HIGH-LEVEL CONTROL OF FES-ASSISTED WALKING USING PATH EXPRESSION

Jonkers, H. and Schoute, A.L.

Department of Informatics, University of Twente,
Enschede, NETHERLANDS

ABSTRACT

In order to restore the walking function of paraplegic patients, artificial stimulation of the leg muscles can be applied (Functional electrical stimulation, FES). Since a few years, studies have been made of the high level control of these stimulation systems. In this paper high level control of different aspects of gait is analyzed and represented by discrete event-type decision models. The models are in first place represented by finite state diagrams. Relations between these models are specified separately. For the actual implementation of the walking models use is made of a special operating system in which synchronization is based on so-called path expressions. Path expressions are "sentences" of an expression language, by which restrictions are imposed upon the order in which actions are allowed to occur. The advantage of path expressions is that expressions can be formulated separately for different aspects of gait, whereas the relations between these aspects can be specified by path expressions as well. The above-mentioned operating system has been extended with extra actions corresponding with the phase of the gait. By simulation, it can be checked whether the used control strategy results in the desired behaviour. Because the path expressions can be modified easily, it is possible to experiment with different model ion this way.

INTRODUCTION

Attempts to restore the walking function of paraplegic patients by means of functional electrical stimulation (FES) have been made for over 25 years. Up to now, simple control mechanisms have been used for this kind of artificial walking. For instance, a hand-switch is used to initiate a step, which has to be triggered by the patient. Moreover, no use is made of sensors for checking the correct progress of step cycle. However, since a few years studies have been made of the high-level control of FES systems (the group at Biomedical engineering department, University of Twente and Rehabilitation Center "Het Roesingh", in Enshede, the Netherlands) [1].

For control of FES-assisted artificial walking high-level and low-level control can be distinguished. The high level control can be provided, voluntarily, by the patient. This can be done directly by means of manual commands (for instance switches to

initiate a step or to select between different modes of walking), or indirectly by means of sensors which detect body movements. In the second place, automatic control [2] can be used, which can't be influenced by the patient; an example of this kind of control is the use of feedback information from sensors. Usually two kinds of control are combined. Low level control is focused on specific joints; it has the function to stabilize or move these joints in a predetermined way, provided that this is enabled by the high level control. Figure 1. shows a block diagram of a possible FES system with hierarchical control. This paper mainly handles about the *High level control* block; parts of it could also be classified ion the *Intention detection* block.

First, some examples of control models of FES controlled gait will be given, in which the patient's wishes are automatically passed to the system by means of sensors. These models are decision models, they are not physical models of human gait. Rather than one all-embracing model, different sub-models will be made for the different aspects of artificial walking; the relations between those models will have to be specified separately.

The model can be represented by means of finite state diagrams. A problem with this representation is what, when several finite state diagrams are combined in order to model the total behaviour, the number of states and transitions soon becomes very large and the overall survey can easily be lost.

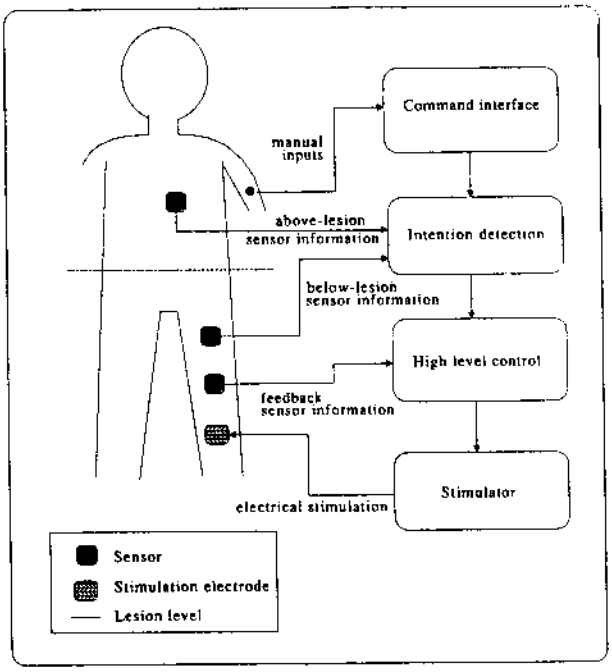


Figure 1. Block diagram of a FES system with hierarchical control

For that reason it is tried to represent the walking models by so-called *Path expressions*. Path expressions are *sentences* of an expression language, in which different operators are used to impose restrictions upon the order in which actions occur. Because different expressions can be made for different aspects of a model, the modularity will be enhanced, which results in a more comprehensible solution. A prototype operating system has been developed at the Computer Science department, The University of Twente. The process control in this system is attained by means of path expressions (Path Expression Based Operating System, PBOS). With the aid of this operating system the path expression representations of the models can be executed.

Finally, computer simulations of gait according to different control models were performed to check whether the control strategy results in the desired behaviour. This was done by extending PBOS with extra actions which represent the phases of gait on a screen by means of icons and text.

MODELS OF HUMAN GAIT

Human gait can be divided in different phase, which occur subsequently. Depending on how detailed a walking model has to be, more or less phases can be distinguished. In the simplest case, a distinction is made between the so -called *double support phase*, in which both feet are in contact with floor, and the *swing phase*, in which only one foot touches the floor and the other leg makes a forward swing phase. The swing phase can be subdivided in two parts: the *initial* and the *terminal swing*. During the initial swing, the knee flexes and the leg makes a slight forward swing movement. During the terminal swing the knee is brought back in extension by means of a quadriceps activation.

