Towards Real-time Adaptive Approximation

Raheleh Biglari, Joost Mertens, Joachim Denil

Towards Real-time Adaptive Approximation

Raheleh Biglari
Cosys-lab, University Of Antwerp
Flanders Make@Uantwerpen
Raheleh.Biglari@uantwerpen.be

Joost Mertens
Cosys-lab, University Of Antwerp
Flanders Make@Uantwerpen
Joost.Mertens@uantwerpen.be

Joachim Denil
Cosys-lab, University Of Antwerp
Flanders Make@Uantwerpen
Joachim.Denil@uantwerpen.be

***Abstract*—Cyber-physical systems (CPS) are real-time systems that operate in dynamic and non-deterministic environments. Models are often used for control and prediction, however do not reason on the trade-off between real-time constraints and uncertainty. This paper presents a conceptual model to reason on adaptive approximation in such systems. Furthermore, we envision a framework to allow the adaptivity of models, balancing between uncertainty and the real-time behavior of the system.**

***Index Terms*—cyber physical systems, real-time, uncertainty, adaptation, abstraction**

I. Introduction

Cyber-physical systems (CPS) are engineered systems that have tight integration between the cyber part (computation and networking) and its physical components [1]. Examples include but are not limited to industry 4.0, automotive, and aerospace. Engineered systems have a goal to achieve in the context of the system, e.g. an autonomous vehicle needs to pilot the environment while not harming anyone. To achieve this goal, multiple decision models are needed and combined. For example, an autonomous vehicle has low-level control to accelerate and brake, tactical decision-making models for path planning, and strategic models to decide which roads to avoid, e.g., as an accident occurred. All of these decision models implement some form of (feedback) control.

Cyber-physical systems of systems are CPS that exhibit the features of a system of systems (SoS). They are large and spatially distributed, have distributed control, and autonomic behavior where parts of the SoS can join or leave the system [2]. As such it is a system composed from different CPS where each part of the system contributes to the overall goal of the CPSoS. The engineering of such a CPSoS has to address the complex situations and, environments of the system, which is characterized by ambiguity, high uncertainty and emergence [3]. CPSoS have to allow for collaborative decision making and, as such, need to be aware of the state of the other constituents of the system [2].

CPS and CPSoS operate in a very dynamic environment where lots of uncertainty is present [4]. Uncertainty points to the lack of information that is available about the system or its environment. An autonomous vehicle might have uncertainties about its own position in the system and about the direction and velocities of other vehicles and road users. Cyber-physical systems are also real-time systems which means that the time at which a decision is made is as important as the decision itself. This means that during design time, the system is analyzed to ensure that all the deadlines of the control components are met in the worst-case.

Raheleh Biglari is funded by the BOF fund at the University of Antwerp. Joost Mertens is funded by the Research Foundation - Flanders (FWO) through strategic basic research grant 1SD3421N.

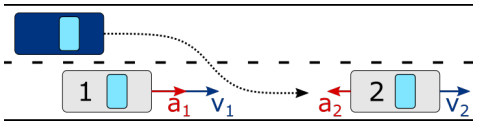

Fig. 1. Lane changing scenario.

One way to deal with the contradiction of better performance and reduced cost in CPS is to allow adaptivity at run-time and to change the underlying decision and estimation models such that they are sufficient for computing the control actions but at the same time computationally less intensive. Techniques that abstract or approximate models are commonly available in the literature, e.g., surrogate modeling [5].

In this short paper, we look at different dimensions of the problem to allow for such a run-time adaptation of the underlying control and decision models with more abstract and approximate models. The rest of the paper is organized as follows: section II introduces our running example, section III describes the challenges in our work. We also present use case evaluation results. In section IV we describe our strategy to dealing with the challenges. We provide related works in section V and section VI presents our conclusion and discusses future directions.

II. Running Example

We use a lane change control algorithm to show the challenges of introducing adaptive abstraction and approximation in a real-time context. While we do not aim precisely at this class of control algorithms, lane changing allows for visual and intuitive reasoning over the problem space.

