A Case Study of Systematic Top-down Design of Cyber-physical Models with Integrated Validation and Formal Verification

Christoph Luckeneder
Technische Universität Wien
Vienna, Austria
christoph.luckeneder@tuwien.ac.at

Hermann Kaindl
Technische Universität Wien
Vienna, Austria
hermann.kaindl@tuwien.ac.at

ABSTRACT

Abstract models are required to handle the complexity for designing and verifying large-scale systems. An open problem is to consistently and systematically derive a more concrete model from an abstract model with regard to verification of its behavior against certain properties. Based on our recently proposed workflow for systematic top-down design of models of a Cyber-physical System (CPS), we present an in-depth case study of Adaptive Cruise Control (ACC). It includes both *verification through model checking* and *validation* in the sense that a refined model is checked for its fit with reality. This approach works top-down for designing a concrete model by starting from an abstract model. The resulting concrete model was validated and indirectly verified in this case study. In addition, we made a cross-check by verifying it directly on the concrete level. Hence, our case study provides some empirical evidence on the feasibility of this new workflow for top-down design of models.

CCS CONCEPTS

• **Software and its engineering** → *Software design engineering*; *Formal software verification*;

KEYWORDS

Top-down design, formal verification, behavioral models, CPS

ACM Reference Format:
Christoph Luckeneder and Hermann Kaindl. 2019. A Case Study of Systematic Top-down Design of Cyber-physical, Models with Integrated Validation and Formal Verification. In *The 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19), April 8–12, 2019, Limassol, Cyprus.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3297280.3297460

1 INTRODUCTION

Complexity is a major reason for *designing* in a *top-down* fashion, by starting with an abstract model and refining it. Verification in the course of such a design approach often means checking lower-level artefacts against higher-level ones. We make use of behavioral verification against properties through model checking in such a context.

SAC '19, April 8–12, 2019, Limassol, Cyprus

ACM ISBN 978-1-4503-5933-7/19/04…$15.00
https://doi.org/10.1145/3297280.3297460

An approach to design and model-check an abstract qualitative model of a cyber-physical system (CPS) was proposed in [1]. More specifically, this was an *Adaptive Cruise Control* (ACC) model, studied from the perspective of a composition of features. It includes both *Cruise Control* (CC), as widely used in cars, and *Distance Control* (DC) of a vehicle A following another vehicle B. Since ACC per se and its inherent feature interactions are well understood already, it was possible for this work to focus on automated verification of feature coordination. A minimalist qualitative model of the behavior of ACC in the sense of a high-level representation of the ACC behavior was verified through model checking against a formula in temporal logic representing a rear-end collision accident.

In [2], we proposed a new workflow for systematic top-down design of cyber-physical models with integrated validation and formal verification. Their results guide model changes on different levels of abstraction. This workflow also uses refinement techniques from [3], but in a top-down design approach rather than bottom-up verification.

The major motivation for studying and using this workflow is the inherent guarantee on behavioral consistency between a higher- and a lower-level model. If the higher-level model satisfies certain properties, then the lower-level model is guaranteed to satisfy them as well. This is particularly important for top-down design of safety-critical systems.

In this paper, we present a case study of applying our systematic workflow to the abstract qualitative ACC model. It resulted in a concrete quantitative model, which we validated (primarily according to simple physics). According to the workflow, it was also formally verified indirectly. As a cross-check, we additionally verified it directly through model-checking.

The remainder of this paper is organized in the following manner. First, we present some general background material and related work, as well as the workflow, in order to keep this paper self-contained. Then we describe our case study, including lessons learned. In addition, we present a direct verification of the resulting concrete model. Finally, we discuss this approach more generally and conclude.

2 BACKGROUND

The top-down design approach used heavily relies on mappings from an abstract (and qualitative) to a concrete (and quantitative) model. Such a mapping does not only define several parameters of the concrete model but also an abstraction function from the concrete to the abstract model. It is key to preserving the statement made about a specific property checked in the abstract model, in our case whether a collision can occur or not. Hence, the mapping from

the abstract to the concrete model needs to define an abstraction function that guarantees this.

In this regard, we build on the theory of [3], more precisely its part on abstraction techniques. Clarke et al. distinguish between two systematic abstraction techniques, *over-* and *under-approximation*. These techniques are designed in a such a way that the information loss inherent in abstraction causes only one-sided error. Over-approximation and under-approximation techniques only cause false negatives and false positives, respectively.

Which type of approximation has to be used depends on the reasoning needed. Since we use over-approximation, let us focus on it as required background. Over-approximation guarantees that, if a temporal logic expression evaluates to true in an abstract model, then it is true in the concrete model as well. If it evaluates to false in the abstract model, however, no conclusion can be drawn for the concrete model in this regard. For determining whether the abstraction function defined by a mapping is an over-approximation, we strongly build on the framework of existential abstraction presented by Clarke et al. [4]. Intuitively, an abstract model is an over-approximation of a concrete model, if it allows for all the behavior of the latter and possibly more. In the course of an abstraction, states of the concrete model are clustered into abstract states. This may already lead to an increase of behavioral options through the transitions between clustered states in the abstract model. However, no transition in the abstract model must be removed so that a possible behavioral option in the concrete model is not available in the abstract model.

