Preprints of the Fourth International
Symposium on Experimental Robotics, ISER'95
Stanford, California, June 30–July 2, 1995

Specification, Formal Verification and Implementation of Tasks and Missions for an Autonomous Vehicle

Konstantinos Kapellos*, Sofiane Abdou**, Muriel Jourdan*, Bernard Espiau*

* INRIA Rhône-Alpes
41, rue Félix-Viallet
38031 Grenoble Cedex - France
{name}@inria.fr

** INRIA Rocquencourt
Domaine de Voluceau,
78150 Le Chesnay Cedex - France
abdou@inria.fr

Abstract

This paper describes the use of modern approaches of formal verification of behavioral and temporal temporties in an experimentation of automatic vehicle following. We firstly recall the key concepts of the proposed specification method: the robot-task, which allows to describe in a structured way an automatic control law and the associated discrete-event handling; the robot-procedure, which is the specification of complex missions from the logical point of view. We then present the used environment (ORCCAD) and verification tools, based on the use of synchronous languages for the reactive part of the system. This is followed by a description of the conducted experiments, which involves two electric cars, within the PRAXITÈLE project. Results and foreseen future works are given in the conclusion.

1. Introduction

Robotic systems are hybrid systems operating in real time and handling events as well as "continuous computations". Reliable and easy programming of these systems at the "task level" requires a systematic method for the specification of the missions, formal verification of their execution from a continuous and discrete-time point of view and efficient implementation over the target architecture. Models proposed in the literature cannot handle in an *unified* way discrete events parts and highly complex continuous time expressions, like the control equations used in advanced robots. Nevertheless, for every component of the system, wellknown theories could be applied: automatic control theory for the design and analysis of the control law, reactive systems theory [2] for the discrete events aspects. In the robotic area the first theory is widely used, while this is not the case for the second one. As a consequence, a discrete events controller cannot in general be clearly isolated in an application. It is therefore not possible to apply formal verification methods to check its correctness, although this be required, for truely autonomous robots, in order to be as sure as possible before launching that the system will behave correctly.

The work presented here constitutes an extension of the ORCCAD ([7]) framework, which allows the specification, the simulation and the implementation of robotic elementary actions (called *Robot-tasks*), by harmoniously integrating discrete and continuous aspects. In this paper we propose a method for the specification, the validation by using *formal* verification methods (not simulation), and implementation of the events handling parts of robotic missions. This is achieved by composing Robot-tasks using a set of operators, the final result being called a *Robot-procedure*. The systematic translation of a *Robot-procedure* into suitable synchronous languages (see below) provides the robot "program" with a nice semantics. This allows us to methodicaly verify a large set of behavioral and quantitative temporal properties, including crucial properties of liveliness and safety, as well as the conformity with missions requirements.

In our approach, the controller behaviour is considered as a reactive system: *a system which maintains a continous interaction with its environment by sending outputs in reaction of inputs.* The family of synchronous languages, which aim at describing the complex ordering and causality relations between the inputs and the corresponding outputs of a reactive system, has been a very important contribution to the domain. These languages may be compiled into a wide class of models, usually labeled transition systems. The models are then used to perform static verification of *properties* (order relations between in-

puts and outputs) of the system.

We illustrate the concept of *Robot-procedure*, its use as a specification of complex robotic missions and the formal verification methods we use, on an example taken from the area of automatic vehicle driving. The long-term objective is to specify, validate and implement a virtual "train" of electric vehicles: each car is expected to closely follow the previous one automatically using dedicated sensors — a vision system is used to locate the previous vehicle in distance and angle — and a computerized system. The first vehicle would be the only one with a human driver. This work is a part of the PRAXITELE Project ([8]).

The paper will be organized as follows. In a first part, we will briefly present the ORCCAD system and its central entity the *Robot-task* (RT). In a second part, we will introduce the concept of the *Robot-procedure* (RP) which allows to combine RTs in a structured way in order to express complex controllers. The formal verification methods will be exposed in a third part. All these concepts and methods will be illustrated through an automatic vehicle driving example. Finally, we will describe the current status and the results of experiments.

2. The Orccad system

ORCCAD is a development environment for specification, validation by formal methods and by simulation, and implementation of robotic applications.

The formal definition of a robotic action is a key concept in the ORCCAD framework. It is based on two principles: a) most physical actions to be performed by robots can be stated as automatic control problems which can be efficiently solved in real-time, b) the characterization of the physical action is not sufficient for fully defining a robotic action: starting and stopping times must be considered, as well as reactions to significant events observed during the task execution. In order to capture coherently the diversity of elements involved in these two issues, two entities are defined: the RT, representing an elementary robotic action, where automatic control aspects are predominant although coherently merged with behavioral ones, and the RP, the RTs are the basic element of which, and where only behavioral aspects are considered.