Lane changing algorithms control the lateral direction of the vehicle. The scenario throughout this paper is shown in Figure 1. The ego car (in dark blue) tries to change the lane in between two other cars. The velocity and acceleration vectors of the two cars is shown as vectors v_1,v_2, a_1 and a_2. The front car is lightly decelerating while the back car is lightly accelerating. Multiple algorithms have been proposed in the literature to solve such lane changing problems, e.g. [6]. In the context of this paper, we use a Simulink provided model

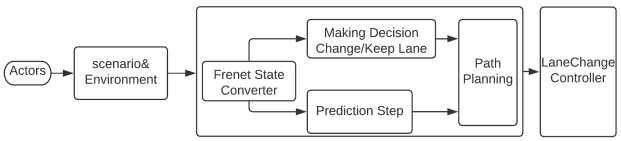

Fig. 2. Causal Block Diagram of the Lane Change Algorithm

to simulate the lane changing behavior. Figure 2 shows the high-level architecture of the lane changing algorithm. The algorithm uses Frenet Coordinate System, which represents the position of the car on the road more intuitively than traditional (x, y) coordinates.This algorithm also uses a prediction step to predict the trajectories of the different actors. Afterward, path planning takes care of finding a good path in the world. Finally, a model-predictive controller steers the vehicle over the planned path.

In the rest of the paper, we focus on the prediction step of the other vehicles. The prediction step simulates a trajectory relative to the ego car for each vehicle in the environment. The component receives information on the lateral and longitudinal position, velocity, and acceleration of the vehicles. In the default case, the prediction step uses a constant velocity model to predict the vehicle's position at multiple time-steps in a three-second window:$s(t) = v * t + s_0$ (with s the relative distance to the ego-car, v the relative speed to the ego vehicle, and s_0 the initial distance). However, we can imagine much more detailed models that also take the vehicle's acceleration into account or even more detailed models that simulate the complex decision-making in each of the vehicles.

III. Challenges

Using a model of a car performing a lane change such as the scenario in Section II, we elaborate on identified challenges. The model is simulated with Simulink.

A. Model Prediction Uncertainty

Each of the different models has a different amount of predictive power. As such, more detailed models typically have lesser prediction uncertainty. Figure 3 depicts the prediction of the scenario with two different models over three time-steps. The top control bar is the uncertain position of the cars using a constant velocity model. The lower control bar shows the uncertain position using a constant acceleration model.

To reason over using different alternative models, we need to map the prediction uncertainty of all the different models. Two types of uncertainty are typically present in modeling and simulation: aleatoric and epistemic uncertainty. Aleatory uncertainty is known as stochastic uncertainty and is due to probabilistic variability. Epistemic uncertainty is the uncertainty that occurs because of the lack of knowledge [7].

With the simulation model, we demonstrate how the constant velocity and constant acceleration models have different predictive power. Figure 4 shows the velocity profile of car 2 and the L2-norm (Euclidean distance) between 4 predictions and the true location of that car. The number behind each

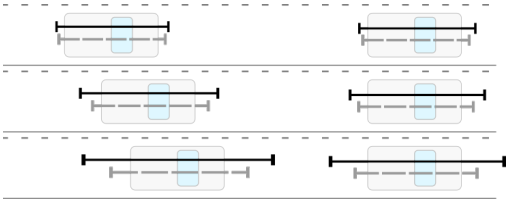

Fig. 3. Prediction of the car position on three time steps within the time window, relative to the leading car. On each car, the prediction uncertainty of two different models is shown, one solid, one dashed.

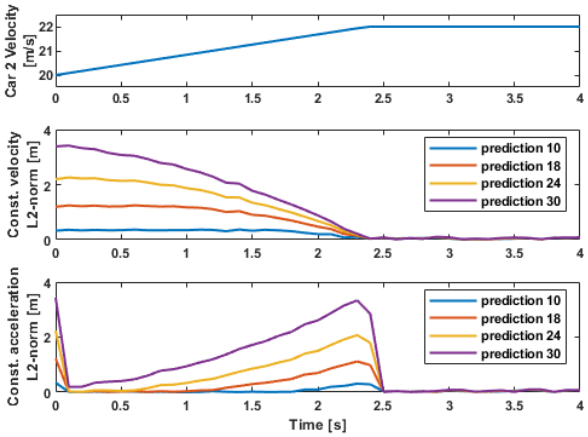

Fig. 4. Car 2's velocity profile, compared with the L2 errors made by the constant velocity and acceleration prediction models.

