ON THE FORMALIZATION OF THE EVOLVING TRANSFORMATION SYSTEM MODEL

by

Oleg Golubitsky

B.Sc./M.Sc. in Math./Applied Math., Moscow State University, 1999

Ph.D. in Math., Moscow State University, 2003

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

Doctor of Philosophy

in the Faculty of Computer Science

THE UNIVERSITY OF NEW BRUNSWICK

March, 2004

Abstract

The central concept of the Evolving Transformation System (ETS) model is structural object representation constructed by the process of inductive inference. The model was proposed in 1990 by Lev Goldfarb to be applied to any pattern learning or classification problem. A formal exposition of the model is presented in this thesis. It defines the concepts that encapsulate the idea of structural representation and includes lemmas and theorems that link these concepts together into a single model. The chosen form of definitions is related to several general postulates about structural representation.

The main feature of this formalization of the ETS model is the presence of an infinite hierarchy of representational levels. At each level, object representations are constructed from primitive constructive transformations (building blocks). Primitive transformations of the next level correspond to complex context-dependent additive transformations of the previous one. This hierarchy allows to reduce the complexity of representation of an object by constructing its higher-level representation through the process of inductive inference. It is argued that it is not possible to obtain this kind of hierarchy within the conventional symbolic approaches based on strings, trees, graphs and the corresponding formal grammars.

The restriction of transformations to context-dependent "additions" simplifies the problem of inductive inference in ETS, as compared to the problem of grammatical inference, in which the transformations include deletions and substitutions and which is known to be intractable. It is proved that the generative power of ETS transformations is sufficient for simulating any string-rewriting system without cyclic derivations.

A stochastic generating process and a typicality measure on representations are introduced, in order to conclude with a formulation of an optimization criterion for inductive inference. The criterion is compared with the minimum description length principle.

Acknowledgments

I am deeply grateful to my supervisor, Lev Goldfarb, for opening my eyes on what I now consider to be one of the most beautiful, challenging, and promising directions in science; for putting an unbelievable amount of effort into my education; for supporting me during all five years of my Ph.D. studies in all respects; for creating a unique research atmosphere in our group; for his encouragement, enthusiasm, and abundance of crazy ideas.

I want to thank all members of the ETS group, John Abela, Muhammad Al-Digeil, David Clement, Sean Falconer, David Gay, Lev Goldfarb, and Dmitry Korkin, for their ideas, critique, and for trust that they have put in me as their collaborator. I am especially thankful to Lev Goldfarb, Dima Korkin, and Dave Gay for the unforgettable experience of numerous brainstorming sessions, without which no progress in our research would have been possible.

Thanks to the members of my committee, David Bremner, Mark Burgin, Joe Horton, Viqar Husain, and Vlad Tasić, for their critical reading of the thesis, suggestions, and comments.

I want to thank all professors and staff of the Faculty of Computer Science of the University of New Brunswick not only for the knowledge and help that I have received from them being a Ph.D. student, but also for accepting me as their colleague and giving a wonderful opportunity to teach undergraduate courses and coach the UNB programming team. I am especially thankful to Rod Cooper, Ghislain Deslongchamps, and Will Hyslop for their interest in our research. Many thanks to the Dean of the Faculty, Jane Fritz, for supporting and inspiring me. I am deeply grateful to Sean Falconer for trusting me to supervise his honors thesis research, putting a great amount of effort into it, and successfully completing his project.

Thanks to the UNB Toastmasters club for teaching me to organize and express my thoughts.

Many thanks to my parents, Tamara and Dmitry Golubitsky, for their love and support during this wonderful and important period of my life. Special thanks to my father for reading the thesis and giving helpful comments. Many thanks to my landlady, Patricia Des Champs, for being such a good friend to me and having the best house in Fredericton. I am very grateful to all of my friends in Canada and in Russia for letting me be part of their lives and being part of mine.

Contents

List of Figures

Chapter 1

Introduction

In the following nine sections, I present my thoughts on the main ideas, postulates and concepts of the ETS model. They are presented here, in order to introduce the reader to the way I was thinking while working on the formalization of the ETS model and, hopefully, give more sense to the formal definitions and theorems that constitute this thesis.

I had a strong temptation to explain *why* the model should be formalized the way it is formalized here, and, in fact, attempted to justify its ideas and postulates in the introduction. What I have obtained in the end, still looks like a presentation of the way I think about the model, and, to some extent, like a logical analysis of this way of thinking. A critically minded reader will find a number of statements in this introduction, which require further logical analysis, factual support, and references to the literature in various areas of science. I have to apologize here for not providing this to the extent the reader might want to have. This is not only because of space limitation, but also because the purpose of my research was to *formalize* and not to justify the model. However, I hope that the reader will find enough material in this introduction to build a rough but consistent picture of the model and, with this picture is mind, have an easier time with the formal part.

Trying to achieve a certain level of transparency, I have omitted from the picture most of the references to the literature. This is because most of the references that I know either talk about the same issues in a different language, or talk about different but related issues. In other words, there are very few *direct* references, which I could mention as a foundation or starting point of my research. It is really a diverse and not quite comprehensible variety of different sources, many of which can only be related to the whole picture, but not to any of its parts. Moreover, a number of important ideas and concepts presented in this thesis emerged during the extensive discussions by the members of the ETS group at the University of New Brunswick and have not yet appeared in press. For this reason, I decided to give a literature overview separately, after the introduction into the model (in Section 1.10).

1.1 Mathematics as a language of scientific representation

The issue of representation is and has always been central in all areas of science. Every achievement in science has been due to a new ingenious outlook at a known phenomenon or a description of a new one. To describe things or to look at them differently means to construct a new representation for them.

First of all, this representation is constructed in the scientists' minds. Then, it is communicated among the scientists by means of a scientific language. Arguably, any kind of language, including scientific languages, can also be a part of our internal mental representations. And also quite undoubtedly, a large part of mental representations is not described by any of the existing languages. Thus, we can say that we think in English or in terms of images, formulas, numbers, but by no means this list will cover all kinds of our thoughts. In other words, our representational capabilities include but are not limited to the (current) languages of communication.

Languages can be informal, such as English or pictorial language, or formal, i.e., mathematical language. Here we are concerned with the formal language only.

It is quite an interesting question, to which extent mathematical language is a language of *representation* in science, i.e., what part of scientific knowledge can be described solely in mathematical language and what part relies on other means of representation.

Among all scientific areas, physics is undoubtedly the one that uses the mathematical language most extensively. Point, line, sphere, wave, distance, velocity, vector field, probability, etc. are purely mathematical concepts employed by physicists to describe, and even perhaps to think about, real-world phenomena. There is no other area of science, where this would be the case.

This is not a coincidence, since part of mathematical formalisms has arisen directly from the study of physical phenomena. It is well-known that the concepts of continuous function, differentiation, and integration, are the result of the generalization made by Newton: he assumed that “all geometrical magnitudes might be conceived as generated by continuous motion” [Bal01], thus creating a new formalism based on the intuition about moving objects. Vice versa, mathematics has been directing physics toward new discoveries. Examples of such influence include Maxwell’s electrodynamics, Dirac’s quantum mechanics, quantum electrodynamics and Einstein’s theory of relativity.

Other sciences—chemistry, geography, geology, economy, linguistics, history, biology, psychology, cognitive science—certainly rely on mathematical language to some extent, but it is not as significant as in physics in two aspects. First, very few scientists in the above areas would claim that they *think* in mathematical terms. And second, very few discoveries can be traced back to mathematical concepts.

The reason for this is that historically the mathematical language has been designed to operate with concepts describing quantities obtained through the process of numeric measurement and their derivatives. Physical phenomena are the ones that admit an accurate numeric representation (I think that this statement actually defines physics), but this is not the case with the other areas of science. Indeed, when one thinks about a chemical molecule, a continent on our planet, a sentence from a natural language, a biological cell, or human mind, one tends to think that there is more to them than just a sequence of numeric values. In other words, not only numeric concepts do not currently serve the purpose of representing objects in any area of science except physics, but there is a good chance that they never will.

1.2 Numeric representation vs structural representation

It is not our purpose here to get involved in a philosophical argument about the inadequacy of numeric representation for the purposes of the above scientific areas. Neither we are going to carry out such an argument in case of pattern recognition, inductive learning, and classification. There are two reasons for it. First, this has already been done in [Go90, Go96, Go03]. This view is very far from being universally accepted in the scientific community and, probably, it will not be accepted based on purely philosophical arguments. A formal model tested on practical problems is needed. This is the second reason. Though, a few points need to be mentioned here, as an introduction to the author's way of thinking about structural representation.

A numeric representation is an embedding of the data into a vector space, usually Euclidean. Using numeric representation, one can classify objects by grouping together those that are close to each other according to the Euclidean metric or those that can be separated by a decision surface (see Fig. 1.1). Artificial neural networks, support vector machines, *k*-nearest neighbors would be the obvious examples of numeric models.

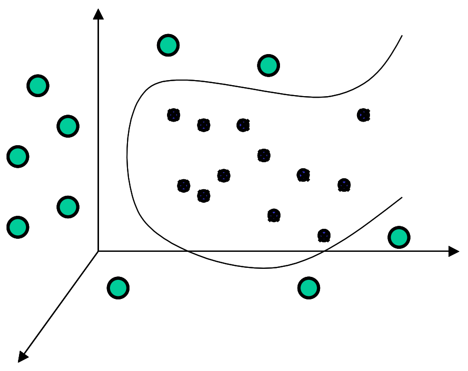

Figure 1.1: Numeric object representation and classification based on Euclidean metric or separation of data by a decision surface.

We would like to expand the concept of a numeric model by including all models, in which objects are represented by feature vectors and features are either real numbers or elements of a finite set. Then bayesian networks, semantic networks, and logical formalisms also fall into the category of numeric models, since they all represent objects by vectors, whose components are either boolean vales or elements of a finite set.

All above mentioned numeric models have their advantages and disadvantages and are widely used in various applications. It is well known in pattern recognition and other areas, how important is the selection of good features for numeric models. For this reason, a successful application of a numeric model to a particular classification problem does not indicate yet that the model itself is robust. Indeed, given a particular (i.e., not very general) classification problem, it is conceivable that one will eventually come up with good features by applying the method of trial and error or any other, independent of the model, considerations. As a result, one may obtain features that almost directly encode the class to which the object belongs, in which case any reasonable classification model would succeed. On the other hand, for very general and complex applications such as protein classification and drug design, translation of natural languages, and motion control in unrestricted environments, no existing models yet can solve classification problems adequately. Which does not mean that they will not in the future, but it is also not clear why they should. Thus, instead of judging about numeric models based on applications, we would like to carry out some theoretical investigation of whether they fit our general intuition about object representation and classification. We have already seen a number of scientific areas, in which numeric representation is too rigid to represent objects in such a way that their essence is captured and accurate classification is possible. Let me try to give an abstract formulation of the reason for this.

I consider it important that numeric representation does not preserve the part/whole relation between objects. That is, given a numeric vector representing an object (say, a house), it is often hard to obtain the vector representing any of its *parts* (say, a window). Unless, of course, these parts are directly encoded in the features, which is clearly impossible for any object of a reasonable complexity. Yet, one has to be able to represent the parts of objects, in order to be able to model *transformations* of the objects. For example, one has to have a representation for the window in order to model the transformation "open the window". In general, this is because parts of objects are *contexts* of the transformations, i.e., they define the necessary conditions for the transformations to apply. Finally, one has to be able to model transformations of objects, in order to *classify* the objects.[1] A house belongs to the class of houses, because it shares with other houses certain transformations that have been used to construct this house (say, build the base, the frame, the walls, lay the roof) or can be applied to the house later on. This suggests an evolutionary description of objects, which consists in the description of the object's past, in terms of the transformations used to construct the object, and the object's future, in terms of the transformations that can be applied to the object later (see Fig. 1.2). A collection of such transformations endowed with a generative mechanism that applies them is thus a class description (see Fig. 1.3). A representation that explicitly includes the transformations used to construct the objects from a certain class is called an *evolutionary structural representation.*

1.3 Universal class description

An immediate consequence of the above definition is that evolutionary structural representation of an object cannot exist without the description of a class to which the object belongs. Thus, a description of an initial class, in which objects can be represented has to be postulated in advance. This initial class should be thought of as a very broad universal class that contains all kinds of different objects, in fact, all

[1]The proponents of numeric models will have to disagree with this statement, since numeric classification is not based on modeling of transformations of objects.

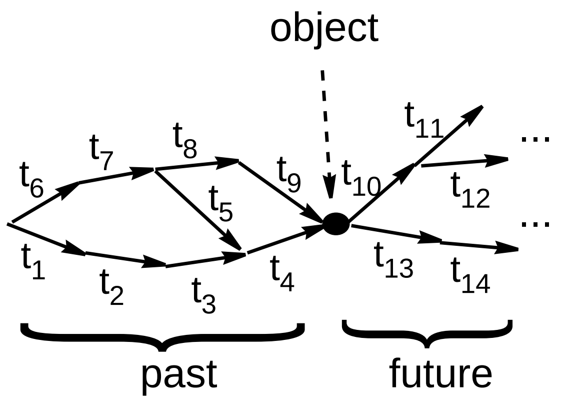

Figure 1.2: Evolutionary description of an object: labeled arrows denote transformations (some of them may have the same structure, even though labeled differently). An object can have several possible constructive histories and several possible future developments.

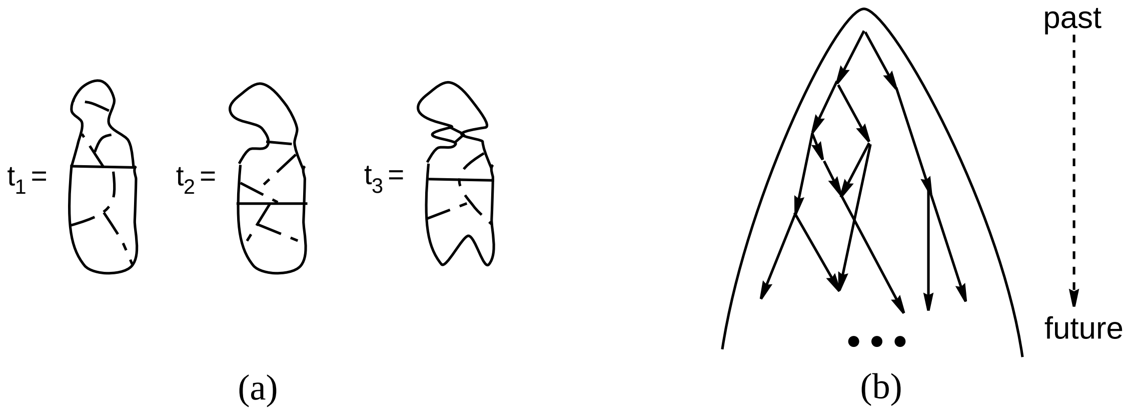

Figure 1.3: Structural description of a class as a list of labels and structures of the class transformations (the structure will be explained later) (a) and the corresponding generating process that generates class objects by applying class transformations (b).

objects considered in a certain scientific problem. For example, it can be the class of all chemical compounds, organic compounds, or proteins.

