Making Sense of the Sensory Data – Coordinate Systems by Hierarchical Decomposition

Attila Egri-Nagy and Chrystopher L. Nehaniv

BioComputation Research Group
School of Computer Science
University of Hertfordshire
College Lane, Hatfield, Hertfordshire AL10 9AB, United Kingdom
{A.Egri-Nagy, C.L.Nehaniv}@herts.ac.uk

Abstract. Having the right sensory channels is an important ingredient for building an autonomous agent, but we still have the problem of making sense of the sensory data for the agent. This is the basic problem of artificial intelligence. Here we propose an algebraic method for generating abstract coordinate system representations of the environment based on the agent's actions. These internal representations can be refined and regenerated during the lifespan of the agent.

1 Introduction

According to the so-called Good Old-Fashioned Artificial Intelligence approach we have to build a system with a reasonably accurate representation of its environment to make it behave intelligently. But this just does not work. The hard-wired model is rigid, even cannot cope with small changes of the environment, or it the representation should contain all details with all the possible changes, thus combinatorial explosions pop up. Moreover, the basic assumption itself that we have complete knowledge about the environment beforehand can hardly be defended. Therefore the Artificial Intelligent (AI) community has come up with the counterintuitive idea that we do it better without any representation [1]. Clearly, this is a fruitful method showing that one can have complex behavior without complex inner structure. But it is also clear that we cannot get too far without representations [2].

Here we adopt the viewpoint that we often need representations of the environment in order to realize artificial intelligence, but the representation should be flexible and dynamically changing over time and obtained by the artificial system on its own by recognizing regularities of the real world around. We propose an algebraic method for generating abstract coordinate system representations of the environment based on the agent's actions. Sampling the transitions through the sensory channels after the actions of the agent allows us to build a finite state automaton description, from which we can generate abstract coordinate systems using the algebraic hierarchical decomposition of finite state automata.

The general ideas of applying automata decompositions as formal models of understanding were proposed several times [3, 4], but now they are closer to fulfilment. The mathematical theory behind this is the algebraic hierarchical decomposition of finite state automata, the so-called Krohn-Rhodes Theory. For forty years there was no computational implementation for the hierarchical decomposition of automata. However, in the electronic circuit industry there are many different decomposition methods and implementations, but they are not hierarchical since there are several physical constraints on circuit design and the cascaded composition appears not to be the most efficient in terms of power consumption, area and delay minimization [5]. Though it may be very appropriate for understanding such systems [3]. Recently the authors have implemented two methods for the holonomy decompositions [6–8].

2 Hierarchical Decomposition: The Krohn-Rhodes Theory

Here we present the very basic underlying ideas of algebraic hierarchical decomposition of finite state automata. We use the minimum amount of mathematical notation here. For precise definitions see [9, 4].

2.1 Reversible and Irreversible Processes

Roughly speaking, we have two different kinds of computational operations: reversible and irreversible ones. For instance, if we move some content of the memory to another empty location, that is reversible, since we can move it back. But if we overwrite a nonempty part of the memory, then it is irreversible, since there is no way to restore the previously stored data. Closer to a formal definition we can say that irreversible processes reduce the size of the set of possible future states, while reversible ones do not. A map $f : A \to A$ of a set A is called a permutation (reversible) if it is a bijection, otherwise it does collapse elements ($a \in A$ is an image of more than one element), therefore it is irreversible.

Algebraically the distinction is more immediate. A *permutation group* is a set G of bijective mappings together with the *state set* A on which the mappings act. A *transformation semigroup* (A, S) has a similar structure, but S consists of general functions, not only bijective maps. Roughly speaking we consider finite automata as transformation semigroups. The elements of the semigroup are the transformations of the state set induced by the input symbols. This way the problems in automata theory are transfered into the algebraic domain.

2.2 The Prime Decomposition Metaphor

For explaining the Krohn-Rhodes Theory, the best way is to present it by a metaphor. Basically we do the same as the prime decomposition for integers, but instead of numbers we do it for more complicated structures, namely finite state automata (considered as transformation semigroups). The similarities can be summarized the following way:

	Integers	**Automata**
Factors	Primes	Flip-flop Automaton Permutation Automata
Composition	Multiplication	Wreath Product
Precision	Equality	Division, Emulation

The basic building blocks are the simple[1] permutation groups (for the reversible computation) and only one component for the irreversible computation, the so-called flip-flop automaton. It is like a one-bit sized memory.