prediction tells us how many steps (of $0.1s$) in the future this prediction is. The simulation includes uncertainty on the sensor inputs of the ego vehicle, which results in slightly jagged error traces. The velocity profile shows that the car accelerates to 22 m/s. Afterward, it holds a constant velocity. We observe that under acceleration, the constant velocity model has a continuous error that only diminishes as the vehicle approaches constant velocity. The constant acceleration model performs better when conditions are constant, that is, positive, negative, or 0 acceleration, and shows the largest error when the predictions cross transitions in acceleration. Such transitions can be seen at $t = 0s$, where the initial acceleration measurement is 0, yet the car is already accelerating at $0.8m/s^2$, and at $t = 2.4s$, when the car stops accelerating. Under constant velocity, both models perform equally. From the results, we can say that the constant acceleration model has better predictive power, given the generally smaller errors.

B. Dynamic Environment

The environment in which systems operate is dynamic. Cars enter and leave the operating environment of the system. Some highways have more lanes than others. Even more, not all of the actors within the environment are of the same type. In a traffic environment, we have pedestrians, bicycles, cars, trucks, and buses that all behave differently. The prediction component of the lane change algorithm has to set up this dynamic

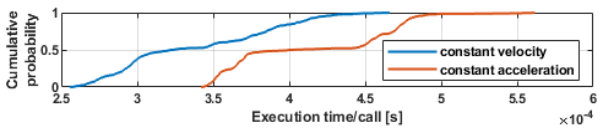

Fig. 5. Experimental CDF of the profiled execution times of both models.

environment each time and select the most appropriate models to include.

C. Real-time Constraints

Often our cyber-physical systems are also real-time systems. This means that the time at which the computation result is available is as important as that result itself. During the design of such systems, care is taken that the computations are finished before the expiry of the deadline of the computation. Different analytical techniques are available to check that this is correct. However, in the dynamic environment described above, this is difficult. How many cars, pedestrians, and bicycles are in the system's environment is unknown at design time. However, we still need a prediction of the behavior of each of these actors in the example within a certain time-span.

With the simulation model, we can demonstrate the impact computations can have on the number of predictions. Figure 5 shows the experimental cumulative distribution function of the profiled execution times of the state predictor for the constant velocity and acceleration models. 400 simulations were run for each model. Although profiling a Simulink model does not yield real-time results, it does allow us to relatively compare the computational burden of its blocks, We observe that, on average, the constant acceleration model requires 22.74% more computation time than the constant velocity model. This implies that for predicting 4 other cars with constant acceleration, 5 cars with constant velocity could be computed.

IV. Approach

This section details our approach to handle these challenges. In the first part of this section, we look at a conceptual manner on reasoning over substitutability of models. We look at the effect of approximating a model with a surrogate model and how it impacts a decision making algorithm. The conceptual framework is used as the foundation for a framework that allows for the run-time adaptation of a system where models can be swapped by more approximate models based on the context of the system and the available library of models.

A. Language Engineering Framework

To reason on the challenges, we propose an adapted version of the conceptual framework proposed by Barroca, Kuhne and Vangheluwe [8]. The conceptual framework in Figure 6 integrates ontological and language engineering. We extended the framework with two different models with different approximations, and the layer of the decision-making algorithms. The linguistic level starts with a simulation model. The simulation model has semantics (denoted by [[.]]), which results in a

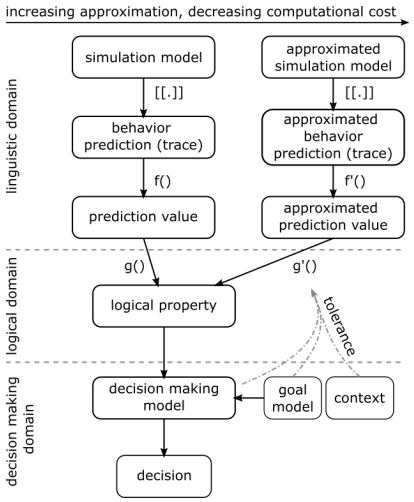

Fig. 6. Conceptual framework.

simulation behavior trace. However, the trace might not be the quantity of interest of the decision-making algorithm or for reasoning over the logical behavior of the system. A function $f()$ (e.g. integrating the signal) transforms the trace into some quantity of interest, here called the prediction value. A logical property is on the ontological level where it gets a real-world meaning, e.g. will the two cars collide? The logical property is a Boolean value; either the cars collide, or they do not collide. Another function $g()$ is used to transform between the quantity of interest and the logical property. Decision algorithms typically work directly with the prediction value but implicitly encode the transform within its algorithm. We, therefore, show the direct link between the logical property and the decision-making model.