For a broad class like this we may already have a good idea about the transformations that construct its objects. In fact, we may have an informal description of these transformations. For example, some chemical reactions that build compounds from atoms and smaller compounds can be understood as constructive transformations for these compounds; biochemical processes that build proteins from amino acids can be understood as constructive transformations for proteins; rules of composition of characters, syllables, phonemes, words, etc. can be understood as linguistic constructive transformations as well. It remains to translate the informal descriptions of these transformations into a formal language of evolutionary structural representation, and we obtain a formal description of a universal class.

1.4 Structural measurement

Once the description of the universal class is obtained, the problem of finding representations of particular objects in this class arises. For example, how can we find the sequence of transformations that construct the molecule in our flask? Again, such sequences may already be known for certain molecules, so the corresponding formal representations for them can be obtained.

What if the constructive sequence for the molecule is unknown? Then one can guess a constructive sequence, predict certain properties of its result, and then compare with the results of measurements of these properties for the molecule in the flask. In this scenario, the existing measurement devices play the role of extensions of the generative mechanism mentioned above. Note, however, that most of the existing measurement devices are numeric and (probably, for this reason) the results of measurements can only be matched with the properties that are predicted for the entire constructive sequence. A part of this sequence often gives little information about the properties. Thus, we face the problem of combinatorial search for the sequence of constructive transformations, to which a greedy strategy does not apply (the search space is shown in Fig. 1.3b above). The search complexity becomes exponential in terms of the length of the constructive history, hence only short histories can be obtained in practice. Unless, of course, we have a model that guides this search.

An interesting example of a non-numeric measurement process capable of measuring large objects is the DNA sequencing process. The sequencing process was proposed by Sanger as a method for determining the nucleotide sequence of DNA molecules [Cam93]. The DNA molecule consists of two complementary strands, each of which can be thought of as a sequence of characters A, C, T, G for now, where A is complementary to T and C is complementary to G. The sequencing method alters the DNA replication process as follows. The first step, the separation of the two strands, remains unchanged. Then each strand serves as a template, to which the complementary nucleotides are attached sequentially (these complementary nucleotides are floating among the DNA molecules in abundance). In Sanger's method, one also introduces a modified version of one of the nucleotides, say A', in some concentration. A' behaves exactly as A, that is it attaches to T, except it also terminates the replication process. Thus, instead of a complete complementary strand, we obtain only its prefix. Now, by determining the length of this prefix, we can obtain the position of A in the strand (the length can be determined by separating the DNA strands by electrophoresis on a polyacrylamide gel, which can separate strands differing by one nucleotide in length [Cam93]). Similarly, by introducing C', T', G', one can obtain the positions of the other nucleotides in the strand.

It is important to note that, although sequencing does produce a sequence of transformations that can be understood as constructive for the DNA molecules, these are not the transformations that define the classes of these molecules. The transformations that do define these classes are the evolutionary transformations that have produced the molecules (and also the ones that might occur to the molecules later), i.e., mutations. These mutations occurred to the molecule *before* the replication process simulated by the sequencing method. Thus, the sequencing process does determine a part of constructive history of the DNA molecule, but this is only the most recent part, whereas the classes of DNA's are determined by the earlier parts of the historic process. In other words, in order to be able to classify DNA molecules, one has to construct their representation based on transformations corresponding to mutations, which will clearly look different from the sequential attachments of nucleotides. That is, the sequencing method is an example of a structural measurement process in a certain universal class, but this is not the class, within which we can find the subclasses of DNA's created by evolution.

1.5 Conventional data structures and object representation

Experience of using conventional data structures such as strings, trees, and graphs, shows that it is possible to describe some of the evolutionary transformations as transformations of these data structures. For example, certain point mutations and crossovers can be described as formal production rules, rewriting rules, splicing rules, etc. However, these descriptions have a lot of exceptions. The exceptions are of two kinds: the actual evolutionary transformations may not be representable by rules of the selected type and, vice versa, many of the rules may have nothing to do with the actual evolutionary transformations. Both kinds of exceptions, if encountered often, cause problems: the first simply does not allow to encode transformations and the second makes the search for the transformations intractable.

Noam Chomsky has introduced a hierarchy of types of formal grammars, known as Chomsky's hierarchy. By moving along this hierarchy from regular (type 3) to context-free (type 2), context-sensitive (type 1) and unrestricted (type 0) grammars, we decrease the number of exceptions of the first kind but simultaneously increase the one of the second. Thus, we might hope to find an intermediate type, for which both kinds of exceptions are reasonably rare. But this is apparently not the case. For example, context-free grammars describe only a small part of evolutionary transformations, yet are already extremely hard to search for. This suggests that we should be moving along a different hierarchy.

Surprisingly enough, the extension of Chomsky's theory to graphs suggests an idea about this new hierarchy. It is known that there are context-dependent string languages that, if the strings are encoded by linear labeled graphs, can be generated by context-free graph grammars (see [Ro97]). In other words, by choosing a different data structure instead of a different type of rules, we may hope to increase the generative power and keep reasonable the complexity of the search for the rules.[2]

Still, the question of how to choose the appropriate data structure in a systematic way remains open. And, once this data structure is chosen, which transformations should be considered on it? The purpose of the ETS model is to answer these questions.

[2]This is not to say that we should try to represent evolutionary transformations by graph-grammatic rules. In fact, the parsing problem for context-free graph grammars is known to be NP-complete, and there is no advantage of graph grammars that I am aware of as far as the representation of evolutionary transformations of genetic molecules is concerned. But the idea of changing the data structure instead of the generative mechanism, in my opinion, is important.

1.6 ETS constructive histories and object representations

Both questions are answered in the ETS model simultaneously. It is postulated that once a data structure is fixed, its transformations are fixed as well. This postulate is consistent with the general approach to data structures and abstract data types in computer science. It is just that for some reason this approach has not been strictly followed in case of strings, trees, or graphs—there is no fixed definition of a transformation of these structures or, in other words, different kinds of transformations exist.

The definition of the ETS data structures is essentially based on the concept of the object evolutionary history (the idea was proposed by Lev Goldfarb). This history, which is a sequence of constructive transformations, is explicitly present in the central data structure of the model (see Fig. 1.4a). Once the data structure corresponding to a single constructive history is introduced, it becomes inevitable that *objects should be viewed as collections of constructive histories* (see Fig. 1.4b).

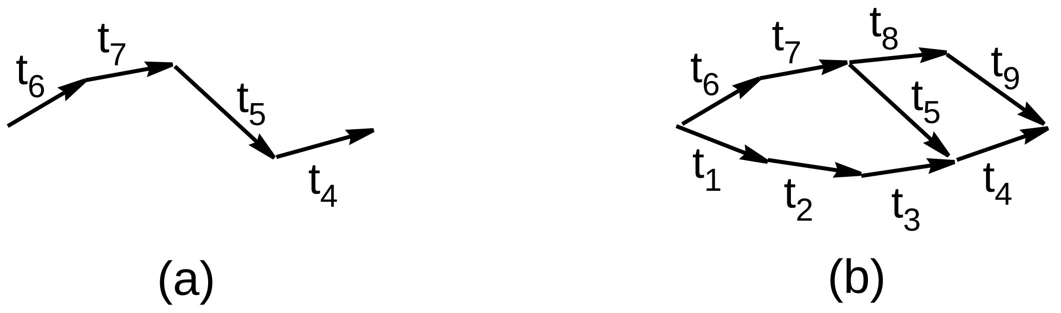

Figure 1.4: A single constructive history represented as a sequence of constructive transformations (a), and an object represented as a collection of constructive histories (b). The structure of transformations will be explained later.

This assertion reflects the fact that objects usually have several possible constructive histories. For example, there can be several ways to obtain a certain chemical compound through reactions. In other words, there are several constructive histories, which have the same result, and this result *is* the compound. Or, if we want to be more accurate here, we have to assume that the results of different constructive histories may be different, but they *appear* to us and, more importantly, to various chemical reactions to be the same.

Formally, one can express the fact that certain constructive histories produce the same result by introducing an equivalence relation on the set of constructive histories; the equivalence classes then *are* representations of objects. These equivalence classes turn out to be a rich source for data structures. In particular, it is shown in this thesis how to obtain strings, trees, and graphs in this way. It will also become clear that the equivalence classes are much more general than these conventional data structures (see Fig. 1.5).

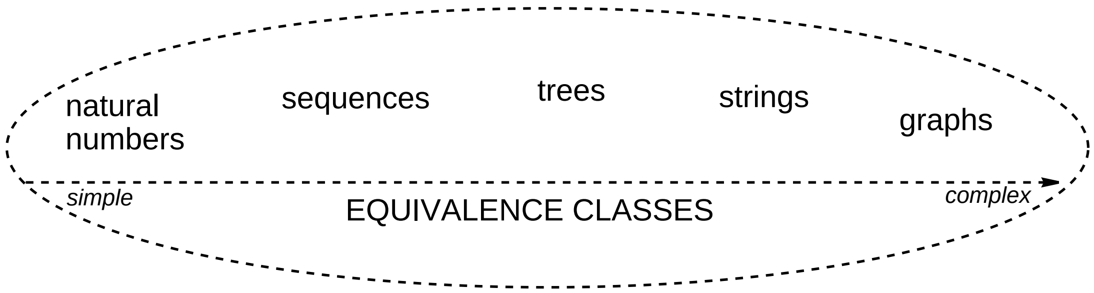

Figure 1.5: Equivalence classes of constructive histories and conventional data structures depicted as particular cases of the equivalence classes.

The choice of the equivalence relation depends on the particular application and requires to incorporate existing scientific knowledge about the objects into the definition of the equivalence relation. Sometimes this can be done easily, if the existing knowledge is already expressed as an equivalence relation (for example, the resonance effect in chemistry—see Fig. 1.6). In other cases, it may require substantial effort (see the definition of an equivalence relation on the histories corresponding to organic compounds in Dmitry Korkin's PhD thesis [Ko03]). But, in any case, the definition of the equivalence relation on histories is not what we mean by a *systematic* selection of the data structure for object representation. This is because this definition has to be done in one step, and systematic approach means at least several steps guided by a model. In fact, the definition of the equivalence relation is only the initial step in the process of construction of object representation, which has to be made by a human. The remaining steps are supposed to be made by an algorithm based on the ETS model. Each of these steps modifies the initial representation by applying the inductive learning algorithm.

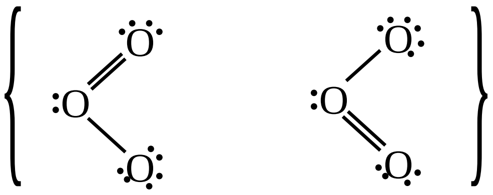

Figure 1.6: "The resonance description of ozone ... the molecule has a structure represented by the superposition of the two structures shown" [Paul75].

1.7 Inductive learning and modification of object representation in ETS

It is time now to recall that the initial representation is the representation of objects in a large universal class. It is the description of this class that is specified, when we define the constructive histories and the equivalence relation on them. Recall also that, because of the exponential complexity of the search for a representation, we can only assume that representations of small objects, whose histories consist of just a few constructive steps, can be obtained.[3]

From now on, we will call the constructive steps in the universal class *primitive transformations*. Complex objects, together with decomposition of their history into primitive transformations, usually also admit decomposition into larger chunks, which we will call *composite transformations* (see Fig. 1.7). For example, a protein can be decomposed into atoms or into amino acids, which can themselves be decomposed into atoms. Thus, intuitively, composite transformations are composed out of primitive transformations.

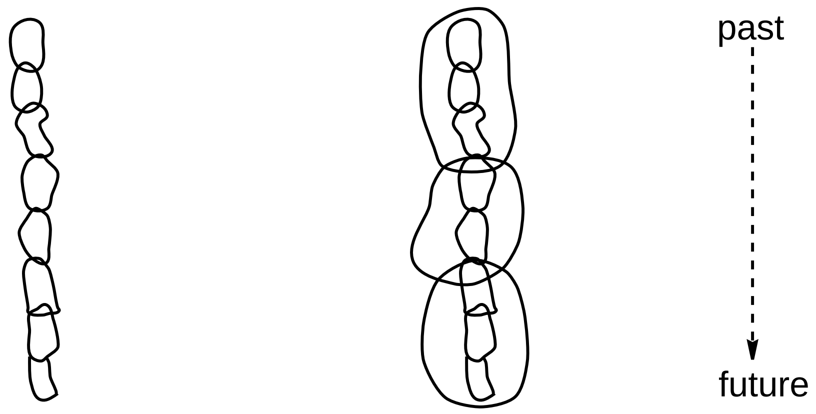

Figure 1.7: Decompositions of a constructive history of an object into primitive transformations (left) and composite transformations (right).

Now, suppose we have several objects that, on the one hand, are small enough to allow their representation in the universal class and, on the other hand, are large

[3]Strictly speaking, the number of steps in different histories for the same object may vary, but this is not important for us now.

enough to contain composite transformations in them. Then, by applying the inductive learning algorithm, we can extract these composite transformations from the given object representations.[4] Once the composite transformations are extracted, one can obtain *new* representations for the objects as follows: the new constructive histories are sequences of composite transformations, and the new object representations are equivalence classes of them.

Note that the new representations are simpler in terms of the number of transformations in the constructive histories. This applies to any objects from the class generated by the extracted composite transformations.[5] Thus, within this subclass of the universal class, we are now capable of constructing representations of more complex objects, compared to the ones that we could obtain before learning. For example, once we discover amino acids from simple compounds, we can represent much more complex compounds, such as proteins. In other words, the composite transformations learned from the training set allow to lift the representations of objects from the subclass, to which this training set belongs, to a new level of the *representational* hierarchy (see Fig. 1.8).[6]

1.8 ETS representational hierarchy

The representational hierarchy and inductive learning algorithm constitute the systematic and algorithmic approach to the construction of representation of complex objects, which was discussed above. Once a subclass is learned, and therefore we are

[4]The inductive learning problem is a very important and difficult problem, thus the claim that composite transformations can indeed be extracted from the training data needs a thorough justification. We will return to this question in a moment, in Section 1.9 below.

[5]Recall our discussion of structural class description at the end of Section 1.1.

[6]Note the difference between this hierarchy and the Chomsky's hierarchy.

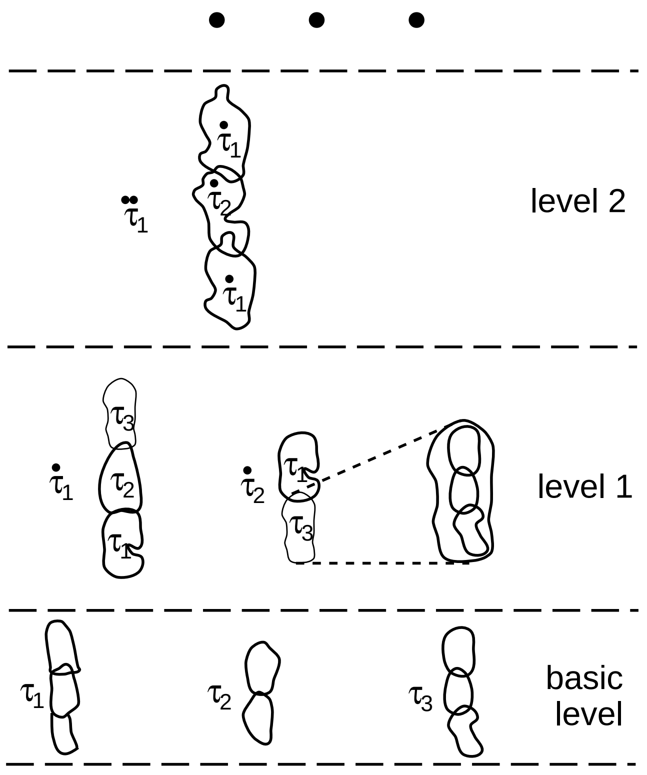

Figure 1.8: Representational hierarchy (the number of dots over a transformation label specifies the level, to which the transformation belongs).

capable of representing its more complex objects, we can take a new, more complex, training set and learn the composite transformations of the next level, which will be compositions of the current composite transformations. This process, which we will call the *inductive learning process*, can be continued to a potentially unlimited number of representational levels.

