On ETS Representation of Human Movement

Lev Goldfarb, Ian Scrimger

Faculty of Computer Science
University of New Brunswick
Fredericton, Canada

July 30, 2007

Abstract

To partly compensate for the present lack of applications in our main ETS paper [1], we present an illustrative and tentative example of the application of ETS to human movement, i.e., to the representation of a leg as it is engaged in various manners of walking.

In this limited domain and to a limited extent, it should shed light on the methodology of the application of ETS.

For simplicity as well as for the convenience of the reader, we decided to focus on the representation of a single leg's movement only, while modelling the leg in as simple a manner as possible for this purpose. We view leg movement as resulting from a sequence of (predefined) *changes/transformations* each affecting the current limb movement processes locally, i.e., the movement of limb parts attached to the same joint. Most of these changes—each represented as an ETS primitive transformation—modify the flow of movement in at most two joint-adjacent limb parts (and hence the corresponding primitive modifies the "flow" of its initial primal processes, each representing the movement of the corresponding limb parts).

In a sense, the main objectives of the paper are to touch on the role of sensors, as well as on the compositional role smaller/constituent classes of movements play in the representation of larger classes of movements.

Familiarity with the main ETS paper is assumed.

1 Introduction

As we mentioned in the abstract, our objective here is to shed at least some light on the methodology of the application of ETS. Above all, we wish to illustrate how, in a relatively simple setting, one goes from observations to the corresponding ETS representation, i.e., how one chooses primitives and the corresponding sensors, and how one builds structs.

First, it is important to keep in mind that even within the same application, an ETS "sensor" can be implemented in a variety of ways: in software, hardware, or their combination. Of course, although special-purpose ETS sensors are always preferable, at present

conventional hardware and software could be used for this purpose. The main point to grasp is that—independent of the technical means used to capture ETS events—consistent with the role of conventional mathematics ETS proposes a *formal* framework for organizing (via structs) information associated with natural and man-made phenomena. In particular, ETS postulates that it is the *classes* of relevant "objects"—which are now viewed as processes—and the associated *class representations* that should play the central (organizing) role.

In that context, one might consider ETS as the first formalism for "symbolic" representation, with primitives being the symbols that have the same structure as the corresponding real events that they capture. Such considerations led us to suggest that biological neurons might play the role of various event sensors.

Given our objective towards representing the class of leg movements when a leg is involved in "walking" on a *level* surface (e.g., pavement, floor), we chose as simple a model of leg movement as possible. Even for this simplified model it is quite conceivable that similar representations might be useful in a number of settings: computer generated animation, autonomously generated robot movement, motion capture, or in any other area where it is important to generate complex classes of movement out of a set of simpler ones.

Another of our objectives is to draw to one's attention that in the context of pattern recognition, machine learning, and data mining, ETS is the first representational formalism that allows and promotes *systematic* reliance on the structure of already-learned simpler classes, i.e., on their class representations, when learning more complex classes. This, for the first time, introduces genuine modularity into the learning process, which is consistent with a wide variety observations in biology and psychology. In the case of movement, as is the case with program design, having a modular system to embody movement (either in a robot or an animated character) substantially reduces the development time as well as improves the reliability of a system capable of complex movement. Moreover, the generative nature of class representation in ETS would allow such a system to generate *new kinds of movement* which were not previously observed by or introduced to the system.

Finally, we would like to mention that this preliminary application is added to the special issue at the last minute, since, at this time, even a simple illustration (of the substantially improved *just completed* version) of ETS is definitely better than none at all. Obvious time constraints prevented us from addressing the more advanced topics, especially those related to movement learning, or the ETS concept of multi-staged representation.

2 Primitives and their sensors

As far as sensing of, or "measuring", real objects/processes is concerned, ETS obviously suggests a radically different approach. One might call such kinds of measurements "structural measurements", since the result of a measurement is either a concrete primitive—which is the basic structural unit in ETS—or the identification of one of a primitive's primal processes. The temporally recorded and appropriately connected outputs of such sensors produce ETS structs. Note that once the initial-stage representation of the objects/processes has been recorded via sensors, one may or may not need to rely on any other, higher-level or higher stage sensors, depending on the efficiency needs and the sophistication of available hardware.

Obviously, the initial-stage sensors are a must, while the higher level/stage sensors are a luxury that one should be able to live without, since non-primal classes and transformations are always first learned inductively.

First, we note that, *as much as possible*, the design of primitives should proceed independently of their implementation via sensors. This, of course, does not mean that the implementational issues should be ignored. In other words, although the "convenience" of the implementation of primitives should be considered during their design, the latter should not drive the design. Moreover, during the design of primitives, one should consider a reasonable variety of their possible implementations.

Also, one should be prepared for the situation in which, within the same application, different families of primitives would require quite different hardware solutions.

Second, we would like to draw one's attention to the following interesting and, if corroborated, important experimental observation: "a dolphin could . . . recognize the shape of an object through one sense (either echolocation or vision) that it had earlier interrogated through the alternate sense" [2]. This observation would indicate that independent of the chosen modality—each with its own, fundamentally different from each other, primitives—quite different representations of the same object could be equally successful. Thus, generalizing the above observation, one can draw the following "common sense" conclusion: even for a given modality there might/should be several satisfactory choices for the basic primitives.

2.1 On the implementation of ETS sensors

It is only natural to think of a sensor (or a related family of sensors) as designed for detecting a particular abstract primitive(s), even though the concrete output of such a sensor is a concrete primitive. Moreover, depending on the application domain, it might sometimes be convenient to rely on small software modules to implement some or all sensors, especially in various conventional information retrieval areas (where, typically, the number of such abstract primitives might be substantial but quite manageable).

For each abstract primitive, two general sensor architectures are conceivable, indirect and direct: in the first, the event is recorded on the basis of particular changes in the flow of its primal processes detected by sensors monitoring these processes, and in the second, *in addition* to the primal process sensors, a special sensor is used for detecting the transforming event itself (i.e., the event which this primitive is standing for). In some exceptional cases, where the relationship between primitive events is absolutely transparent, one may not need process sensors if one has adequate event sensors. Note that the sensors for primal processes allow one to properly link the primitives to each-other, i.e., to build structs.

2.2 Our primal classes and primitives for leg movement during walking

What is the situation with our illustrative example? In Figure 1, we show our primal classes and their symbolic notation. For a particular leg, any primal process is associated either with a "regular" movement of a leg part or with its fixed position: of the hip (relative to the other hip), of the thigh (relative to the hip), and of the lower leg (relative to the thigh).