A model based in this division of gait has been described by Andrews [3]. A finite state diagram of this model is given in Figure 2. In this model, a close loop system for stability of the knee is implicit; during double support phase both artificial knee extending reflexes are active, during a swing phase only the artificial reflex of the supporting leg is active. Three types of sensors are assumed to be present. In the first place the foot soles contain pressure sensors to detect a weight shift and a foot contact. Further there are four goniometers. Goniometers for measuring the hip angles serve to detect a transition from the initial; to the terminal swing; goniometers for measuring the knee angle can be used to check whether a knee is in extension, as part of an artificial knee extending reflex. If crutches are used, they can also contain pressure sensors or switches to be used in the step intention conditions. A timer is required to generate a time-out.

This model has some important limitations. First, the only relationship between a right and a left step, is that they can't occur at the same time; there is no restriction at all to the order of the steps. One of the reasons it that no distinction is made between the situation with the feet position next to each other and the situation with one feet before the other in the double support phase. Further, there are only few direction possible to failures that can occur; the only unusual situation that is considered is the leg flex time-out. Another important limitation of this model is, that possible parallelism of phases is not shown explicitly.

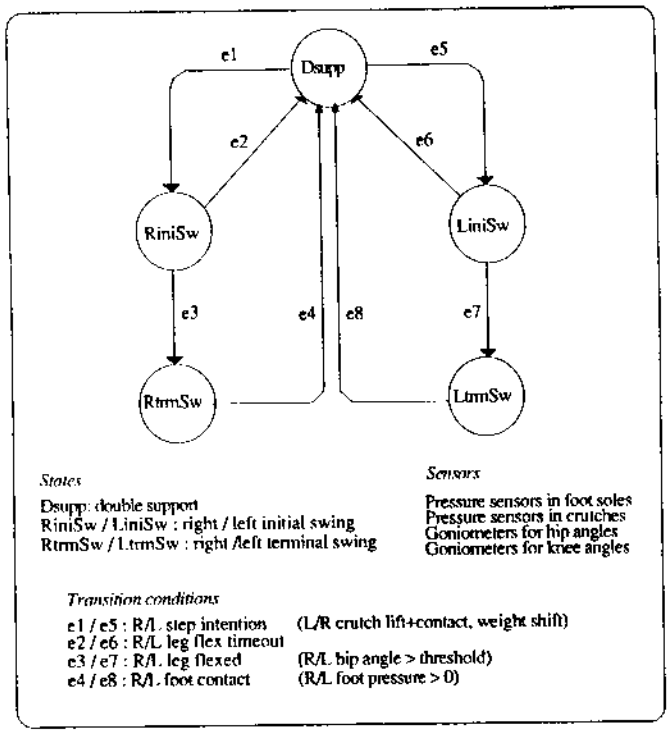


Figure 2. Simple model of Andrews

A second model based on another article of Andrews [4], can be considered as an extension of the first model and does not have these limitations. A finite state diagram of this model is given in Figure 3. The same sensors are assumed to be present as in the first model. However, the artificial knee extending reflex has been made explicit by dividing the support phases in the phase with and the phase without quadriceps stimulation [5]. Moreover, the step intention has been added as an extra phase which simplifies the definition of the conditions for state transitions. In the step intention phase one of the condition for transition to the swing phase has become true (crutch load or weight shift). It the second condition becomes tues as well the actual transition of the swing phase will be made.

There are two separate finite state diagrams, one for the right leg and one for the left leg, which work in parallel independently of each other. A step with right leg and a step with a left leg can never be initiated at the same time though because a weight shift to the left leg can not take place at the same time as a weight shift to right leg. For a similar reason a step with one leg can't be initiated during the swing phase of the other leg.

In the described model different positions of one leg in respect to the other are distinguished. This is an extension to the original model as described by Andrews. A parameter k is introduced which classifies the difference between the right and left hip angle as a measure of the relative position of the legs. This parameter has an integer