Assume-Guarantee Abstraction Refinement (AGAR) [5] is a variant of *Counterexample-guided Abstraction Refinement* (CEGAR) [3]. Both approaches make model-checking more efficient based on automated use of such abstraction techniques. While CEGAR is defined for transition systems and Kripke structures, AGAR is defined for labeled transition systems (LTS). Since Finite State Machines (FSMs) are used in our work, which are labeled transition systems, we can rely on the definitions in [5].

3 RELATED WORK

Since model checking intrinsically faces combinatorial explosion, abstraction techniques were studied, e.g., in [3, 4, 6], to reduce the state-space. These techniques provide a systematic way to generate an abstract model from a concrete one with one-sided error. In contrast to the approach used here, this work considers the concrete model as given and fixed. Hence, these techniques start from a given concrete model, while our workflow starts from an abstract model either given or defined in due course. Still, these abstraction techniques can be used in the course of enacting our workflow.

Clarke et al. [3, 4] distinguish in CEGAR between *spurious* and real counterexamples, where the former only occur in the abstract model but not in the concrete one, while the latter occur in both. Given a spurious counterexample, CEGAR provides a systematic and automated way to refine the abstract model. If the counterexample is real, the algorithm stops and states that the system does not fulfil the property checked. If no counterexample can be found, however, the concrete system is considered safe. Either way, the technique can be used to make a statement whether a concrete model fulfils a given property, by model-checking it against abstract models derived from the concrete one. In contrast, the goal of our workflow is to systematically construct a concrete model that fulfils the property. Since the design starts at the abstract level, also real counterexamples are used in our approach to fix the abstract model. In effect, all counterexamples are used as sources of information for (manually) creating a concrete and quantitative model.

Wang et al. [7] and Tian et al. [8] proposed meanwhile technical improvements on CEGAR with regard to the detection of spurious counterexamples and the refinement of the model. For our case study, we employed the original techniques (manually). For automating, it could be interesting to apply these improvements.

The seminal CEGAR approach has been applied to different verification tasks for already given systems. Nellen et al. [9, 10] used CEGAR to verify an already given programmable logic controller (PLC) against safety properties, proposing two approaches for that. Stursberg et al. [11] built on CEGAR for verification of a cruise control system using counterexample-guided *search*. Hybrid automata representing an already given ACC implementation are used for its verification in a closed-loop setting. In addition to the integrated verification (based on the same theory), our design workflow includes steps for constructively designing such a CPS in the first place, including, e.g., the determination of system parameters.

Clarke et al. [12] extended CEGAR for verification of *hybrid systems*. While we derived a quantitative model from a qualitative one in the course of our case study, it is still a finite system since we restrained it, e.g., to a finite number of distinct speed values. Hence, we did not have to use these extended techniques for hybrid systems. Still, applying them should be possible in the context of our workflow.

Software and CPS design in practice integrates verification as well, of course, both informally and formally. However, we are not aware of any previous approach that would systematically determine changes of models on different levels of abstraction based on verification results like our workflow.

4 THE MODEL DESIGN WORKFLOW USED

Now let us explain our very recently proposed workflow for systematic top-down model design used in our case study, as illustrated in Figure 1. It uses theory on over-approximation and refinement techniques from [3, 4] in a different context and in a different way.

First, an initial abstract model has to be provided (or used, when already given like the abstract qualitative model of a CPS proposed in [1]). This activity requires design skills and domain knowledge as well as a vision. Still, it should be easier to create an abstract model first than a concrete one, without having to take all the details into account upfront. Model-checking the abstract model may already at this stage reveal property violations in the form of counterexamples, and it is more efficient than model-checking a more concrete model. These counterexamples can be used as a source of information for modifying the abstract model with respect to the violated properties. Hence, the counterexamples have to be analyzed and the model modified, until model-checking does not find any counterexample.

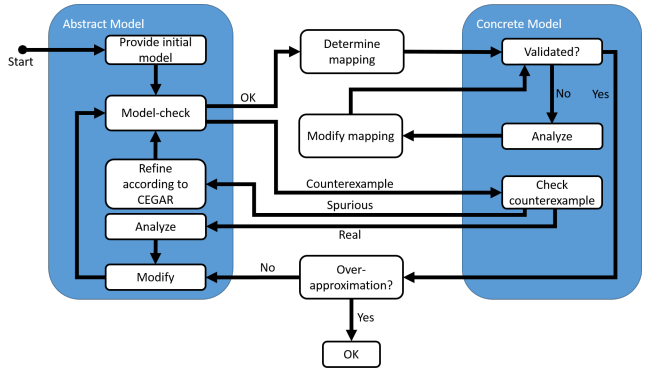

Figure 1: Workflow for systematic top-down model design, redrawn from [2]

For a verified abstract model, a mapping to a concrete model is to be determined. The resulting concrete model should be immediately validated whether it is realistic or not. If the validation is not successful, the reasons must be analyzed and the mapping modified accordingly, until a concrete model is considered realistic.

For answering the question whether the concrete model fulfils the properties that the abstract model has already been verified against successfully, the abstract model needs to be an *over-approximation* of the concrete model. This must be checked, and such a check can be intricate when done manually. Even if this holds with respect to the originally mapped concrete model, the mapping may actually have changed due to changes of the concrete model regarding its validation. Hence, the over-approximation check must be performed bottom-up and separately.