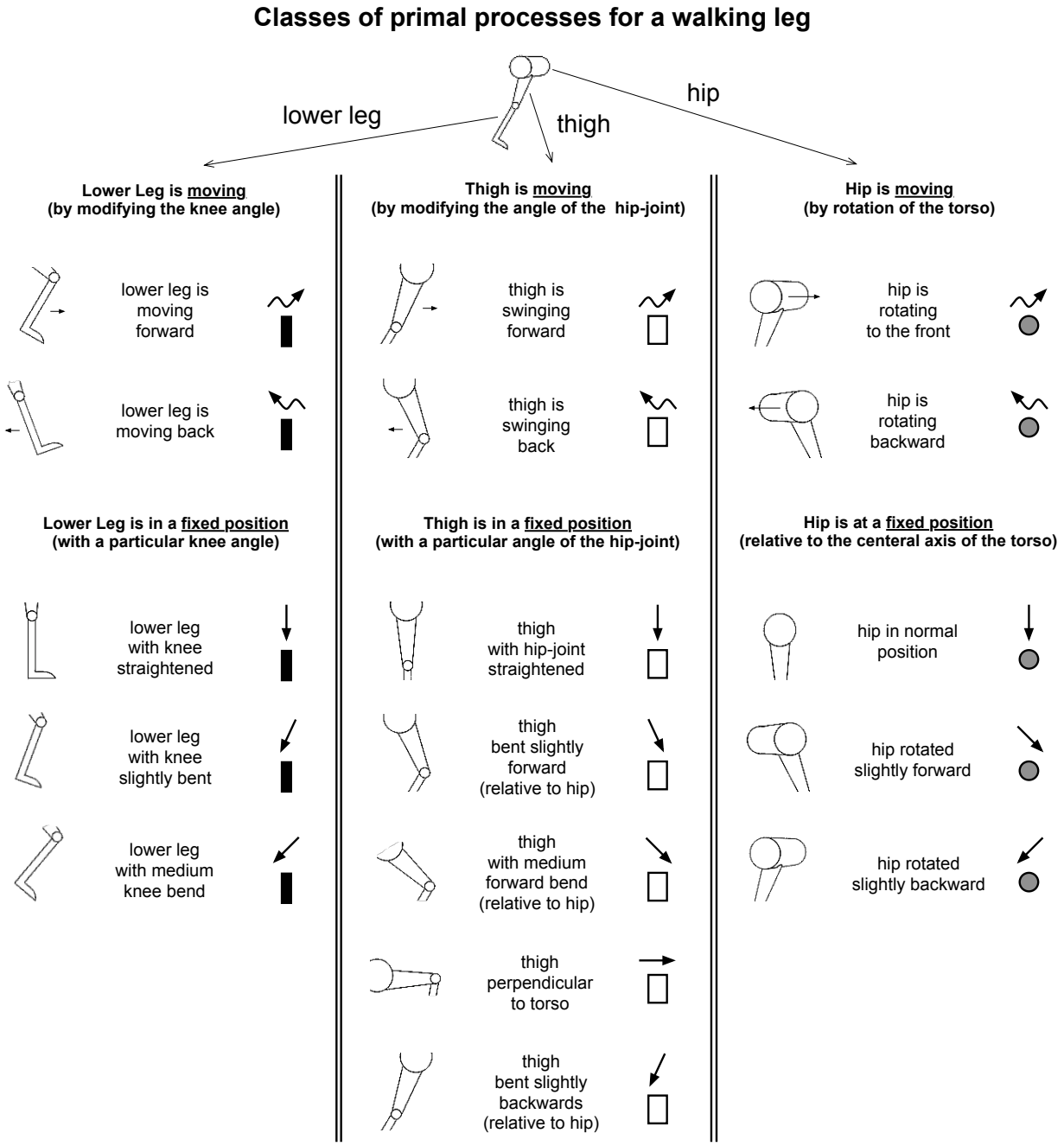

Figure 1: Pictorial representation of seventeen primal classes used in our illustrative example. On the right of *each* of the three main columns are the symbols used to denote each class of primal process, where the *arrows are an integral part of this notation*. The straight arrows indicate the approximate *fixed* position of the corresponding limb part, while a curved arrow indicates the limb part *undergoing* the appropriate movement process. (Note that for simplicity a hip is depicted as a cylinder and its position is exaggerated).

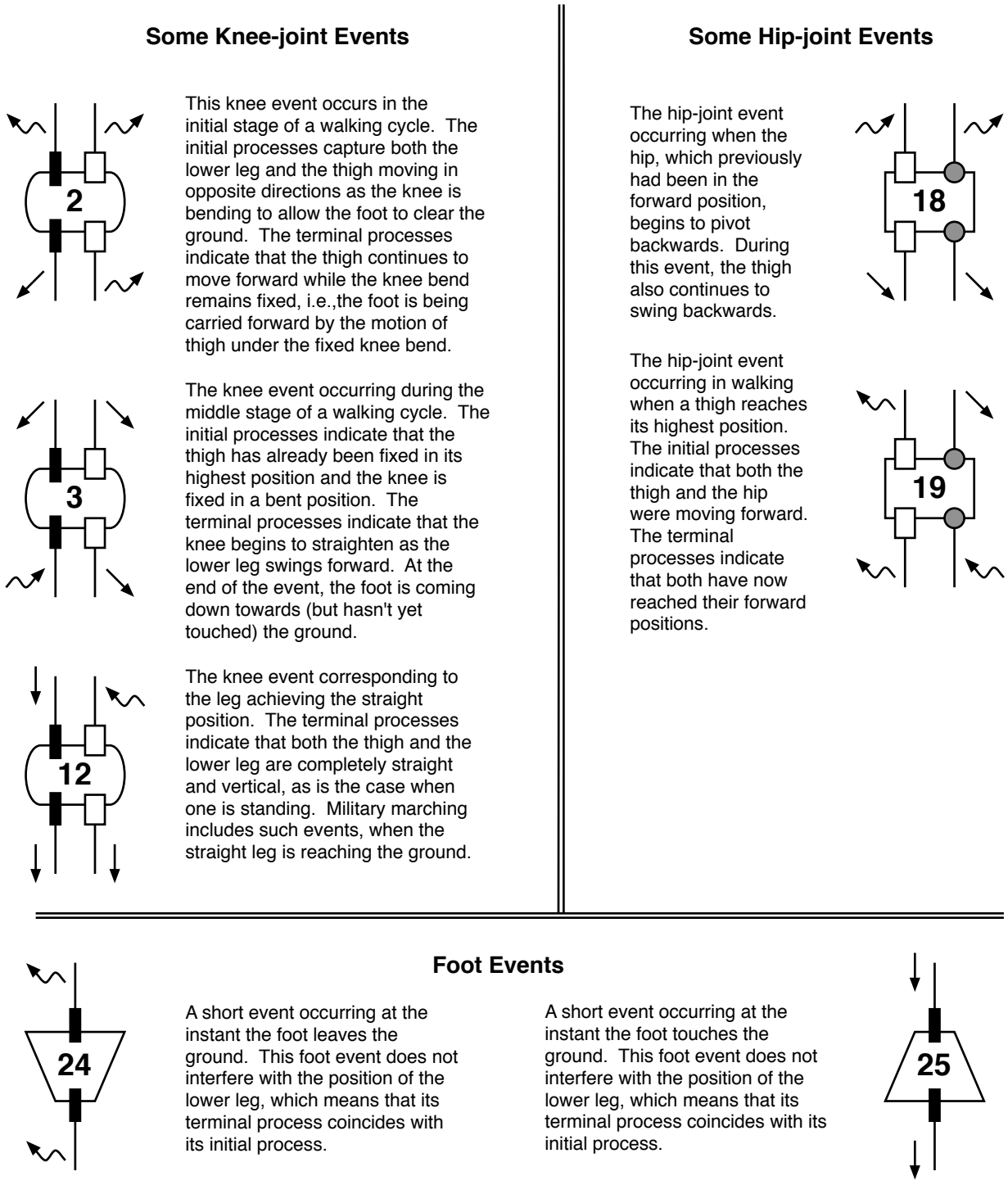

Figure 2: Seven selected abstract primitives (out of 25) used in our illustrative example. We chose the primitives with the rounded corners to denote events relating the lower leg and the thigh, and the square ones to denote events events relating the thigh and the hip. Note that for the lower two primitives are we have chosen still different shapes because they involve just lower leg (i.e., the foot) and the ground. To simplify the reading of the primitives, each abstract primitive is given a numeric identifier instead of the usual subscripted π. For a leg movement involving several of these primitives, see Fig. 3.