The RT entity has now been well studied and validated through multiple experiments (see [6, 9, 7]). In the following, we therefore only present its behavioral aspects, since they are used to compose RPs, which are more extensively presented in section 2.2.

2.1. The Robot-task

A RT is formally defined as the *parametrized specification of an elementary control law, i.e. the activation of a control scheme structurally invariant along the task duration, and a logical behavior associated with a set of signals(events) which may occur just before, during and just after the task execution.*

From the control point of view, RTs specification requires the explicitation of the set of functions, models and parameters appearing in the analytical expression, in continuous time, of the control outputs to be applied to the actuators in order to perform the desired physical action. Besides, the specification of the logical behavior is obtained by setting the events to be considered and their treatment. In order to make that specification easier to people from the robotics community who are not familiar with this approach, the events and the associated processings are typed. We distinguish:

• the pre-conditions: their occurence is required for starting the servoing task. A temporal watchdog can be associated with each pre-condition.

• the exceptions: they are generated during the execution of the servoing task and indicates a failure detection. Their processing is as follows:

- type 1: the reaction to the received exception is limited to the modification of the value of at least one parameter within the control scheme,
- type 2: the exception requires the activation of new RTs. The reaction consists in killing the current one and reporting the causes of the disfunction to the adequate level. The recovering process to activate is known and specified, as discussed in section 2.2,
- type 3: the exception is considered as fatal. The overall application is stopped and the robotic system must be driven to a safe position.

• the post-conditions: often related to the environment, they are handled as conditions for a normal termination of the RT. Watchdogs can be associated to their waiting. Normal or forced termination of the RT are accompanied by the diffusion of specific synchronization signals.

Finally, a RT is completely specified by the explicitation of implementation aspects including temporal properties, like the sampling rates. This is done by implementing each RT in terms of communicating real-time computing tasks, called Module-tasks, which each implement a part of the control law or of the behavior.

The ORCCAD system provides also a graphical environment that offers a series of functionalities, ranging from specification of RTs by instanciation of its

object-oriented model and automatic generation of behavior-related programs, to its validation using SIMPARC ([7]) and realtime code generation running under VXWORKS.

2.2. The Robot-procedure

The aim in designing the RP entity is to be able to define a representation of a robotic action that could fit any abstraction level needed by the mission specification system. In its simplest expression, it coincides with a RT, while the most complex one might represent an overall mission. Briefly speaking, it specifies in a structured way a logical and temporal arrangement of RTs in order to achieve an objective in a context-dependent and reliable way, providing predefined corrective actions in the case of unsuccessful execution of RTs. More formally a RP is the full specification of

- a main program, which characterizes the nominal execution of the action and is composed of RTs, RPs and conditions,
- a set of triplets (exception event, processing, assertion), which specifies the processing to be applied for handling the exception, and the information to transmit to the planning level (if any), and
- a local behavior defining the logical coordination of the previously considered items.

The composition of RTs and RPs in the main program is obtained through operators which express the sequence, the parallelism, the conditional, the iteration, the rendez-vous and various levels of preemption.

2.2.1. The Application

Let us informally describe the considered application. We have two electric vehicles, the leader being driven and the second having to follow it like in a virtual train. Initially, the undriven car tries to catch the right signal in order to locate the first one. When done, the expected nominal execution is that the undriven car follows the driven one until all be stopped by an operator intervention. During the execution the video signal may be lost or irrelevant. In this case a parking manoeuvre is started. The driven car is supposed to come back and the train to be reformed. In addition, other situations must be monitored like physical damaging of crucial components.

This informal specification could be translated into a RP named FOLLOWME (see fig.1). Its nominal execution is described in the RP main program specified as an infinite loop the body of which begins with the test of the external condition *TargetFound* which indicates that the driven car is detected by the second one. Whenever it is satisfied, within a specified elapsed

```
Name : FollowMe
Pre-cond:OkInit[30ms],AutoMode[30ms],Start[5mn]
Main program:
        Loop
            wait TargetFound [5mn]
            start (GuardedFollow)
        EndLoop
T3 exceptions: Auto2man, MecFaill, ...
Post-cond : Stop [60mn]
```

Figure 1. FOLLOWME RP specification

time, a RP named GUARDEDFOLLOW, detailed below, is activated to control the second car. Let us emphasize that the programming is structured, in the sense that we can use a RP inside the definition of another one. Before starting the FOLLOWME nominal execution, a set of three preconditions must be satisfied before the indicated delays; the initialization phase (motors, sensors, ...) must have been achieved without detecting errors, the automatic mode must be activated and the "human" operator has to give the start order. The nominal execution of this main RP is stopped in two cases: either the supervisor gives a stop order or the manual mode is activated. In the first case, the RP ends normally; in the second one it is interrupted by a global T3 exception.