Note that the description of the inductive learning process as a process that creates a multi-level representational hierarchy is consistent with the biological understanding of the evolutionary process in Nature. I see the result of this process in the hierarchy of atoms, molecules, cells, tissues, organs, organisms, societies, etc...—each concept in this list corresponds to a level in the representational hierarchy. This description is also consistent with the psychological understanding of learning performed by children, which involves construction of more complex concepts out of previously learned simpler ones. Finally, it is consistent with the process of development of scientific concepts and theories. Thus, according to the ETS model, the general laws guiding the construction of object representations should be identical with those guiding the construction of objects themselves! In a broader perspective, this assertion expands the meaning of the term “representation”. For example, biological organisms can be considered as representations of the inorganic environment they live in, constructed within the universal class of organic compounds. In general, any kind of interaction results in construction of a new representation within a certain class, which triggers another step of the inductive learning process, i.e., of the global evolutionary process.

1.9 ETS transformations and inductive learning

As it is clear from above, it is crucial for the ETS model (and any other model of classification) that the inductive learning problem is solved efficiently. Probably, the unpopularity of symbolic approaches in the areas of pattern recognition and classification (compared with numeric ones) is largely due to the fact that in many cases learning in symbolic environments is extremely difficult. For example, this is the case with regular and context-free grammars, even though their generative power is quite limited.

Of course, a learning algorithm with a good theoretical upper bound on its time complexity and/or good practical performance would be a definite asset of the model it is based upon. But right now, when the formalization of the model has just been completed, it is too early to talk about learning in algorithmic terms. In fact, because the model is so general, it unlikely that it will be possible to develop a universal learning algorithm. It is more reasonable to direct the research toward concrete applications, consider particular universal classes, and design particular learning algorithms for them. Therefore, the design of the learning algorithm is really inseparable from the applied side of the model. All I can do now, when a substantial amount of work on practical application and testing of the model is still ahead, is give some clues on why I think that the learning problem in the ETS model will be resolved more successfully than it has been for other symbolic approaches. I hope, the reader will allow me to discuss briefly the future of the model, rather than what has been achieved so far.

The two key features of the model that, in my opinion, should result in an efficient learning algorithm, are explicit representation of the constructive histories and the representational hierarchy.

It is pretty clear how the representational hierarchy works: it allows to subdivide a difficult learning problem into several simpler steps and takes care of linking those steps together. The reason why we should be successful in making a single learning step is in the explicit representation of the constructive histories.

Recall that a single learning step consists in constructing composite transformations based on a training set. The fact that objects from the training set are defined as collections of constructive histories leaves very little freedom to the choice of the form of transformations. Indeed, it implies that transformations act on constructive histories and, by acting, continue them. Thus, transformations have to be constructive, i.e., they can add something to constructive histories but cannot delete anything. A transformation may also require that a certain segment of constructive history is present before it can be applied. All these considerations imply the definition of a transformation as a *context-dependent attachment of a part of constructive history* (see Fig. 1.9).[7]

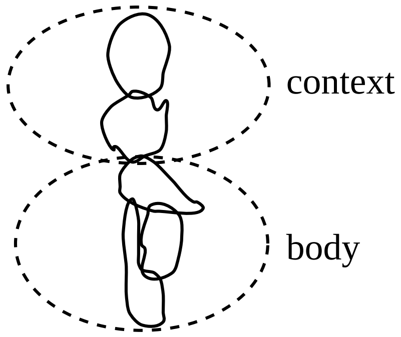

Figure 1.9: A transformation defined as a context-dependent attachment of a part of constructive history.

The definition of transformations of constructive histories is naturally extended to transformations of objects, since the latter are collections of histories. In case of strings, for example, we obtain context-dependent insertions of substrings, and in case of graphs context-dependent attachments of subgraphs.

By defining transformations as attachments (i.e., prohibiting substitutions and deletions), we guarantee that the transformations are explicitly present in the objects generated by them. This explicit presence of transformations in the training set is the feature that should make the search for these transformations a tractable problem. In fact, for certain special cases of non-deleting transformations on strings efficient learning algorithms have been developed in Sandeep Nigam's and John Abela's theses

[7]This requires to define a part/whole relation on histories and objects. But this relation is something that we began our discussion with (see Section 1.2), so it is essential to the model anyway.

[Ab02, Ni93].

Now it may look as though the transformations have been defined in such a way that the learning is as easy as possible, but no care has been taken for their generative power. Where is the guarantee that any interesting classes can be described using the formalism? I think, the statement that this is possible *in principle* is justified by a theorem proved in this thesis. The theorem states that a broad class of string-rewriting systems (or unrestricted grammars), namely all systems whose rules do not allow to produce a part of an object from the object, can be simulated by ETS transformations, given that the representation of constructive histories and the equivalence relation on them are chosen appropriately (see Theorem 14).

Finally, the ETS model provides an optimization criterion for inductive learning, which is based on another central concept defined in this thesis, the stochastic generating process. This process is associated with every class and is therefore specified by the class description (which is a set of transformations) plus numeric parameters that control the flow of the process. The process induces a numeric measure of typicality on the class objects and on the transformations that are composed of class transformations. As a result, we obtain a criterion for learning of a subclass from a training set, which forces to maximize both the typicality of the objects from the training set with respect to this subclass and the typicality of the transformations constituting the subclass description. These two typicalities "balance each other" (similarly to the minimum description length principle), thereby yielding a non-trivial subclass description. The trivial subclasses, such as the entire universal class and just the training set, are excluded because one of the typicalities becomes very low.

1.10 Literature overview

The ETS model was proposed as a framework for structural object representation and classification in 1990 by Lev Goldfarb. His group has published several papers (see, for example, [Go90, GN94, Go96, GGK01, GG01, Go03]), in which the main ideas and postulates of the model are formulated and a comparison between numeric and structural representations is carried out (the structural representation considered in these papers corresponds to the evolutionary structural representation discussed in this thesis; the term "evolutionary" is added to clarify and emphasize the difference from other uses of the word "structure" in science). The terminology for the central concepts in this thesis is consistent with these papers. In the graduate theses written by the members of the ETS group at the University of New Brunswick (e.g. [Cha92, Dew91, Sa92, Ni93, Kam95, Ab02]), one can find concrete learning algorithms designed for some special cases of structural representations and classes. The model is applied to classification problems in computer vision in [Des96, Ho98] and to classification of molecules in [Ko03].

The idea of a generative class description goes back to Noam Chomsky's "Syntactic Structures" [Cho65] (or, perhaps, even further), in which he proposed formal grammars to describe the classes of syntactically correct sentences in natural languages. Formal grammars [RS97] have found numerous applications in computational linguistics and computer science. As an alternative approach (but not very different from formal grammars, in my opinion), one can consider string-rewriting systems [Bo93]. Other rewriting mechanisms are discussed in [Paun98] in connection with biological computations. String rewriting models have been later generalized to graphs, resulting first in inductive descriptions of graph classes (for example, see [Bat85] for an inductive description of the class of all planar graphs), and then in the theory of (hyper-)graph grammars [Ro97] (or graph-rewriting systems), which has also found its applications [Bl95].

The importance of representation of object's constructive history has been emphasized by Leyton in [Le92]. This book considers numerous examples of various objects, such as geometric shapes or natural language sentences, from the historic perspective.

The multi-level hierarchy of representations has been discussed by Sloman (see, e.g., [Slo00]). The concepts of *i-trajectory* and *e-trajectory* introduced in [Slo00] correspond quite closely to the ETS concepts of constructive processes at lower and higher representational levels, respectively. Indeed, i-trajectories are "trajectories that are possible for an individual which adapts or changes itself", whereas e-trajectories are the ones "that are not possible for an individual machine or organism but are possible across generations" [Slo00]. In other words, "e-trajectories for *individuals* can be thought of as i-trajectories for a *species*, or a larger encompassing system, such as an ecosystem". A simpler example of such larger encompassing system would be an organism, whose i-trajectories are at the same time e-trajectories for its constituents, e.g. cells. Interestingly enough, Sloman mentions in [Slo00] in conclusion that "the ideas discussed here deal with phenomena which still seem to be too ill defined for mathematical formulation and computational modeling. However, that may change." Hopefully, the ETS model will help this change to come sooner. I also agree with Sloman that a mathematical formulation should precede computational modeling. For this reason, the logical completeness and general mathematical clarity of the model are the primary goals of this thesis, whereas the computational questions are touched only briefly, and some of them are even left beyond its scope.

In papers [Fo88, Ai97], Fodor, Pylyshyn and Aizawa argue for the inadequacy of numeric representations as a model of mental representations, since the latter possess certain properties which the former do not. Following [Fo88], we formulate these properties in linguistic terms, however, it should be noted that these arguments can be extended far beyond linguistics. So, mental representations are

(a) *systematic*, which means that our "ability to produce/understand some sentences is intrinsically connected to the ability to produce/understand certain others" [Fo88];

(b) *compositional*, which means that there are "semantic relations between words *and the expressions of which they are constituents*" [Fo88]. In particular, "compositionality implies that (some) expressions *have* constituents" [Fo88], thereby explaining systematicity. Indeed, then our ability to produce/understand a sentence implies the ability to produce its constituents and, as a consequence, all other sentences composed of these constituents;

(c) *productive*, i.e., under appropriate idealization, our representational capacities are unbounded, even though they admit a finite description. The property of productivity is a consequence of systematicity and compositionality, if we assume that primitive constituents admit arbitrary large compositions.

All these properties are very much related to our discussion of evolutionary structural object representation, the necessity of modeling of transformations on objects for classification, structural class description, and the form of transformations. In a sense, all these ideas go back to the generative approach to classification proposed by Chomsky.

Some formal results about the limitations of numeric representations are presented in [Be01].

The importance of inductive learning for object representation and classification needs not be emphasized here. I would only like to quote Hermann von Helmholtz: "Inductive inferences, executed by the unconscious activity of memory, play a commanding part in the formation of intuitions. It may be doubted that there is any indication whatsoever of any other source or origin for the ideas possessed by a mature individual."

The minimum description length (MDL) principle, which is very similar to the maximum entropy principle [Gr98], can be derived as a special case from the theory of Kolmogorov complexity [Ri89, Li97]. The MDL principle has been used as a principle for the design of various optimization criteria for inductive learning (see [Ke97] for example). These criteria always include two components corresponding to the complexities of the descriptions of the class and of the training set, assuming that the training set is described in terms of the class description. The presence of these two components brings some arbitrariness to the applications of the MDL principle, since the components usually have different nature and form (for example, in case of grammatical inference, the components are the complexities of a grammar and of a set of sentences, which are computed quite differently). Because of this difference, there is no guiding principle which would tell us how to combine the two components in one criterion in a consistent manner (since the units of complexities can be chosen arbitrarily). The only way to resolve this problem is to come up with a model, in which class descriptions and training set descriptions would have the *same* nature and form. Although the forms of transformations and objects are different in the ETS model (the former have contexts and the latter don't), they are still similar enough to allow a consistent measurement of their complexities and, therefore, a consistent criterion based on the MDL principle.

1.11 Structure of the thesis

In Chapter 2, the basic representational level is introduced and the concept of a basic level inductive structure is defined. This includes a formal description of a constructive process (Section 2.1) and a definition of an object representation (Section 2.2) based on the underlying constructive processes. In Section 2.3, several examples of basic level inductive structures are considered and related to some conventional data structures, including natural numbers, sequences, strings, trees, and graphs. Based on the explicit presence of the constructive processes in the representation, we explain why certain computational problems on conventional data structures, including the problem of inductive inference, are intractable. We also suggest how to resolve this intractability—which leads us to the definition of higher representational levels. In Section 2.4, we introduce the concept of a *struct tuple*, which serves as a technical tool for the formal definition of the the infinite hierarchy of representational levels in Chapter 3.

A comparison of various generative formalisms, including Chomsky grammars [Cho65, RS97], string-rewriting systems [Bo93], and graph grammars [Ro97], with the ETS formalism is carried out in Sections 2.3.4 and 3.1. The arguments are supported by several theorems, most importantly Theorem 2 of Section 2.2.6, Theorem 11 of

Section 3.1.6, Theorem 13 of Section 3.2.1, and Theorem 14 of Section 4.1. Examples of inductive structures corresponding to graph languages in ETS are considered in Section 4.2. These examples are followed by a conjecture that any recursive language can be generatied by a particular first level inductive structure.

In Chapter 5, we discuss briefly the *stochastic generating process*, which is intended to model the process of construction of representations in time. A detailed study of this process, including its formal definition as a continuous parameter Markov chain, is presented in [Golub02]. A measure of typicality, induced by this process, is also defined in Chapter 5, followed by the formulation of an optimization criterion for inductive inference. The *structural measurement process* is briefly discussed in this section, the details can be found in [GG01]. Finally, the optimization criterion is compared with the minimum description length principle [Gr98].

A conclusion and an overview of future research directions are presented in Chapter 6.

The following results of this thesis can be considered as the most important ones from the point of view of their nontriviality or "distance" from definitions: Theorem 2 in Section 2.2.6 about struct finiteness and its higher-level analogue, Theorem 11 in Section 3.1.6; Theorem 5 in Section 2.3.8 about undecidability of part/whole distinguishability and struct finiteness conditions; Theorem 14 in Section 4.1 establishing a correspondence between ETS transformations and strongly acyclic string-rewriting systems; a collection of lemmas and theorems listed in the introduction to Chapter 3 that provides the formal basis for the inductive construction of the infinite representational hierarchy.

Chapter 2

Basic representational level

The formalization of the ETS model begins with the definition of the basic representational level and the corresponding inductive structure. This includes a formal description of a constructive process (Section 2.1) and a definition of an object representation (Section 2.2) based on the underlying constructive processes. In Section 2.3, several examples of basic level inductive structures are considered and related to some conventional data structures, including natural numbers, sequences, strings, trees, and graphs. Based on the explicit presence of the constructive processes in the representation, we explain why certain computational problems on conventional data structures, including the problem of inductive inference, are intractable. We also suggest how to resolve this intractability—which leads us to the definition of higher representational levels. In Section 2.4, we introduce the concept of a *struct tuple*. This concept serves as a technical tool for the formal definition of the the infinite hierarchy of representational levels in Chapter 3. In order to ensure logical consistency of our definitions—which allows to work with the concepts of any representational level in the same abstract manner—a number of level-invariant statements are proved.

2.1 Process of construction of object representation

We introduce the concept of a *primitive type*, which is a formal equivalent of the notion of a primitive constructive operation, and the concept of a *formation*, which corresponds to the process that involves several primitive operations and thus constructs a representation of a physical object.