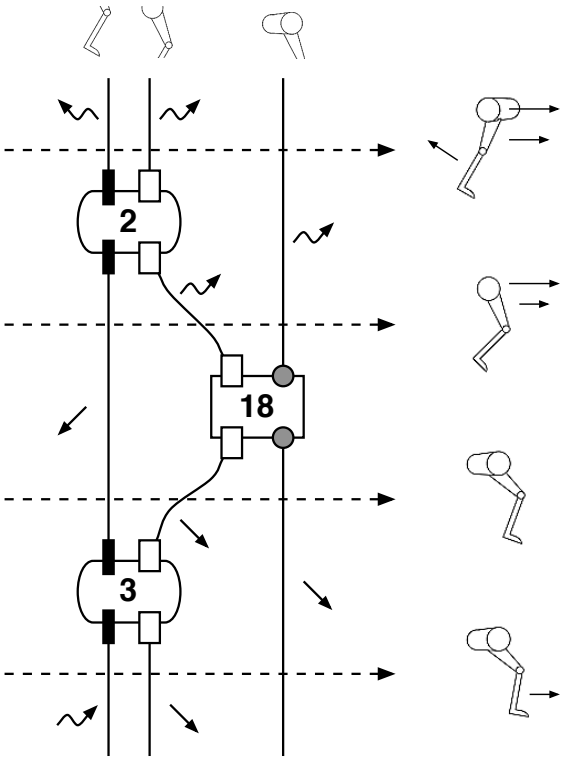

Figure 3: An *abstract* struct representing a segment of the forward movement of an entire leg (involved in a common walk). We added dashed arrows each pointing to a snapshot of the leg undergoing the movement process between two consecutive transformations. The latter movement process consists of the three movement processes, each of them corresponding to a constituent limb part.

For example, during some, usually short, period of time, a hip can move forward, backward, or stay still in one of the three indicated positions (see the last three primal processes in Fig. 1). It is important to keep in mind that we do not treat the hip *itself* as a primal process, but rather the *moving hip* becomes a primal process; a "fixed hip" is treated as a special case of a "moving hip" with the hip remaining relatively stationary. In other words, the adjective "fixed" should be read as meaning that the corresponding body part is briefly maintaining relative fixity between two consecutive movement processes. In that sense, the "regularity" of a primal process means that it simply maintains its form/pattern of activity until an appropriate event affects it.

Thus, the following point should also be kept in mind: various "fixed" positions of a limb part are always relative to another limb part. For example, the "lower leg with medium knee bend" process is observed whenever the knee is bent at the appropriate angle, independent of the position of the thigh. We thought that such a hypothesis is not too inconsistent with the organization of the nervous system as well as with the visual perception of movement by an external observer.

Not having adequate physiological or motion capture expertise, we attempted to decompose leg movement into "natural" unit processes in which the homogeneity of movement is

maintained both from physiological (internal) and the visual perception (external) points of view. Our primal processes are chosen with these considerations in mind. A more careful analysis of motion capture data would be desirable to make more informed decisions about the exact positions and therefore events that need to be captured.

Moving on to primitives and keeping in mind that a primitive is a transformative, or restructuring, event, we note that our primitives are the events that alter the regular activity of (typically) two joint-adjacent leg parts (see Fig. 2, where several selected primitives are shown). We want to draw attention to the following *implicit* feature of most primitives. To allow recovery of leg structure without complicating the structure of the primitives, the limb part connectivity is silently embedded into each of them. If necessary (as might be the case with a robot's detachable limbs), the limb part connectivity could be modelled more explicitly, with more complex primitives. In other words, 23 (of 25) of our primitives each implicitly carry additional information about the connectivity relationship between the *two* primal processes participating in each such primitive event, i.e., the the corresponding two limb parts are connected via a shared joint, which also explains the particular structure of these 23 primitives involving two initial processes and two terminal processes. This also means that if necessary the connectivity structure of an entire leg can be recovered from a corresponding struct of any reasonable length.

Obviously, the chosen primitives are leg-independent, i.e., they are used in representing the movement of either leg, left or right.

2.3 On sensors for leg movement

First, we must mention that, at this stage of our research, we have not chosen any particular sensor system, and so in this very brief section we simply want to mention several possibilities. Second, since our illustrative example does not require any sophisticated sensors, many commercially available motion capture technologies could be adapted for this purpose: e.g., the Moven motion capture suit by xsens [3] or the Gypsy motion capture suit by MetaMotion [4], to mention just two of them. Alternatively, if one is interested in replicating human motion in a robot, the relevant sensors could be implemented in a much larger variety of ways, since these sensors can now be embedded *inside* the relevant (artificial) joints and body parts. Moreover, in the case of a robot, monitoring its structural integrity, as was suggested above, becomes more important and can easily be implemented via a few additional sensors.

Given that the current motion capture technology is geared towards *numeric* recordings of body positions over a period of time, when relying on them, one would need to add several *very simple* software modules to convert their output data to the above ETS events. For example, in order to be able to record primitive π_3 (see Fig. 2), one needs to be able to capture the following moment: the lower leg and the thigh are in the relevant *fixed* positions and (only) the lower leg *begins* to move forward.

Finally, note that to capture the foot reaching or lifting off the ground, one may need—depending on whether the adapted motion capture technology allows for this or not—extra, pressure-based, sensors to account for the above two primitives (see prims π_{24} and π_{25} in Fig. 2).

3 Representing leg movement by a struct

In Figure 4 we show three structs each representing one full cycle of movement for a single leg involved in a common walk, a lazy walk (small steps), and a military march (with the lifted leg completely straightened), see also Fig. 5. The data was not obtained via sensors but is synthetic, i.e., recorded directly based on our visual observations. In general, events were recorded in the "natural" order, but in some cases when the events are close to each other temporally, the order was chosen by the rule that events associated with upper limb parts precede those for lower ones. The latter was necessitated by the lack of real sensors which would have determined the temporally order more accurately. However, in general, given satisfactory sensors, if it turns out that two *attached* primitives are in temporal conflict, it means that one has to go back and redesign at least those primitives.

It is important to keep in mind that the time segments between primitive events are not necessarily of equal lengths and the lengths of the corresponding lines in all figures do not relate to each-other in any simple manner. Thus, as always, such a drawing should be interpreted in a structural manner only, i.e., as a structural pattern/sequence of events. Moreover, although the above figure depicts the movement of a single leg, one should assume the other leg is engaged in an identical movement cycle.

As was discussed in Section 2.1, when real (rather than synthetic) data is being collected, the identity of concrete primal processes is established via the corresponding sensors.