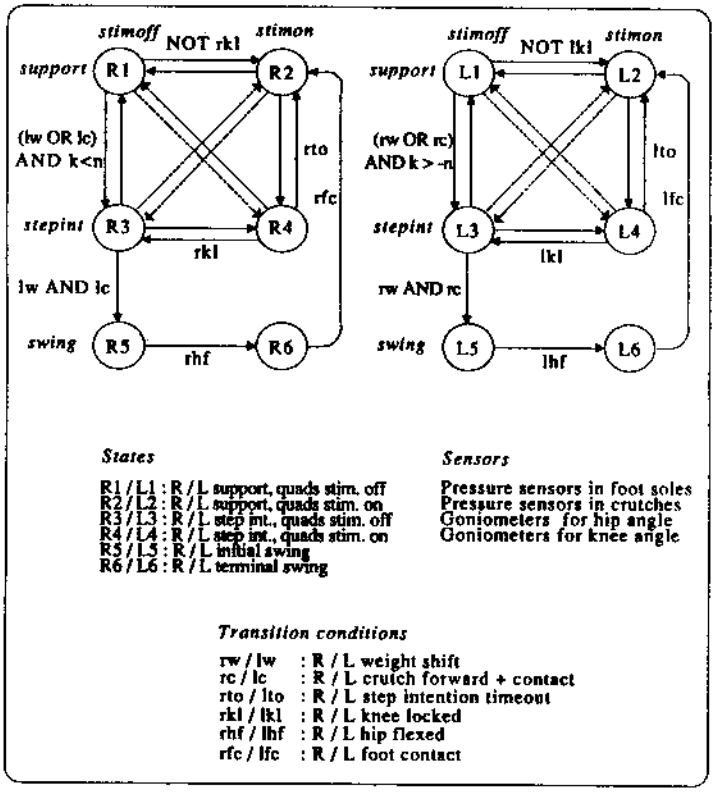


Figure 3. Extended model of Andrews

value between *-n* and *n*. The distance between the legs can be classified more accurately if a greater value of *n* is chosen. The different possible values of *k* have the following meaning:

- *if k > 0, the right leg is positioned before the left leg,*
- *if k< 0, the left leg is positioned before the right leg,*
- *if k = 0, the legs are positioned next to each other.*

The double support phases (both right and left support) for every value of *k* between *-n* and *n* for *n* = 2 are shown in Figure 4.

With this subdivision the possibility of taking the step can be restricted in order to enhance the patient safety. For instance, the conditions for phase transitions can be defined in such a way that the absolute value of *k* can never be come grater than *n*:

- *if k = n, a step with the right leg is prohibited,*
- *if k = -n, a step with the left leg is prohibited,*
- *the step size is chosen in such a way that k will stay within the permitted boundaries.*

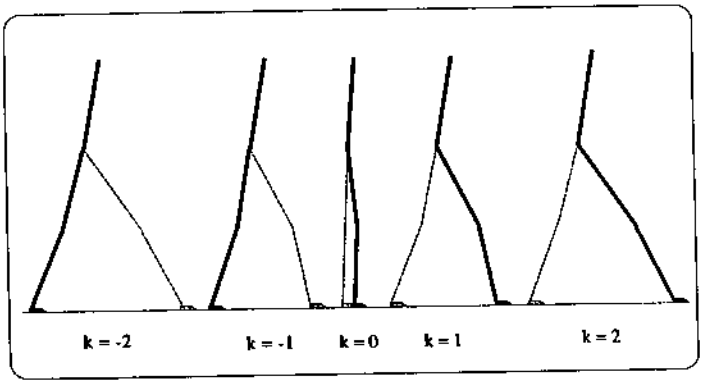


Figure 4. Double support for different values of *k*

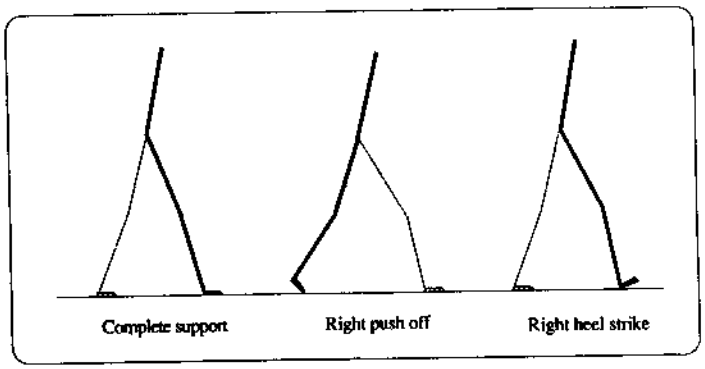


Figure 5. Subdivision of the double support

The transition from the double support phase to the swing phase and inversely the position of the feet can be considered as well. For instance, the push off phase can be added, in which the toes still touche the floor but the heel is lifted, and the heel strike phase in which the heel touches the floor but the toes are of the floor. The push off and heel strike phases are actually part of the double support phase, because both feet are in contact with the floor. The situation in which both feet are flat, that is when both heels and all toes touch the floor, will henceforth be called complete support. Figure 5 illustrates this subdivision of the double support. In order to be able to use the information about the subdivision of the double support in control strategies each foot sole must contain at least two switches or pressure sensors.

A model for the use of foot switches is given in Figure 6. This model can not be used without another model because no sensors to detect patient's wish to take a step are considered; however the model can be use in combination with other models as an extra guide of the correct progress of the step. There are two switches for both feet:

one at the heel (RH rest. LL) and one at the toes (RH rest. LET). The state in which the switch is closed is called state 1, the state in which it is open is called state 2.