For technical reasons, it turns out to be convenient to introduce auxiliary concepts of a *primitive*, which may be thought of as an instance of a primitive type and thus corresponds to an instance of a primitive constructive operation, and a *composite*, which is a particular composition of primitives and thus corresponds to an instance of the above constructive process. Instances are necessary to specify *how* the primitive types are composed with, or connected to, each other. Alternatively, connections could be specified directly, for example by a mapping between connection sites. For our purposes, it is more convenient to define primitives that already "know" how to connect to each other. This allows to compose them via a binary operation of attachment. In Section A we draw a parallel between the two approaches and suggest a simple mental image for primitives and composites.

Several primitives sequentially attached to each other form a composite. Composites that correspond to the same compositions of primitive types form an equivalence class, called a formation.

2.1.1 Primitives

Definition 1. A **primitive** is a 3-tuple $\pi \overset{\text{def}}{=} \langle \alpha, I, T \rangle$, where α is the **label of primitive** π, and I, T are disjoint finite linearly ordered sets.

For a primitive $\pi = \langle \alpha, I, T \rangle$, we use the following notation:

$$\begin{array}{rcll} \text{init}(\pi) & \stackrel{\text{def}}{=} & \{i \mid i \in I\} & \text{set of } \textbf{initial sites} \text{ for } \pi \\ \text{term}(\pi) & \stackrel{\text{def}}{=} & \{i \mid i \in T\} & \text{set of } \textbf{terminal sites} \text{ for } \pi \\ \text{sites}(\pi) & \stackrel{\text{def}}{=} & \text{init}(\pi) \cup \text{term}(\pi) & \text{set of (all) } \textbf{sites} \text{ for } \pi. \end{array}$$

►

Note that the sets init(π), term(π), and sites(π) are unordered.

Pictorially, it is convenient to represent a primitive $\langle \alpha, I, T \rangle$ as a circle with $|I|$ points marked on its upper part and $|T|$ points marked on its lower part. Each point is labeled by a letter denoting the corresponding site. The left-to-right ordering of the points in the picture specifies the linear orderings on I and T. In Fig. 2.1, four primitives are shown:

$$\begin{aligned} \pi_1 &= \langle \alpha, \langle a, b \rangle, \langle c \rangle \rangle \\ \pi_2 &= \langle \beta, \langle c \rangle, \langle d, e, f \rangle \rangle \\ \pi_3 &= \langle \beta, \langle c \rangle, \langle d, f, e \rangle \rangle \\ \pi_4 &= \langle \beta, \langle c \rangle, \langle a, f, e \rangle \rangle. \end{aligned}$$

Note that different primitives can share their labels and/or sites and also that $\pi_2 \neq \pi_3$.

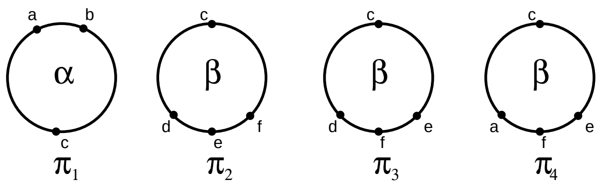

Figure 2.1: Pictorial representation for primitives.

In addition to the abstract illustrations like Fig. 2.1, I would like to illustrate each concept of the basic representational level with an example, which will run through this and the next section and result in the ETS definition of string data structure.

Consider the set $\mathbb{S}$ of strings over the alphabet $\{a, b, c\}$ (the construction below can be easily extended to arbitrary finite alphabets), and assume that each string is obtained as a result of a sequence of insertions of characters, starting from the empty string. The set of specifying primitives for strings, $\Pi_{\mathbb{S}}$, is shown in Fig. 2.2. Primitives π_a, π_b, and π_c correspond to the elementary operations of insertion of characters a, b, and c at any position in the string.

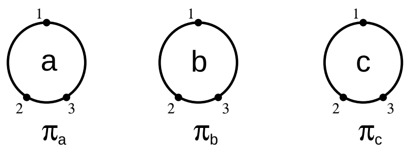

Figure 2.2: Primitives corresponding to insertions of characters.

The reasons for defining a primitive as the above 3-tuple are discussed in the Appendix.

2.1.2 Composites

We have defined primitives in such a way that no additional concepts are required to define compositions of them, since the information about the potential connections of a primitive to other primitives is contained in its sites. We can call two primitives attached, if they share sites properly. More precisely, the only sites that can be shared by attached primitives are terminal sites of the first primitive and initial sites of the second.

Definition 2. An ordered pair of primitives $\langle \pi, \sigma \rangle$ satisfies the **attachment condi-**

tion, denoted $\pi \dashv \sigma$ if and only if

$$\text{sites}(\pi) \cap \text{sites}(\sigma) = \text{term}(\pi) \cap \text{init}(\sigma). \tag{2.1}$$

▶

Note that the intersections in (2.1) may be empty, in which case we obtain an "empty" attachment. In Fig. 2.1, the only two pairs that satisfy the attachment condition are $\langle \pi_1, \pi_2 \rangle$ and $\langle \pi_1, \pi_3 \rangle$.

Composites are defined as finite[1] sequences of primitives satisfying the attachment condition. The way *how* the primitives are attached is already specified by their sites.

Definition 3. A **composite** is a finite sequence of primitives $\gamma = \langle \pi_1, \ldots, \pi_n \rangle$ such that if $i < j$, then $\pi_i \dashv \pi_j$. The empty composite is denoted by λ. ▶

Pictorially, it is convenient to represent a composite by connecting the shared sites of its primitives (see Fig. 2.3). The position of a primitive in the sequence corresponds to its vertical position in the picture (thus, no primitives can be have the same vertical position).

An example of a composite representing a constructive history for the string "aacb" is shown in Fig. 2.4.

Composite is a new abstract data type.[2] It differs from the conventional data types such as strings, trees, graphs, and hypergraphs, in one important aspect. A composite explicitly contains all structural information about the process that has

[1]Finiteness is necessary for practical use of composites for representation of constructive processes. Infinite composites may prove (and actually have proved) to be useful in description of various properties of classes of objects, but this is beyond the scope of the present thesis.

[2]Two operations on composites will be defined later.

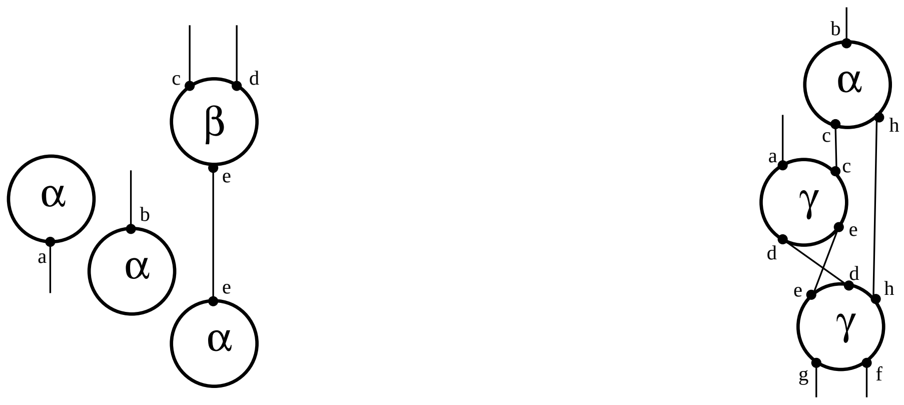

Figure 2.3: Two examples of composites.

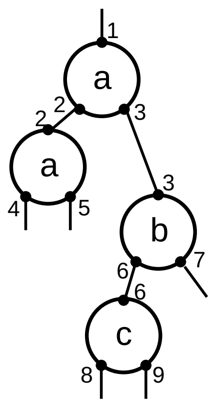

Figure 2.4: A composite representing a particular constructive process that constructs string "aacb". In this process, first, character 'a' is created, then another 'a' is inserted before it, then 'b' is inserted after the first 'a' and, finally, 'c' is inserted before 'b'.

constructed it, namely, which primitives and in which order have been attached by this process. For the conventional data types, on the contrary, it is often hard to tell what that process was. In fact, depending on a particular problem or application of the data type, different processes may be assumed (the above example shows that a string of length n has $n!$ possible constructive histories, if only insertions are allowed; in reality, this assumption is not necessarily made, so the number of possible histories for a string is actually unlimited).

Formally, one can consider composites as hypergraphs [Ro97], but the following distinctions should be kept in mind, as they relate to the above mentioned requirement of explicit representation of the constructive history. First, the sets of initial and terminal sites of primitives are ordered (the reason for the introduction of this ordering is explained in the Appendix), as opposed to the unordered set of nodes of a hyperedge. Second, the primitives in a composite are also ordered. One of the reasons for introducing new terminology in [GGK01] and, consequently, in this thesis is that the emphasis is on these orders and not on the other characteristics of primitives and composites, which make them somewhat similar to hyperedges and hypergraphs. It is these orders, that, in my opinion, invoke the right way of thinking about primitives and composites and eventually leads to a generalization of these concepts to the higher representational levels, which is presented in Chapter 3. The difference of those, generalized, concepts (see, for example, Def. 39) from hypergraphs should become even more apparent.

2.1.3 Composition of composites

Composites can be constructed out of other composites via the operation of composition.

Definition 4. An ordered pair of composites $\langle \alpha, \beta \rangle$ satisfies the **composition condition**, if for all primitives $\pi \in \alpha, \sigma \in \beta$, $\pi \dashv \sigma$.

For a pair of composites $\langle \alpha, \beta \rangle$ satisfying the composition condition, the **composition** $\alpha \lhd \beta$ is defined as the concatenation of sequences α and β. ▶

Remark. Considering composites as ordered multisets of primitives (indeed, ordered multisets and sequences are the same concepts), one can unambiguously use the set notation, such as $\pi \in \gamma$, where π is an element of sequence γ.

In Fig. 2.5, two composites representing particular constructive histories for strings "ab" and "bc" and their composition are shown. The composition represents a constructive history for the string "abcb", which is the result of insertion of "bc" into "ab" (between 'a' and 'b').

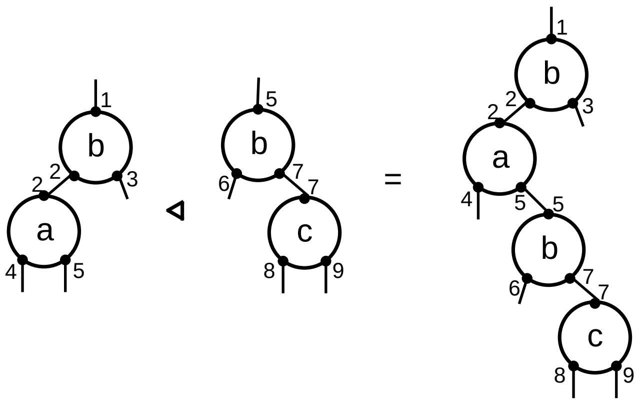

Figure 2.5: Composition of two composites representing constructive histories for strings "ab" and "ba'.

Lemma 1. A pair of composites $\langle \alpha, \beta \rangle$ satisfies the composition condition if and only if the concatenation of sequences α and β is a composite.

Proof. If a pair of composites $\langle \alpha, \beta \rangle$ satisfies the composition condition, then, as a direct consequence of definitions of a composite (Def. 3) and composition (Def. 4), for all $\pi_i, \pi_j \in \alpha \lhd \beta$ $(i < j)$, $\pi_i \dashv \pi_j$. Hence $\alpha \lhd \beta$ is a composite.

Conversely, consider $\pi_i, \pi_j \in \alpha \lhd \beta$. If $\pi_i \in \alpha$ and $\pi_j \in \beta$, then $i < j$. Hence, if $\alpha \lhd \beta$ is a composite, then, according to Def. 3, $\pi_i \dashv \pi_j$. Therefore, the composition condition is satisfied. ■

The following two lemmas follow directly from the properties of concatenation of sequences:

Lemma 2. For any composite γ, $\lambda \lhd \gamma = \gamma \lhd \lambda = \gamma$ (where λ is the empty composite).

Lemma 3. If $\alpha \lhd \gamma \lhd \beta = \gamma$, then $\alpha = \beta = \lambda$.

Composition of composites is associative:

Lemma 4. If α, β, γ are composites such that pairs $\langle\alpha, \beta\rangle$ and $\langle\alpha \lhd \beta, \gamma\rangle$ satisfy the composition condition, then the pairs $\langle\beta, \gamma\rangle$ and $\langle\alpha, \beta \lhd \gamma\rangle$ also satisfy the composition condition and

$$(\alpha \lhd \beta) \lhd \gamma = \alpha \lhd (\beta \lhd \gamma). \tag{2.2}$$

One can verify associativity of composition directly using the definition of composition, or, alternatively, apply the composition criterion introduced below.

Definition 5. For a composite γ,

$$\begin{aligned}
\text{init}(\gamma) &\overset{\text{def}}{=} \textstyle\bigcup_{\pi\in\gamma} \text{init}(\pi) \setminus \bigcup_{\pi\in\gamma} \text{term}(\pi) \\
\text{term}(\gamma) &\overset{\text{def}}{=} \textstyle\bigcup_{\pi\in\gamma} \text{term}(\pi) \setminus \bigcup_{\pi\in\gamma} \text{init}(\pi) \\
\text{sites}(\gamma) &\overset{\text{def}}{=} \textstyle\bigcup_{\pi\in\gamma} \text{sites}(\pi).
\end{aligned}$$

▶

For the composite shown in Fig. 2.4, the sets of sites are:

$$\text{init}(\gamma) = \{1\}, \quad \text{term}(\gamma) = \{4, 5, 7, 8, 9\}, \quad \text{sites}(\gamma) = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}.$$

Lemma 5. For any pair of composites $\langle\alpha, \beta\rangle$ satisfying the composition condition, the following relationships hold:

$$\text{init}(\alpha \lhd \beta) = \text{init}(\alpha) \cup [\text{init}(\beta) \setminus \text{term}(\alpha)] \tag{2.3}$$

$$\text{term}(\alpha \lhd \beta) = [\text{term}(\alpha) \setminus \text{init}(\beta)] \cup \text{term}(\beta) \tag{2.4}$$

$$\text{sites}(\alpha \lhd \beta) = \text{sites}(\alpha) \cup \text{sites}(\beta). \tag{2.5}$$

Lemma 6. **(Composition Criterion)** A pair of composites $\langle \alpha, \beta \rangle$ satisfies the composition condition if and only if

$$\text{sites}(\alpha) \cap \text{sites}(\beta) = \text{term}(\alpha) \cap \text{init}(\beta).$$

2.1.4 Site replacement

The operation of site replacement plays two roles:

1. For some disconnected composites, we would like to be able to replace some of their sites, in order to connect them.

2. Intuitively, two composites that differ only in their sites but have the same internal structure, represent the same constructive processes. Thus, a constructive process is uniquely represented by an equivalence class of composites modulo site replacements, rather than by a composite.

Definition 6. For a primitive $\pi = \langle \alpha, I, T \rangle$ and an injective mapping

$$h : \text{sites}(\pi) \to \mathcal{S},$$

called **site replacement**, the primitive $\pi\langle h \rangle$ is defined as

$$\pi\langle h \rangle \overset{\text{def}}{=} \langle \alpha, h(I), h(T) \rangle.^3$$

[3] Here $h(I)$ and $h(T)$ should be understood as linearly ordered sets, where the orderings are induced from those on I and T by the injective mapping h.