At this point, given the simplicity of our setting and relative structural uniformity of our primitives, the following representational question might be asked: Would conventional strings over a 25-letter alphabet do the same representational job? First of all, one should note that with ETS primitives, syntax and semantics are merged and are embodied by the primitives themselves. Since a letter (from a finite alphabet) does not and cannot carry such information, because it lacks any structure, one is forced to introduce all kinds of production rules, including context sensitive ones, to address this underlying representational deficiency by capturing with such rules the corresponding syntax (and semantics). However, given an alphabet of a substantial size (as is already the case with our *simple example*), this undertaking becomes highly involved and unwieldy. For example, take primitive #6. Why is it difficult to model it by a letter? First, as a necessary (but not sufficient) condition, one would need to capture the fact that #6 should typically have among its predecessors one or two of {#1, #5, #6, #13, #16, #20, #24}, while among its successors, {#1, #6, #17}. At the same time, having primitive #1, for example, among predecessors of #6 is not sufficient, since the presence of a primitive #2, for example, between #1 and #6 may "negate" the presence of #1 and hence may negate the above sufficiency condition. (In fact, one can already see the non-linearity of the proposed representation from the structs in Figure 4, where some primal processes realize "non-linear" connections between primitives, e.g., the primal process connecting primitives #17 and #21 in the middle struct of Figure 4.) Of course, the above difficulties are further compounded when representing the movement of both legs (see Fig 11), where there are several temporally "independent" primitives.

Moving on to classes, a simple perusing of Figure 4 reveals a definite *similarity* among the three structs depicted there, which leads us into our next section on classes.

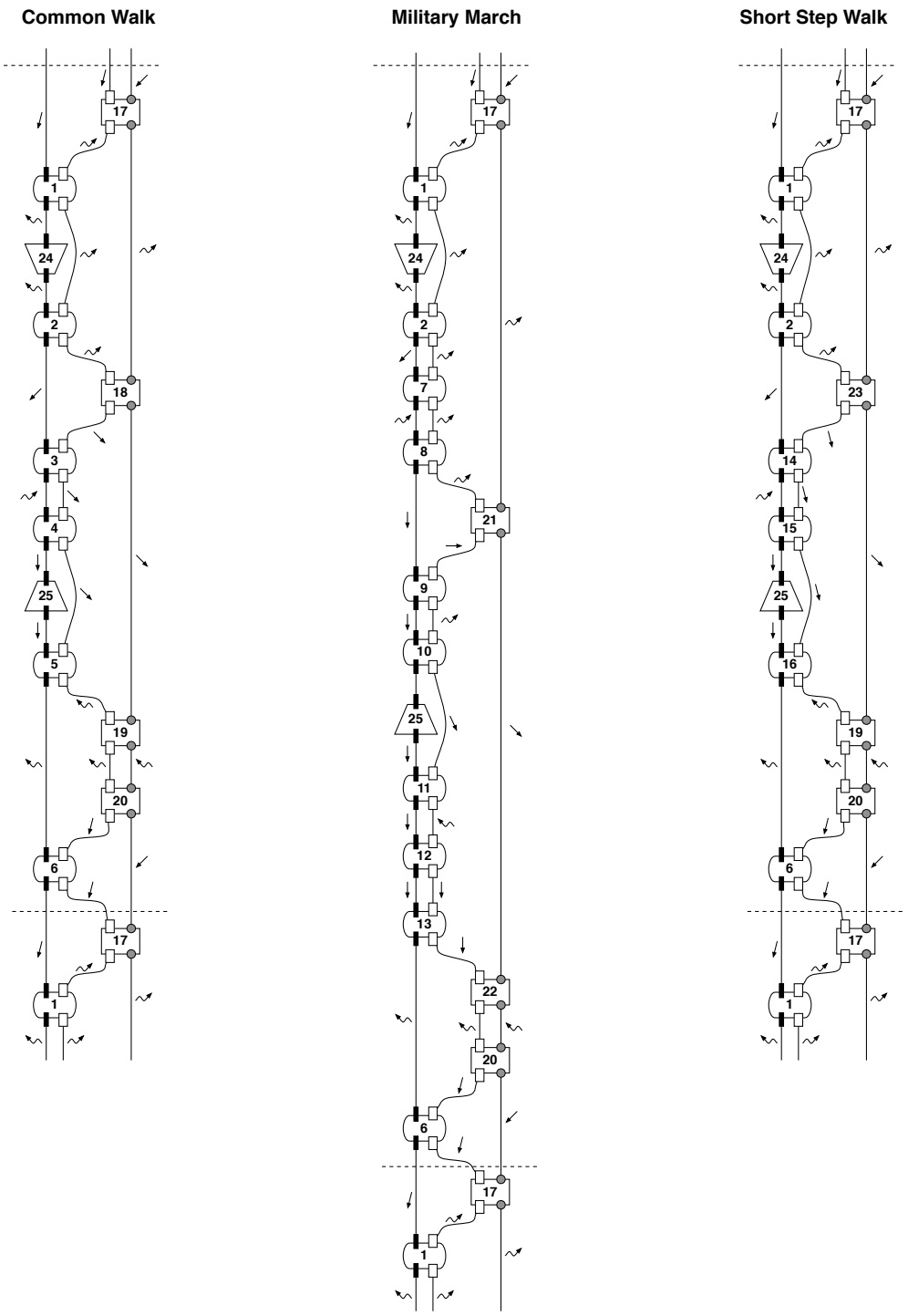

Figure 4: Three *abstract* structs each representing one full *cycle* of movement for a single leg involved in a common walk, a short step walk, and a military march. Note that the struct in Fig. 3 is a substruct of the common walk struct. See also Fig. 5.

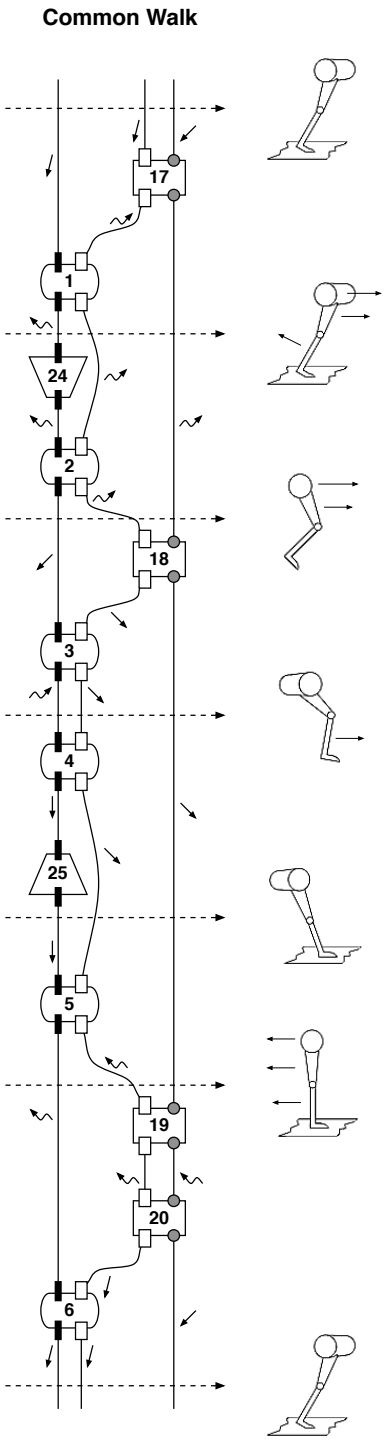

Figure 5: An abstract struct representing one full cycle of movement for a single leg involved in a common walk, with additions of snapshots of the leg similar to those in Fig. 3

4 Class of leg movement in walking

We thought it quite natural to view a particular kind of a *single leg's movement*—involved in (upright) human forward locomotion via *cyclical* and *identical* movement of both legs—as a single class called "single leg movement in walking". Such class of single leg movements could also be justified on the basis of evolutionary considerations: forms of movement from the same class serve a common evolutionary locomotive purpose. Marching could also be placed in the class we have chosen, even though it serves an additional, exhibitionary, purpose, since it is attained as a relatively simple enhancement of the other two forms of walking given in Figure 4. Moreover, there is no reason to assume that the chosen class is restricted to just those three forms of walking.