If this check fails, the abstract model needs to be modified accordingly and model-checked again. If a counterexample is found, it is checked whether it is *spurious* (according to [3]). This is the case, if it is not a counterexample for the concrete model, i.e., it is an artefact of the abstraction. Hence, the abstract model is to be *refined*, and this can be done systematically according to [3, 4.4]. If the counterexample is real then the abstract model needs to be analyzed and modified accordingly, and model-checked again, etc.

Once the check for over-approximation is successful, the workflow can stop with the result that the concrete model is verified successfully against the same properties that hold for the abstract model.

5 CASE STUDY

Our case study started with the qualitative ACC model presented in [1], the FSMs are given in Figure 2. In order to make this paper self-contained, the whole model is also presented in this paper. However, we modified its representation, in order to make changes in the course of the case study easier to spot. In the following, all tables regarding transition conditions belong to the qualitative model. Hence, all changes in these tables or the FSMs are changes in the abstract model.

The transition conditions are partly given in Table 2. For the latter, we created a compressed representation in Table 1, where each row contains a different speed value of vehicle A, and each column a different speed value of vehicle B. The table entries specify whether the distance is kept (held), increased or decreased, depending on the current speed values of vehicle A and vehicle B. For example, if the speed value of both vehicles is currently Low, then the distance is kept. This corresponds to ($LowSpeedA \wedge LowSpeedB$) in Table 2, which is given there in the $Dist\_H$ row. Generally, the entry in each cell of Table 1 corresponds to the row of Table 2 where the term ($xSpeedA \wedge ySpeedB$) appears.

Table 1: Transition conditions of Table 2 compressed

Speed A\B	Low	Medium	High
Low	H	I	I
Medium	D	H	I
High	D	D	H

H . . . Hold distance
I . . . Increase distance
D . . . Decrease distance

The transition conditions for both DC and the coordinator are equivalent to the conditions given in [1], still we decided to represent them in the form of a table, see Tables 3 and 4.

This qualitative model has been successfully verified through model-checking in [1] against the rear-end collision accident property in Eq. 1, which means that it is always (globally) true that the physical distance is different from a collision:

$$AG(state\_phy\_dist \neq COLLISION) \tag{1}$$

Since this property holds in this abstract model it should be possible to reason that it also holds in a more concrete model. To achieve this, according to [3] the abstract model needs to be an *over-approximation* of the concrete model.

5.1 First Mapping to a Quantitative Model

Our first mapping was from the qualitative ACC model introduced in [1], which we briefly explained above, to a quantitative model. According to our mapping approach outlined above, we first selected the total range of speed values and partitioned it according to the number of speed classes in the qualitative model. More precisely, we chose three concrete speed values ($15m/s = 54km/h$, $22.5m/s = 81km/h$, and $30m/s = 108km/h$), corresponding to the three speed values defined in [1] (LowSpeed, MediumSpeed, and HighSpeed). Since we selected the same concrete speed values for both vehicles, there is no change in their distance in the quantitative model whenever they are in the same speed class.

Then we defined the cycle time for the model-checking runs as $1s$. In one time-step, a change from one speed class to an adjacent class is allowed. Consequently, the possible speed changes are $-7.5m/s$, $0m/s$, and $+7.5m/s$, i.e., possible accelerations of $-7.5m/s^2$, $0m/s^2$, and $+7.5m/s^2$.

Based on all that, we mapped the Distance Classification of the qualitative model [1] to real distances in our first quantitative model. In particular, the maximum speed difference between the two vehicles of $15m/s$ results in a maximum change of their distance in a single time-step of $15m$. Taking also the constraints on such a mapping into consideration as given above, we defined the following mapping of distances:

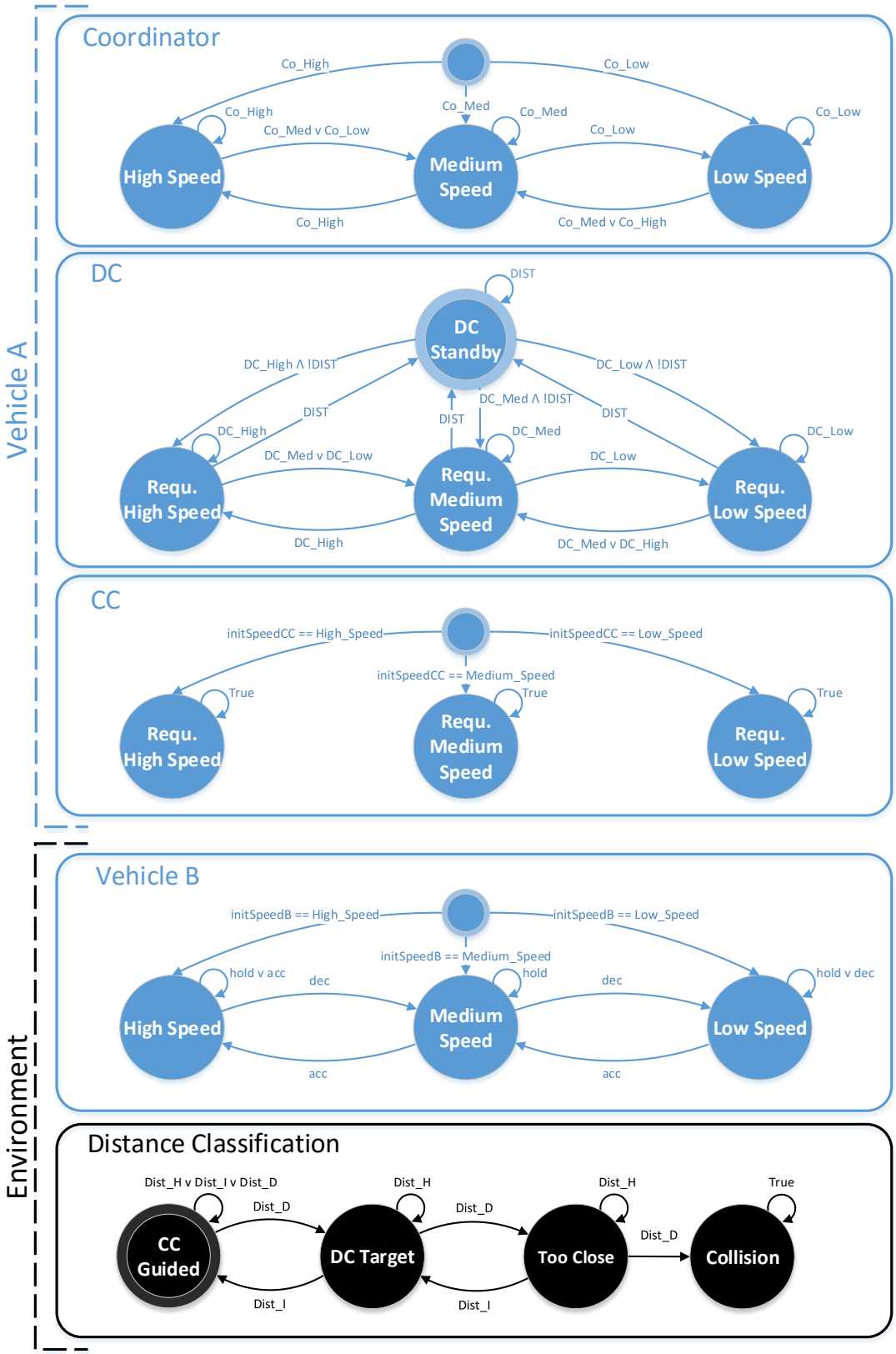

Figure 2: FSMs according to [1], with changed representation

Table 2: Conditions for Transitions of FSM of the Distance Classification according to [1]

Name	Condition
Dist_H	$(LowSpeedA \wedge LowSpeedB) \vee (MediumSpeedA \wedge MediumSpeedB) \vee (HighSpeedA \wedge HighSpeedB)$
Dist_D	$(MediumSpeedA \wedge LowSpeedB) \vee (HighSpeedA \wedge LowSpeedB) \vee (HighSpeedA \wedge MediumSpeedB)$
Dist_I	$(LowSpeedA \wedge MediumSpeedB) \vee (LowSpeedA \wedge HighSpeedB) \vee (MediumSpeedA \wedge HighSpeedB)$

Table 3: Transition conditions related to the DC FSM of [1]

Distance\Speed B	Low	Medium	High
DC Target	DC_Low	DC_Med	DC_High
Too Close	DC_Low	DC_Low	DC_Med

Table 4: Transition conditions related to the coordinator FSM of [1]

Speed DC\CC	Low	Medium	High
Low	Co_Low	Co_Low	Co_Low
Medium	Co_Low	Co_Med	Co_Med
High	Co_Low	Co_Med	Co_High

- CC Guided = [Inf, 35m)
- DC Target = [35m, 20m)
- Too Close = [20m, 0m)
- Collision = [0m, -inf]

In addition, it was necessary to provide the quantitative models with the Newtonian equations for the physics involved, e.g., $Distance_{n+1} = (Speed_B - Speed_A) * \Delta t + Distance_n$. They may not be violated and provide an essential basis for the validation in the course of the investigated workflow.

5.2 Detailed Analysis and Model Modification

Analyses of the physical behavior of the quantitative model resulting from our first mapping showed that its behavior is not fully covered by the original qualitative model. According to physics, the distance interval is not always changed when, e.g., vehicle A drives LowSpeed and vehicle B MediumSpeed. The distance may also be kept within an interval, depending on where exactly it was before. Since physics cannot be changed, of course, we had to modify the qualitative model. Actually, only certain transition conditions had to be changed. Based on our analyses, the conditions must be as given in Table 5, where the letters in orange indicate the additional possibilities as compared to Table 1.

Table 5: Transition conditions according to physics with distance ranges

Speed A\B	Low	Medium	High
Low	H	H/I	I
Medium	H/D	H	H/I
High	D	H/D	H

H . . . Hold distance
I . . . Increase distance
D . . . Decrease distance

This adapted table indicates indeterministic state transitions in the Distance Classification FSM at the bottom of Figure 2. This leads to a collision when model-checking the adapted qualitative model with the new transition conditions, with the following counterexample output by the tool NuSMV [13]:

(1) Vehicle A drives HighSpeed and Vehicle B MediumSpeed. → Distance changes from CCGuided to DCTarget.
(2) Vehicle A drives MediumSpeed (because of the DC Request) and Vehicle B accelerates to HighSpeed. → Because DC Target is now a range, the distance DCTarget may be kept, and in this example, it is kept (while in the previous model the distance would change to CCGuided).
(3) Vehicle A drives HighSpeed (DC recognizes HighSpeed of Vehicle B from the previous step) and Vehicle B decelerates to MediumSpeed. → Distance changes to Too Close.
(4) Vehicle A decelerates to MediumSpeed and Vehicle B decelerates to LowSpeed. → Distance changes to Collision.