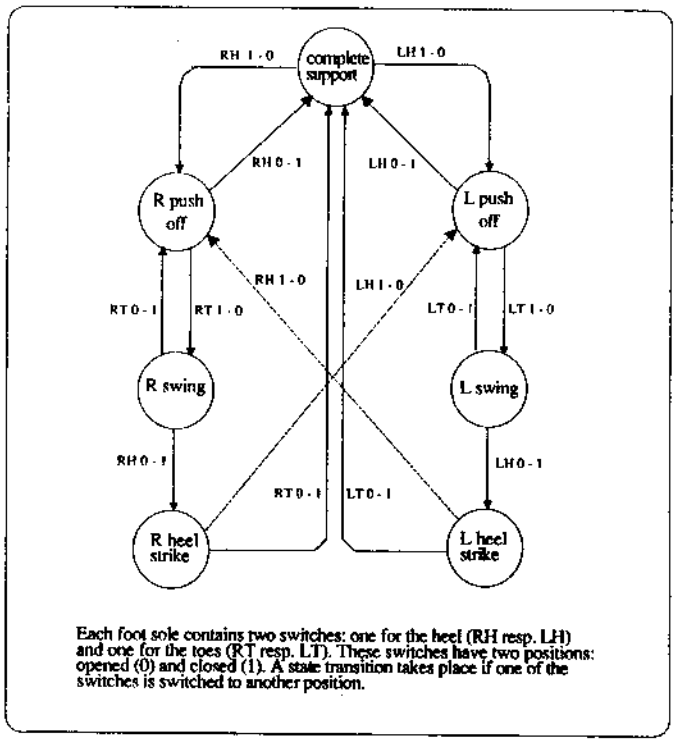


Figure 6. Model for foot switches

Next to the positions of the legs and feet, the use of crutches can be considered separately. This offer extra information about the progress of the step cycle. There are two basic forms of walking of crutches. In the first form the crutch on the opposite side is advanced before the step is taken with the leg. This means the crutch and legs will allow easy be moved alternately. In the second form which is called the swing through method, both crutches are advanced first, after which a step is taken with each leg. In both cases a crutch is loaded, that is an increased force is applied on the crutch, before a swing of the leg on the same side takes place. The situation in which both crutches touch the floor with normal force is called *double crutch support*; this phase usually appears approximately at the same time as the double support phase of the legs.

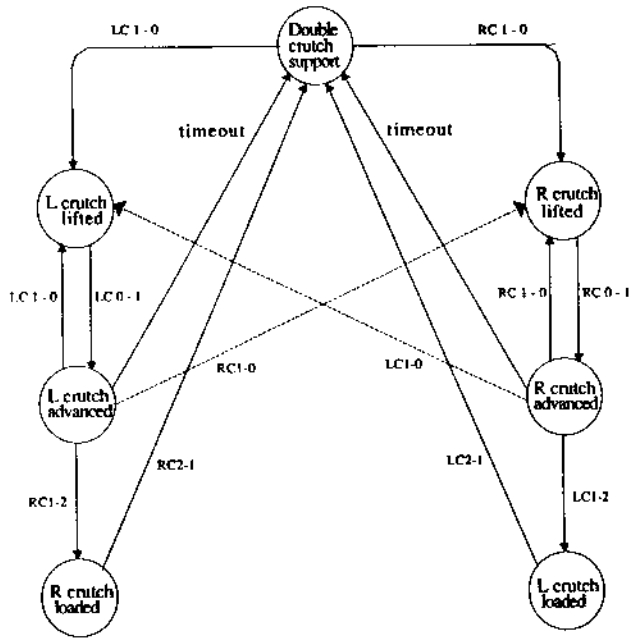
In figure 7 a model for the use of crutches is given, which can only be used in combination with other models, like in the model for foot switches. If this model is used in combination with the first model of Andrews, the conditioned connected with crutches in that model can be left out. It is assumed that the crutches contain switches or pressure sensors, and that three states can be distinguished:

State 0: the crutch is free from floor;

State 1: the crutch touches the floor with normal force;

State 2: an increased force is applied to the crutch.

The first basic form of walking with crutches is assumed to be used in this model. Another model would be necessary to control the swing through method with crutches or to allow both forms.



Both crutches contain a pressure sensor (resp. RC and LC); the measured pressure is divided in three classes: no pressure (0), pressure for normal floor contact (1) and pressure for a loaded crutch (2). A state transition takes place if the pressure class of one of the sensors changes.

Figure 7. Model for crutches

In the preceding, models have been derived for the different aspects of FES-controlled artificial walking. The advantage of using separate models for the different aspects instead of one all-embracing model is that the small models can be designed and tested independently; this enhances the comprehensibility and reduces the chance to make mistakes. However, in order to get a working combination of models, it does not suffice to give an implementation for each model separately; the relations between the different models will have to be specified as well. One solution for finite state diagram is indicating which phase in one model corresponds completely or partly with which phase in another model. In most cases this is not sufficient; other restrictions will have to be imposed upon the order of phases from different models. As will be shown in the next section, this can be done rather easily when using path expressions.

MODELLING WITH PATH EXPRESSIONS

Path expressions, introduced by Habermann and Campbell [6] as a means to attain process synchronization, offer a compact, algebraic notation to express the order of actions in a parallel processing environment. The path expressions used here resemble predicate path expressions, which were introduced by Andler [7].

Path expressions can be considered as a generalisation of regular expressions. Regular expressions are well-known in the area of formal languages and compilers [8]. Whereas a regular expression defines a set of possible *strings* over some alphabet of symbols (or in other words defines a language of sentences), path expressions define possible *evolutions* over an alphabet of operations or actions (that is a language of *behaviour*). Path expressions therefore provide a convenient way to specify process behaviour [9].

The behaviour imposed by path expressions, can be derived by controlling the evocation of actions on-line in a fully automated way. As such, path expressions can be used as a control mechanism within a multi-taking operating system. A prototype of a path expression based operating system has been developed and is described in the next section. With such a system it is possible to control FES-assisted walking at a high-level. The purpose of this section is to give a general idea of path expressions and show the examples how the gait models of the previous section can be described in terms of them. A more complete treatment can be found in [10].

The basic path expressions constructions that we will consider are:

- ***sequential*** composition (using the sequence operator ';')
- ***parallel*** composition (using the parallel operator '&')
- ***alternative*** operator (using the choice operator ',').