Simple analysis of the three structs in Figure 4 representing three elements of the class of single leg movements suggests that this structure of this class is based on the following constituent classes of movement.

4.1 Constituent movement classes

Each of the Figures 6, 7, and 8, gives a *pictorial* specification/description of a single-level class generating system for a *constituent class* participating in the formation of a two-level class representation for the above class of single leg movement in walking.

Briefly, without having adequate physiological expertise, we attempted to decompose single leg movement into "natural" sub-movements, and it turned out that there are three of them, $\mathfrak{C}_1$, $\mathfrak{C}_2$, $\mathfrak{C}_3$, which were arrived at by inspecting the structs in Figure 4 and relating them to the actual walking process suggested this is based on their role in walking.

We decided not to give a precise specification of constraints to allow for a possible expansion or modification of the classes involved based either on the physiological considerations or further observational evidence (i.e., of new forms of "walking" from the chosen class).

4.2 Our main class of movement

In Figure 9, we present a pictorial specification/description of a two-level class generating system for the class $\mathfrak{C}^1$, "single leg movement in walking", based on the above three constituent classes, $\mathfrak{C}_1$, $\mathfrak{C}_2$, $\mathfrak{C}_3$.

The continuation of the caption to Figure 9: the actions of the class generating system might then look as follows: 1) choose some *level 1 struct* consisting of a single element of class $\mathfrak{C}_1$ satisfying the *level 1 constraint* consisting of a single class element link A and applicable to the working struct, and assemble it with the working struct 2) choose some level 1 struct consisting of a single element of class $\mathfrak{C}_2$ satisfying the level 1 constraint consisting of class element links *C, E, F* and applicable to the working struct, and assemble it with the working struct 3) choose some level 1 struct consisting of a single element of class $\mathfrak{C}_3$ satisfying the level 1 constraint consisting of class element links from *J, K, L* and applicable to the working struct, and assemble it with the working struct; go to 1).

It is important to note that there isn't much room for variety inside each constituent class, since, for example, permuting the order of the constraints would lead to non-sensible

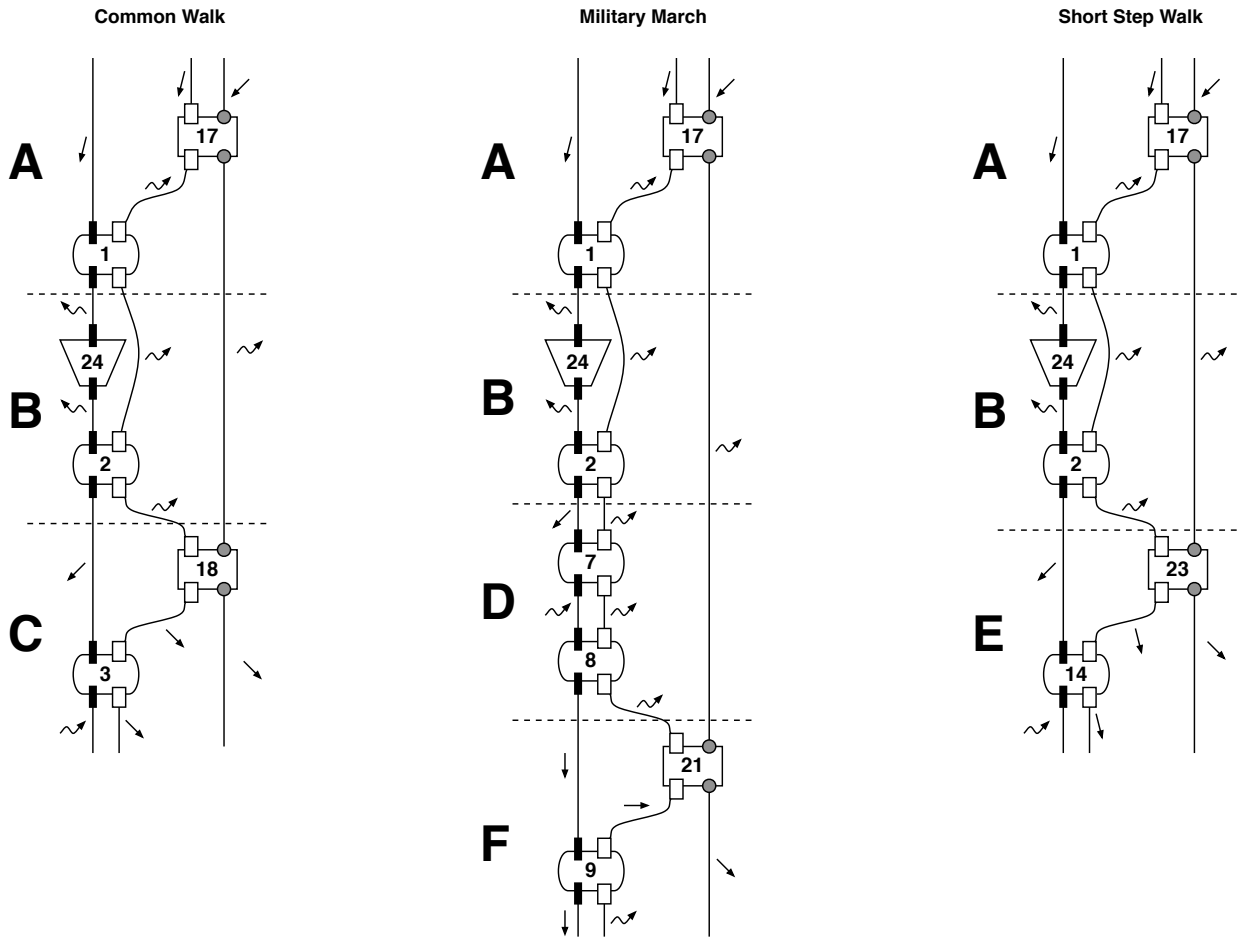

Figure 6: A pictorial description of a single-level class generating system for the class $\mathfrak{C}_1$, "step commencement (by the back leg)". Although the precise constraints are not specified, each letter denotes a constraint and is *located in the place where the struct satisfying this constraint would be added* during the generating process (in the thee chosen examples, all structs satisfying the same constraint happen to be structurally identical). The actions of the class generating system might then look as follows: 1) choose some struct satisfying constraint A and applicable to the working struct and add it to the working struct 2) choose some struct satisfying constraint B and applicable to the working struct and add it to the working struct 3) choose some struct satisfying one of the constraints from {C, D, E} and applicable to the working struct and add it to the working struct 4) if it exists, choose some struct satisfying constraint F and applicable to the working struct and add it to the working struct.

