INŽENÝRSKÁ MECHANIKA 2005
NÁRODNÍ KONFERENCE
s mezinárodní účastí
Svratka, Česká republika, 9. - 12. května 2005

DESIGN OF COORDINATION MECHANISM OF WALKING ROBOT GAIT CONTROLLERS

T. Březina[1], V. Ondroušek[2]

Summary: *The contribution deals with the arrangement of coordination mechanism of gait controllers of four legged robot. Robot system is modeled as discrete events dynamic system (DEDS). Formation of control strategies is based on controller set whose instances are able to create independent robot behaviors. Based on formal logic methods initial set of controllers is extended by composite controllers. Their determination is always the parallel achievement of several control aims. Coordination mechanism of feedback controllers is then organized as finite nondeterministic state machine. Activation of all control laws in every state of system is underlying idea. Appropriate complexity of problem is reached through implementation of security limitations as well as limitations which imply from solved problems domain. Using the limited set of input information a suitable problem algorithmization is possible to obtain full automated arrangement of coordination mechanism.*

1. Introduction

The integral part of the walking robot design is the implementation of an autonomous system that will be able to work and move in an unknown and unpredictable environment. While designing the control system it is important to achieve the appropriate adaptivity without modelling complicated and unpredictable situations. One of the possible approaches (Huber, 2000) to this issue uses successfully the chaining of composite controllers activations. The instance of controllers generates simple independent behaviour of the robot. Using propositional logic we can deduce a set of composite controllers. Their aim is to achieve several particular control objectives simultaneously. Coordination mechanism design is strongly facilitated by both appropriate state space discretization and the description of control activation influence on the robot state. The model of all possible behaviours was implemented using non-deterministic finite machine the transition function of which is implemented by an oriented graph. The safety and task constraints implementation markedly facilitated the complexity of model.

[1] Tomáš Březina, doc., RNDr., Ing., CSc., VUT v Brně, FSI ÚAI, Technická 2, 61669 Brno, ČR
e-mail: brezina@fme.vutbr.cz

[2] Vít Ondroušek, Ing., VUT v Brně, FSI ÚAI, Technická 2, 61669 Brno, ČR
e-mail: yondro00@stud.fme.vutbr.cz

2. Used approach

The contribution describes our experiences with the algorithmization of the design of coordination mechanism of four-legged robot controllers. This work uses the results (Huber, 2000). Control system architecture is based on a small set of closed-loop control laws generating mutually independent cases of behaviour. That is why this set, $\{C\}$, is called control basis. In the case of robot walking task it contains contact force controller, Φ_f, and posture controller, Φ_k, (Bor, 2004; Březina, 2003). The instance of controller, Φ_i, is the base controller to which a concrete set of input resources (sensors), $\underline{\sigma}$, and output resources (actuators), $\underline{\tau}$, are assigned. The aim of the contact force controller is to achieve a stable position of the robot on three legs thanks to a position change of the point of landing of one of these three legs (to achieve triangular static stability). The aim of posture controller is to achieve, thanks to a change of orientation of the robot, such positions of leg actuators that are as close as possible to their central positions, which can be considered as the kinematically optimal position.