In this example, there are six involved RTs: two for the driving and steering motors of the car using exteroceptive sensor information, which virtually links the driven leader car to the non driven following one (named SENSLOC and SENSDIR respectively); two allowing the undriven car to track a reference trajectory on the basis of odometry information only (respectively nammed CARTLOC and CARTDIR); one (named BRAKE) using the foot-brake of the car in order to impose a desired deceleration. The RT SENSLOC is choosen to be presented in section 4.

The RP GUARDEDFOLLOW (see fig.2) is built from the parallel composition of the three RTs SENSLOC, SENSDIR and BRAKE . The RT BRAKE is activated every time the leading car imposes strong decelerations, indicated by *MoreBrake* event. Let us note that the first two RTs may be forced to stop if the exception of type 2 concerning the loss of the video signal between the two cars is detected. The RP GUARDEDFOLLOW handles this situation by starting a recovery program made of the RP PARKING, the specification of which is analog to GUARDEDFOLLOW RP. However, RTs based on odometry information only are used instead of the sensor based ones.

The RP formalism allows an user to program at the "task-level", without worring about the coding of

```
Name: GuardedFollow
Nominal execution :
    Parallel
       start (SensLoc)
       start (SensDir)
       Loop
        if  MoreBrake then start (brake)
       EndLoop
    EndParallel
T2 Exception: (TargetLost, start (parking))
```

Figure 2. GuardedFollow RP specification

tricky things, like signal exchange between elementary tasks. A systematic translation to the Esterel language ([1]) is provided, minimizing in that way the risk of errors. Even so, the complexity of programmed applications and their critical character require the use of formal methods for ensuring the correctness at the *specification* level.

3. Formal verification in Orccad

In the Orccad framework formal verification is used to prove the largest possible set of assertions about the execution of the RTs and RPs and, ideally, directly from their definition. A complementary objective is to provide the end-user, who is not supposed to be an expert in verification, with a systematic way to express the property to verify, to interpret the results and therefore to analyse the behavior of complex robotic actions.

Verification methods we used are model-based. They suppose that 1) the system is modelled by a labelled transition system; 2) the property is expressed either by another labelled transition system — in which case the property and the model are compared modulo a well-chosen equivalence relation — or by a formula from a temporal logic — in which case the formula are evaluated by computation on the model.

For the first point, the use of synchronous languages is very helpful, since their semantics are expressed in terms of labelled transition systems. This explains why we translate the RT and RP formalisms into a suitable synchronous langage. It is important to choose the good one since the type of properties we would like to check depends on the form of the model used to express the semantics of the language. Moreover their environment does not offer exactly the same type of verification tools.

We have chosen to translate our formalisms into: 1)Esterel, because it is friendly interfaced with Auto, a tool aimed to prove behavioral properties by reducing/comparing boolean automata; 2) the timed extension of Argos [5], since its semantics is expressed in terms of temporized automata (automata extended with real variables) which allow to prove time dependent properties ([4]). This is achieved by using Kronos [3] which is a model-checking tool for the real-time temporal logics tctl.

We now present how behavioral properties on RTs and RPs, and time-dependent properties on RPs can be verified. This last is justified by the fact that a lot of explicit delays appear in a RP specification.

3.1. Robot-task

Concerning the RT, we proved ([6]) that its specified behavior is non-blocking and satisfies the liveliness property (a successful termination of the RT can be reached from any state of its evolution), and the safety property (any fatal exception is appropriately handled by the emission of a specific signal driving the system to a safe situation). Taking advantage of the synchronous assumption on which Esterel is based, the RT behavior is encoded in a generic way, i.e. independently of any particular specification, a dedicated instance being obtained by an appropriate substitution of input/output signals. These properties are verified on the *generic* specification by observing the resulting behavior. Therefore the basic objects to be composed for designing complex robotic actions are ensured to have "good" properties.

3.2. Robot-procedure

Contrary to the RT behavior, which is proved to be always correct, the user is in charge of most of the RP verification. Nevertheless, the **crucial properties** of safety and liveness, and a time-dependent property which states that the maximal execution time of the RP is bounded, can be checked automatically.