The way of putting together the components, the so-called *cascaded* or *wreath* product, is hierarchical and no feedback is allowed from deeper levels to upper levels (see Fig. 1). The reason, why we choose this special way of composition, is that the following special properties of hierarchy render the composed structure more comprehensible.

– Information flow between levels is restricted enabling modularity (also within one level with parallel components).
– Generalization and specialization are natural operations realized by taking subsets of levels in either direction up or down the hierarchy

Note that we allow parallel components on one hierarchical level.

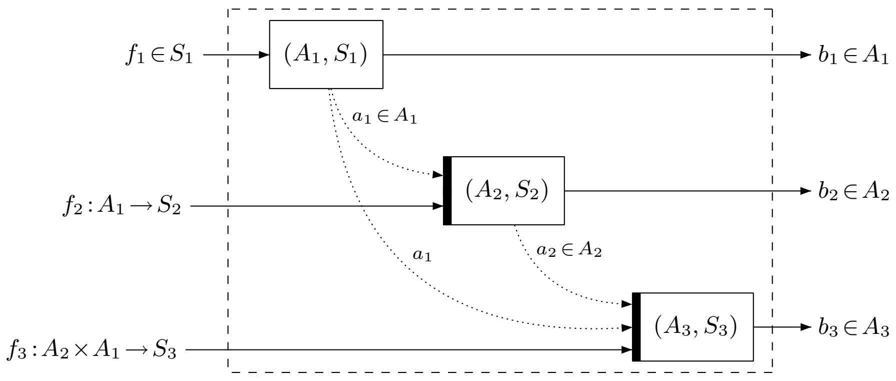

Fig. 1. State transition in the wreath product $(A_3, S_3) \wr (A_2, S_2) \wr (A_1, S_1)$. An input determines a transformation (f_3, f_2, f_1) which maps the state (a_3, a_2, a_1) yielding the new state $(b_3, b_2, b_1) = (a_3 \cdot f_3(a_2, a_1), a_2 \cdot f_2(a_1), a_1 \cdot f_1)$. The black bars denote the applications of functions f_2, f_3 according to hierarchical dependence. Note that the applications of these functions happen exactly at the same moment since their arguments are the previous states of other components, therefore there is no need to wait for the other components to calculate the new states. We use the state as the output of the automaton. It is often misunderstood to have some time delay during the state transition. This is not the case, the state transition in the wreath product is instantaneous.

[1] This has a well defined meaning in group theory.

2.3 Coordinates, Hierarchical Dependence

Hierarchical decompositions provide a coordinate system for the original phenomenon described as an automaton. For each coordinate position we have transformation semigroup components and their state sets are the possible values for that position. Due to its hierarchical nature the order of the coordinates does matter. What happens on deeper levels is determined by the states of the levels above. The simplest example to describe *hierarchical dependence* is a bidirectional counter. Imagine a device which keeps track how many times you press a button, and you have two other buttons set the operating mode. You start from zero in adding mode then as a check whether the resulting number is the correct value, you switch to subtracting mode and count again, but this time downwards, until you reach zero again. For instance to count the number of passengers on an airplane while walking along the aisle. The operation of this device can be represented with the following simple coordinate system: (n, mode), where modes are $+$ and $-$ corresponding to adding and subtracting. The mode coordinate is the top level of the hierarchy. There are three operations: counting c, switching to adding mode m_+, and switching to subtracting mode m_-. For instance

$$(9,+) \cdot c = (10,+)$$

$$(9,+) \cdot m_- = (9,-)$$
$$(9,+) \cdot m_+ = (9,+)$$
$$(9,-) \cdot c = (8,-)$$

Hierarchical dependence: the counting operation does different things depending on the top level coordinate.

2.4 Computational Implementations

Now we have available implementations for Krohn-Rhodes Theory [7] and we can start exploring the vast space of the decomposition of computational structures including actions and sensory activity of agents. However before applying the method to large-scale problems we need to solve some scalability issues. Currently we are working on a new incremental version of the algorithm, which starts at the top level and goes down to decompose further levels when they are feasible. This way we get some information about the hierarchical structure immediately, instead of trying to calculate the first phase of the whole decomposition, which may fail due to combinatorial complexity.

3 Building Coordinate Systems from Sensory Data Based On Actions

For acting meaningfully in a complex environment an agent may need a representation, a model of that environment. The model is used to predict the outcome of certain actions. But where does this representation come from? The widely accepted answers are evolution or learning, since having a predefined and fixed representation often exhibits very unintelligent behaviour. If we want to make representation hardwired, we might not have complete knowledge (if we have any at all) about the environment, and also the environment can be changing. Moreover, the most important thing is that the agent needs a model from its viewpoint (not from our viewpoint), i.e. that is appropriate for its 'Umwelt' (cf. [10]). Therefore we can conclude that agent should build its representation of the environment primarily based on the data coming through the sensors.