According to the workflow, the counterexample was checked whether it is spurious or real. Since the check revealed that it is spurious, CEGAR refinement was applied, which led to the distance classification depicted in Figure 3.

Model-checking by using the adapted FSMs did not reveal any collision.

5.3 Second Mapping to a Quantitative Model

After having successfully adapted the original qualitative model from [1], we started another iteration by defining our second mapping to a quantitative model. More precisely, we mapped the model with the new Too Close 1/2 state defined in the FSM in Figure 3 with its transition conditions defined in Table 5 as well as the adapted DC FSM transition conditions in Table 6 to a quantitative model with the values given in Table 7. This was also done using the CEGAR refinement that the workflow adopts, since it is also defined for the concrete model. Note, that the distance of DC Target has not been increased in the course of this change of the quantitative model.

Table 6: Transition conditions of the DC FSM corresponding to the Distance Classification with two Too Close states

Distance\Speed B	Low	Medium	High
DC Target	DC_Low	DC_Med	DC_High
Too Close 1/1	DC_Low	DC_Low	DC_Med
Too Close 1/2	DC_Low	DC_Low	DC_Low

Since the values look realistic according to physics, in particular the distances and the accelerations ($-7.5m/s^2$, $0m/s^2$, and $+7.5m/s^2$) as compared to the concrete speed values, we were tempted at first to consider this model *validated*. And since the abstract qualitative model was an *over-approximation* of the concrete

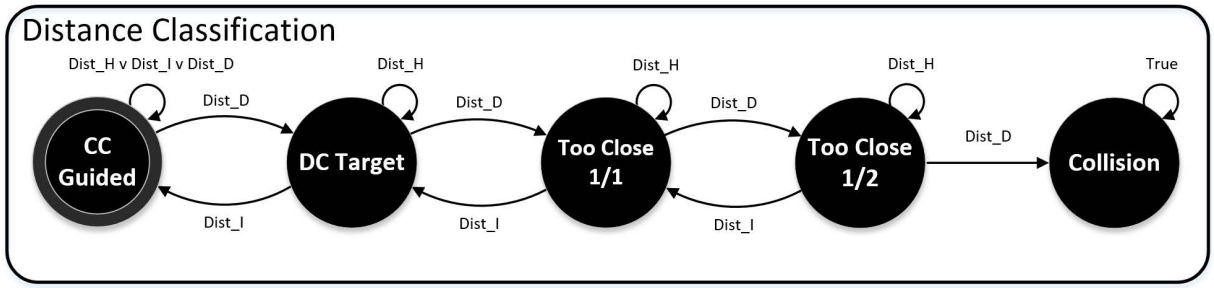

Figure 3: Distance Classification with Too Close 1 split according to CEGAR refinement

Table 7: Summary of mapping found to be valid for specific speed values

	Value/Range
HighSpeed	30m/s
MediumSpeed	22.5m/s
LowSpeed	15m/s
Time-step	1s
CC Guided	[Inf, 35m)
DC Target	[35m, 20m)
Too Close 1/1	[20m, 7.5m)
Too Close 1/2	[7.5m, 0m)
Collision	[0m, -inf]

quantitative one, the workflow would already have been finished successfully.

Still, there is a serious validation problem with this model, since it is obviously unrealistic in the real world to allow only select few concrete speed values and no speed values in between. This results from our mapping where each speed class is mapped to exactly one speed value only.

5.4 Introducing Speed Ranges

To overcome this shortcoming, speed classes in the qualitative model were mapped to a range of speed values each, such as follows:

- High-Speed = [30m/s, 25m/s)
- Medium-Speed = [25m/s, 20m/s)
- Low-Speed = [20m/s, 15m/s)

In order to avoid having to deal with the additional techniques for hybrid systems, we simply define these ranges through the integer values in between.

Unfortunately, when each vehicle randomly picks one speed value from the same speed range, e.g., it is no longer guaranteed that the distance stays the same, of course. Hence, the abstract qualitative model is *not* an over-approximation, since it does not have the corresponding transitions. As a consequence, correctness of the qualitative model does not imply correctness of the quantitative one.

Hence, the qualitative model has to be modified again. The adapted transition conditions in Table 8 reflect these changes. Model-checking with these adaptations generated a counterexample, i.e., collision is possible, even with the previously adapted Distance Classification (with Too Close 1/1 and Too Close 1/2) as shown in Figure 3.

Table 8: Transition conditions according to speed ranges

Speed A\B	Low	Medium	High
Low	H/I/D	H/I	H/I
Medium	H/D	H/I/D	H/I
High	H/D	H/D	H/I/D

H . . . Hold distance
I . . . Increase distance
D . . . Decrease distance

5.5 Analysis and Another Model Modification

Since checking the counterexample revealed that it is real, we performed a detailed analysis of the qualitative model from [1] as adapted by Table 8.