The very important class of properties representing the **conformity** of the RP specification to the mission constraints must be handled by the user. In general, such constraints can be expressed in terms of relations between events and events, events and actions or actions and actions. Let us illustrate that by a few examples from our case study. Mission specification requires that the target is searched each time after its loss. This could be tested by specifying a behavioral property which is of the event-event type and involves *TargetLost* and *TargetFound* events. The property expression and the verification procedure can be automatically derived by simply indicating the relevant events. It is left to the user to interpret, visually in this case, the results of the method (fig.3).

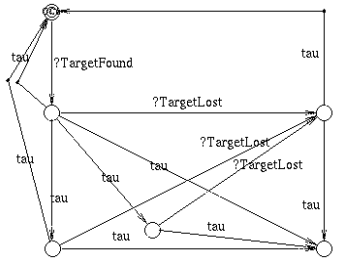

Figure 3. Verification process resulting automaton

Another example, which is of the type action-action and is time-dependent, consists of the proof that the time-lag between the starting points of the wheel execution control law and of the motor execution law is bounded. This property could be expressed by instantiating a well-chosen generic TCTL formula.

Finally, we consider verification methods also as a way of obtaining **abstract views** during the stage of RPs specification, in order to help the user in its mission design. Actually, every mismatch in the behavior specification is reflected by the resulting automaton. So, by projecting a complex behavioral activity in a simpler one at a relevant level of abstraction, useful (i.e easy to understand) views of the overall behavior can be given.

4. Experimental Issues

4.1. Description of the Facilities

We use two LIGIER electrical cars. A set of infrared emitters is mounted on the back of the first one, while the second is equipped with a vision system. This last consists of a linear CCD camera with acquisition frequency 1 Khz and a PC with an image processing board. It is connected to a VME rack, which includes a Motorola MVEM 162 board with four IP-modules (IP-DAC, IP-ADC, IP-QUADRATURE, IP-DIGITAL) and is dedicated to the car control [1].

The ORCCAD environment is running on a Sun Sparc workstation; the generated realtime code is loaded on the on-board resources through ETHERNET.

4.2. SensLoc Robot-task design

In order to understand how the elementary parts of the application are designed and how automatic driving is achieved, we now present the SENSLOC RT. The aim of this RT is to move the non-driven car longitudinally using information on the speed and on the distance between the two cars obtained through the vision system. The required action is: to start, after having checked that the connexion with the driving motors is OK and that the automatic mode is on. During its execution it is also required to detect a possible change of mode (switch to manual mode), a possible defect in the system and to verify that the leader car is always in sight of the camera. Let us now detail some aspects of its specification:

Continuous-time specification The control input a_c is the driving motors acceleration, given by ([8]): $a_c = (\frac{1}{h})\Delta v + \frac{1}{h^2}(\Delta x - hv - d_{min})$ where v is the velocity of the following car, d_{min} is the minimal distance and h, Δx, Δv denote differences between the two cars, in time, position and speed respectively. For its computation we use the following functional modules:

- PC: interprets the results of the image processing in order to compute Δx and Δv,
- EST: estimates the slopes,
- GT: computes the desired acceleration a_c,
- CO: converts the a_c in current and send it to the DAC,
- VOIT: reads the car state (wheels position, speed).

The event-based behavior of the task is expressed through two pre-conditions and two exceptions (PREC, EXC, ATR modules). Let us notice that the *TargetLost* exception is handled by a type-2 processing, which consists in stopping the RT and activating the recovery program specified during the design of the GUARDEDFOLLOW RP.

Time constrained specification For the implementation, a sampling period of 10ms is assigned to all periodic tasks, except for CO which runs at 5ms. In addition, non-blocking message passing mechanisms are selected for inter-tasks communication.

The full specification is validated by simulations using SIMPARC and, after that, effectively executed.

4.3. Experimental results

Software Issues The RTs used in the application are gathered in a library. The RP FOLLOWME was translated in a methodic way to an ESTEREL program. This last produces after compiling the automaton which controls the RTs. Its size is of 300 states and corresponds to 78751Kb of "C" code produced by the ESTEREL v4.40 compiler. The realtime software used for the execution is running under VXWORKS. In addition, a prototype environment integrating tools for the specification, analysis and verification of RPs is currently designed.

