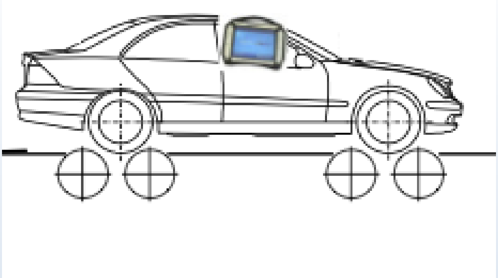
Conclusion



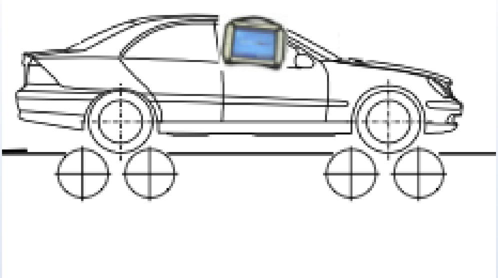
- Example of a system with real-time characteristics
- Challenge: Breaking at the right point in time, so that the tires stop between the rolls
- Problematic: communication delays and error-prone pose measurement of the car



- Example of a system with real-time characteristics
- Challenge: Breaking at the right point in time, so that the tires stop between the rolls
- Problematic: communication delays and error-prone pose measurement of the car



- Example of a system with real-time characteristics
- Challenge: Breaking at the right point in time, so that the tires stop between the rolls
- Problematic: communication delays and error-prone pose measurement of the car



- Example of a system with real-time characteristics
- Challenge: Breaking at the right point in time, so that the tires stop between the rolls
- Problematic: communication delays and error-prone pose measurement of the car

What is the problematic of developing a safety-critical embedded system with real-time characteristics?

- You have to verify and also certify the correct behaviour
- There exists many formal approaches on the verification of embedded control systems [1] [2]
- The physical or technical system is mapped to a context-specific model

BUT ...

- ... verifying safety properties within the model only hold at modeling level
- ... on implementation level, you have to „reverify“

Goal

Refinement of the context-specific model, so that the verification of its safety properties also holds at the implementation level.

What is the problematic of developing a safety-critical embedded system with real-time characteristics?

- You have to verify and also certify the correct behaviour
- There exists many formal approaches on the verification of embedded control systems [1] [2]
- The physical or technical system is mapped to a context-specific model

BUT ...

- ... verifying safety properties within the model only hold at modeling level
- ... on implementation level, you have to „reverify“

Goal

Refinement of the context-specific model, so that the verification of its safety properties also holds at the implementation level.

What is the problematic of developing a safety-critical embedded system with real-time characteristics?

- You have to verify and also certify the correct behaviour
- There exists many formal approaches on the verification of embedded control systems [1] [2]
- The physical or technical system is mapped to a context-specific model

BUT ...

- ... verifying safety properties within the model only hold at modeling level
- ... on implementation level, you have to „reverify“

Goal

Refinement of the context-specific model, so that the verification of its safety properties also holds at the implementation level.

What is the problematic of developing a safety-critical embedded system with real-time characteristics?

- You have to verify and also certify the correct behaviour
- There exists many formal approaches on the verification of embedded control systems [1] [2]
- The physical or technical system is mapped to a context-specific model

BUT ...

- ... verifying safety properties within the model only hold at modeling level
- ... on implementation level, you have to „reverify“

Goal

Refinement of the context-specific model, so that the verification of its safety properties also holds at the implementation level.

What is the problematic of developing a safety-critical embedded system with real-time characteristics?

- You have to verify and also certify the correct behaviour
- There exists many formal approaches on the verification of embedded control systems [1] [2]
- The physical or technical system is mapped to a context-specific model

BUT ...

- ... verifying safety properties within the model only hold at modeling level
- ... on implementation level, you have to „reverify“

Goal

Refinement of the context-specific model, so that the verification of its safety properties also holds at the implementation level.

What is the problematic of developing a safety-critical embedded system with real-time characteristics?

- You have to verify and also certify the correct behaviour
- There exists many formal approaches on the verification of embedded control systems [1] [2]
- The physical or technical system is mapped to a context-specific model

BUT ...

- ... verifying safety properties within the model only hold at modeling level
- ... on implementation level, you have to „reverify“

Goal

Refinement of the context-specific model, so that the verification of its safety properties also holds at the implementation level.

What is the problematic of developing a safety-critical embedded system with real-time characteristics?

- You have to verify and also certify the correct behaviour
- There exists many formal approaches on the verification of embedded control systems [1] [2]
- The physical or technical system is mapped to a context-specific model

BUT ...