3.1 Experimenting with the Environment

In order to apply hierarchical decompositions, first we need to build a finite state automata description. The state set of the automaton consists of the possible states of the environment from the viewpoint of the agent, i.e. the perception, the data coming from the sensory channel. The accurate definition of the state depends on the actual hardware setup of the sensors. The input symbols are the actions of the agent. That is why we say that the automaton model is built from the agent's perspective. The agent carries out basic experiments: first it determines the current sensory state, next it carries out an action, then determines the resulting sensory state again. This elementary experiment is recorded as one state transition in the finite state automata being built.

After finite many repeats of the basic experiments, we have a sequence

$$\text{perception}\rightarrow\text{action}\rightarrow\text{perception}\rightarrow\text{action}\rightarrow\text{perception}\rightarrow \\ \ldots\rightarrow\text{action}\rightarrow\text{perception}.$$

This sequence is usually called the perception-action loop, and can be studied by using information theoretical tools [11]. The state transitions define the automaton, and we can do the decomposition.

Let's suppose we have a perception-action automaton and we do not really know what it is doing (though by knowing its generators we fully describe it implicitly), as in Fig. 2. Is it doing some complex computation? Calculating its holonomy decomposition we find that it can be emulated by a cascaded automaton with two levels (for details and visualisation see Figures 3-4). Now if we ask the question, 'What is the automaton doing roughly?', then we can answer very easily just by looking at the top level (Fig. 4). We have three states there and the component is not a reversible one, which means that there are actions of the agent that induce decisive changes in the environment, and those changes cannot be undone. Going further down to the second level we find that depending on the state above we either have a reversible component or another irreversible change. The actual reversible component is a permutation of two states of the original automaton, corresponding to actions that can be repeated. This illustrates the idea of having a coordinate system for understanding.

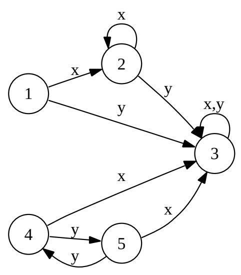

Fig. 2. An example perception-action automaton $\mathcal{A}$ for an agent with two actions x and y showing the transitions these actions induce on sensory states. The actions determine the state transformations $x = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 2 & 3 & 3 & 3 \end{smallmatrix}\right)$, $y = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 3 & 3 & 5 & 4 \end{smallmatrix}\right)$.

3.2 Integrating Sensory Channels

Depending on the granularity level on which we define states, we can either have composite states by somehow integrating sensory channels, or we can build the hierarchical model for each channel. The latter seems to be more interesting since we have different models for different modalities and we have the possibility of comparing the different coordinate systems.

3.3 Hidden States

It may be very well possible that starting from a state a the same action x applied many times yields different results. That is the indication that the sensor does not capture some important aspects of the process in the environment. There are some "hidden states", that the agent cannot see. This clearly complicates the finite state automata description by making it nondeterministic[2] and may be converted into a deterministic one (with the cost of introducing more states, for instance the states that are not detected by the sensors). As an extreme case we can consider an agent only with one action, the observation. For this problem of hidden states we have more sophisticated approaches, like the ϵ-machine reconstruction [12], where the state transitions of the automaton are based on histories, not just on single states of the environment.

3.4 Stochasticity

Our basic assumptions is that the environment of the agent can be described by finite state automata. It is debatable whether our assumption holds for agent

[2] Nondeterminism in the context of automata theory means only that in a given state the same action may have several outcomes, whereas stochasticity concerns the assignment of probabilities to such transitions.

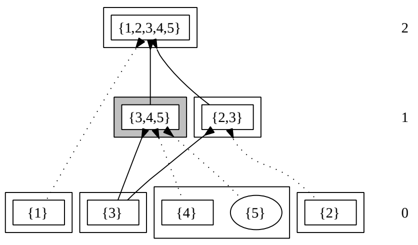

Fig. 3. The structure of the holonomy decomposition of $\mathcal{A}$. The numbers on the right denote the hierarchical levels (the level 0 is present just to show the states of the components on the first level, it does not appear as a hierarchical level in the decomposition). The nodes are subsets of the state set, rectangular nodes represent the components of the decomposition. Shaded components denote the existence of some reversible computation. The arrows going into the component come from the component's states. On the first level we have parallel components.

put into real world situations, or by using discrete non-stochastic models we abstract away important layers of the real processes. However, our approach is justified by the very idea of a model, which should be simpler than the modeled phenomenon.