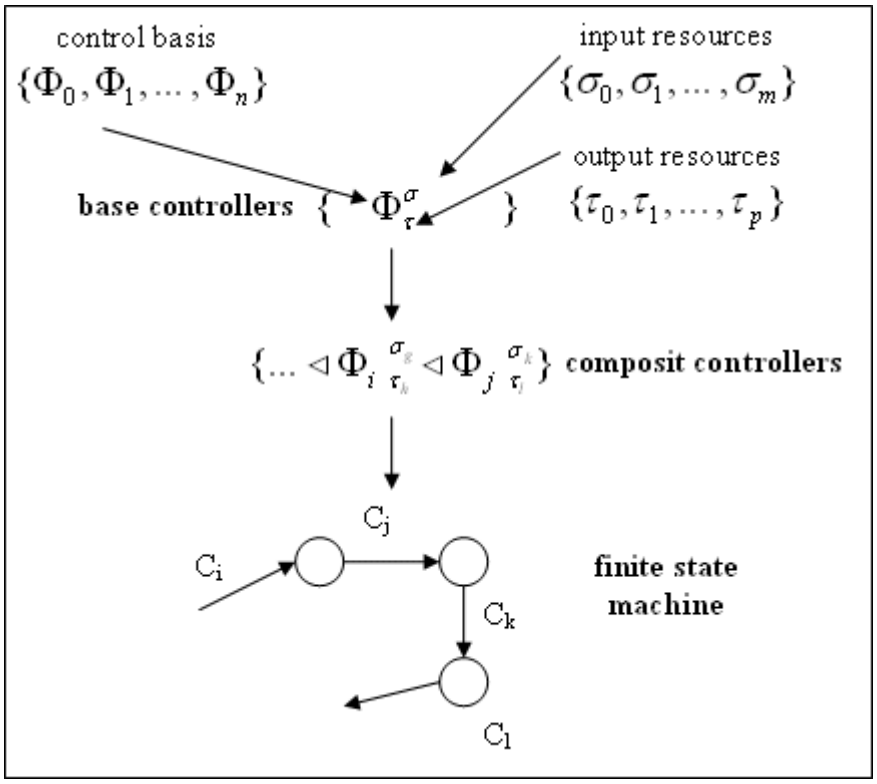

Fig.1 Coordination mechanism design process (Huber, 2000)

To determine the influence of closed-loop control laws activation on the robot state, it is necessary to find the appropriate discretization of the robot state space. The state is described by state vector $q = (p_0, p_1, ..., p_n)$, where $p_0, p_1, ..., p_n$ are predicates determining which of the particular control objectives are achieved. For example, be *0,1,2,3* points of landing of the robot´s legs and * a substitute meaning that any point of landing of any leg can be used, then predicate $p_0 \leftarrow \Phi_1 \, {}^{1,2,3}_{\underline{*}}$ is 1 (true) if triangular static stability is achieved with legs *1,2,3*. This example shows the way a discrete state is related to achieving objectives by control laws.

To achieve more complicated control objectives, composite activations of base controllers are used. These composite controllers are designed using the "subject to" operator, $\triangleleft$. For example, $\Phi_P \triangleleft \Phi_N$, where Φ_P is the underlying controller and Φ_N is the dominant controller. The underlying base controller can accomplish only such actions that cannot conflict with the operations of the dominant controller. The set of base controllers, $\{C\}$, can be extended by the set of composite controllers $C_m : (... \triangleleft (\Phi_i \, {}^{\sigma_j}_{\tau_j} \triangleleft \Phi_k \, {}^{\sigma_l}_{\tau_l})...)$. Whereas the set of base controllers must be defined by the designer, the set of composite controllers can be defined automatically using propositional logic, i.e. based on formal deduction only; for example, it has proved inapplicable to use composite controllers consisting of more than two position controllers, Φ_1, as it is impossible to achieve the objective of control law in more than two static stability triangles simultaneously. It is also impossible to use simultaneous activation of base controllers the objectives of which are conflicting, i.e. physically unachievable. Using this formal deduction we get a set of syntactically appropriate controllers $\{C^\circ\}$ that can be used to design a model of all possible behaviours of robot.

To prevent the designed model from being unacceptably complicated, it is advisable to reduce the set, $\{C^\circ\}$, using safety and task constraints. The constraints can be represented as logical expressions containing state vector predicates. For example, constraint $(p_0 \vee p_1)$ says that the first or the second state vector component at least is true, i.e. at least one of the control objectives of the base controllers that determine these predicates is achieved. The other vector components are not influenced by this constraint. The robot states complying with given conditions are then classified as inadmissible and unachievable, respectively. States without at least one statically stable posture are classified as inadmissible, in the task of robot walking. The configuration in which the objectives of position controller, Φ_1, would be achieved in the opposite static stability triangles, is considered as unachievable.