- ... verifying safety properties within the model only hold at modeling level
- ... on implementation level, you have to „reverify“

Goal

Refinement of the context-specific model, so that the verification of its safety properties also holds at the implementation level.

What is the problematic of developing a safety-critical embedded system with real-time characteristics?

- You have to verify and also certify the correct behaviour
- There exists many formal approaches on the verification of embedded control systems [1] [2]
- The physical or technical system is mapped to a context-specific model

BUT ...

- ... verifying safety properties within the model only hold at modeling level
- ... on implementation level, you have to „reverify“

Goal

Refinement of the context-specific model, so that the verification of its safety properties also holds at the implementation level.

Agenda

Motivation

Compactor Scenario

Reconsideration of the model

Case Study

Conclusion

Compactor Scenario I

- Approach to formulate a formal model for collision avoidance in the context of autonomous driving [5]
- A one-dimensional robotic system between two objects
- One of the objects moves with a constant velocity towards the robotic system

Question

Under which conditions will the robotic system not collide with the moving object?

Compactor Scenario I

- Approach to formulate a formal model for collision avoidance in the context of autonomous driving [5]
- A one-dimensional robotic system between two objects
- One of the objects moves with a constant velocity towards the robotic system

Question

Under which conditions will the robotic system not collide with the moving object?

Compactor Scenario I

- Approach to formulate a formal model for collision avoidance in the context of autonomous driving [5]
- A one-dimensional robotic system between two objects
- One of the objects moves with a constant velocity towards the robotic system

Question

Under which conditions will the robotic system not collide with the moving object?

Compactor Scenario I

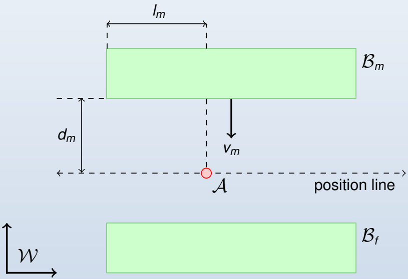
- Approach to formulate a formal model for collision avoidance in the context of autonomous driving [5]
- A one-dimensional robotic system between two objects
- One of the objects moves with a constant velocity towards the robotic system

Question

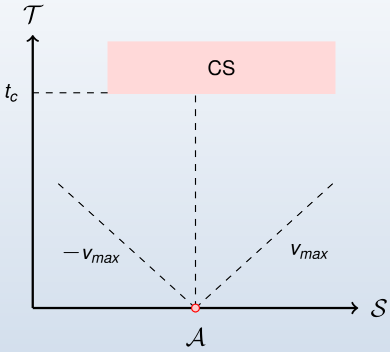
Under which conditions will the robotic system not collide with the moving object?

Compactor Scenario II

- $\mathcal{W}$: workspace
- $\mathcal{A}$: robotic system
- $\mathcal{B}_f$: static object
- $\mathcal{B}_m$: moving object
- v_m: velocity of the moving object $\mathcal{B}_m$
- d_m: distance between $\mathcal{A}$ and $\mathcal{B}_m$
- l_m: minimal escape distance

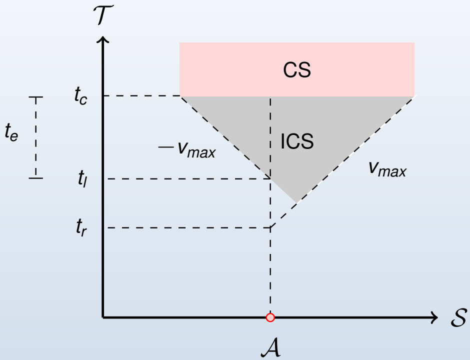


Compactor Scenario III



- t_c: time to collision
- t_l / t_r: last possible time to escape in left/right direction.
- t_e: time to escape using the minimal escape route).

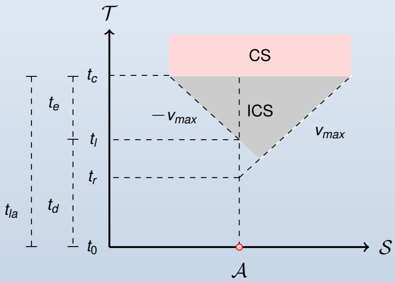
Compactor Scenario III



- t_c: time to collision
- t_l / t_r: last possible time to escape in left/right direction.
- t_e: time to escape using the minimal escape route).