Let us illustrate the use of each of these operators. In Andrews' models, the swing phase has been subdivided into a initial swing and a terminal swing. Let the actions corresponding to the subphases be denoted by *XiniSw*, where *X* stands for R and L (right or left leg). The *sequential order* of these actions is expressed by:

XiniSw;XtrmSw

After the double support phase, either a right initial swing or a left initial swing can follow; these alternatives can be described with the *choice* operator:

RiniSw,LiniSw

Assume the walking steps of the right and left leg are described separately by path expressions *R-leg* and *L-leg*. The simultaneous behaviour is specified by the *parallel* composition:

R-leg & L-leg

A special notation is used for *repetitive* forms of sequential and parallel orders:

- sequential repetition: *{path}-min+max*
- parallel repetition: *{path}-min+max*

The repetition bounds specify the minimum and maximum number of sequential respectively parallel activations of the repeated path expression. If omitted the minimum is zero and the maximum is unlimited.

With the notation for sequential repetition, the simple model of Andrews can be described as:

{DSupp;((RiniSw;{RtrmSw}+1),(LiniSw;{LtrmSw}+1))} (P1)

Note that {P} + 1 represents an *optional* path expression P, which may be skipped or executed at most one time. Because the terminal swing phase can be skipped in case of a time-out, this phase is optional.

The activation of a path expression P may be made dependent upon a predicate by introducing a so called *conditional* path expression: *IF predicate THEN P*.

In predicate conditional path expression, the evolution of the path *P* is postponed until the predicate becomes true. The predicate has the effect of a guard: if it is false and an alternate path *Q* is enabled, *Q* will be activated instead of *P*.

Conditional path expressions can for example be used to check conditions for state transitions in the gait model:

{DSupp; (IF cR THEN (RiniSw;{RtrmSw}+1),
IF cL THEN (LiniSw;{LtrmSw}+1))} (P2)

with *cR* and *cL* specifying the conditions for a right and left swing.

In the prototype system predicate expressions may contain relations between two kinds of special variables: *event counters* and *time stamps*, which are associated with each action name. Also predicate functions may be part of predicate expressions. Predicate functions are boolean function procedures which test application dependent conditions. The value of a predicate function may change as a result of asynchronically occurring events or interrupts.

If conditions for the activation of actions are independent of their position on the path expression, they could be given by separate path expressions. The following path expression rule specifies a parallel repetition of all possible actions. In principle such a construction does not impose any restriction on the order of evolution of the actions. However, by guarding the actions with predicates, an action may only occur when the predicate condition is met.

[IF e2 OR e4 OR e6 OR e8 THEN Dsupp, (P3)
IF e1 THEN RiniSw,
IF e3 THEN RtrmSw,
IF e5 THEN LiniSw,
IF e7 THEN LtrmSw]

The predicates *e1* to *e8* refer to the conditions on the finite stated model of figure 2. Note that the conditions for several transitions leading to an action may be taken together in one conditional path expression, connected with an *OR* operator.

More than one path expression can be specified to control the order of actions; in that case, action may occur only in an order compatible with all path expressions. For example, the combination of path expressions P1 and P3 (this is sometimes called the *intersection* of the two expressions [11], represents precisely the simple walking model as given in figure 2. Multiple path expression interact with each other in *common* action names are specified.

In the simple walking model, the aspect of controlling both legs in parallel has been ignored. The double support phase should in fact be seen as simultaneous left and right support. When, for example, a left swing is activated, the right leg support has to be maintained.

In order to model this aspect we need two finite state diagrams executing in parallel, one for each leg (see figure 3). In terms of path expressions this corresponds to the parallel composition of path expressions components that describes the legs separately:

R-leg={Rcontact;RiniSw;RtrmSw} (P4)
L-leg={Lcontact;LiniSw;LtrmSw}
R-leg & L-leg

Here, we also introduce a new feature of our path expression language, namely a *macro* facility. A path expression may be defined as a macro in the following way:

name=path expression

When the *name* appears anywhere in a path expression, it will have the same effect as when the right hand path expression would be substituted. So, *R-leg & L-leg* is equivalent to *{Rcontact; RiniSw; RtrmSw} & {Lcontact; LiniSw; LtrmSw}*. Macros introduce the possibility of recursion in the language. An infinite process of alternating actions a and b could be defined recursively: *proc = a; b; proc*.

The parallel behaviour of both legs must be restricted by the fact that a right and left swing may not happen at the same time. This restriction may be implicit if the corresponding actions are guarded by predicates which exclude each other. The restriction may, however, be imposed explicitly by an additional path expression rule:

{(RiniSw;RtrmSw),(LiniSw;LtrmSw)} (P5)

At the beginning of each cycle path expression P5 will allow either a right or left initial swing. However, when say a left swing is initiated by action *LiniSw*, P5 only allows action of *LtrmSw*, whereas it will inhibit *RiniSw* (and also *RtrmSw*). It should be noted that the path expressions only restrict actions of which the name appears in the expression. Hence the action *Dsupp* is not constrained in P5. The synchronized behaviour is obtained by combining both path expressions P4 and P5.