3. Implementation

Considering what has been written above, behaviour of the robot can be viewed as a chaining of dominant and composite controllers activations. That is why the coordination mechanism follows the state of the robot step by step and chooses a subset of such control laws from the appropriate controllers set, $\{C^\circ\}$, the activation of which cannot lead to inadmissible robot states, not even in the case if the appropriate control objectives are not achieved. The upper layer of the control system decides about which control law from the set will be activated.

The coordination mechanism is implemented using finite state machine, $A = (Q, \Sigma, \delta, q_0, F)$. The final set of internal states, Q, is determined by all admissible states of the robot. The input symbols set, Σ, is the same as the set of syntactically appropriate controllers $\{C^\circ\}$. The initial state, q_0, can be chosen at random. The final states set, F, depends on the domain of the tasks being solved. For robot walking task, no state can be determined as final because the process can be stopped in any random state. The transition function, $\delta : Q \times E \to P(Q)$, is implemented using a graph.

It is an oriented graph where each edge and node is linked to data. Nodes data represent state vector, *p,* and edge data represent control actions which generate the transition between the given nods. Closed-loop control law in the given state can result in several distinct states. This means that there are some nodes in the graph from which edges linked to the same data lead to different nods. That is why this machine is non-deterministic. The main idea of the design of the graph representing the transition function of non-deterministic machine was the activation of all control laws in all system states. Simultaneously with the new nods expansion, safety constraints were applied so as to avoid generating of an unuseably complicated structure. The used algorithm of the graph design is:

1) Save the initial state to FIFO of the states waiting to be processed.
2) Read the state from FIFO of the states waiting to be processed. If FIFO is empty - then end, if the state not included in graph - then add it.
3) for I:= 1 to (control laws number) do
    a) apply I-th control law on the actual state and design a set of states into which the control law activation can result, if any of the newly created states belongs among the inadmissible states, go for processing another control law, if the state is not among the states already used then add it into FIFO of the states waiting to be processed and into the graph structure,
    b) if there is no edge between the actual and newly created state then add the edge into the graph structure,
    c) add new data (I-th control law) to this edge
4) continue with step no. 2.

To be able to experimentally verify the accuracy of coordination mechanism, test software was designed using Borland Delphi. The software is able to design coordination mechanism in the way stated above for different types of tasks. Input data contain state vector predicates, a set of base controllers and (safety and task) constraints. To assess the capabilities of coordination mechanism and to tell if it has completed the given objectives, a method testing the program was implemented in it. Each test generates different behaviours of the robot as a chaining of states and control laws activations. The algorithm can be described as following:

1) Choose the initial state of the system and mark it as "actual"
2) For the actual state choose at random one of the control laws the activation of which is allowed by the coordination mechanism, and carry out its activation
3) Set up a set of all direct followers of the actual state which the system can reach by control law activation.
4) Choose a state at random from the set created in the previous step and mark it as the "actual system state". If this state is final, then end, if not - continue with step no. 2.

A random definition of control law that is to be activated is used to replace the decision of the upper layer of hierarchical structure of the control system in which the usage of one of the

algorithms of machine learning, for example Q-learning (Coelho, 2001; Huber, 2000) is expected.

4. Achieved results

The appropriacy and correctness of the described algorithm was tested while simulating three tasks: four-legged robot walking, peg-in-hole insertion using mechanical arm, manipulation task using four-fingerd hand.

For the first task, a controllers coordination mechanism for the task of four-legged robot walking was designed. The set of possible resources is formed by the legs of the robot, marked *0,1,2,3*, and the orientation of robot´s body, φ, see fig.2. The control basis is formed by 2 controllers: contact force controller, Φ_f, and posture controller, Φ_k. Through the binding of input and output resources, with control basis, the initial set of controllers, containing thirteen control laws was establishe $C = \{\Phi_f \frac{a,b,c}{\underline{a}}, \Phi_k \frac{0,1,2,3}{\underline{\varphi}}\}, a \neq b \neq c \in \{0,1,2,3\}$. The robot state was represented by state vector having 5 components:

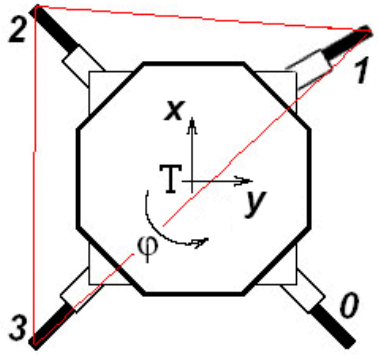

Fig.2 Robot walking task

$$p_0 \leftarrow \Phi_f \frac{1,2,3}{*}, p_1 \leftarrow \Phi_f \frac{0,2,3}{*}, p_2 \leftarrow \Phi_f \frac{0,1,3}{*}, \quad p_3 \leftarrow \Phi_f \frac{0,1,2}{*}, p_4 \leftarrow \Phi_k \frac{0,1,2,3}{*}. \tag{1}$$

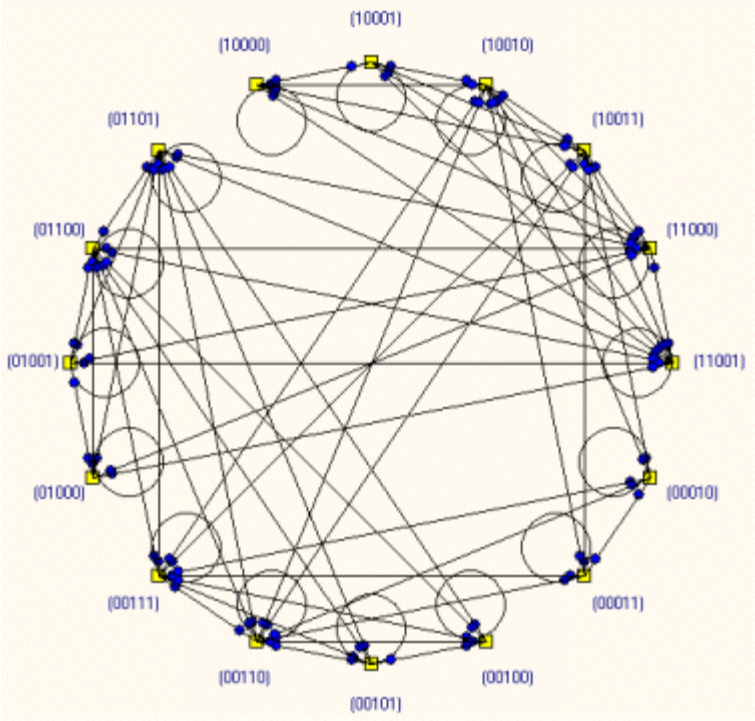

Fig.3 Model of all possible behaviours: robot walking task

To achieve an acceptable complexity of the model of all possible behaviours, see fig. 3, two constraints were used, $p_0 \vee p_1 \vee p_2 \vee p_3$ and $\neg(p_0 \wedge p_2) \wedge \neg(p_1 \wedge p_3)$. All the information was used as input data of the above stated software to design the coordination mechanism. The originally given set of control laws was automatically extended by 48 syntactically appropriate composite controllers. The resulting model contained 16 states, complying with constraints, and by 128 transitions between them. Out of the total number of 61 control laws, 25 were used for the model. The activations of other

control laws were not possible due to the constraints given.