Compactor Scenario IV

- t_d: time to decide about the minimal escape route
- t_{la}: lookahead time

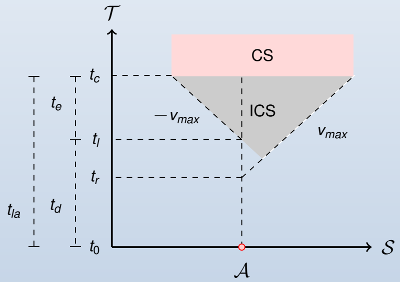


Compactor Scenario IV

- t_d: time to decide about the minimal escape route
- t_{la}: lookahead time

Constraint

A collision is avoided, if the following constraint holds: $t_d \leq t_c - t_e$



Agenda

Motivation

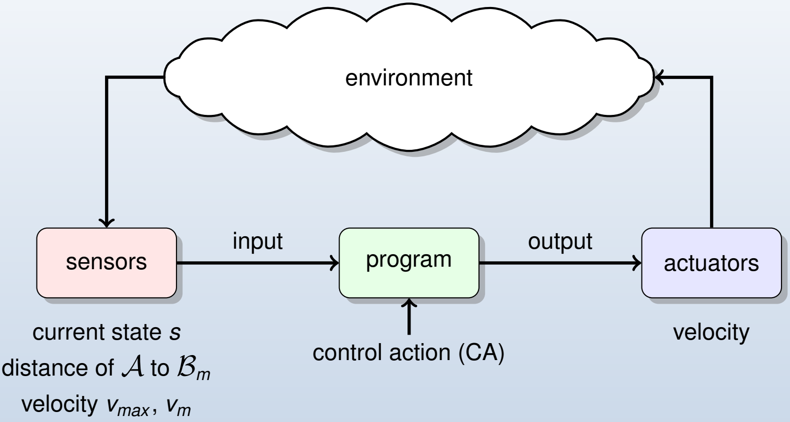
Compactor Scenario

Reconsideration of the model

Case Study

Conclusion

How can we map the model to an implementation?



How can we map the model to an implementation?

- The measured values of the sensors are used to calculate a correct control action.
- The actuators offer the interface to implement this control action within the environment.
- The important question, which arises is:

Question

How looks the implementation of the control action?

Answer?

```
if ( t_d > t_c - t_e ) {
  // Collision occurs eventually
}
```

How can we map the model to an implementation?

- The measured values of the sensors are used to calculate a correct control action.
- The actuators offer the interface to implement this control action within the environment.
- The important question, which arises is:

Question

How looks the implementation of the control action?

Answer?

```
If ( t_d > t_c - t_e ) {
  // Collision occurs eventually
}
```

How can we map the model to an implementation?

- The measured values of the sensors are used to calculate a correct control action.
- The actuators offer the interface to implement this control action within the environment.
- The important question, which arises is:

Question

How looks the implementation of the control action?

Answer?

```
If ( t_d > t_c - t_e ) {
  // Collision occurs eventually
}
```

How can we map the model to an implementation?

- The measured values of the sensors are used to calculate a correct control action.
- The actuators offer the interface to implement this control action within the environment.
- The important question, which arises is:

Question

How looks the implementation of the control action?

Answer?

```
If ( t_d > t_c - t_e ) {
  // Collision occurs eventually
}
```

How can we map the model to an implementation?

- The measured values of the sensors are used to calculate a correct control action.
- The actuators offer the interface to implement this control action within the environment.
- The important question, which arises is:

Question

How looks the implementation of the control action?

Answer?

```
if ( t_d > t_c - t_e ) {
  // Collision occurs eventually
}
```

How can we map the model to an implementation?

- The measured values of the sensors are used to calculate a correct control action.
- The actuators offer the interface to implement this control action within the environment.
- The important question, which arises is:

Question

How looks the implementation of the control action?

Answer?

```
If ( t_d > t_c - t_e ) {
  // Collision occurs eventually
}
```

How can we map the model to an implementation?

- `t_c = d_m / v_m`
- `t_e = l_m / v_max`
- `t_d ?`

Problematic

- The measured values `d_m`, `v_m` and `d_e` are error-prone
- The measured values are ageing
- Setting the value `v_max` to the motors does not necessarily result in an exact movement of $\mathcal{A}$ with a velocity v_{max}
- It consumes time until the motor of $\mathcal{A}$ receives the command to drive into a specified direction

How can we map the model to an implementation?

- `t_c = d_m / v_m`
- `t_e = l_m / v_max`
- `t_d` ?

Problematic

- The measured values `d_m`, `v_m` and `d_e` are error-prone
- The measured values are ageing
- Setting the value `v_max` to the motors does not necessarily result in an exact movement of $\mathcal{A}$ with a velocity v_{max}
- It consumes time until the motor of $\mathcal{A}$ receives the command to drive into a specified direction

How can we map the model to an implementation?