The framework is analog for a model that is approximated. It reasons on the same logical property, however, using a more approximate model introduces uncertainty. This means that the function $g'()$ should give the same answer as the original model, except with more uncertainty. If the uncertainty is within bounds, the original model can be substituted with the more approximate one. This bound is defined by some metric, which is the tolerance to this change in the logical property (and hence later in the decision that was made by the decision-making model). The tolerance depends on a number of factors. In our lane changing example, the tolerance is based on (a) the goal model: do I actually want to change lanes, or do I want to stay in the same lane? When staying in the same lane, the cars on other lanes are maybe less important. (b) the context or environment of the system: is the car close-by (less tolerance) or far away (more tolerance) (c) the decision-making model itself: is the algorithm itself more or less tolerant to uncertainties?

B. Adaptive abstraction and approximation for real-time systems

Based on the insights provided by the conceptual framework, we propose an architecture for adaptive abstraction and

approximation for real-time systems. The architecture is shown in Figure 7. The high-level architecture is based on the MAPE-K architecture [9]. MAPE-K is a high-level control loop for self-adaptive systems. The *managed system* is, in our case, the embedded system running the control application for lane changing.

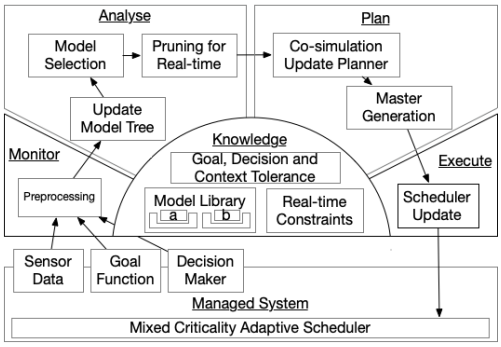

Fig. 7. Real-time Adaptive Abstraction and Approximation Architecture

The MAPE-K loop starts with a **M**onitor-phase where the necessary (processed) sensor data, the current goal function and the decision-maker are communicated to the adaptation mechanism. Here some processing occurs to allow for easier **A**nalysis. The analysis phase starts by updating a model tree. This model tree contains, in the ordered branches, all real-world objects that are passed by the sensor data. In the running example, we only have two objects: the two cars. It uses the (processed) sensor data to add or remove objects from the model tree incrementally. The Model Selection activity updates or selects the different models for each object in the model tree. The update is based on the goal, context and decision model, the quantified tolerances, and the available models with uncertainty quantification. The combination of these **K**nowledge items result in a set of rules to decide if a certain model can or cannot be used in this specific instance. Furthermore, the tree is also ordered with the most important objects in the first branches, and the most appropriate model in the first branch. In our case, the acceleration model is placed before the constant velocity model. If another car was present at a larger distance, the constant vehicle model would be first, and an empty model second branch. Using the accelerated model would not make sense as the tolerance for the decision-maker is high. The empty model is present, as the non-computing of the model reduces the execution time significantly but reduces the performance of the system. Finally, real-time constraints are imposed on top of the model tree to prune infeasible solutions. After the analysis phase, a set of feasible solutions is left. Based on this set of solutions, the co-simulation scheme is adapted and a master is generated that enables execution of the simulation. Note that each co-simulation unit contains the different possible models (possibly an empty model) for fast adaptation. Finally, in **E**xecute, the scheduler in the Managed system is updated.

The scheduler itself is an adaptive mixed-criticality scheduler based on [10]. This allows for responsiveness to overload conditions and imperfections in the execution time measurement of each co-simulation unit.

C. Discussion

This paper shows how to deal with the trade-off between uncertainty and real-time behaviour using models at different approximation levels. Consider if the number of cars increases in a scenario, the ECU will not have sufficient time to compute a prediction of all the different models.

The proposed conceptual model solves the dynamic environment where you cannot reason over real-time behaviour as worst-case bounds or where fallbacks in the decision logic are necessary. This solution needs computation of the MAPE-K loop and switching of models.

In this research, we address Tolerance Quantification. We need to be able to evaluate the tolerance of the decision making algorithm, since it is one of the model selection criteria. Off-line experiments are needed to see how a decision making algorithm responds to uncertainty. In the example, it is the tolerance to the uncertain distance and velocity vector that must be quantified.