This analysis revealed that, whenever vehicle A is in the Too Close 1/2 state, its speed is LowSpeed. The critical case occurs when vehicle B also drives LowSpeed. Since we abstract the exact speed values of both vehicles, we do not know whether vehicle A is approaching or not. The transition conditions of this qualitative model take that into account by allowing the transition from Too Close 1/2 to Collision.

In order to solve this problem, we changed the conditions according to Table 9. This adaptation of the qualitative model reflects a mapping of LowSpeed to exactly one concrete speed value, so that, when both vehicles have this speed, the distance can only stay the same. Hence, there are fewer transitions again, and model-checking with this adaptation did not lead to a counterexample.

Table 9: Transition conditions when LowSpeed is mapped to exactly one concrete value

Speed A\B	Low	Medium	High
Low	H	H/I	H/I
Medium	H/D	H/I/D	H/I
High	H/D	H/D	H/I/D

H . . . Hold distance
I . . . Increase distance
D . . . Decrease distance

Table 10: Transition conditions related to the coordinator FSM with AbsLowSpeed

Speed DC\CC	AbsLow	Low	Medium	High
AbsLow	Co_AbsLow	Co_AbsLow	Co_AbsLow	Co_AbsLow
Low	Co_AbsLow	Co_Low	Co_Low	Co_Low
Medium	Co_AbsLow	Co_Low	Co_Med	Co_Med
High	Co_AbsLow	Co_Low	Co_Med	Co_High

Table 11: Transition conditions related to the DC FSM when the Distance Classification has two Too Close states and four speed classes are used

Distance\Speed B	AbsLow	Low	Medium	High
DC Target	DC_AbsLow	DC_Low	DC_Med	DC_High
Too Close 1/1	DC_AbsLow	DC_AbsLow	DC_Low	DC_Med
Too Close 1/2	DC_AbsLow	DC_AbsLow	DC_AbsLow	DC_Low

5.6 Modification of the Mapping

As indicated above, keeping the distance when both vehicles drive with LowSpeed is only possible in a mapped quantitative model with both vehicles having the same concrete speed value for LowSpeed. Hence, we modified the mapping as follows, to take this latest change of the qualitative model into account:

- High-Speed = [30m/s, 22,5m/s)
- Medium-Speed = [22,5m/s, 15m/s)
- Low-Speed = 15m/s

According to the workflow, the question had to be answered, whether this model can be *validated*. Our changes of the speed ranges (while keeping the cycle time at 1s) have actually increased the possible accelerations, which are in this model in the range $(-15m/s^2, +15m/s^2)$. At least for current automotive vehicles, we judged them as unrealistic.

To reduce these acceleration values, the mapping was changed in such a way that the overall speed range was divided into four instead of three sub-ranges. In a nutshell, this led to a validated quantitative model, which the qualitative one is clearly not an over-approximation of, however, since it does not have the fourth speed class yet. Hence, we had to add it into the qualitative model.

After all this, the correspondence can be summarized as follows, where the new speed class corresponds to an absolute low speed (AbsLowSpeed), identical for both vehicles (while LowSpeed is mapped to a range of speed values again):

- HighSpeed = [30m/s, 25m/s)
- MediumSpeed = [25m/s, 20m/s)
- LowSpeed = [20m/s, 15m/s)
- AbsLowSpeed = 15m/s

Due to the inclusion of the new speed class AbsLowSpeed, all FSMs had to be updated. Of course, the transition conditions had to be updated, too. The new conditions for the Distance Classification are given in Table 12 and the transition conditions corresponding to the FSMs of vehicle A in Tables 10 and 11.

Model-checking this qualitative model shows that a collision can occur. We did not expect that, since according to our intuition introducing an additional speed class into the abstract model reduces the approximation error. However, it entails the need for a more detailed Distance Classification as well. Keep also in mind, that the distance can change in this model, although the speed classes are the same (except for the case of both vehicles driving AbsLowSpeed).

Table 12: Transition conditions with AbsLowSpeed

Speed A\B	AbsLow	Low	Medium	High
AbsLow	H	H/I	H/I	H/I
Low	H/D	H/I/D	H/I	H/I
Medium	H/D	H/D	H/I/D	H/I
High	H/D	H/D	H/D	H/I/D

H . . . Hold distance
I . . . Increase distance
D . . . Decrease distance

(1) Vehicle B drives MediumSpeed and Vehicle A drives HighSpeed → Target changes from CCGuided to DCTarget.
(2) Vehicle B decelerates to MediumSpeed and Vehicle A drives HighSpeed (DC sees HighSpeed of Vehicle B from the previous step) → Distance changes to Too Close 1/1.
(3) Vehicle B decelerates to LowSpeed and Vehicle A decelerates to MediumSpeed → Distance changes to collision, since the difference in the vehicle speeds allows 'jumping over' the Too Close 1/2 state.

However, this is a spurious counterexample due to the abstraction.

5.7 Modification of the Qualitative Model and the Mapping

According to CEGAR refinement, the state Too Close 1/1 has to be split. This results in Too Close 1/1/1 [20m, 10m) and Too Close 1/1/2 [10m, 7.5m). Due to lack of space and analogy with some of the above, we omit the concrete state machines and their corresponding tables in the following.