- `t_c = d_m / v_m`
- `t_e = l_m / v_max`
- `t_d` ?

Problematic

- The measured values `d_m`, `v_m` and `d_e` are error-prone
- The measured values are ageing
- Setting the value `v_max` to the motors does not necessarily result in an exact movement of $\mathcal{A}$ with a velocity v_{max}
- It consumes time until the motor of $\mathcal{A}$ receives the command to drive into a specified direction

How can we map the model to an implementation?

- `t_c = d_m / v_m`
- `t_e = l_m / v_max`
- `t_d` ?

Problematic

- The measured values `d_m`, `v_m` and `d_e` are error-prone
- The measured values are ageing
- Setting the value `v_max` to the motors does not necessarily result in an exact movement of $\mathcal{A}$ with a velocity v_{max}
- It consumes time until the motor of $\mathcal{A}$ receives the command to drive into a specified direction

How can we map the model to an implementation?

- `t_c = d_m / v_m`
- `t_e = l_m / v_max`
- `t_d` ?

Problematic

- The measured values `d_m`, `v_m` and `d_e` are error-prone
- The measured values are ageing
- Setting the value `v_max` to the motors does not necessarily result in an exact movement of $\mathcal{A}$ with a velocity v_{max}
- It consumes time until the motor of $\mathcal{A}$ receives the command to drive into a specified direction

How can we map the model to an implementation?

- `t_c = d_m / v_m`
- `t_e = l_m / v_max`
- `t_d` ?

Problematic

- The measured values `d_m`, `v_m` and `d_e` are error-prone
- The measured values are ageing
- Setting the value `v_max` to the motors does not necessarily result in an exact movement of $\mathcal{A}$ with a velocity v_{max}
- It consumes time until the motor of $\mathcal{A}$ receives the command to drive into a specified direction

How can we map the model to an implementation?

- `t_c = d_m / v_m`
- `t_e = l_m / v_max`
- `t_d` ?

Problematic

- The measured values `d_m`, `v_m` and `d_e` are error-prone
- The measured values are ageing
- Setting the value `v_max` to the motors does not necessarily result in an exact movement of $\mathcal{A}$ with a velocity v_{max}
- It consumes time until the motor of $\mathcal{A}$ receives the command to drive into a specified direction

How can we map the model to an implementation?

- All these points mentioned before must be considered inside the model, so that the verification of safety properties holds at the implementation level

- Finally the are two different categories of refinement to include inside the model:
  1. errors in values
  2. [illegible] scheduling

Requirement

There is a need of a dedicated method to describe this kind of refinements.

How can we map the model to an implementation?

- All these points mentioned before must be considered inside the model, so that the verification of safety properties holds at the implementation level

- Finally the are two different categories of refinement to include inside the model:
  1. errors in values
  2. deviation in time, e.g. caused by process communication and process scheduling

Requirement

There is a need of a dedicated method to describe this kind of refinements.

How can we map the model to an implementation?

- All these points mentioned before must be considered inside the model, so that the verification of safety properties holds at the implementation level

- Finally the are two different categories of refinement to include inside the model:
  1. errors in values
  2. deviation in time, e.g. caused by process communication and process scheduling

Requirement

There is a need of a dedicated method to describe this kind of refinements.

How can we map the model to an implementation?

- All these points mentioned before must be considered inside the model, so that the verification of safety properties holds at the implementation level
- Finally the are two different categories of refinement to include inside the model:
  1. errors in values
  2. deviation in time, e.g. caused by process communication and process scheduling

Requirement

There is a need of a dedicated method to describe this kind of refinements.

How can we map the model to an implementation?

- All these points mentioned before must be considered inside the model, so that the verification of safety properties holds at the implementation level

- Finally the are two different categories of refinement to include inside the model:
  1. errors in values
  2. deviation in time, e.g. caused by process communication and process scheduling

Requirement

There is a need of a dedicated method to describe this kind of refinements.