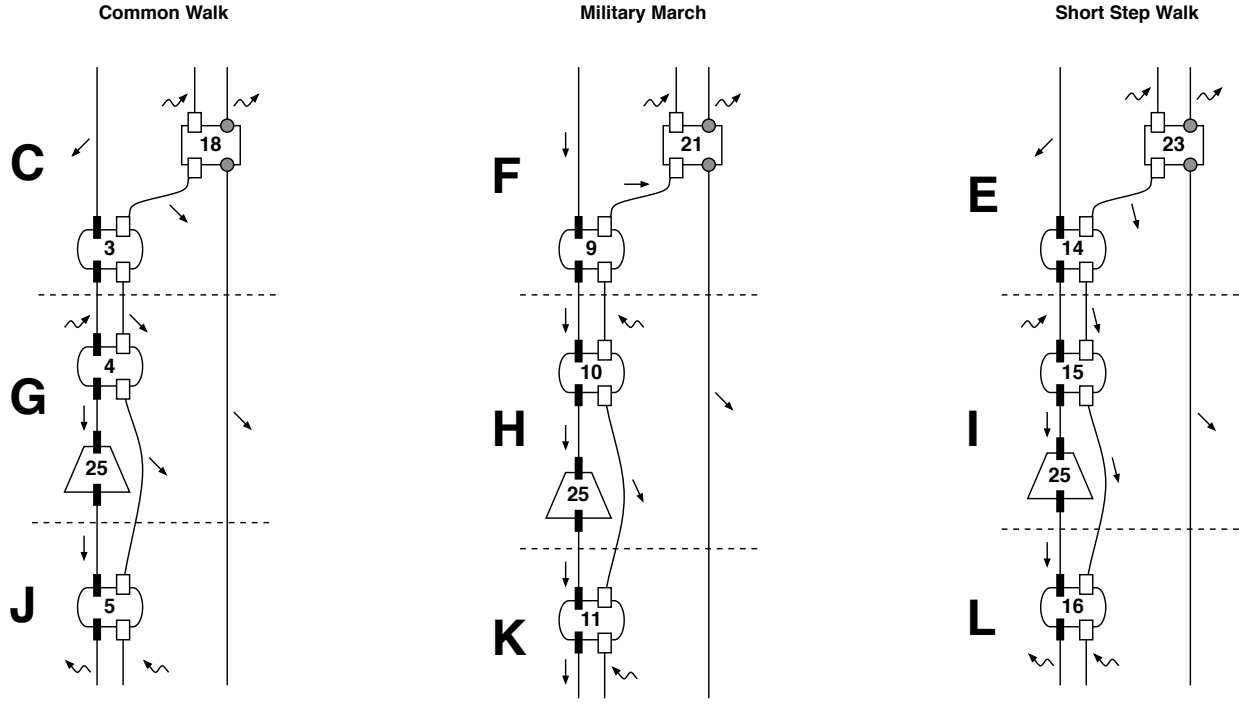

Figure 7: A pictorial description of a single-level class generating system for the class $\mathfrak{C}_2$, "foot reaching the ground". Although the precise constraints are not specified, each letter denotes a constraint and is *located in the place where the struct satisfying this constraint would be added* during the generating process (in the three chosen examples, all structs satisfying the same constraint happen to be structurally identical). The actions of the class generating system might then look as follows: 1) choose some struct satisfying one of the constraints from {C, D, E} and applicable to the working struct and add it to the working struct 2) choose some struct satisfying one of the constraints from {G, H, I} and applicable to the working struct and add it to the working struct 3) choose some struct satisfying one of the constraints from {J, K, L} and applicable to the working struct and add it to the working struct.

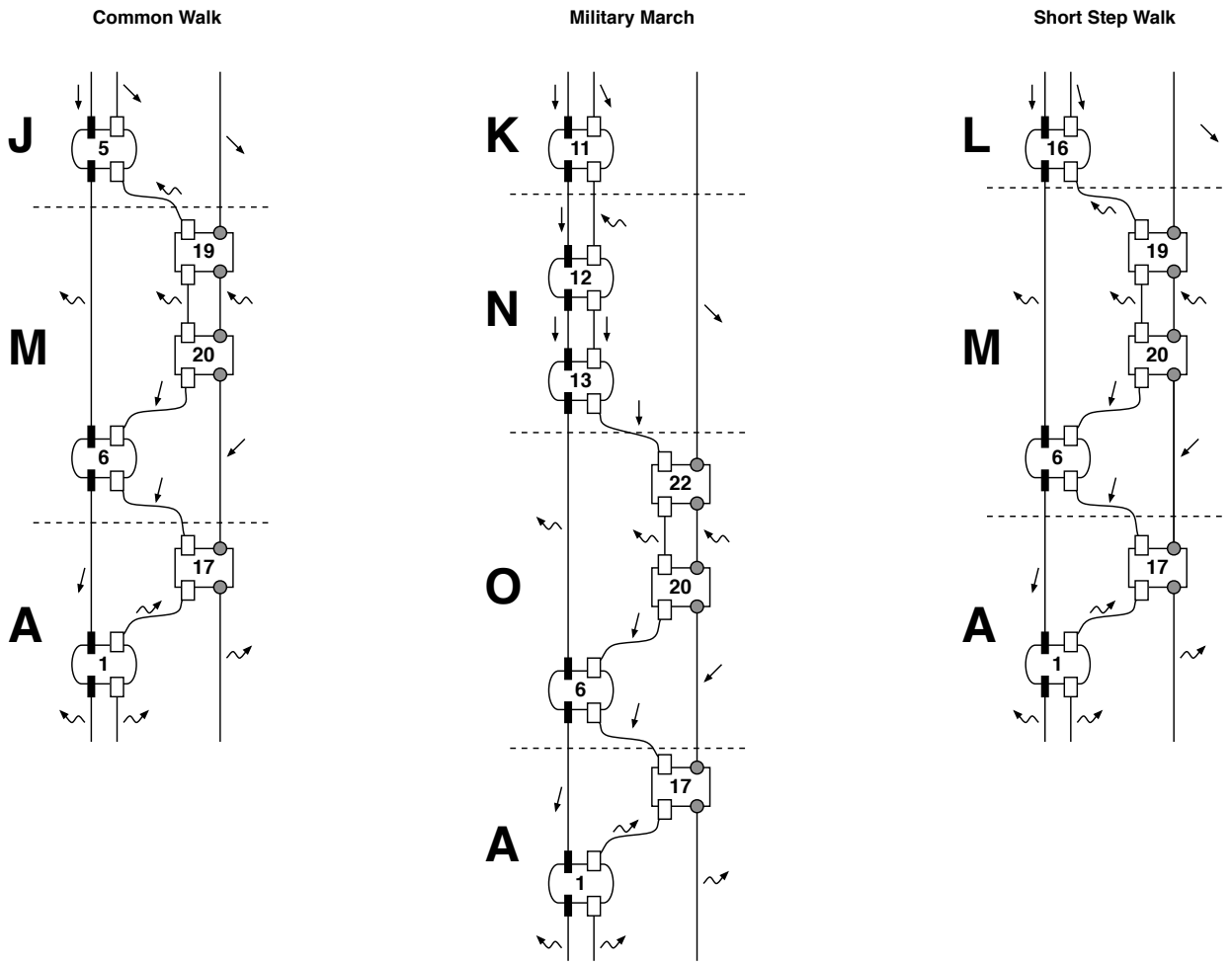

Figure 8: A pictorial description of a single-level class generating system for the class $\mathfrak{C}_3$, "leg supporting the shifting upper body (while the other leg is making a step)". Although the precise constraints are not specified, each letter denotes a constraint and is *located in the place where the struct satisfying this constraint would be added* during the generating process (in the three chosen examples, all structs satisfying the same constraint happen to be structurally identical). The actions of the class generating system might then look as follows: 1) choose some struct satisfying one of the constraints from {J, K, L} and applicable to the working struct and add it to the working struct 2) if it exists, choose some struct satisfying constraint N and applicable to the working struct and add it to the working struct 3) choose some struct satisfying one of the constraints from {M, O} and applicable to the working struct and add it to the working struct 4) choose some struct satisfying constraint A and applicable to the working struct and add it to the working struct.