4 Future Work and Discussion

We presented a framework for using sensory channels to build models of the environment on the fly. We did not mention many difficult issues that are expected to come up for real experiments (e.g. with physically built robots). The difficulties can be the definition of state for each sensory channels, the resolution of time for actions and perceptions, the number of actions needed for building the model, etc., these should be solved by future attempts. The next steps are to automate the construction of perception-action automata and the related ϵ-machines arising from real-world examples. By having a computational implementation we are getting closer to those very promising and possibly successful applications.

References

1. Brooks, R.A.: Cambrian Intelligence: The Early History of the New AI. MIT Press (A Bradford Book) (1999)
2. Steels, L.: Intelligence with representation. Philosophical Transactions: Mathematical, Physical and Engineering Sciences **361**(1811) (2003) 2381–2395
3. Rhodes, J.L.: Applications of Automata Theory and Algebra via the Mathematical Theory of Complexity to Finite-State Physics, Biology, Philosophy, Games, and Codes. World Scientific Press (to appear 2007) Foreword by Morris W. Hirsch, edited by Chrystopher L. Nehaniv (Original version: University of California at Berkeley, Mathematics Library, 1971).

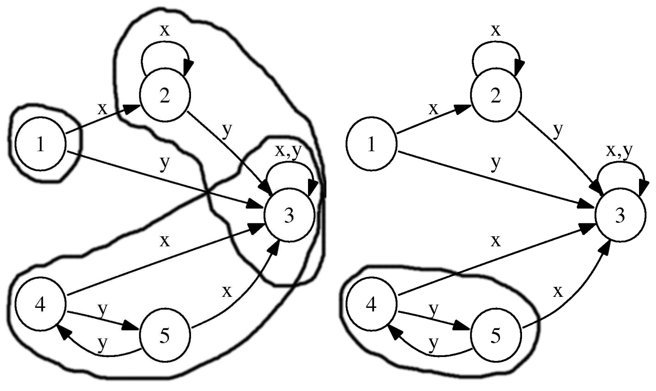

Fig. 4. The states of the top (level 2) component of the decomposition of $\mathcal{A}$ are overlapping subsets of the state set of the transformation semigroup being decomposed (on the left). The identified reversible computation at the second level of the decomposition of $\mathcal{A}$ (on the right).

4. Nehaniv, C.L., Rhodes, J.L.: The evolution and understanding of hierarchical complexity in biology from an algebraic perspective. Artificial Life **6** (2000) 45–67
5. Devadas, S., Newton, A.R.: Decomposition and factorization of sequential finite state machinces. IEEE Transactions on Computer-Aided Design **8**(11) (1989) 1206–1217
6. Egri-Nagy, A., Nehaniv, C.L.: Algebraic hierarchical decomposition of finite state automata: Comparison of implementations for Krohn-Rhodes Theory. Conference on Implementations and Applications of Automata CIAA 2004, Lecture Notes in Computer Science **3317** (2004) 315–316
7. Egri-Nagy, A., Nehaniv, C.L.: GrasperMachine, Computational Semigroup Theory for Formal Models of Understanding. (`http://graspermachine.sf.net`). (2003)
8. Egri-Nagy, A.: Algebraic Hierarchical Decomposition of Finite State Automata – A Computational Approach. PhD thesis, University of Hertfordshire, School of Computer Science, United Kingdom (2005)
9. Krohn, K., Rhodes, J.L., Tilson, B.R.: The prime decomposition theorem of the algebraic theory of machines. In Arbib, M.A., ed.: Algebraic Theory of Machines, Languages, and Semigroups. Academic Press (1968) 81–125
10. von Uexküll, J.: Environment [Umwelt] and inner world of animals. In Burghardt, G.M., ed.: Foundations of Comparative Ethology. Van Nostrand Reinhold, New York (1985) 222–245
11. Klyubin, A.S., Polani, D., Nehaniv, C.L.: Organization of the information flow in the perception-action loop of evolved agents. In Zebulum, R.S., Gwaltney, D., Hornby, G., Keymeulen, D., Lohn, J., Stoica, A., eds.: Proceedings of 2004 NASA/DoD Conference on Evolvable Hardware, IEEE Computer Society (2004) 177–180
12. Crutchfield, J.P.: The calculi of emergence: Computation, dynamics, and induction. Physica D **75** (1994) 11–54