For the other task a coordination mechanism for peg-in-hole insertion using a mechanical arm was designed. Following resources were used: orientations of the peg, $\varphi_x, \varphi_y, \varphi_z$, orientations of the hole, ξ_x, ξ_y, ξ_z, and scalar ξ_0, defining the distance of the peg to the ground, see fig. 4. The control basis of this task is formed by three base controllers, $\{c_{of}, c_{al}, c_{in}\}$. The aim of this controller, c_{of}, is a change of the position of the peg in the orientation of vector, ξ_z, i.e. this controller attempts to achieve distance, d, between the peg and the ground but the orientation of vector ξ_z does not change. The controller, c_{al}, attempts to align the orientation and position of the peg with respect to the hole, the distance of the peg from the ground does not change. Control objective of controller, c_{in}, is to minimize the distance between the peg and the hole, and perform the insertion. The state of the system can be represented by state vector, $\vec{q} = (p_0, p_1, p_2)$, that has three components. Each of the predicates indicates the achievement of an objective of one of the controllers stated above in the following way $p_0 \leftarrow c_{of}$, $p_1 \leftarrow c_{al}$, $p_2 \leftarrow c_{in}$. For the design of coordination mechanism two constraints were applied, $\neg(p_0 \wedge p_2)$ and $p_0 \vee p_1$, that substantially simplified the resulting model of all possible behaviours.

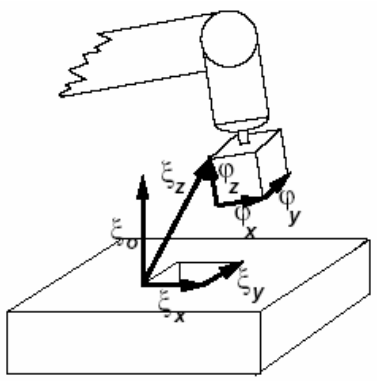

Fig.4 Peg-in-hole insertion

The information given above was used as the input data for the designed software. The software extended the control basis by 4 composite controllers:

$$C^{\circ} = \{c_{of}, c_{al}, c_{in}, (c_{of} \lhd c_{al}), (c_{al} \lhd c_{of}), (c_{al} \lhd c_{in}), (c_{in} \lhd c_{al})\} \quad (2)$$

Mentioned software used a set of syntactically appropriate controllers to design the model of all possible behaviours, see fig. 5. The resulting model consisted of only 4 states, remaining 4 states were marked as inadmissible and unachievable based on the given constraints. Between the states 12 various transitions were designed each of which carries four different control activations at average. The instance of controller, $c_{in} \lhd c_{al}$, was not used as the only one from the set of syntactically appropriate controllers.

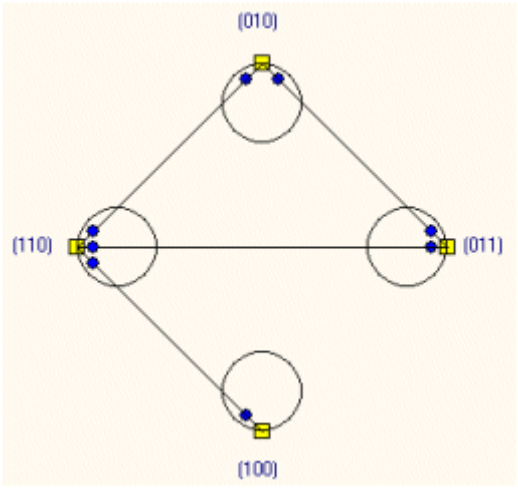

Fig.5 Model of all possible behaviours: Peg-in-hole insertation

For the next task, coordination mechanism was designed for the manipulation with an object using four-fingered hand. The set of input and output resources consists of the hand´s fingers marked as 0,1,2,3, and the orientation of the hand, φ. Similarly to the robot walking task, control basis consists of controllers

Φ_f and Φ_k. The instance of controller Φ_f, e.g. $\Phi_f \frac{0,1,2}{1}$, achieves a stable grasp with fingers 0, 1, 2 and optimizes the grasp and contact forces distribution, respectively, by moving finger 1. The instance of controller Φ_k, e.g. $\Phi_k \frac{0,1,2,3}{\varphi}$ attempts to achieve, by changing the orientation of the hand, such positions of finger´s actuators that are as close to the central positions as possible. For the design of the model two constraints were used. The first is a safety constraint, $p_0 \vee p_1 \vee p_2 \vee p_3$, that stipulates that for each step there must exist at least one statically stable three-fingered grasp. The second is a task constraint, $\neg(p_0 \wedge p_2) \wedge \neg(p_1 \wedge p_3)$, that defines which of the three-fingered grasps can occur at the same time. The complexity of the resulting model of all possible behaviours is comparable to the complexity of robot walking task. Out of 32 possible states 14 were assessed as inadmissible and 2 as unachievable. In the model 13 instances of controllers and 12 composite controllers were used, see fig. 6.