More detailed modelling of the leg support phase is described in the previous section to the following subspecification of *Xcontact:*

Xcontact={Xsupport;Xsteint} & {Zstimon;Xstimoff}

The *support* and pint phases take place alternately; at the same time, there is no alteration of the *stimon* and *stimoof* phases (quadriceps stimulation on and off). The four states *X1* to *X4* (with X either R or L) in figure 3 are related to these actions by:

Xsupport={X1,X2}
Xstepint={X3,X4}
Xstimon={X2,X4}
Xstimoff={X1,X3}

As mentioned before, one may keep track of relative positions of the legs by introducing a discrete parameter ranging from -n (left leg completely advanced) to n

(right leg completely advanced). In order to prohibit left and right swings in these extreme cases, one may control the leg movements by the following macro definitions:

$$pos(-n) = \{RiniSw\};RtrmSw;pos(-n+1) \quad \text{(P6a)}$$

$$pos(k) = \{RiniSw,LiniSw\};\{RtrmSw;pos(k+1),LtrmSw;pos(k-1)\} \quad \text{(P6b)}$$

$$pos(n) = \{LiniSw\};LtrmSw;pos(n-1) \quad \text{(P6c)}$$

where the macro *pos(k)* in P6b must be defined for $k=-n+1, .., n-1$

We assume that a change of the relative position (by one unit) happens as a result of the terminal swing phase. Because initial swing can occur without actually leading to a terminal swing (the step fails), the path expressions must allow multiple occurrence of left and right initial swing. However, in the extreme positions an initial swing with the 'wrong' leg is prohibited. The position control, as given by the macros, has to be initiated by a path expression pos(0), assuming that in the beginning both legs are positioned next to each other. If steps of different sizes (one or more units) are distinguished, the description will become more complex. It may even be questioned if the relative position should be taken into account at the path expression control level. The distance parameter k could remain hidden within the procedures that implement the actions. Initial swing actions could be guarded by predicates in order to prevent impossible steps. The macro definitions P6 nevertheless illustrate how parameter dependant behaviour can be modelled in a recursive manner.

Up to now, path expressions are derived form the walking models by interpreting the finite state diagrams in an informal manner. In case of more complex finite state diagrams, such as the model for foot switches and crutches, this may not be so easy. However, a systematic method exists to convert finite state diagrams into path expression form by using macros. For each state X of a finite state diagram a macro may be defined that describes the transition behaviour when being at state X. The possible transitions are represented by an alternate expression in which all subsequent states Y (similarly represented by a macro) are listed, possibly with a predicate condition *Cxy* and associated action *Txy*.

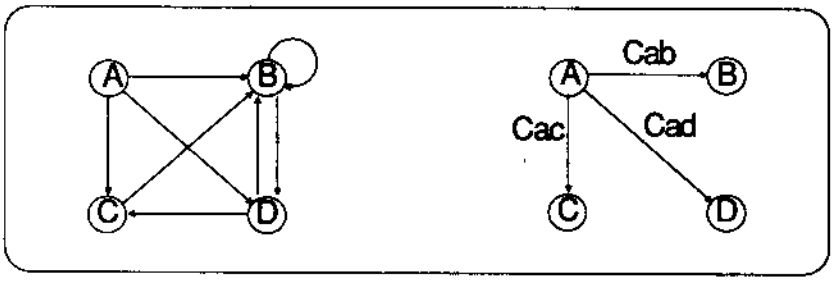


Figure 8. Simple finite state diagram (left) and transition from state A (right)

The finite state machine of figure 8 is described by a set of four macro definitions, one for each state:

A=IF Cab THEN(Tab;B), IF Cac THEN(Tac;C), IF Cad THEN(Tad;D)

B=IF Cbb THEN(Tbb;B), IF Cbd THEN(Tbd;D)

C=IF Ccb THEN(Tcb;B)

D=IF Cdb THEN(Tdb;B), IF Cdc THEN(Tdc;C)

Note that a transition from a state to itself presents no problems. A finite state diagram starting in A, is simply given by path expression A. The representation of a finite state diagram is given above in its most extensive form. Generally, not all transitions are subjects to conditions, ant not all transitions execute a transition action.

We will omit a complete exposition of all path expressions covering the various finite state models. A recursive macro representation, derived by systematic conversion by intelligent translation by hand, appears to be ther best approach.

In order to model the total walking behaviour, we have to combine the different behaviourial aspects as given by the finite state diagrams. At this point the power of path expression may become clear. Different finite state models are easily combined by joining their path expression representations. Or, in other words, complex behaviour can be disentangled into separate rules, which describe partial behaviour and are possibly based on simple finite state diagrams. Whereas a single finite state machine only describes sequential behaviour, parallel behaviour of multiple finite state machines can be easily obtained by the parallel composition of their individual path expression representations.

In combining models typically some path expression rules have to be added that impose restrictions upon the order of related phases in different models. For example, in a combined model that has been worked out in [10] an expression is added such that after every heel strike a double crutch support occurs before a crutch lift can be activated. Another path expression takes care that a swing can only take place in the corresponding crutch has been loaded.

THE PATH EXPRESSION BASE OPERATING SYSTEM (PBOS)

On-line control according to path expressions has been made possible by a prototype system named PBOS [11]. It has been built on top of a multitasking system on the IBM/PC written in Turbo-Pascal. The idea of PBOS is to evoke and control all actions within the system by means of path expressions. Actions mentioned in these path expressions are basic system commands or application dependant tasks that can be specified at the user command level.