V. Related Work

In previous work [11]–[13], we worked on the adaptivity of large scale simulations with surrogates. However, none of these techniques use a quantification of tolerance or reason over the real-time behaviour of the system.

The two most related techniques to our defined methods are mixed criticality systems and the imprecise computation model. Most of the complex embedded systems like automotive and avionic industries are mixed criticality systems. These systems deal with task-priority scheduling regarding execution time [14]. The imprecise computation is also a technique to deal with transient overload and improve real-time fault tolerance. This approach ensures that all critical tasks never miss their deadlines [15]. Both techniques do not take the uncertainty of the models into account but allow for adaptation based on real-time citeria.

VI. Conclusion and Future Work

In this paper, we presented a conceptual model for reasoning on adaptive approximation in a dynamic environment. We used a Simulink model for the simulation of the lane change control algorithm as a visual and tangible use case. Guided by it, we identify 3 main challenges faced for real-time adaptive approximation: model prediction uncertainty, dynamicness of environments and real-time constraints. To handle those challenges, we envisioned an integrated framework for adaptive approximation in CPS that balances the uncertainty and real-time behavior of the system. In the future, we aim to implement the framework with a supporting architecture, methods, and techniques to reason over the use of self-adaptive approximations and abstractions at runtime. We also want to create an appropriate modeling language and supporting techniques to find better runtime deployment solutions.

REFERENCES

[1] E. A. Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC)*. IEEE, 2008, pp. 363–369.

[2] S. Engell, "Cyber physical sos-definition and core research and development areas," Working paper of the Support Action CPSoS. Retrieved from http://www. cpsos . . . , Tech. Rep., 2014.

[3] A. Sousa-Poza, S. Kovacic, and C. Keating, "System of systems engineering: an emerging multidiscipline," *International Journal of System of Systems Engineering*, vol. 1, no. 1-2, pp. 1–17, 2008.

[4] M. Zhang, B. Selic, S. Ali, T. Yue, O. Okariz, and R. Norgren, "Understanding uncertainty in cyber-physical systems: a conceptual model," in *European conference on modelling foundations and applications*. Springer, 2016, pp. 247–264.

[5] D. Caughlin and A. F. Sisti, "Summary of model abstraction techniques," in *Enabling Technology for Simulation Science*, vol. 3083. International Society for Optics and Photonics, 1997, pp. 2–13.

[6] D. Bevly, X. Cao, M. Gordon, G. Ozbilgin, D. Kari, B. Nelson, J. Woodruff, M. Barth, C. Murray, A. Kurt, K. Redmill, and U. Ozguner, "Lane change and merge maneuvers for connected and automated vehicles: A survey," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 105–120, 2016.

[7] W. L. Oberkampf and C. J. Roy, *Verification and validation in scientific computing*. Cambridge University Press, 2010.

[8] B. Barroca, T. Kühne, and H. Vangheluwe, "Integrating language and ontology engineering." in *MPM@ MoDELS*. Citeseer, 2014, pp. 77–86.

[9] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[10] F. Guan, L. Peng, L. Perneel, H. Fayyad-Kazan, and M. Timmerman, "A design that incorporates adaptive reservation into mixed-criticality systems," *Scientific Programming*, vol. 2017, 2017.

[11] S. Bosmans, S. Mercelis, P. Hellinckx, and J. Denil, "Towards evaluating emergent behavior of the internet of things using large scale simulation techniques (wip)," in *Proceedings of the 4th ACM International Conference of Computing for Engineering and Sciences*, ser. ICCES'18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3213187.3213191

[12] ——, "Reducing computational cost of large-scale simulations using opportunistic model approximation," in *2019 Spring Simulation Conference (SpringSim)*, 2019, pp. 1–12.

[13] S. Bosmans, T. Bogaerts, W. Casteels, S. Mercelis, J. Denil, and P. Hellinckx, "Adaptivity in multi-level traffic simulation using experimental frames," *Simulation Modelling Practice and Theory*, vol. 114, p. 102395, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X2100099X

[14] A. Burns and R. Davis, "Mixed criticality systems-a review," *Department of Computer Science, University of York, Tech. Rep*, pp. 1–69, 2013.

[15] J. W. Liu, W.-K. Shih, K.-J. Lin, R. Bettati, and J.-Y. Chung, "Imprecise computations," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 83–94, 1994.