▶

Definition 7. For a composite $\gamma = \langle \pi_1, \ldots, \pi_n \rangle$ and an injective mapping

$$h : \mathrm{sites}(\gamma) \to \mathcal{S} \ ,$$

called **site replacement**, the composite $\gamma\langle h \rangle$ is defined as follows:

$$\gamma\langle h \rangle \overset{\mathrm{def}}{=} \langle \pi_1 \langle h\big|_{\mathrm{sites}(\pi_1)} \rangle, \ldots, \pi_n \langle h\big|_{\mathrm{sites}(\pi_n)} \rangle \rangle.$$

▶

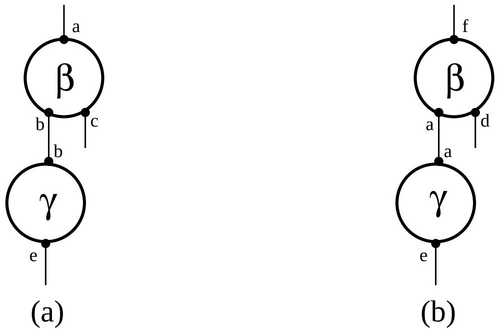

Figure 2.6: Composite (a) and its site replacement (b).

For a composite γ with $\mathrm{sites}(\gamma) = \{s_1, \ldots, s_k\}$, a site replacement $h : \mathrm{sites}(\gamma) \to \mathcal{S}$ is specified by the list of its values as follows (see also Fig. 2.6):

$$s_1 \to h(s_1), \ldots, s_k \to h(s_k).$$

In Fig. 2.6, a composite γ and the result of its site replacement $\gamma\langle h \rangle$ are shown, where the site replacement mapping $h : \mathrm{sites}(\gamma) \to \mathcal{S}$ is defined by

$$a \to f, \ b \to a, \ c \to d, \ e \to e.$$

The following lemmas are straightforward.

Lemma 7. Let γ be a composite, and let $h_1 : \text{sites}(\gamma) \to \mathcal{S}_1$, $h_2 : \text{sites}(\gamma\langle h_1 \rangle) \to \mathcal{S}_2$ be site replacements. Then,

$$(\gamma\langle h_1 \rangle)\langle h_2 \rangle = \gamma\langle h_2 \circ h_1 \rangle.$$

Lemma 8. Let $\gamma' = \gamma\langle h \rangle$. Then there exists a site replacement $h' : \text{sites}(\gamma') \to \mathcal{S}'$ such that $\gamma'\langle h' \rangle = \gamma$.

The operations of composition and site replacement are related to each other:

Lemma 9. If $\langle \alpha, \beta \rangle$ is a pair of composites satisfying the composition condition, and $h : \text{sites}(\alpha \lhd \beta) \to \mathcal{S}$, $h_1 = h\big|_{\text{sites}(\alpha)}$, $h_2 = h\big|_{\text{sites}(\beta)}$ are site replacements, then the pair $\langle \alpha\langle h_1 \rangle, \beta\langle h_2 \rangle \rangle$ also satisfies the composition condition and

$$\alpha\langle h_1 \rangle \lhd \beta\langle h_2 \rangle = (\alpha \lhd \beta)\langle h \rangle. \tag{2.6}$$

For composites α, β, we shall write $\alpha \approx \beta$ if and only if there exists site replacement $h : \text{sites}(\alpha) \to \text{sites}(\beta)$ such that $\beta = \alpha\langle h \rangle$.

From Lemmas 7,8, it follows that relation $\approx$ on a set of composites is an equivalence relation, i.e., it is reflexive, symmetric, and transitive.

2.1.5 Formations

A formation is an equivalence class of composites modulo site replacements. It is a new data type, whose instances correspond uniquely to processes that construct object representations. Formally, it is necessary to have such a data type, in order to be able, for example, to count the *number of processes* that can possibly construct a given object representation. Later on, based on the ETS representation of strings and graphs, we show that they implicitly admit exponentially many constructive processes. A consequence is the intractability of the learning problems on strings and graphs.

Let Π be a set of primitives (its cardinality can be arbitrary), and let Γ_Π be the set of all composites that are composed of primitives from Π. One can think about these sets as corresponding to a particular environment (see Section A for a discussion about environments).

Let also $\approx_\Pi$ be the relation obtained via restriction of $\approx$ to Γ_Π:

$$\approx_\Pi \overset{\text{def}}{=} \{\langle \alpha, \beta \rangle \mid \alpha, \beta \in \Gamma_\Pi \text{ and } \alpha \approx \beta\}.$$

An equivalence class with respect to $\approx_\Pi$ containing composite γ is called a **formation**, denoted $\bar{\gamma}$ or $[\gamma]$. The following is the characteristic property of formations:

$$\bar{\alpha} = \bar{\beta} \;\Leftrightarrow\; \alpha \approx \beta.$$

The set of all formations, i.e., the quotient set $\Gamma_\Pi/_{\approx_\Pi}$, is denoted $\bar{\Gamma}_\Pi$.

A pictorial representation for a formation can be obtained from that for a composite by removing the site labels. Since formations do not have sites, the operations of composition and site replacement cannot be defined for them either. For example, two formations shown in Fig. 2.7a can be "composed" in four different ways (Fig. 2.7b).

For a countable set of primitives Π, the set of composites Γ_Π is countable, as a subset of the set of finite sequences over Π. For the set of formations $\bar{\Gamma}_\Pi$ to be countable, it is only required that the set of *labels* of primitives from Π is countable.

The intuition behind the concept of formation is quite similar to the one behind the concept of *abstract graph* in the theory of graph grammars [Ro97, Section 3.5.2]:

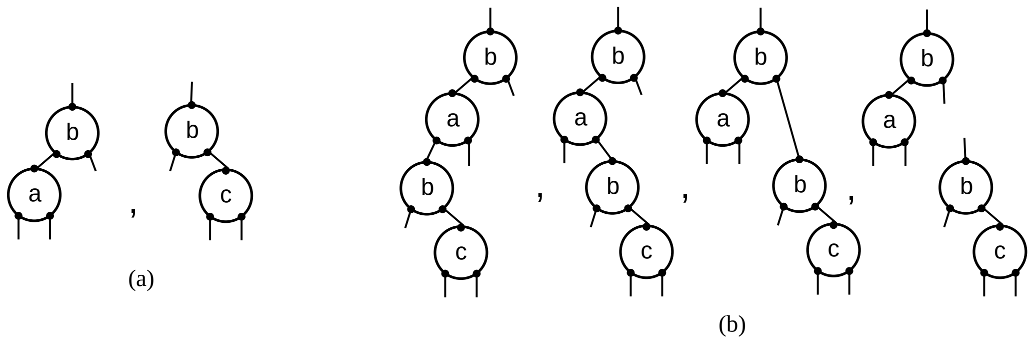

Figure 2.7: Two formations (a) and four ways to compose them (b).

"Almost invariably, two isomorphic graphs are considered as representing the *same* system state. In fact, such a state is determined by the topological structure of the graph ... while the true identity of the nodes is considered just as a representational detail"; likewise, the constructive process is determined by the "topological structure" of a composite, and particular site labels are just a representational detail. Therefore, "it is very natural to look for an equivalence which equates all isomorphic graphs" (an equivalence class w.r.t. it is called an abstract graph). In ETS, such an equivalence is expressed via site replacements; in fact, the latter play the role of standard isomorphisms [Ro97, Def. 3.5.13] between graphs. However, an analogue of Lemma 9 cannot be expressed directly for graphs, since there is no sequential composition operation on them; a similar property called *allowance for sequential composition* is expressed in terms of *abstract derivations* [Ro97, Proposition 3.5.16].

2.1.6 Part/whole relation on composites and formations

The part/whole relation on composites is introduced, in order to obtain the part/whole relation on formations. The latter represents the part/whole relation on the object constructive processes.

Definition 8. A composite α is a **part** of composite β, denoted $\alpha \preceq \beta$, if there exist composites γ_1, γ_2 such that $\beta = \gamma_1 \lhd \alpha \lhd \gamma_2$. The set of all parts of a composite β is denoted Parts(β). ▶

An example of a composite representing a constructive history for the string "babaa" and of a part of this composite is shown in Fig. 2.8. This part represents a constructive process, which produces two subsequences of the original string, "b" and "ba".

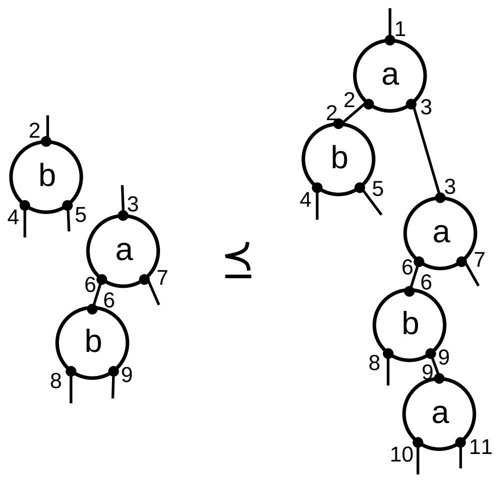

Figure 2.8: A part of a composite representing a constructive history for string "babaa".

It follows from the definition that the properties of a partial ordering relation hold for $\preceq$, i.e. it is reflexive, antisymmetric and transitive.[4]

The following lemma relates the part/whole relation with site replacements:

Lemma 10. If $\alpha \preceq \beta$ and $h : \text{sites}(\beta) \to \mathcal{S}$ is a site replacement, then

$$\alpha\langle h|_{\text{sites}(\alpha)}\rangle \preceq \beta\langle h\rangle.$$

[4]Intuitively, a relation named *part*/whole should be a *part*ial ordering.

Proof. Since $\alpha \preceq \beta$, $\beta = \gamma_1 \lhd \alpha \lhd \gamma_2$ for some composites γ_1, γ_2. Then, by Lemma 9,

$$\gamma_1 \langle h\big|_{\text{sites}(\gamma_1)} \rangle \lhd \alpha \langle h\big|_{\text{sites}(\alpha)} \rangle \lhd \gamma_2 \langle h\big|_{\text{sites}(\gamma_2)} \rangle = \beta\langle h \rangle,$$

hence $\alpha\langle h\big|_{\text{sites}(\alpha)}\rangle \preceq \beta\langle h\rangle$. ■

Definition 9. A composite α is called an **ancestor** of composite β, if there exists composite γ such that

$$\beta = \alpha \lhd \gamma.$$

The set of all ancestors of a composite β is denoted $\text{Anc}(\beta)$. ▶

Obviously, for any composite β, $\text{Anc}(\beta) \subseteq \text{Parts}(\beta)$. Both sets are finite, since they are contained in the set of all subsequences of the finite sequence β.

Now we extend the concepts of a part and of an ancestor to formations.

Definition 10. A formation $\bar{\alpha} \in \bar{\Gamma}_\Pi$ is called a **part** (**ancestor**) of a formation $\bar{\beta} \in \bar{\Gamma}_\Pi$, if there exist composites $\alpha \in \bar{\alpha}$ and $\beta \in \bar{\beta}$ such that α is a part (ancestor) of β. We use the same notation, $\bar{\alpha} \preceq \bar{\beta}$, $\text{Anc}(\bar{\beta})$, and $\text{Parts}(\bar{\beta})$ for formations. Relation $\preceq$ on $\bar{\Gamma}_\Pi$ is called the **part/whole relation on formations**. ▶

Lemma 11. For any set of primitives Π, the part/whole relation on formations is a partial ordering.

Proof.

- *Reflexivity* of $\preceq$ on $\bar{\Gamma}_\Pi$ directly follows from the reflexivity of $\preceq$ on composites.
- *Antisymmetry* of $\preceq$ on $\bar{\Gamma}_\Pi$. Let $\bar{\alpha}, \bar{\beta}$ be two formations such that $\bar{\alpha} \preceq \bar{\beta}$ and $\bar{\beta} \preceq \bar{\alpha}$. Then, by definition of $\preceq$ on $\bar{\Gamma}_\Pi$, there exist composites $\alpha_1, \alpha_2 \in \bar{\alpha}$, $\beta_1, \beta_2 \in \bar{\beta}$ such that $\alpha_1 \preceq \beta_1$ and $\beta_2 \preceq \alpha_2$. Since composites α_1, α_2 belong to the

same formation, there exists site replacement g such that $\alpha_1 = \alpha_2\langle g\rangle$. Similarly, there exists site replacement h such that $\beta_2 = \beta_1\langle h\rangle$. Thus, by Lemma 10,

$$\alpha_2\langle g\rangle\langle h\big|_{\text{sites}(\alpha_1)}\rangle \preceq \beta_2.$$

By transitivity of $\preceq$ on composites, we have

$$\alpha_2\langle h\big|_{\text{sites}(\alpha_1)} \circ g\rangle \preceq \alpha_2,$$

which, by definition of site replacement, implies that

$$\alpha_2\langle h\big|_{\text{sites}(\alpha_1)} \circ g\rangle = \alpha_2.$$

By antisymmetry of $\preceq$ on composites, $\alpha_2 = \beta_2$, which implies $\bar{\alpha} = \bar{\beta}$.

- *Transitivity* of $\preceq$ on $\bar{\Gamma}_\Pi$. Let $\bar{\alpha}, \bar{\beta}, \bar{\gamma}$ be three formations such that $\bar{\alpha} \preceq \bar{\beta}$ and $\bar{\beta} \preceq \bar{\gamma}$. Then, by definition of $\preceq$ on $\bar{\Gamma}_\Pi$, there exist $\alpha \in \bar{\alpha}$, $\beta_1, \beta_2 \in \bar{\beta}$, $\gamma \in \bar{\gamma}$ such that $\alpha \preceq \beta_1$ and $\beta_2 \preceq \gamma$. Since composites β_1 and β_2 belong to the same formation, there exists site replacement h such that $\beta_2 = \beta_1\langle h\rangle$. Hence, by Lemma 10, $\alpha\langle h\big|_{\text{sites}(\alpha)}\rangle \preceq \beta_2$. By transitivity of $\preceq$ on composites, $\alpha\langle h\big|_{\text{sites}(\alpha)}\rangle \preceq \gamma$, hence $\bar{\alpha} \preceq \bar{\gamma}$.

■

Lemma 12. For every formation $\bar{\gamma} \in \bar{\Gamma}_\Pi$, the set $\text{Parts}(\bar{\gamma})$ is finite.