Refinement of the model I

- As in [4] and [3] we divide two different entity types to represent the data in our system:
  1. Real-Time Entities, e.g. the current velocity of $\mathcal{B}_f$
  2. Observerd Entities, e.g. the measured velocity of $\mathcal{B}_f$
- Real-Time Entities are from the view of the technical system or the environment
- Observerd Entites are from the view of the implementation
- Sensors and actuators are the interfaces to transform Real-Time Entities to Observed Entities and vice versa
- A model which uses only the Real-Time Entities exists already
- The challenge is to develop the model from the view of the implementation, using the observed entities

Refinement of the model I

- As in [4] and [3] we divide two different entity types to represent the data in our system:
  1. Real-Time Entities, e.g. the current velocity of $\mathcal{B}_f$
  2. Observerd Entities, e.g. the measured velocity of $\mathcal{B}_f$
- Real-Time Entities are from the view of the technical system or the environment
- Observerd Entites are from the view of the implementation
- Sensors and actuators are the interfaces to transform Real-Time Entities to Observed Entities and vice versa
- A model which uses only the Real-Time Entities exists already
- The challenge is to develop the model from the view of the implementation, using the observed entities

Refinement of the model I

- As in [4] and [3] we divide two different entity types to represent the data in our system:
  1. Real-Time Entities, e.g. the current velocity of $\mathcal{B}_f$
  2. Observerd Entities, e.g. the measured velocity of $\mathcal{B}_f$
- Real-Time Entities are from the view of the technical system or the environment
- Observerd Entites are from the view of the implementation
- Sensors and actuators are the interfaces to transform Real-Time Entities to Observed Entities and vice versa
- A model which uses only the Real-Time Entities exists already
- The challenge is to develop the model from the view of the implementation, using the observed entities

Refinement of the model I

- As in [4] and [3] we divide two different entity types to represent the data in our system:
  1. Real-Time Entities, e.g. the current velocity of $\mathcal{B}_f$
  2. Observerd Entities, e.g. the measured velocity of $\mathcal{B}_f$
- Real-Time Entities are from the view of the technical system or the environment
- Observerd Entites are from the view of the implementation
- Sensors and actuators are the interfaces to transform Real-Time Entities to Observed Entities and vice versa
- A model which uses only the Real-Time Entities exists already
- The challenge is to develop the model from the view of the implementation, using the observed entities

Refinement of the model I

- As in [4] and [3] we divide two different entity types to represent the data in our system:
  1. Real-Time Entities, e.g. the current velocity of $\mathcal{B}_f$
  2. Observerd Entities, e.g. the measured velocity of $\mathcal{B}_f$
- Real-Time Entities are from the view of the technical system or the environment
- Observerd Entites are from the view of the implementation
- Sensors and actuators are the interfaces to transform Real-Time Entities to Observed Entities and vice versa
- A model which uses only the Real-Time Entities exists already
- The challenge is to develop the model from the view of the implementation, using the observed entities

Refinement of the model I

- As in [4] and [3] we divide two different entity types to represent the data in our system:
  1. Real-Time Entities, e.g. the current velocity of $\mathcal{B}_f$
  2. Observerd Entities, e.g. the measured velocity of $\mathcal{B}_f$
- Real-Time Entities are from the view of the technical system or the environment
- Observerd Entites are from the view of the implementation
- Sensors and actuators are the interfaces to transform Real-Time Entities to Observed Entities and vice versa
- A model which uses only the Real-Time Entities exists already
- The challenge is to develop the model from the view of the implementation, using the observed entities

Refinement of the model I

- As in [4] and [3] we divide two different entity types to represent the data in our system:
  1. Real-Time Entities, e.g. the current velocity of $\mathcal{B}_f$
  2. Observerd Entities, e.g. the measured velocity of $\mathcal{B}_f$
- Real-Time Entities are from the view of the technical system or the environment
- Observerd Entites are from the view of the implementation
- Sensors and actuators are the interfaces to transform Real-Time Entities to Observed Entities and vice versa
- A model which uses only the Real-Time Entities exists already
- The challenge is to develop the model from the view of the implementation, using the observed entities

Refinement of the model I

- As in [4] and [3] we divide two different entity types to represent the data in our system:
  1. Real-Time Entities, e.g. the current velocity of $\mathcal{B}_f$
  2. Observerd Entities, e.g. the measured velocity of $\mathcal{B}_f$
- Real-Time Entities are from the view of the technical system or the environment
- Observerd Entites are from the view of the implementation
- Sensors and actuators are the interfaces to transform Real-Time Entities to Observed Entities and vice versa
- A model which uses only the Real-Time Entities exists already
- The challenge is to develop the model from the view of the implementation, using the observed entities

Concept of the approach I

- First, we need an invariant I_{RT} representing the correctness of a safety property of our system
- Outgoing point of our calculation is the *CA* inside the implementation
- We divide between the set of ICS and ACS (Avoidable collision states).
- Every state inside ACS matches I_{RT}, so this states are safe

Concept of the approach I

- First, we need an invariant I_{RT} representing the correctness of a safety property of our system
- Outgoing point of our calculation is the *CA* inside the implementation
- We divide between the set of ICS and ACS (Avoidable collision states).
- Every state inside ACS matches I_{RT}, so this states are safe