A special feature in PBOS, not present in other path expression based systems, is that an action name may be qualified as being either an *activator* or an *acceptor*. This qualification depends on the fact whether the action itself or an enclosing (sub)expression is defined as activator or as acceptor. A path expression P may be qualified explicitly as activator by writing P! and as acceptor by writing P?. The qualification of a path P may be determined implicitly dependent upon whether P is entered into the system as *command* (typed immediately as a command line), in which

case it is treated as activator, or as *rule* (entered by means of special system tasks), in which case it is treated as acceptor. The collection of path expressions that at any moment controls the system behaviour is called the *rulebase*. If a command or rule is successfully compiled according to the path expression syntax, it is added to the rulebase.

A run-time mechanism determines which actions are enabled or disabled according to the evaluation state of the path expressions in the rulebase. If an activator is enabled in some path and the action is not disabled by any other path, it will be activated. The execution of an activator leads to the forking of the appropriate task. Actions may correspond to standard system tasks or to application defined tasks. When the actions of a command are executed and the command terminates, it is removed from the rulebase.

Special system tasks exist to add path expression s as rules to the rulebase, to remove arid to modify them. Actions that are unknown are simulated by a process that prints a start message, sleeps a (small) random time and then print a stop message. The control system is very rigorous in the sense that also these system actions have to obey the rules in the rulebase.

Stimulation control with on/off phases can serve as example to illustrate the use of activators and acceptors in PBOS. To start or stop a stimulation sequence the commands *start* and *stop* can be given to the PBOS command shell. The actions *start* and *stop* have no further effect. The actions *stimon* and *stimoff* will execute phases with stimulation respectively without stimulation. The following rule can be defined to get the desired result:

{start;{stimon;stimoff}!;stop}

First, the action *start* is enabled, but it is not activated yet because the action has the function of an acceptor. When the command *start* is given in a command line, the action is activated though, because an action in a command line is an activator by default. After that, the actions *stimon* and *stimoff* are activated alternately, for they have the function of an activator in the rule. After each termination of *stimoff*, the action *stop* is also enabled; therefore, this action is activated after *stimoff* if the command *stop* has been given in the command line. After this, only the action *start* is enabled again, so this action has to be invoked in order to restart the stimulation.

SIMULATION OF FES-CONTROLLED GAIT

In order to be able to see if the implementation of the models of FES-controlled gait results in the desired behaviour, a computer simulation of artificial walking has been performed [10]. A new shell module was made, which implements the actions added to PBOS. The added actions correspond with the different phases of gait, and show a graphical representation of the phase on the screen, together with the name of the phase and the value of *k*. Further, two actions to switch to the graphical mode and back to the next mode are available. The added actions are summed up in table 1.

graph	enter the graphical mode	**rtrmsw**	right terminal swing
text	return to the text mode	**ltrmsw**	left terminal swing
csupp	complete support	**dcsup**	double crutch support
rpoff	right push off	**rlift**	right crutch lifted
lpoff	left push off	**llift**	left crutch lifted
rhs	right heel strike	**radv**	right crutch advanced
lhs	left heel strike	**ladv**	left crutch advanced
rinisw	right initial swing	**rload**	right crutch loaded
linisw	left initial swing	**lload**	left crutch loaded

Table 1. Actions that are simulated graphically (names and their meaning)

Because of the limitations that the PBOS system still has, the influence of most sensor inputs were not yet simulated. Only simple input form the hand switch box was used to the represent the foot switches in one of the simulations. For the other simulations the action evolution was observed as specifies by the path expressions without guarding the phase transitions by external conditions.

The gait procedures show a picture corresponding with the current phase of the gait in the screen depending on the value of k together with some explaining text. Crutches are treated separately: if the crutches free from the floor, cross is drawn next to the body above the floor line; if the crutch touches the floor an arrow is drawn to the floor next to the body. The length of this arrow depends on the ground reaction force applied to the crutch; this force can be in one or two classes: normal force and increased force. Both for the legs and for the crutches, the left side is represented by

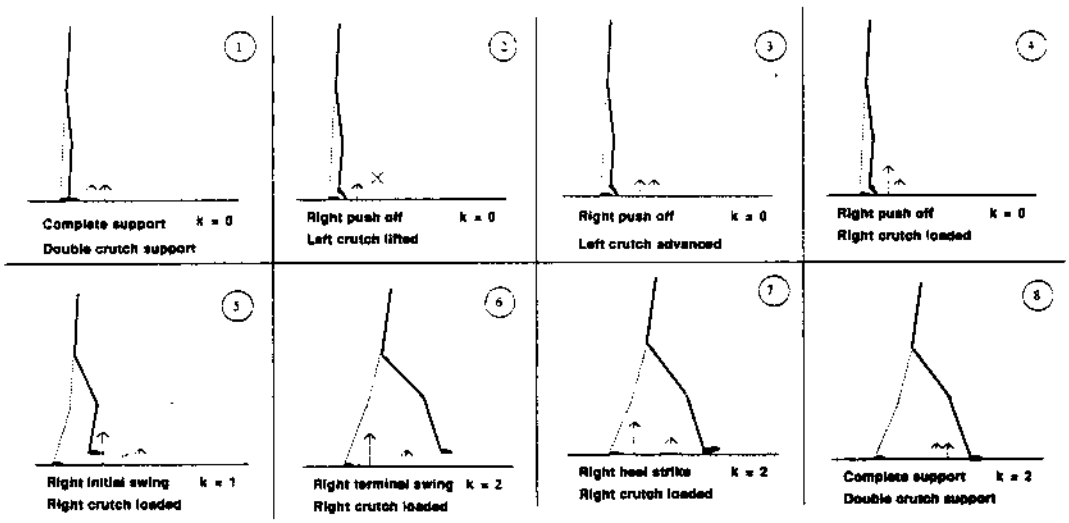


Figure 9. Sample sequence of simulation results.

dashed lines, the right side by solid lines. Figure 9 shows the possible sequence of pictures shown on the screen during the simulation.