Proof. Suppose, $\text{Parts}(\bar{\gamma})$ is infinite. Fix a composite $\gamma_0 \in \bar{\gamma}$. By Def. 10, for each $\bar{\alpha} \in \text{Parts}(\bar{\gamma})$, there exist composites $\alpha \in \bar{\alpha}$ and $\gamma \in \bar{\gamma}$ such that $\alpha \in \text{Parts}(\gamma)$. Since both γ and γ_0 belong to the same formation, $\gamma_0 = \gamma\langle h\rangle$ for some site replacement h. Then, by Lemma 10, $\alpha_0 \overset{\text{def}}{=} \alpha\langle h\big|_{\text{sites}(\alpha)}\rangle \preceq \gamma_0$. Since formations are equivalence classes of composites, all α_0 corresponding to distinct $\bar{\alpha} \in \text{Parts}(\bar{\gamma})$ are distinct. However, the set of parts for any composite is finite, contradiction! ■

2.1.7 Specification of a set of primitives

Consider the practical problem of specification of an infinite set of primitives. This has to be done before we can consider composites or formations. Arbitrary infinite sets of primitives cannot be specified by finite descriptions (since the set of all possible finite descriptions is countable, and the collection of all possible infinite sets of primitives is not). For our purposes, it suffices to consider only sets of primitives closed under site replacements:

Definition 11. A set of primitives Π is **closed under site replacements**, if for all $\pi \in \Pi$, if σ is a primitive such that $\mathrm{sites}(\sigma) \subset \cup_{\pi\in\Pi} \mathrm{sites}(\pi)$ and $\sigma \approx \pi$, then $\sigma \in \Pi$. ▶

Two primitives π and σ are called **equivalent**, if for the composites $\langle\pi\rangle$ and $\langle\sigma\rangle$, we have $\langle\pi\rangle \approx \langle\sigma\rangle$. In other words, π and σ are equivalent, if their labels coincide, $|\,\mathrm{init}(\pi)| = |\,\mathrm{init}(\sigma)|$, and $|\,\mathrm{term}(\pi)| = |\,\mathrm{term}(\sigma)|$. It is easy to see that equivalence classes of primitives are primitive types (see Def. 51).

To specify a set of primitives Π closed under site replacements, it is sufficient to fix a set of sites $\mathcal{S}$ and a set of primitive types $\bar{\Pi}$. Each primitive type can be either specified as in Def. 51 by a label and two integers, or by choosing any primitive that belongs to it, called the **specifying primitive**. In what follows, we specify sets of primitives in the latter way.

To summarize this section:

- primitives and a composites have been defined;
- a countable set of primitives Π induces countable sets of composites Γ_Π and a countable set of formations $\bar{\Gamma}_\Pi$;

- the associative operation of composition and the operation of site replacement on composites have been defined;
- a formation is an equivalence class of composites with respect to site replacements; a formation is a formal concept that corresponds to the process that constructs an object representation;
- parts and ancestors of composites and formations have been defined;
- the part/whole relation on formations is a partial ordering; every composite and every formation has finitely many parts.

2.2 Representation of objects

Composites and formations have been defined in order to describe formally the constructive processes. Now, thinking about an object representation as a result of a constructive process, we define the corresponding formal concept. Rather than defining this result directly, we introduce the *semantic equivalence* relation on formations; the equivalence classes modulo this relation are called *structs* and correspond to the results of the constructive processes.

The definition of semantic equivalence is based on the following postulate: sequential composition of constructive processes p_1 and p_2 is semantically equivalent to (i.e., produces the same result as) the sequential composition of the processes p_1' and p_2', if p_1 is equivalent to p_1' and p_2 is equivalent to p_2'. In other words, the semantic equivalence relation is a congruence with respect to composition. This postulate implies that the semantic equivalence relation is naturally induced, or generated, by a fixed set of equivalent formation pairs called *semantic identities*. An equivalence class of formations, called a *struct*, is the formal concept corresponding to a single object representation.

It should be noted that our definition of object representation as an equivalence class of representations of constructive processes is conceptually similar to the definition of a quotient group (or ring) w.r.t. a congruence induced by a system of identities, which correspond to our semantic identities (see, for example, [Ku65]). A well known theorem states that any group (ring) can be defined in this way. One has to keep in mind that, when objects are defined via factorization w.r.t. a congruence, the problem of equality of these objects may become undecidable (this problem is known as the word problem). We will address this issue by imposing the requirement of finiteness on the corresponding equivalence classes and study the consequences of this requirement.

A fixed set of primitives and a set of semantic identities can be considered as a complete formal specification of the basic representational level. The following section is devoted to the general properties and examples of such specifications.

Throughout this section, we assume that a set of primitives Π is fixed and closed under site replacements, and that all composites and formations belong to the sets Γ_Π and $\bar{\Gamma}_\Pi$, respectively.

2.2.1 Formation tuples

Recall that a formation corresponds to a single constructive process. A formation tuple corresponds to a collection of dependent, or interrelated, constructive processes. The processes in the collection may be related to each other in various ways. For example, they can be parts of a larger process, or they can be semantically equivalent processes. To specify this relation, we first define the collections of process instances as tuples of composites, where the relation between the composites in a tuple is uniquely specified by their shared sites, and then consider the equivalence classes of composite tuples modulo site replacements. The latter are called formation tuples.

Formation 2-tuples are used to define the semantic equivalence relation. Later on, *composable formation tuples* are involved in the definition of higher-level representations.

The concept of site replacement and the relation $\approx$ can be extended from composites to tuples of composites as follows:

Definition 12. For an n-tuple of composites $C = \langle \gamma_1, \ldots, \gamma_n \rangle$, let the following set

$$\mathrm{sites}(\langle \gamma_1, \ldots, \gamma_n \rangle) \overset{\mathrm{def}}{=} \bigcup_{i=1}^{n} \mathrm{sites}(\gamma_i)$$

be called the **set of sites**. An injective mapping

$$h : \mathrm{sites}(\langle \gamma_1, \ldots, \gamma_n \rangle) \to \mathcal{S}$$

is called an **n-tuple site replacement**. The set

$$[C] \overset{\mathrm{def}}{=} [\gamma_1, \ldots, \gamma_n] \overset{\mathrm{def}}{=} \{\langle \gamma_1 \langle h\big|_{\mathrm{sites}(\gamma_1)} \rangle, \ldots, \gamma_n \langle h\big|_{\mathrm{sites}(\gamma_1)} \rangle \rangle \mid h \text{ is an } n\text{-tuple site replacement}\}$$

is called a **formation n-tuple**. ▶

There is a difference between a formation tuple $[\gamma_1, \ldots, \gamma_n]$ and an n-tuple of formations $\langle [\gamma_1], \ldots, [\gamma_n] \rangle$: If composites γ_1 and γ_2 have a common site s, then so do composites γ'_1 and γ'_2 for all composite tuples $\langle \gamma'_1, \ldots, \gamma'_n \rangle \in [\gamma_1, \ldots, \gamma_n]$, whereas this is not generally the case for arbitrary composites $\gamma''_1 \in [\gamma_1]$ and $\gamma''_2 \in [\gamma_2]$. For $n = 1$, the concept of a formation 1-tuple coincides with that of formation.

Next, we define the concept of projection, first for a tuple of composites, and then for a formation tuple.

Definition 13. Let $C = \langle \gamma_1, \ldots, \gamma_n \rangle$ be an n-tuple of composites, and let $\mathbf{i} = \langle i_1, \ldots, i_k \rangle$ be a subsequence of $\langle 1, 2, \ldots, n \rangle$.

The **projection of composite tuple** C onto $\mathbf{i}$ is defined as the following k-tuple of composites:

$$C_{\mathbf{i}} \overset{\mathrm{def}}{=} \langle \gamma_{i_1}, \ldots, \gamma_{i_k} \rangle.$$

▶

Definition 14. Let $D = [\gamma_1, \dots, \gamma_n]$ be a formation n-tuple, and let $\mathbf{i} = \langle i_1, \dots, i_k \rangle$ be a subsequence of $\langle 1, 2, \dots, n \rangle$.

The **projection of formation tuple** D onto $\mathbf{i}$ is defined as the set of projections of all composite n-tuples from D onto $\mathbf{i}$:

$$D_{\mathbf{i}} \overset{\text{def}}{=} \{C_{\mathbf{i}} \mid C \in D\}.$$

▶

The following lemma states that any projection of a formation tuple is a formation tuple itself.

Lemma 13. **(Formation Projection)** For any formation n-tuple $D = [\gamma_1, \dots, \gamma_n]$ and any non-empty subsequence $\mathbf{i} = \langle i_1, \dots, i_k \rangle$ of $\langle 1, 2, \dots, n \rangle$, the projection of D onto $\mathbf{i}$, $D_{\mathbf{i}}$, is a formation k-tuple.

Proof. Let $C, C' \in D_{\mathbf{i}}$. Then there exist composite tuples $\tilde{C}, \tilde{C}' \in D$ such that $\tilde{C}_{\mathbf{i}} = C$, $\tilde{C}'_{\mathbf{i}} = C'$. Since $\tilde{C}, \tilde{C}' \in D$, there exists a site replacement $h : \text{sites}(\tilde{C}) \to \text{sites}(\tilde{C}')$ such that $\tilde{C}' = \tilde{C}\langle h \rangle$. Then

$$C' = C\langle h\big|_{\text{sites}(C)} \rangle,$$

so $[C] = [C']$, hence $D_{\mathbf{i}} \subseteq [C]$.

Vice versa, if $C \in D_{\mathbf{i}}$ and $C' = C\langle h \rangle$, then take $\tilde{C} \in D$ such that $C = \tilde{C}_{\mathbf{i}}$ and consider $\tilde{h} : \text{sites}(\tilde{C}) \to \mathcal{S}$ such that $\tilde{h}\big|_{\text{sites}(C)} = h$. Then $C' = \tilde{C}\langle \tilde{h} \rangle_{\mathbf{i}}$, hence $C' \in D_{\mathbf{i}}$ and $[C] \subseteq D_{\mathbf{i}}$.

Altogether, $D_{\mathbf{i}} = [C]$. ■

Unlike vectors, formation tuples are not uniquely specified by their single-component projections. For example, there exist 26 different formation tuples (Fig. 2.9a) having

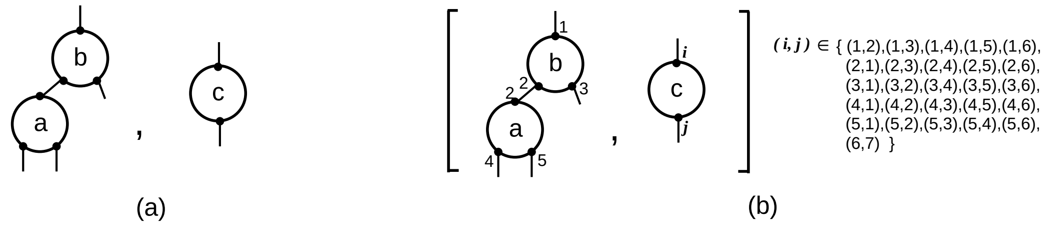

Figure 2.9: Two formations (a), which are single-component projections of 26 different formation pairs (b).

projections shown in Fig. 2.9b. However, the number of formation tuples having a given set of projections is always finite:

Theorem 1. *For any formations $\bar{\alpha}_1, \dots, \bar{\alpha}_n$, the following set of formation n-tuples*

$$S = \{[\alpha_1, \dots, \alpha_n] \mid \alpha_1 \in \bar{\alpha}_1, \dots, \alpha_n \in \bar{\alpha}_n\}$$

is finite.

Proof. The proof is by induction on n. For $n = 1$, S consists of one element, $[\alpha_1]$. Assume that the set

$$S_{n-1} = \{[\alpha_1, \dots, \alpha_{n-1}] \mid \alpha_1 \in \bar{\alpha}_1, \dots, \alpha_{n-1} \in \bar{\alpha}_{n-1}\}$$

is finite. For each formation tuple $D_{n-1} \in S_{n-1}$, select one $(n-1)$-tuple of composites $C^0_{n-1} \in D_{n-1}$, and let $\mathbf{C}^0_{n-1}$ be the set of these selected tuples. Then, by Formation Projection Lemma,

$$S_n = \{[C^0_{n-1}, \alpha_n] \mid C^0_{n-1} \in \mathbf{C}^0_{n-1},\ \alpha_n \in \bar{\alpha}_n\} = \bigcup_{C^0_{n-1} \in \mathbf{C}^0_{n-1}} \{[C^0_{n-1}, \alpha_n] \mid \alpha_n \in \bar{\alpha}_n\}.$$

The set $\mathbf{C}^0_{n-1}$ finite, hence it remains to show that each of the sets $\{[C^0_{n-1}, \alpha_n] \mid \alpha_n \in \bar{\alpha}_n\}$ is finite. If $\bar{\alpha}_n$ is finite, which can only be the case when the set of primitives

Π is finite, since Π is closed under site replacements, then $\{[C^0_{n-1}, \alpha_n] \mid \alpha_n \in \bar{\alpha}_n\}$ is obviously finite as well. Otherwise, Π is infinite and, since it is closed under site replacements, there exists $\alpha^0_n \in \bar{\alpha}_n$ such that

$$\text{sites}(\alpha^0_n) \cap \text{sites}(C^0_{n-1}) = \varnothing.$$

Let $\mathcal{S} = \text{sites}(\alpha^0_n) \cup \text{sites}(C^0_{n-1})$. Then,

$$\{[C^0_{n-1}, \alpha_n] \mid \alpha_n \in \bar{\alpha}_n\} = \{[C^0_{n-1}, \alpha_n] \mid \alpha_n \in \bar{\alpha}_n,\ \text{sites}(\alpha_n) \subset \mathcal{S}\},$$

and the latter set is finite. ■

2.2.2 Direct convertibility and equivalence

As mentioned above, the postulated requirement for the semantic equivalence relation to be a congruence with respect to composition implies that this relation can be induced, or generated, by a fixed set of formation pairs. Here we define a relation induced by a single formation pair, applying the idea of rewriting (see, for example, [Bo93, Ro97]). The direct convertibility relation defined here corresponds to the single-step reduction relation from the rewriting theory. As before, it will be first defined for composites, and then extended to formations. An equivalence relation on formations is obtained as the reflexive and transitive closure of the direct convertibility relation.

Definition 15. Composites α and β are called **directly convertible** via a composite pair c, denoted $\alpha \overset{c}{\leftrightarrow} \beta$, if there exist composites γ_1, γ_2 such that

$$\begin{aligned} \alpha &= \gamma_1 \lhd \gamma \lhd \gamma_2 \\ \beta &= \gamma_1 \lhd \gamma' \lhd \gamma_2, \end{aligned}$$

where either $c = \langle \gamma, \gamma' \rangle$, or $c = \langle \gamma', \gamma \rangle$. ►

An example of directly convertible composites, each of which represents a constructive process for a string, is shown in Fig. 2.10.

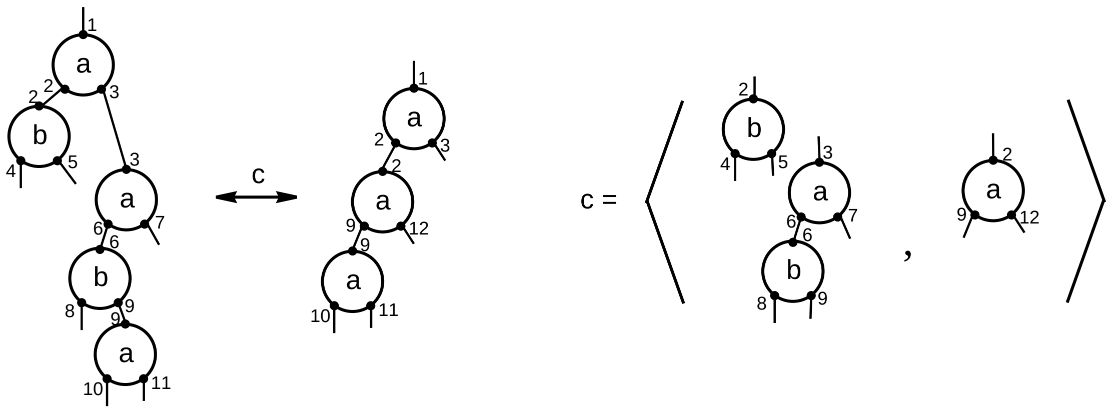

Figure 2.10: Directly convertible composites.