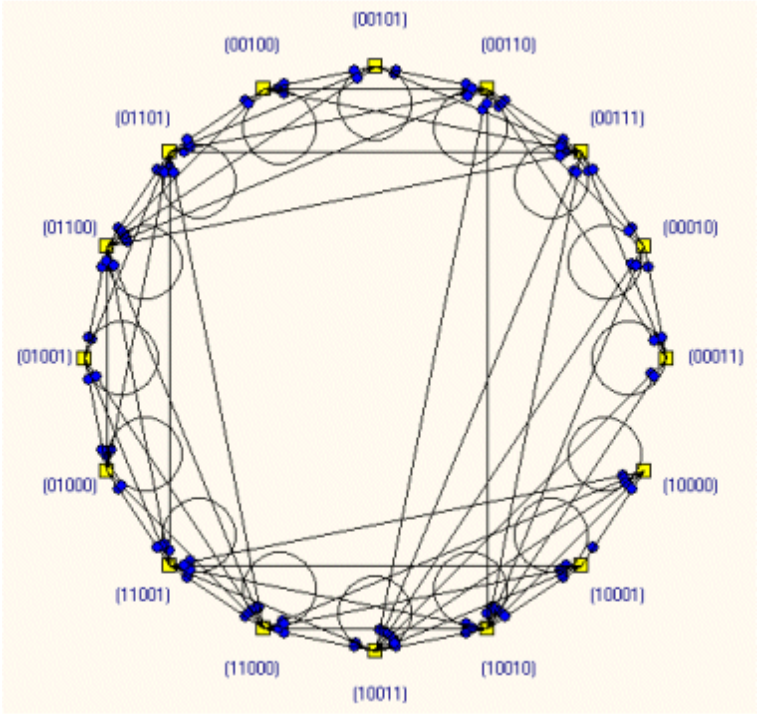

Fig.6 Model of all possible behaviours: manipulation task using four-fingerd hand

5 Conclusion

The experimental tasks have shown that the above described method of coordination mechanism design can be algorithmized. Based on a small amount of input data and using propositional logic it is possible to deduce a large number of composite controllers. It is possible to use these parallel activations together with the given instances of base controllers to design the model of all possible behaviours of the system. Substantial minimization of model´s complexity is achieved using constraints. It is necessary to distinguish between safety constraints defining the inadmissible states and task constraints defining the unachievable states. None of the coordination mechanism tests did freeze. None of the control activations did result in an impossible state that would not comply with the given constraints. Coordination mechanism has met the requirements. It can be used for an upper layer design using some of the methods of machine learning. Further research should focus on the extension of control basis by other controllers for the walking robot task, and the upper layer design.

6 Acknowledgement

This contribution was composed within the framework of research projects MSM 0021630518 "Simulation modeling of mechatronic systems ".

7 References

Bor K., Návrh a implementace kompozitního řídicího členu chůze 4-nohého robotu, Diplomová práce, FSI VUT v Brně, 2004.

Březina T., Houška P., Sedlák P., Singule V.: Walking gait of Four-Legged robot obtained through Q-learning, Sborník národní konference Inženýrská mechanika, Svratka, 2003.

Coelho, Jr. J.A.: Multifingered Grasping: Grasp Reflexes and Control Context, Ph.D. Dissertation, Univ. of Massatchusetts, Amherst, 2001.

Huber M.: A Hybrid Architecture for Adaptive Robot Control. Ph.D. Dissertation, Univ. of Massatchusetts, Amherst, 2000.

Ondroušek V.: Návrh a implementace koordinačního mechanismu řídicích členů chůze robotu, Diplomová práce, FSI VUT v Brně, 2004.