CONCLUSIONS

Gait models of artificial walking have been designed in order to use them as a specification of a high level FES control system. In first instance the model description is given in terms of finite state diagrams. Finite state diagrams are useful and illustrative methods to represent discrete event type decision models, but only if models are not to complicated. Fort more complex models or if models are combined the number of a states and transitions in a diagram will soon became very large and the overall survey will be lost.

Path expressions appear to offer a convenient way to implement control models for FES-assisted artificial walking. The most important advantage of the use of path expressions is that modelling can be done in a modular way: different path expressions can be specified for different aspects of walking and relations between these sub-models can be also be specified with expressions. Finite state diagrams still may be used as an intermediate model representation, because they can always be converted into a set of path expressions. As shown, path expressions may be derived from finite state diagrams by informal means or by a standard method which uses recursive macro definitions. However, in many cases it is easy to derive path expressions directly from a given situation and constrains.

The describe gait models are of an exploratory nature, there are not the only possible models. By using the path expression based operating system PBOS, any set of path expressions can be entered as rules and/or commands into the system and will be executed. The PBOS system will control the basic actions mentioned in the path expressions, in a fully automated way. Their effect will depend on the implementation at n underlying level.

In order to experiment with different kind of models, basic actions have been simulated graphically. Control rules can be changed interactively; it can be checked if the model is still correct after the modification, or advantages or disadvantages of alternative models can be found. The system can therefore be useful for phototyping purposes.

REFERENCES

1. Hermens,H.J. et al, (1989) Development of Practical Hybrid FES Systems, 3rd Vienna, Intern. Workshop on Functional Electrostimulation, Vienna

2. Thrope,G.B., P.H.Peckham and P.E.Crago, (1985), Computer-controlled multi-channel stimulation system for laboratory using in functional neuromuscular stimulation, IEEE Trans. on Biomed.Eng., Vol BME-32-6:363-370

3. Andrews,B.J., (1988) Rule-based control of hybrid FES orthoses, Proc. IFAC Symp. on Modelling and Control in Biomed.Eng.

4. Andrews,B.J. et al, (1988) Hybrid FES orthosis incorporating closed-loop control and sensory feedback, J.Biomed.Engineer, Vol 110-4:189-195

5. Mulder,A.J., H.B.K.Boom, H.J.Hermens and G.Zilvold, (1990) Artificial reflex stimulation for FES induced standing with minimum quadriceps force, to appear in Med. & Biol. Engineer. & Comp., 1990

6. Campbell,R.H. and A.N.Habermann, (1974), The specification of process synchronization by path expressions, **Lecture Notes in computer science**, 16, Springer Verlag

7. Andler,S., (1979), Predicate path expressions, Sixth annual ACM symposium on principles of programming languages, San Antonio

8. Aho,A.V. and J.D.Ullman, (1988), **Principles of Compiler Design**, Addison Wesley, Reading Mass.

9. Schoute,A.L. and J.J.Luursema, (1990) Real-time System control by means of path expressions, Euromicro workshop on real time systems, Horsholm, Denmark, IEEE Computer Society Press

10. Jonkers,H., (1990), High-level control of FES-assisted walking using path expressions, Ms.thesis, Dep.of Computer Science, University of Twente, The Netherlands

11. Schoute,A.L., (1988), Path expression based operating system, Memorandum, INF-88-68, Dep.of Computer Science, University of Twente (ISBN 90-365-0240-3)

4. Andrews,B.J. et al, (1988) Hybrid FES orthosis incorporating closed-loop control and sensory feedback, J.Biomed.Engineer, Vol 110-4:189-195

5. Mulder,A.J., H.B.K.Boom, H.J.Hermens and G.Zilvold, (1990) Artificial reflex stimulation for FES induced standing with minimum quadriceps force, to appear in Med. & Biol. Engineer. & Comp., 1990

6. Campbell,R.H. and A.N.Habermann, (1974), The specification of process synchronization by path expressions, **Lecture Notes in computer science**, 16, Springer Verlag

7. Andler,S., (1979), Predicate path expressions, Sixth annual ACM symposium on principles of programming languages, San Antonio

8. Aho,A.V. and J.D.Ullman, (1988), **Principles of Compiler Design**, Addison Wesley, Reading Mass.

9. Schoute,A.L. and J.J.Luursema, (1990) Real-time System control by means of path expressions, Euromicro workshop on real time systems, Horsholm, Denmark, IEEE Computer Society Press

10. Jonkers,H., (1990), High-level control of FES-assisted walking using path expressions, Ms.thesis, Dep.of Computer Science, University of Twente, The Netherlands

11. Schoute,A.L., (1988), Path expression based operating system, Memorandum, INF-88-68, Dep.of Computer Science, University of Twente (ISBN 90-365-0240-3)