Obviously, relation $\overset{c}{\leftrightarrow}$ is symmetric.

The extension of the direct convertibility relation to formations is straightforward:

Definition 16. For a formation pair $\bar{c}$ and two formations $\bar{\alpha}$, $\bar{\beta}$, we say that $\bar{\alpha}$ is **directly convertible to $\bar{\beta}$ with respect to $\bar{c}$** and write $\bar{\alpha} \overset{\bar{c}}{\leftrightarrow} \bar{\beta}$, if there exist composites $\alpha \in \bar{\alpha}$, $\beta \in \bar{\beta}$, and a composite pair $c \in \bar{c}$ such that $\alpha \overset{c}{\leftrightarrow} \beta$. ▶

The following lemma states that direct convertibility of two formations implies a stronger statement: namely, in the above definition, one of the composites $\alpha \in \bar{\alpha}$, $\beta \in \bar{\beta}$, or the composite pair $c \in \bar{c}$ can be chosen arbitrarily.

Lemma 14. If $\bar{\alpha} \overset{\bar{c}}{\leftrightarrow} \bar{\beta}$, then

1. For all $c \in \bar{c}$, there exist $\alpha \in \bar{\alpha}$, $\beta \in \bar{\beta}$ such that $\alpha \overset{c}{\leftrightarrow} \beta$.

2. For all $\alpha \in \bar{\alpha}$, there exist $\beta \in \bar{\beta}$, $c \in \bar{c}$ such that $\alpha \overset{c}{\leftrightarrow} \beta$.

Proof. According to the definition of $\bar{\alpha} \overset{\bar{c}}{\leftrightarrow} \bar{\beta}$, there exist $\alpha_0 \in \bar{\alpha}$, $\beta_0 \in \bar{\beta}$, $c_0 \in \bar{c}$ such that $\alpha_0 \overset{c_0}{\leftrightarrow} \beta_0$.

Depending on the case of the lemma statement, find a site replacement h such that

1. $h : \text{sites}(c_0) \longrightarrow \text{sites}(c)$, $c_0\langle h\rangle = c$.
2. $h : \text{sites}(\alpha_0) \longrightarrow \text{sites}(\alpha)$, $\alpha_0\langle h\rangle = \alpha$.

Extend h to an injective mapping $f : \text{sites}(\alpha_0) \cup \text{sites}(\beta_0) \longrightarrow \mathcal{S}$ (note that $\text{sites}(c_0) \subset \text{sites}(\alpha_0) \cup \text{sites}(\beta_0)$. Depending on the case of the lemma statement, take

1. $\alpha = \alpha_0\langle f\big|_{\text{sites}(\alpha_0)}\rangle$, $\beta = \beta_0\langle f\big|_{\text{sites}(\beta_0)}\rangle$.
2. $\beta = \beta_0\langle f\big|_{\text{sites}(\beta_0)}\rangle$, $c = c_0\langle f\big|_{\text{sites}(c_0)}\rangle$.

Then it follows from the definition of $\alpha_0 \overset{c_0}{\leftrightarrow} \beta_0$ and Lemma 9 that $\alpha \overset{c}{\leftrightarrow} \beta$. ■

Definition 17. For a set of formation pairs $\mathcal{I}$, define the direct convertibility relation on formations as follows:

$$\bar{\alpha} \overset{\mathcal{I}}{\leftrightarrow} \bar{\beta} \quad \Longleftrightarrow \quad \exists\, \bar{c} \in \mathcal{I} \;\; \bar{\alpha} \overset{\bar{c}}{\leftrightarrow} \bar{\beta}.$$

The **equivalence relation** induced by $\mathcal{I}$, denoted $\sim_{\mathcal{I}}$, is defined as the reflexive and transitive closure of $\overset{\mathcal{I}}{\leftrightarrow}$, i.e., a minimal equivalence relation that contains $\overset{\mathcal{I}}{\leftrightarrow}$. ▶

2.2.3 Equivalence on formation tuples

The equivalence relation on formation tuples is a straightforward extension of that on formations.

It will be used in the formulate the statement about the congruence of the equivalence relation on formations with respect to composition. The equivalence classes with respect to this relation, called struct tuples, plays the role of the main technical concept in the definitions of higher representational levels.

Definition 18. Let $c = \langle \gamma, \gamma' \rangle$ be a pair of composites. Composite n-tuples C and C' will be called **directly convertible via** c, denoted $C \overset{c}{\leftrightarrow} C'$, if there exists $i \in \{1, \ldots, n\}$ such that

$$C_i \overset{c}{\leftrightarrow} C'_i \quad \text{and} \quad C_{\{1,\ldots,n\}\setminus\{i\}} = C'_{\{1,\ldots,n\}\setminus\{i\}}.$$

For a formation pair $\bar{c}$ and two formation n-tuples $\bar{C}$, $\bar{C}'$, we say that $\bar{C}$ is **directly convertible to $\bar{C}'$ with respect to** $\bar{c}$ and write $\bar{C} \overset{\bar{c}}{\leftrightarrow} \bar{C}'$, if there exist composite tuples $C \in \bar{C}$, $C' \in \bar{C}'$, and $c \in \bar{c}$ such that $C \overset{c}{\leftrightarrow} C'$.

For a set of formation pairs $\mathcal{I}$, define the direct convertibility relation on formation n-tuples as follows:

$$\bar{C} \overset{\mathcal{I}}{\leftrightarrow} \bar{C}' \iff \exists\, \bar{c} \in \mathcal{I} \;\; \bar{C} \overset{\bar{c}}{\leftrightarrow} \bar{C}'.$$

The **equivalence relation** induced by $\mathcal{I}$, denoted $\sim_{\mathcal{I}}$, is defined as the reflexive and transitive closure of $\overset{\mathcal{I}}{\leftrightarrow}$, i.e., the minimal equivalence relation that contains $\overset{\mathcal{I}}{\leftrightarrow}$. ▶

The following generalization of Lemma 14 to formation tuples is obvious:

Lemma 15. If $\bar{C} \overset{\bar{c}}{\leftrightarrow} \bar{C}'$, then

1. For all $c \in \bar{c}$, there exist $C \in \bar{C}$ and $C' \in \bar{C}'$ such that $C \overset{c}{\leftrightarrow} C'$.

2. For all $C \in \bar{C}$, there exist $C' \in \bar{C}'$ and $c \in \bar{c}$ such that $C \overset{c}{\leftrightarrow} C'$.

2.2.4 Semantic identities and semantic equivalence

Here we formulate a condition on formation pairs in $\mathcal{I}$, necessary and sufficient for the equivalence relation $\sim_{\mathcal{I}}$ to be a congruence with respect to composition. We also prove that congruence implies compatibility of equivalence relation with the part/whole relation.

Consider a composite pair $c = \langle \gamma, \gamma' \rangle$ and the corresponding direct convertibility relation $\overset{c}{\leftrightarrow}$. The statement that $\overset{c}{\leftrightarrow}$ is a congruence with respect to composition on composites requires that if α and β satisfy the composition condition and, say, $\alpha \overset{c}{\leftrightarrow} \alpha'$, then α' and β must satisfy the composition condition as well, and $(\alpha \lhd \beta) \overset{c}{\leftrightarrow} (\alpha' \lhd \beta)$. Similarly, if $\beta \overset{c}{\leftrightarrow} \beta'$, then $(\alpha \lhd \beta) \overset{c}{\leftrightarrow} (\alpha \lhd \beta')$.

The composition criterion (Lemma 6)

$$\text{sites}(\alpha) \cap \text{sites}(\beta) = \text{term}(\alpha) \cap \text{init}(\beta)$$

implies that the following requirement

$$\text{init}(\gamma) = \text{init}(\gamma'),\ \text{term}(\gamma) = \text{term}(\gamma'),\ \text{sites}(\gamma) = \text{sites}(\gamma') \qquad (*)$$

is sufficient, in order for $\overset{c}{\leftrightarrow}$ to be a congruence. Indeed, if equalities $(*)$ hold for $c = \langle \gamma, \gamma' \rangle$ and $\alpha \overset{c}{\leftrightarrow} \alpha'$ or $\beta \overset{c}{\leftrightarrow} \beta'$, then they also hold for composite pairs $\langle \alpha, \alpha' \rangle$ and $\langle \beta, \beta' \rangle$, hence the composition condition still holds for $\langle \alpha', \beta \rangle$ and $\langle \alpha, \beta' \rangle$. Vice versa, assume that one of the above equalities does not hold for $\langle \gamma, \gamma' \rangle$. Then we can choose $\alpha, \beta, \alpha', \beta'$ to show that the congruence property is violated as follows. Consider two special composites, δ_s and δ^s shown in Fig. 2.11.

Due to the symmetry of the relation $\overset{c}{\leftrightarrow}$, without loss of generality, we can assume that one of the following cases applies:

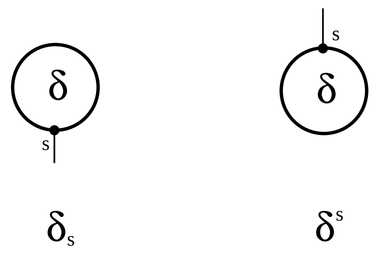

Figure 2.11: Composites δ_s and δ^s.

1. There exists $s \in \text{init}(\gamma) \setminus \text{init}(\gamma')$, $s \in \text{sites}(\gamma')$. Then, if $\beta \overset{c}{\leftrightarrow} \beta'$, we obtain that $\delta_s \dashv \beta$ but $\delta_s \not\dashv \beta'$.

2. There exists $s \in \text{init}(\gamma) \setminus \text{init}(\gamma')$, $s \notin \text{sites}(\gamma')$. Then, if $\alpha \overset{c}{\leftrightarrow} \alpha'$, we obtain that $\alpha' \dashv \delta^s$ but $\alpha \not\dashv \delta^s$.

3. The case when $\text{term}(\gamma) \neq \text{term}(\gamma')$ is symmetric to the case of $\text{init}(\gamma) \neq \text{init}(\gamma')$ considered above.

4. $\text{init}(\gamma) = \text{init}(\gamma')$, $\text{term}(\gamma) = \text{term}(\gamma')$, $s \in \text{sites}(\gamma) \setminus \text{sites}(\gamma')$. Then, if $\beta \overset{c}{\leftrightarrow} \beta'$, due to $s \notin \text{init}(\gamma)$, we have $\delta_s \not\dashv \beta$ but $\delta_s \dashv \beta'$.

In all four cases, congruence with respect to composition is violated.

Conditions $(*)$ are too restricting, because in most cases they mean that composites γ and γ' have to consist of the same primitives, perhaps arranged in different orders, or, at least, that $\alpha \overset{c}{\leftrightarrow} \alpha'$ implies that α and α' must have the same number of sites. Intuitively, one would like to give more flexibility in defining the equivalence relation and, in particular, to allow composite pairs like the ones shown in Fig. 2.12. And indeed, we will show that, in order for the equivalence relation *on formations* to be a congruence, it is sufficient to impose the first two conditions only:

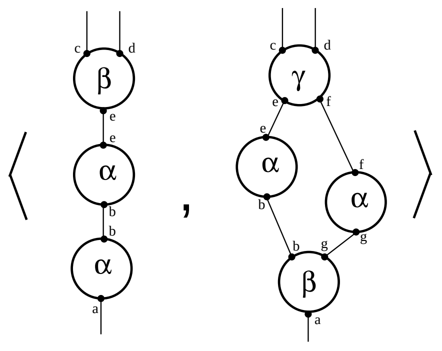

Figure 2.12: A pair of composites that violates the condition sites(γ) = sites(γ').

init(γ) = init(γ'), term(γ) = term(γ').

First, we need a definition of composition on formation pairs:

Definition 19. A formation pair $\bar{c}$ satisfies the **composition condition**, if there exists a composable composite pair $\langle \alpha, \beta \rangle \in \bar{c}$. Then the composition of $\bar{c}$ is the formation $[\alpha \lhd \beta]$. ▶

It follows from Lemma 9 that if a formation $\bar{c}$ satisfies the composition condition, then *every* composite pair $\langle \alpha, \beta \rangle \in \bar{c}$ satisfies the composition condition, and the composition $[\alpha \lhd \beta]$ does not depend on the choice of $\langle \alpha, \beta \rangle \in \bar{c}$.

Next, we would like to relax the formulation of congruence with respect to composition, by requiring it to hold for formation pairs that satisfy the composition condition only. That is, relation $\sim_{\mathcal{I}}$ is congruent with respect to composition, if for all formation pairs $[\alpha, \beta]$, $[\alpha', \beta']$ satisfying the composition condition, $[\alpha, \beta] \sim_{\mathcal{I}} [\alpha', \beta']$ implies $[\alpha \lhd \beta] \sim_{\mathcal{I}} [\alpha' \lhd \beta']$.

Before we impose conditions init(γ) = init(γ'), term(γ) = term(γ') on formation pairs $[\gamma, \gamma'] \in \mathcal{I}$, we would like to explain why, in general, these conditions are necessary.[5] In Fig. 2.13, we show an example of a typical situation when congruence of $\sim_{\mathcal{I}}$ does not hold, because one of the conditions (namely, $\mathrm{term}(\gamma) = \mathrm{term}(\gamma'))$, is violated. Indeed, for composites β and δ shown in this figure, we have $[\beta] \sim_{\mathcal{I}} [\delta]$, since one can transform β to δ by applying semantic identities from $\mathcal{I}$ as shown at the bottom of the figure, but we cannot apply any identity to $\alpha \lhd \beta$ that would disconnect π_c from π_a, thus $[\alpha \lhd \beta] \not\sim_{\mathcal{I}} [\alpha \lhd \delta]$. An example, in which violation of condition $\mathrm{init}(\gamma) \neq \mathrm{init}(\gamma')$ leads to non-congruence of $\sim_{\mathcal{I}}$ can be constructed symmetrically.

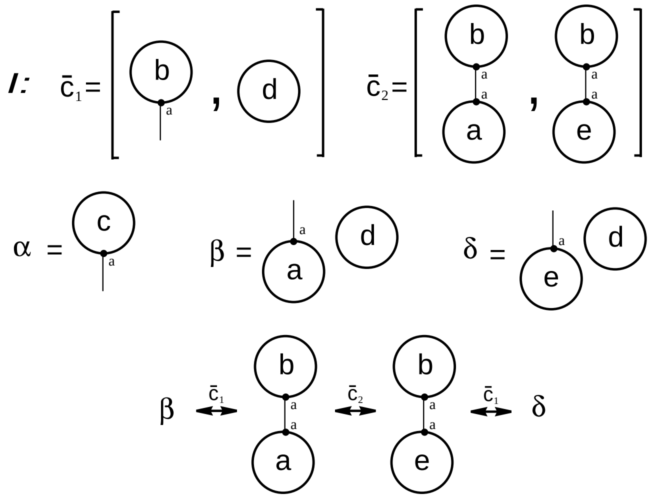

Figure 2.13: A set of formation pairs $\mathcal{I} = \{\bar{c}_1, \bar{c}_2\}$, for which condition $\mathrm{term}(\gamma) = \mathrm{term}(\gamma')$ is violated and equivalence relation $\sim_{\mathcal{I}}$ is not a congruence with respect to concatenation.