A Run Let us finally describe informally the evolution of a 10mn experimentation in the INRIA domain. Some results of part of this experimentation are given

[1] this work is made in cooperation with Aleph Technologies, Grenoble, France

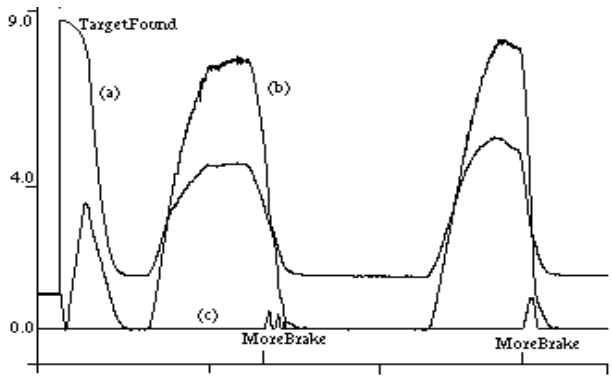

Figure 4. (a) inter-car distance, (b) speed of the following car, (c) brakes pressure

figure 4.
Initially, the inter-car distance was of 9m, the leading car being motionless. After a while the visual target was detected, so TARGETFOUND condition was satisfied, and therefore the GUARDEDFOLLOW RP was activated. Using the visual information, the undriven car after having reach the minimum inter-car distance (1.5m) followed the driven one with a velocity of 30 km/h and a range of environs 4m. At the third mn, the MOREBRAKE event was broadcasted because of a sudden deceleration of the leader vehicle; the RT BRAKE was activated increasing the pressure of the brakes of the following vehicle and therefore decreasing its speed (fig.4). The same situation reproduced later inducing the reactivation of the RT BRAKE. The loss of visual information, signaled by the TARGETLOST event, was immediatly handled by the activation of the recovery program PARKING, which simply drived smoothly the following vehicle at a motionless position. The driven car came back, the visual target was again detected, and the train reformed. The experiment finished when the user emitted the STOP signal.

5. Concluding Remarks

We have shown in this paper how formal specification and verification methods can be used for checking the behavioral and temporal correctness in a complex robotics application. This was illustrated by a non-trivial experiment of automatic vehicle driving in an outdoor environment. From that experimental study, we may draw some conclusions which open future research directions. A first thing is that, even with a sophisticated environment like ORCCAD, it is not obvious for a non-expert user to handle verification tools. It is therefore necessary to greatly improve their interfaces, in order for example to avoid the need for writing TCTL formulas. A second point is that the performances of the compilers and verification tools have to be also improved if we want to consider highly complex applications like underwater or space ones.

Provided that these issues are addressed, we guess that the relevance of the proposed approach is not questionable. It has been proved that specification and programming of rather difficult applications is done with a better robustness, is more efficient and allows easier modifications and evolutions with the proposed framework than using a classical method.

References

[1] F. Boussinot, R. de Simone: *The* ESTEREL *language*, Proccedings of the IEEE, 79(9):1293-1304, September 1991.

[2] D. Harel and A. Pnueli: *On the development of reactive systems*, in Logic and Models of Concurrent Systems, NATO Advanced Study Institute on Logics and Models for Verification and Specification of Concurrent Systems, Springer Verlag, 1985.

[3] T. Henzinger and X. Nicollin and J. Sifakis and S. Yovine :*Symbolic Model-Checking for Real-Time Systems*, LICS'92, IEEE Computer Society Press, 1992.

[4] M. Jourdan :*Integrating formal verification methods of quantitative real time properties into a development environment for robot controllers*, Technical report INRIA 2540, 1995.

[5] M. Jourdan, F. Maraninchi, A. Olivero :*Verifying quantitative real-time properties of synchronous programs*, 5th International Conference on Computer-aided Verfication, LNCS 697, Springer Verlag, 1993.

[6] K. Kapellos, *Environnement de Programmation des Applications Robotiques Réactives*, PhD thesis, Ecole des Mines de Paris, November 1994.

[7] D. Simon, B. Espiau, E. Castillo, K. Kapellos :*Computer-aided design of generic robot controller handling reactivity and real-time control issues*, I EEE Trans. on Control Systems and Technology, vol 1 no 4, dec 1993.

[8] Parent, M. and Daviet, P., *Automatic Driving for Small Public Vehicles., Intelligent Vehicule Symposium* ,Tokyo, July 14-16, 1993.

[9] P. Rives, R. Pissard-Gibollet, and K. Kapellos, *Development of a reactive mobile robot using real time vision*, in ISER'93, Kyoto, Japan, October 1993.

[10] V. Roy, R. de Simone :*Auto and autograph*, In R. Kurshan, editor, proceedings of Workshop on Computer Aided Verification, New Brunswick, June 1990.

[11] V. Roy, R. de Simone :*Auto and autograph*, In R. Kurshan, editor, *proceedings of Workshop on Computer Aided Verification*, New Brunswick, June 1990.

[12] L. Lynch, H.B. Weinberg: *Proving correctness of a vehicle maneuver: deceleration*, Second European Workshop on Real-Time and Hybrid Systems, Genoble, France 1995.