This more detailed Distance Classification fixed the occurrence of this spurious counterexample. However, model-checking revealed another counterexample, a real one. Therefore, we increased the distance between Collision and Too Close 1/2 and labeled it Too

Table 13: Transition conditions related to the DC FSM when the Distance Classification has two Too Close states and four speed classes are used

Distance\Speed B	AbsLow	Low	Medium	High
DC Target	DC_AbsLow	DC_Low	DC_Med	DC_High
Too Close 1/1/x	DC_AbsLow	DC_AbsLow	DC_Low	DC_Med
Too Close 1/2	DC_AbsLow	DC_AbsLow	DC_AbsLow	DC_Low
Too Close 2	DC_AbsLow	DC_AbsLow	DC_AbsLow	DC_AbsLow

Close 2 in the qualitative model. This insertion of a new state required changes in the transition conditions of DC as well, see Table 13.

Model-checking this model did not reveal any counterexample. Hence, we had to determine its mapping to a quantitative model, where both the splitting of Too Close 1/1 and the insertion of Too Close 2 had to be taken into account, resulting in Table 14.

Table 14: Summary of mapping found to be valid for speed ranges

	Range
CC Guided	[Inf, 50m)
DC Target	[50m, 35m)
Too Close 1/1/1	[35m, 25m)
Too Close 1/1/2	[25m, 22.5m)
Too Close 1/2	[22.5m, 15m)
Too Close 2	[15m, 0m)
Collision	[0m, -inf]

This model also passed our validation, but the check for over-approximation revealed that the abstract model had to be modified again to cover the whole behavior of this concrete model. Model-checking this modified abstract model did not reveal any counterexample.

Determining the new mapping and validation of the resulting concrete model were straightforward, since the most recent modification of the abstract model was only related to the most recent modification of the mapping. Finally, the abstract qualitative model was an over-approximation of the concrete quantitative one. Therefore, the workflow finished successfully.

In terms of verification, this means that the concrete quantitative model has been shown indirectly to not run into any collision, since the abstract qualitative model has been formally model-checked against this condition, and it is an over-approximation of the concrete quantitative model.

5.8 Lessons Learned

Through inductive generalization from the case at hand, we generalize the following lessons learned:

- Applying the theory of over-approximation helps to keep the models on different levels of abstraction consistent with each other, hence it can also be utilized in a top-down approach.
- The CEGAR refinement is also applicable and useful in a top-down approach.
- While such formal techniques for verification can be automated, validation is (still) according to human judgment.
- As a side-effect of enacting this workflow, the human(s) doing it gain(s) insight and understanding of the given domain.
- In summary, enacting the workflow in [2] was possible systematically and led to a reasonable quantitative model at the concrete level that was also formally verified indirectly.

6 CROSS-CHECK THROUGH MODEL-CHECKING AT THE CONCRETE LEVEL

In addition to doing the case study per se, we wanted to make a cross-check by directly model-checking the resulting quantitative model at the concrete level. If successful, this provides some evidence on the correctness of the workflow itself and of our enactment in the course of this case study.

The quantitative model was given through the FSMs of the qualitative model together with the mappings. For directly model-checking it with NuSMV, the quantitative physical formulas were used explicitly (instead of implicitly as in the qualitative model).

Fortunately, the mapping limits the possible ranges of values, e.g., the speed range, to decrease the state-space for model-checking. The used limits were determined by the mapping in such a way that for each variable the lowest and highest values were taken. In general, these values correspond to different states in the qualitative model. For example, the minimum speed of vehicle B is 15m/s, which corresponds to ABSLowSpeed, and the maximal speed is 30m/s, which corresponds to HighSpeed. Hence, the code snipped for NuSMV below reflects this through the min/max combination for the variable speed_VB, where speed_VB_tmp is used additionally for the technical reason that its value can be used in the same cycle of the model-checking run. In addition, the mapping defines, together with the model-checking cycle time, that the acceleration is in the range of $-10m/s^2$ and $10m/s^2$. In the code snipped, this is represented through the variable acceleration_delta_VB, i.e., the delta possibly resulting within 1s cycle time for the speed of vehicle B. In every model-checking cycle, every integer value in between is possible to occur.

```
acceleration_delta_VB := -10 .. 10;

speed_VB_tmp := min(max((speed_VB -
    acceleration_delta_VB), 15), 30);

next(speed_VB) := speed_VB_tmp;

...
```

```
distance_tmp := min(max((distance + ((speed_VB_tmp -
    speed_VA_tmp_limited))), -100 ), 500);

next(distance) := distance_tmp;
```

We limited the distance values possible during model-checking as well. Based on our familiarity with the ACC domain gained in the course of this case study, we only considered distances in the range of [-100m, 500m]. Whenever the distance becomes zero or smaller, a collision occurs. Compared to the target distance in the range [50m, 35m), the maximum distance considered here is very high.

In addition, there is an interesting aspect related to the *controller* in the qualitative model, which is specified in Table 13. Through the mappings, of course, no specific controller is defined for the quantitative model. In fact, many controller equations for the quantitative model may comply to this specification.