Concept of the approach I

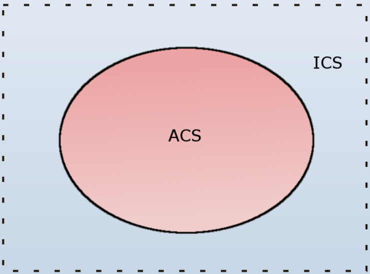
- First, we need an invariant I_{RT} representing the correctness of a safety property of our system
- Outgoing point of our calculation is the *CA* inside the implementation
- We divide between the set of ICS and ACS (Avoidable collision states).
- Every state inside ACS matches I_{RT}, so this states are safe

Concept of the approach I

- First, we need an invariant I_{RT} representing the correctness of a safety property of our system
- Outgoing point of our calculation is the *CA* inside the implementation
- We divide between the set of ICS and ACS (Avoidable collision states).
- Every state inside ACS matches I_{RT}, so this states are safe

Concept of the approach I

- First, we need an invariant I_{RT} representing the correctness of a safety property of our system
- Outgoing point of our calculation is the *CA* inside the implementation
- We divide between the set of ICS and ACS (Avoidable collision states).
- Every state inside ACS matches I_{RT}, so this states are safe



Concept of the approach II

Step 1: Include all time deviations inside the model:

- We have to look into the past, e.g. the age of the measurement of the velocity v_m of $\mathcal{B}_f$
- We have to look into the future, e.g. the time until $\mathcal{A}$ drives with the velocity v_{max} in the specified direction.
- Using only the worst possible values, e.g. the maximum age of a sensor value, we can calculate a new distance d_m from $\mathcal{A}$ towards $\mathcal{B}_f$, causes the size of the set of ACS to shrink

Concept of the approach II

Step 1: Include all time deviations inside the model:

- We have to look into the past, e.g. the age of the measurement of the velocity v_m of $\mathcal{B}_f$
- We have to look into the future, e.g. the time until $\mathcal{A}$ drives with the velocity v_{max} in the specified direction.
- Using only the worst possible values, e.g. the maximum age of a sensor value, we can calculate a new distance d_m from $\mathcal{A}$ towards $\mathcal{B}_f$, causes the size of the set of ACS to shrink

Concept of the approach II

Step 1: Include all time deviations inside the model:

- We have to look into the past, e.g. the age of the measurement of the velocity v_m of $\mathcal{B}_f$
- We have to look into the future, e.g. the time until $\mathcal{A}$ drives with the velocity v_{max} in the specified direction.
- Using only the worst possible values, e.g. the maximum age of a sensor value, we can calculate a new distance d_m from $\mathcal{A}$ towards $\mathcal{B}_f$, causes the size of the set of ACS to shrink

Concept of the approach II

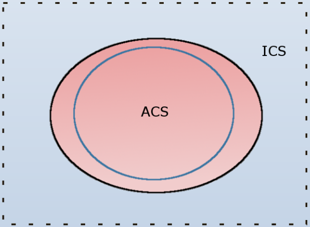
Step 1: Include all time deviations inside the model:

- We have to look into the past, e.g. the age of the measurement of the velocity v_m of $\mathcal{B}_f$
- We have to look into the future, e.g. the time until $\mathcal{A}$ drives with the velocity v_{max} in the specified direction.
- Using only the worst possible values, e.g. the maximum age of a sensor value, we can calculate a new distance d_m from $\mathcal{A}$ towards $\mathcal{B}_f$, causes the size of the set of ACS to shrink

Concept of the approach II

Step 1: Include all time deviations inside the model:

- We have to look into the past, e.g. the age of the measurement of the velocity v_m of $\mathcal{B}_f$
- We have to look into the future, e.g. the time until $\mathcal{A}$ drives with the velocity v_{max} in the specified direction.
- Using only the worst possible values, e.g. the maximum age of a sensor value, we can calculate a new distance d_m from $\mathcal{A}$ towards $\mathcal{B}_f$, causes the size of the set of ACS to shrink



Concept of the approach III

Step 2: Include all measurement errors inside the model:

- We have to regard the errors of the sensors, e.g. the deviation of the measured velocity of $\mathcal{B}_f$ is $\pm 2,5\%$
- We have to regard the errors of the actuators, e.g. the deviation of the calibrated velocity of $\mathcal{A}$ is $\pm 0,1\%$
- Using only the worst possible values, e.g. the minimum velocity of $\mathcal{A}$, causes the size of the set of ACS to shrink

Concept of the approach III

Step 2: Include all measurement errors inside the model:

- We have to regard the errors of the sensors, e.g. the deviation of the measured velocity of $\mathcal{B}_f$ is $\pm 2, 5\%$
- We have to regard the errors of the actuators, e.g. the deviation of the calibrated velocity of $\mathcal{A}$ is $\pm 0, 1\%$
- Using only the worst possible values, e.g. the minimum velocity of $\mathcal{A}$, causes the size of the set of ACS to shrink

Concept of the approach III

Step 2: Include all measurement errors inside the model:

- We have to regard the errors of the sensors, e.g. the deviation of the measured velocity of $\mathcal{B}_f$ is $\pm 2,5\%$
- We have to regard the errors of the actuators, e.g. the deviation of the calibrated velocity of $\mathcal{A}$ is $\pm 0,1\%$
- Using only the worst possible values, e.g. the minimum velocity of $\mathcal{A}$, causes the size of the set of ACS to shrink

Concept of the approach III

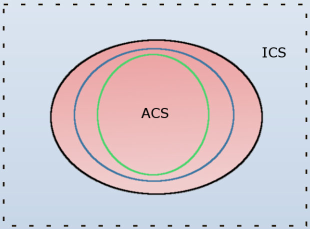
Step 2: Include all measurement errors inside the model:

- We have to regard the errors of the sensors, e.g. the deviation of the measured velocity of $\mathcal{B}_f$ is $\pm 2,5\%$
- We have to regard the errors of the actuators, e.g. the deviation of the calibrated velocity of $\mathcal{A}$ is $\pm 0,1\%$
- Using only the worst possible values, e.g. the minimum velocity of $\mathcal{A}$, causes the size of the set of ACS to shrink

Concept of the approach III

Step 2: Include all measurement errors inside the model:

- We have to regard the errors of the sensors, e.g. the deviation of the measured velocity of $\mathcal{B}_f$ is $\pm 2,5\%$
- We have to regard the errors of the actuators, e.g. the deviation of the calibrated velocity of $\mathcal{A}$ is $\pm 0,1\%$
- Using only the worst possible values, e.g. the minimum velocity of $\mathcal{A}$, causes the size of the set of ACS to shrink



Summary

- We transformed the Observed Entities back into Real-Time Entites
- We can check the invariant I_{RT} on our transformed Real-Time Entities
- Using only pessimistic transformations guarantees the correctness of the left states in ACS

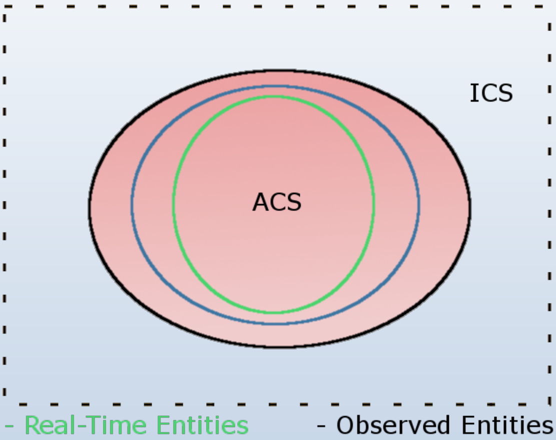
Summary

- We transformed the Observed Entities back into Real-Time Entites
- We can check the invariant I_{RT} on our transformed Real-Time Entities
- Using only pessimistic transformations guarantees the correctness of the left states in ACS

Summary

- We transformed the Observed Entities back into Real-Time Entites
- We can check the invariant I_{RT} on our transformed Real-Time Entities
- Using only pessimistic transformations guarantees the correctness of the left states in ACS

Summary



Concept of the approach IV

What is `t_d`?

`t_d` is the time from the first measurement of a value up to the time, at which the control action takes place inside the environment.

Agenda

Motivation

Compactor Scenario

Reconsideration of the model

Case Study

Conclusion

Instantiation of the model with example values

Description	Symbol	Value	Deviation	Age of the value
Velocity of the dynamic object	v_m	$4\frac{m}{s}$	$\pm 2,5\%$	$[90,\ 140]\ ms$
Distance of $\mathcal{A}$ towards $\mathcal{B}_f$	d_m	$8m$	$\pm 1,1\%$	$[50,\ 90]\ ms$
Minimal escape distance	d_e	$4m$	$\pm 1,5\%$	$[40,\ 130]\ ms$
Execution time of the computational system	Δe	$[15,31]\ ms$	-	-
Delay until the drive maneuver takes place	Δa_m	$[0,200]\ ms$	-	-
Maximal velocity of $\mathcal{A}$	v_{max}	$5\frac{m}{s}$	$\pm 0,1\%$	-