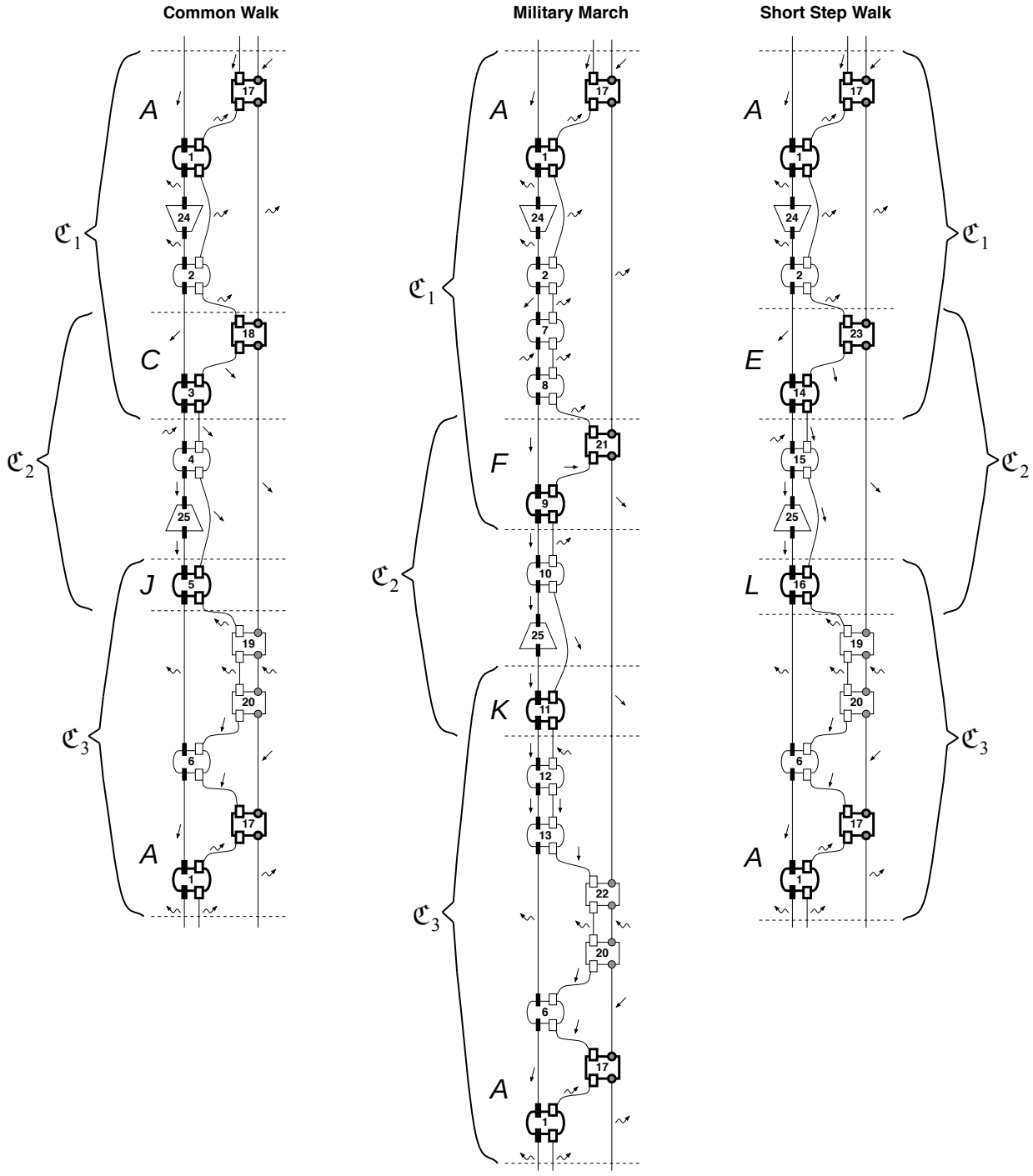

Figure 9: A pictorial description of a two-level class generating system for class $\mathfrak{C}^1$, "single leg movement in walking". Each $\mathfrak{C}_i$ denotes a constituent class and is *located in the place where the corresponding constituent class element would be added* during the generating process. Although the precise level 1 constraints are not specified, each non-subscripted letter now denotes a level one class element link—associated with the corresponding constraint in Figs. 6, 7, and 8—and is located in the place where two appropriate constituent class elements overlap (the overlap primitives are shown in bold). The actions of the class generating system are described in Section 4.2.

movements or put the movement outside the class we seek to capture. More degrees of freedom can be observed, however, at the next level class, when the constituent class elements are composed to form next-level class elements (see Figs. 9 and 10).

4.3 Some comments on our main movement class

First, we want to present two more structs: one of them is a representation of a single leg movement involved in a"lazy walk" (it belongs to our main class and *was constructed relying on the above class representation*), and the other is a representation of a single leg movement involved in a "forward jump" (it does not belong to the main class), see Fig. 10.

A lazy walk can be described as a short step walk with a slight pause between the steps, while the body weight shifts more prominently from one side to another during each step. During such a walk, the supporting leg has to completely straighten. We draw your attention to the fact that, since such a leg movement is an element of our main class, "single leg movement in walking", it can be generated by the corresponding class generating system (see Fig. 9).

Second, we thought it might be useful to give at least one example of a single leg movement that doesn't belong to our main class. The last struct in Figure 10 represents the movement of a single leg involved in forward locomotion by means of jumping forward (simultaneously with both legs). We note that among all presented examples of movement, this is the only one in which the movements of both legs temporally coincide. It is easy to see that the corresponding struct cannot be generated by the main class generating system.

The main class we considered in this section is "single leg movement in walking" and not the corresponding class of movement involving the entire body. Figure 11 suggests that having learned our main class, the job of learning the class of movement involving both legs becomes much easier, independent of several possible approaches to learning this more complex class. In particular, one approach to learning this more complex class of movement may rely on our main class as one of its constituent classes.

5 Conclusion

In some other application domains, for example vision and speech, although the design of primal processes and primitives might be more involved, the overall ETS approach to applications as illustrated in this paper should remain relatively intact at least for the near term (see also [5]). Thus, in contrast to the existing approaches to pattern recognition, machine learning, and data mining, the ETS formalism clarifies both the nature of the classification problem and the context for its pervasiveness.

From the simple example presented, it should become clear that in ETS the concept of "formative history" acquires a meaning much broader than that associated with its usual connotation: as an object's immediate "real/physical" formative history.