In the quantitative model checked directly, we implemented DC as a discrete controller that simply requests the highest possible value of the range given in the corresponding table entry. For example, when Distance is DC Target and Vehicle B is driving MediumSpeed, then the table requests MediumSpeed. In the quantitative model, the request is 25m/s, since this is the highest speed classified as MediumSpeed.

Model-checking this quantitative model directly at the concrete level did not reveal any counterexample. Running it in NuSMV on a usual PC took about 45min. In contrast, each of the model-checking runs of the various qualitative models took less than 5sec.

7 DISCUSSION

Unfortunately, this case study does not show scalability to large problem sizes. Based on our experience from this case study, we think that automation through tool support will be required for efficiently applying our workflow to large-scale systems. Checking formal properties and applying formal refinement manually is simply too demanding for engineers in large real-world projects.

Still, our case study provides some empirical evidence that our workflow is operational, in principle, and that it can be made useful for practice through tool support. After all, CEGAR was fully automated, and based on its tool support, the formal techniques in our workflow can be automated as well.

While we cannot envision yet a fully automated version of this approach to model design, there may be additional automation through applying model-driven technology. It can support automated mappings from the abstract to the concrete model.

The verification approach studied here and in the related work strives for a Boolean result of whether there is a a counterexample or not. In safety verification of real-world applications, however, typically a probabilistic result is desired. In this case, formal verification through *probabilistic model checking* [14] may be used. This approach allows a wide variety of quantitative properties to be specified, such as "the probability of a system failure occurring", etc. Probabilistic safety properties can be used to capture certain properties of probabilistic models, including "the probability of no failures occurring is at least 0.99". It may be interesting to adapt our top-down design approach and its underlying theory for such a setting.

8 CONCLUSION

In this paper, we show the feasibility of our workflow for systematic top-down design of cyber-physical models with integrated validation and formal verification proposed in [2]. The additionally performed cross-check provides some evidence on the correctness of our workflow itself and of our enactment in the course of this case study. For larger problems, it will not be feasible, however, and this emphasizes the importance of our workflow for formal verification. Future work will have to automate its formal steps in order to improve its applicability.

ACKNOWLEDGMENT

The FeatureOpt project (No. 849928), was funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between June 2015 and May 2018. More information can be found at https://iktderzukunft.at/en/.

REFERENCES

[1] M. Rathmair, C. Luckeneder, and H. Kaindl, "Minimalist qualitative models for model checking cyber-physical feature coordination," in *Proceedings of the 23rd Asia-Pacific Software Engineering Conference (APSEC)*, (USA), IEEE, Dec 2016.

[2] C. Luckeneder and H. Kaindl, "Systematic top-down design of cyber-physical models with integrated validation and formal verification," in *ICSE '18 Companion: 40th International Conference on Software Engineering Companion*, ACM, 2018.

[3] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Counterexample-guided abstraction refinement for symbolic model checking," *Journal of the ACM (JACM)*, vol. 50, no. 5, pp. 752–794, 2003.

[4] E. M. Clarke, O. Grumberg, and D. E. Long, "Model checking and abstraction," *ACM Trans. Program. Lang. Syst.*, vol. 16, pp. 1512–1542, Sept. 1994.

[5] M. G. Bobaru, C. S. Pasareanu, and D. Giannakopoulou, "Automated assume-guarantee reasoning by abstraction refinement," in *CAV*, vol. 5123, pp. 135–148, Springer, 2008.

[6] W. Lee, A. Pardo, J.-Y. Jang, G. Hachtel, and F. Somenzi, "Tearing based automatic abstraction for CTL model checking," in *Computer-Aided Design, 1996. ICCAD-96. Digest of Technical Papers., 1996 IEEE/ACM International Conference on*, pp. 76–81, IEEE, 1996.

[7] C. Wang, H. Kim, and A. Gupta, "Hybrid CEGAR: combining variable hiding and predicate abstraction," in *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, pp. 310–317, IEEE, 2007.

[8] C. Tian, Z. Duan, and Z. Duan, "Making CEGAR more efficient in software model checking," *IEEE Transactions on Software Engineering*, vol. 40, no. 12, pp. 1206–1223, 2014.

[9] J. Nellen and E. Abraham, "A CEGAR approach for the reachability analysis of PLC-controlled chemical plants," in *Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on*, pp. 500–507, IEEE, 2014.

[10] J. Nellen, K. Driessen, M. Neuhäußer, E. Ábrahám, and B. Wolters, "Two CEGAR-based approaches for the safety verification of PLC-controlled plants," *Information Systems Frontiers*, vol. 18, no. 5, pp. 927–952, 2016.

[11] O. Stursberg, A. Fehnker, Z. Han, and B. H. Krogh, "Verification of a cruise control system using counterexample-guided search," *Control Engineering Practice*, vol. 12, no. 10, pp. 1269–1278, 2004.

[12] E. Clarke, A. Fehnker, Z. Han, B. Krogh, O. Stursberg, and M. Theobald, "Verification of hybrid systems based on counterexample-guided abstraction refinement," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 192–207, Springer, 2003.

[13] NuSMV, "NuSMV: A new symbolic model checker." http://nusmv.fbk.eu/. [Online; accessed Oct. 11, 2018].

[14] M. Kwiatkowska, G. Norman, and D. Parker, "Probabilistic model checking: Advances and applications," in *Formal System Verification*, Springer, 2017.