Result

Usage of the original model:

$$t_d \leq \Delta t_c - max(t_l, t_r) = 1s.$$

Regarding the age of the measured values:

$$t_d \leq \frac{dl''_m}{v'_m} - \frac{dh''_e}{v_{max}} = 0,458s$$

Regarding additionally errors of sensors and actuators:

$$t_d \leq \frac{odl_m}{ovh_m} - \frac{odh_e}{ovl_{max}} = 0.382s$$

Agenda

Motivation

Compactor Scenario

Reconsideration of the model

Case Study

Conclusion

Conclusion

- Manually we are able to derive all constituents which contribute to the correctness of the implementation.
- Starting with an invariant condition, the steps can be executed rather mechanically.
- The advantages for the programmer are obvisous: Any dependency is comprehensibly documented, verifiable and certifiable respecting the causal order.

Conclusion

- Manually we are able to derive all constituents which contribute to the correctness of the implementation.
- Starting with an invariant condition, the steps can be executed rather mechanically.
- The advantages for the programmer are obvisous: Any dependency is comprehensibly documented, verifiable and certifiable respecting the causal order.

Conclusion

- Manually we are able to derive all constituents which contribute to the correctness of the implementation.
- Starting with an invariant condition, the steps can be executed rather mechanically.
- The advantages for the programmer are obvisous: Any dependency is comprehensibly documented, verifiable and certifiable respecting the causal order.

Outlook

- Any of the mentioned steps are error-prone, so that we are working on tool support.
  - Guiding the user by some sort of syntactic view an asking for any parameter.
  - Giving a readable description of all relevant time- and value-dependent deviations.
- Extending the method to more flexibility.
- Determine correlations within the settings, e.g. changing the priority of the process on the implementation of the control action.

Outlook

- Any of the mentioned steps are error-prone, so that we are working on tool support.
  - Guiding the user by some sort of syntactic view an asking for any parameter.
  - Giving a readable description of all relevant time- and value-dependent deviations.
- Extending the method to more flexibility.
- Determine correlations within the settings, e.g. changing the priority of the process on the implementation of the control action.

Outlook

- Any of the mentioned steps are error-prone, so that we are working on tool support.
  - Guiding the user by some sort of syntactic view an asking for any parameter.
  - Giving a readable description of all relevant time- and value-dependent deviations.
- Extending the method to more flexibility.
- Determine correlations within the settings, e.g. changing the priority of the process on the implementation of the control action.

Outlook

- Any of the mentioned steps are error-prone, so that we are working on tool support.
  - Guiding the user by some sort of syntactic view an asking for any parameter.
  - Giving a readable description of all relevant time- and value-dependent deviations.
- Extending the method to more flexibility.
- Determine correlations within the settings, e.g. changing the priority of the process on the implementation of the control action.

Outlook

- Any of the mentioned steps are error-prone, so that we are working on tool support.
  - Guiding the user by some sort of syntactic view an asking for any parameter.
  - Giving a readable description of all relevant time- and value-dependent deviations.
- Extending the method to more flexibility.
- Determine correlations within the settings, e.g. changing the priority of the process on the implementation of the control action.

Thank you for your attention!

Literature I

[1] Tobias Amnell et al. "TIMES b—A Tool for Modelling and Implementation of Embedded Systems". In: *Tools and Algorithms for the Construction and Analysis of Systems* (2002), pp. 460–464. URL: http://link.springer.com/chapter/10.1007/3-540-46002-0_32 (cit. on pp. 9–16).

[2] B Becker, D Beyer, and H Giese. "Symbolic invariant verification for systems with dynamic structural adaptation". In: *ACM Proceedings of the 28th International conference on Software engineering* (2006), pp. 72–81. URL: http://dl.acm.org/citation.cfm?id=1134297 (cit. on pp. 9–16).

Literature II

[3] H Kopetz. "The time-triggered model of computation". In: *Proceedings of 19th IEEE Real-Time Systems Symposium, 1998.* (1998), pp. 168–177. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=739743 (cit. on pp. 47–54).

[4] H. Kopetz and K.H. Kim. "Temporal uncertainties in interactions among real-time objects". In: *Proceedings Ninth Symposium on Reliable Distributed Systems* (1990), pp. 165–174. DOI: 10.1109/RELDIS.1990.93962. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=93962 (cit. on pp. 47–54).

Literature III

[5] L. Martinez-Gomez and T. Fraichard. “Collision avoidance in dynamic environments: An ICS-based solution and its comparative evaluation”. In: *IEEE International Conference on Robotics and Automation* (May 2009), pp. 100–105. DOI: 10.1109/ROBOT.2009.5152536.