Finally, as we have done throughout the paper, we emphasize an important new applied capability that becomes available with the adoption of ETS: genuine modularity and portability/reusability of the already learned class representations. This should make the development of intelligent systems into a much more manageable and reliable undertaking

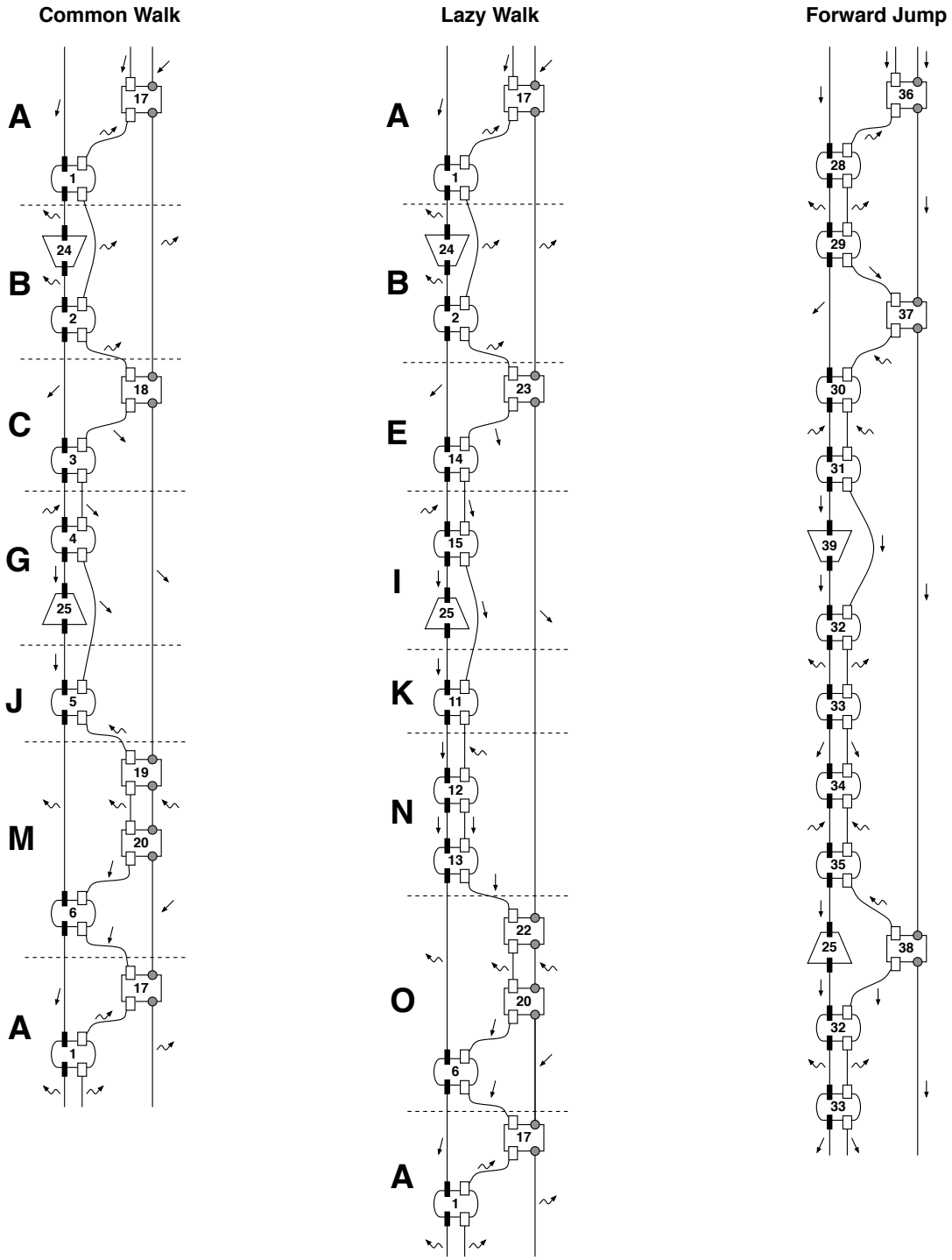

Figure 10: Three abstract structs representing movement of a single leg in a common walk, a lazy walk, and a forward jump, where the first two are supplied with the above constraints since they belong to the same class of "single leg movements in walking". The lazy walk struct is a representation of a "new" class element (from the latter class), *which was constructed following the class representation* sketched in Fig. 9. Note that the last struct required new primitives not mentioned in the text, but their interpretation should be clear since the same primal classes and notation scheme are used.

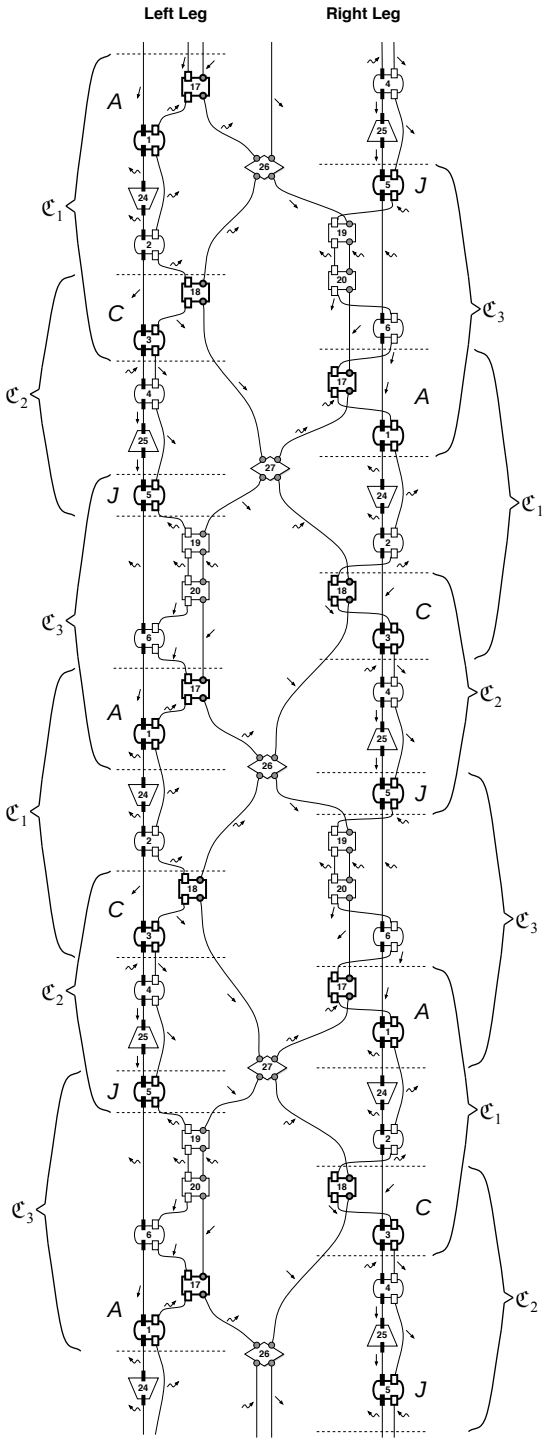

Figure 11: An abstract struct representing movement of both legs involved in a regular walk. Notations are those given in Fig. 9, except for two new primitives (26 and 27) in the shape of a rhombus that relate the left and right hips.

than is presently the case, since gradual expansion of the learning environment becomes a systematic affair (as opposed to starting completely anew with every expansion).

Acknowledgments. We thank Reuben Peter-Paul for his helpful ideas and assistance with the figures, and also Alex Gutkin and Oleg Golubitsky for their comments.

References

[1] L. Goldfarb, D. Gay, O. Golubitsky, and D. Korkin, "What is a structural representation? A proposal for an event-based representational formalism", 2006, in *What is a Structural Representation*, L. Goldfarb, Ed., in preparation.
Available: http://www.cs.unb.ca/~goldfarb/ETSbook/ETS5.pdf

[2] A. A. Pack, L. M. Herman, Dolphins can immediately recognize complex shapes across the senses of echolocation and vision, *The Journal of the Acoustical Society of America* – October 1996 – Volume 100, Issue 4, p. 2610

[3] Xsens Motion Technologies, Moven Mocap Suit, http://www.xsens.com/moven, 2006.

[4] MetaMotion, Gypsy Motion Capture System, http://www.metamotion.com/gypsy/gypsy-motion-capture-system-mocap.htm, 2004.

[5] A. Gutkin, Some thoughts on a formal approach to speech perception in light of the ETS formalism, 2006, in *What is a Structural Representation*, L. Goldfarb, Ed., in preparation.