Definition 20. A formation pair $\bar{c}$ is called a **semantic identity**, if for every composite pair $\langle \gamma, \gamma' \rangle \in \bar{c}$,

$$\begin{aligned} \mathrm{init}(\gamma) &= \mathrm{init}(\gamma') \\ \mathrm{term}(\gamma) &= \mathrm{term}(\gamma'). \end{aligned}$$

▶

[5]In fact, for some particular sets of primitives, the conditions may be relaxed even further, but we shall leave it beyond the scope of the present thesis.

Note that the above conditions are invariant with respect to site replacements, hence, if $[\gamma, \gamma']$ contains at least one composite pair satisfying these conditions, then it is a semantic identity.

The set of semantic identities for strings is shown in Fig. 2.14. The family of

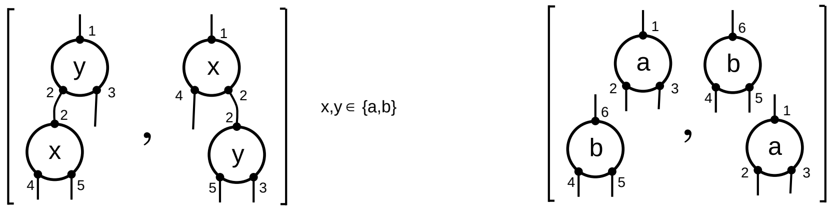

Figure 2.14: Semantic identities for strings.

identities on the left means that insertion of character x, and then of character y after x, into a string has the same result as insertion of character y first, and then of character x before y. The identity on the right allows the permutation of disconnected characters.

Lemma 16. If $\bar{c}$ is a semantic identity and $\alpha \overset{c}{\leftrightarrow} \beta$ for some $c \in \bar{c}$, then

$$\begin{aligned} \text{init}(\alpha) &= \text{init}(\beta) \\ \text{term}(\alpha) &= \text{term}(\beta), \end{aligned}$$

i.e., $[\alpha, \beta]$ is a semantic identity.

Proof. The proof directly follows from the definition of semantic identity and Lemma 5.
■

From now on, we will assume that the equivalence relation $\sim_{\mathcal{I}}$ is generated by a set $\mathcal{I}$ of semantic identities. As it will be shown in the remaining part of this subsection, this implies congruence of $\sim_{\mathcal{I}}$ with respect to concatenation and compatibility with the part/whole relation.

First, we need three auxiliary definitions and a lemma.

Definition 21. For a composite α, define the set of **external sites** as

$$\operatorname{ext}(\alpha) \overset{\text{def}}{=} \operatorname{init}(\alpha) \cup \operatorname{term}(\alpha),$$

and the set of **internal sites** as

$$\operatorname{int}(\alpha) \overset{\text{def}}{=} \operatorname{sites}(\alpha) \setminus \operatorname{ext}(\alpha).$$

►

Definition 22. A pair of composites $\langle \alpha, \beta \rangle$ satisfies the **weak composition condition**, if

$$\operatorname{ext}(\alpha) \cap \operatorname{ext}(\beta) = \operatorname{term}(\alpha) \cap \operatorname{init}(\beta).$$

►

If a pair of composites only satisfies the weak composition condition, but violates the (ordinary) composition condition, their composition is undefined. Moreover, there seems to be no meaningful relaxation of the composition condition for this case. However, for the corresponding formation pair $[\alpha, \beta]$, one can define the result of composition. Note that if a pair of composites $\langle \alpha, \beta \rangle \in [\alpha, \beta]$ satisfies the weak composition condition, then so does every other pair $\langle \alpha', \beta' \rangle \in [\alpha, \beta]$, so it is correct to say that a formation pair $[\alpha, \beta]$ satisfies the weak composition condition.

Definition 23. Let $[\alpha, \beta]$ be a formation pair satisfying the weak composition condition. Take any two site replacements $f : \operatorname{sites}(\alpha) \to \mathcal{S}$ and $g : \operatorname{sites}(\beta) \to \mathcal{S}$ such that

$$f|_{\operatorname{ext}(\alpha)} = \mathrm{id}, \qquad g|_{\operatorname{ext}(\beta)} = \mathrm{id}$$

and the pair $\langle \alpha\langle f\rangle, \beta\langle g\rangle\rangle$ satisfies the composition condition, i.e.,

$$\mathrm{sites}(\alpha\langle f\rangle) \cap \mathrm{sites}(\beta\langle g\rangle) = \mathrm{term}(\alpha\langle f\rangle) \cap \mathrm{init}(\beta\langle g\rangle).$$

Then, the **weak composition** of $[\alpha, \beta]$ is defined as

$$[\alpha \mathbin{\triangleleft\!\!\cdot}\, \beta] \overset{\mathrm{def}}{=} [\alpha\langle f\rangle, \beta\langle g\rangle].$$

▶

Lemma 17. Let $\bar{c}$ be a semantic identity. If $[\alpha, \beta] \overset{\bar{c}}{\leftrightarrow} [\alpha', \beta']$ and $[\alpha, \beta]$ satisfies the weak composition condition, then so does $[\alpha', \beta']$ and

$$[\alpha \mathbin{\triangleleft\!\!\cdot}\, \beta] \overset{\bar{c}}{\leftrightarrow} [\alpha' \mathbin{\triangleleft\!\!\cdot}\, \beta'].$$

Proof. Since $[\alpha, \beta]$ satisfies the weak composition condition, take any site replacements f and g such that

$$f|_{\mathrm{ext}(\alpha)} = \mathrm{id}, \qquad g|_{\mathrm{ext}(\beta)} = \mathrm{id}$$

and the pair $\langle \alpha\langle f\rangle, \beta\langle g\rangle\rangle$ satisfies the composition condition. Then, so does every composite pair from $[\alpha\langle f\rangle, \beta\langle g\rangle]$. According to the Formation Projection Lemma (L. 13), one can find among these pairs a pair of the form $\langle \alpha_1, \beta\rangle$ and obtain

$$[\alpha \mathbin{\triangleleft\!\!\cdot}\, \beta] = [\alpha_1 \triangleleft \beta].$$

By definition, if $[\alpha, \beta] \overset{\bar{c}}{\leftrightarrow} [\alpha', \beta']$, then there exist composite pairs $\langle \alpha, \beta\rangle \in [\alpha, \beta]$, $\langle \alpha', \beta'\rangle \in [\alpha', \beta']$, and $c_0 \in \bar{c}$ such that either $\alpha \overset{c_0}{\leftrightarrow} \alpha'$ and $\beta = \beta'$ or $\beta \overset{c_0}{\leftrightarrow} \beta'$ and $\alpha = \alpha'$.

Due to the symmetry of these two cases, we will only consider the first one. Since $\alpha_1 \in [\alpha]$ and due to Lemma 14, there exists $c \in \bar{c}$ and $\alpha'' \in [\alpha']$ such that $\alpha_1 \overset{c}{\leftrightarrow} \alpha''$. By definition of direct convertibility, this means that there exist composites γ_1, γ_2 such that

$$\begin{aligned} \alpha_1 &= \gamma_1 \lhd \gamma \lhd \gamma_2 \\ \alpha'' &= \gamma_1 \lhd \gamma' \lhd \gamma_2, \end{aligned}$$

where $c = \langle \gamma, \gamma' \rangle$ or $c = \langle \gamma', \gamma \rangle$. It follows from Lemma 16 that

$$\mathrm{ext}(\alpha_1) = \mathrm{ext}(\alpha''), \ \ \mathrm{term}(\alpha_1) = \mathrm{term}(\alpha''),$$

hence $\langle \alpha'', \beta \rangle$ satisfies the weak composition condition. Note that the ordinary composition condition may be violated, if $\mathrm{int}(\alpha'') \cap \mathrm{sites}(\beta) \neq \varnothing$. In this case, take any $s \in \mathrm{int}(\alpha'') \cap \mathrm{sites}(\beta)$. Since $\langle \alpha_1, \beta \rangle$ satisfies the composition condition, $s \notin \mathrm{sites}(\alpha_1)$, hence $s \in \mathrm{sites}(\gamma') \setminus \mathrm{sites}(\gamma)$. Therefore, one can replace s in γ' and α'' by another site s' such that $s' \notin \mathrm{sites}(\beta)$. By doing it for all sites $s \in \mathrm{int}(\alpha'') \cap \mathrm{sites}(\beta)$, we obtain composites γ'_1 and α''_1 such that $c' = \langle \gamma, \gamma'_1 \rangle \in \bar{c}$, $\alpha''_1 = \alpha'' \langle f \rangle$, where $f\big|_{\mathrm{ext}(\alpha'')} = \mathrm{id}$, $\alpha_1 \overset{c'}{\leftrightarrow} \alpha''_1$, and, finally, $\langle \alpha''_1, \beta \rangle$ satisfies the composition condition

$$\mathrm{sites}(\alpha''_1) \cap \mathrm{sites}(\beta) = \mathrm{term}(\alpha''_1) \cap \mathrm{init}(\beta).$$

Moreover, due to the associativity of composition, we have

$$\alpha_1 \lhd \beta \overset{c'}{\leftrightarrow} \alpha''_1 \lhd \beta,$$

hence

$$[\alpha \mathrel{\triangleleft\!\!\cdot} \beta] = [\alpha_1 \lhd \beta] \overset{\bar{c}}{\leftrightarrow} [\alpha''_1 \lhd \beta] = [\alpha' \mathrel{\triangleleft\!\!\cdot} \beta].$$

■

Now we are ready to prove the main result of this subsection. Note that it is a slightly stronger statement than the one of congruence with respect to composition.

Lemma 18. If $\mathcal{I}$ is a set of semantic identities, then relation $\sim_{\mathcal{I}}$ on formations is a congruence with respect to composition, i.e., if $[\alpha, \beta] \sim_{\mathcal{I}} [\alpha', \beta']$ and $[\alpha, \beta]$ satisfies the weak composition condition, then so does $[\alpha', \beta']$ and $[\alpha \lhd \beta] \sim_{\mathcal{I}} [\alpha' \lhd \beta']$.

Proof. Consider a relation R on the set of formation pairs defined as follows: $\langle [\alpha, \beta], [\alpha', \beta'] \rangle \in R$ if and only if either both $[\alpha, \beta]$ and $[\alpha', \beta']$ satisfy the weak composition condition and $[\alpha \lhd \beta] \sim_{\mathcal{I}} [\alpha' \lhd \beta']$, or none of them does so.

Relation R is reflexive, symmetric, and transitive, hence an equivalence relation. Moreover, according to Lemma 17, R contains all pairs $\langle [\alpha, \beta], [\alpha', \beta'] \rangle$ such that $[\alpha, \beta] \overset{\mathcal{I}}{\leftrightarrow} [\alpha', \beta']$. Since $\sim_{\mathcal{I}}$ is, by definition, the minimal equivalence relation containing these pairs, $\sim_{\mathcal{I}}$ is contained in R, which implies the statement of the lemma. ■

Definition 24. If $\mathcal{I}$ is a set of semantic identities, relation $\sim_{\mathcal{I}}$ will be called a **semantic equivalence** relation. ▶

Compatibility of the semantic equivalence relation with the part/whole relation follows directly from its congruence with respect to composition:

Lemma 19. Let $\mathcal{I}$ be a set of semantic identities. If $\bar{\alpha} \sim_{\mathcal{I}} \bar{\alpha}'$, then for all $\bar{\beta} \succeq \bar{\alpha}$ there exists a formation $\bar{\beta}' \succeq \bar{\alpha}'$ such that $\bar{\beta}' \sim_{\mathcal{I}} \bar{\beta}$.

Proof. By definition of the part/whole relation, $\bar{\beta} \succeq \bar{\alpha}$ implies that there exist composites $\beta \in \bar{\beta}$, $\alpha \in \bar{\alpha}$ such that $\beta \succeq \alpha$. By definition of the part/whole relation on composites, this means that there exist composites α_1, α_2 such that $\beta = \alpha_1 \lhd \alpha \lhd \alpha_2$.

Since $\bar{\alpha} \sim_{\mathcal{I}} \bar{\alpha}'$, we also have $[\alpha_1, \alpha] \sim_{\mathcal{I}} [\alpha_1, \alpha']$ for some composite $\alpha' \in \bar{\alpha}'$, as a trivial consequence of the definition of equivalence on formation tuples. Due to the congruence of the semantic equivalence relation with respect to composition (Lemma 18), we obtain that formation pair $[\alpha_1, \alpha']$ satisfies the weak composition condition and $[\alpha_1 \lhd \alpha] \sim_{\mathcal{I}} [\alpha_1 \lessdot \alpha']$. As in the proof of Lemma 17, pick a composite $\alpha_1' \in [\alpha_1]$ such that $[\alpha_1' \lhd \alpha'] = [\alpha_1 \lessdot \alpha']$.

Now, since $[\alpha_1 \lhd \alpha] \sim_{\mathcal{I}} [\alpha_1' \lhd \alpha']$, we also have $[\alpha_1 \lhd \alpha, \alpha_2] \sim_{\mathcal{I}} [\alpha_1' \lhd \alpha', \alpha_2']$ for some $\alpha_2' \in \bar{\alpha}_2$ and hence, due to congruence of semantic equivalence relation,

$$[\alpha_1 \lhd \alpha \lhd \alpha_2] \sim_{\mathcal{I}} [(\alpha_1' \lhd \alpha') \lessdot \alpha_2'].$$

Again, pick a composite $\alpha_2'' \in [\alpha_2]$ such that $[(\alpha_1' \lhd \alpha') \lessdot \alpha_2'] = [(\alpha_1' \lhd \alpha') \lhd \alpha_2'']$ and, using associativity of composition, take

$$\beta' = \alpha_1' \lhd \alpha' \lhd \alpha_2''.$$

Then, by construction, $\bar{\beta}' \succeq \bar{\alpha}'$ and $\bar{\beta} \sim_{\mathcal{I}} \bar{\beta}'$. ■

2.2.5 Structs

Now that we have defined the semantic equivalence relation on formations, we are ready to introduce *structs*. A struct is a formal concept that corresponds to a single object representation. It is defined as an equivalence class of formations with respect to to the semantic equivalence relation. In other words, we think of an object representation as an equivalence class of constructive processes.

Definition 25. Let $\mathcal{I}$ be a set of semantic identities, and let $\sim_{\mathcal{I}}$ be the semantic equivalence relation induced by $\mathcal{I}$ on the set of formations $\bar{\Gamma}_{\Pi}$. For a formation $\bar{\gamma} \in \bar{\Gamma}_{\Pi}$, the **struct** containing $\bar{\gamma}$, $\boldsymbol{\gamma}$, is defined as

$$\boldsymbol{\gamma} \overset{\text{def}}{=} [\bar{\gamma}]_{\sim_{\mathcal{I}}} \overset{\text{def}}{=} [\![\gamma]\!]_{\sim_{\mathcal{I}}} \overset{\text{def}}{=} \{\bar{\gamma}' \in \bar{\Gamma}_{\Pi} \mid \bar{\gamma}' \sim_{\mathcal{I}} \bar{\gamma}\}.$$

The set of all structs is denoted $\Theta_{\mathcal{I}}$. ▶

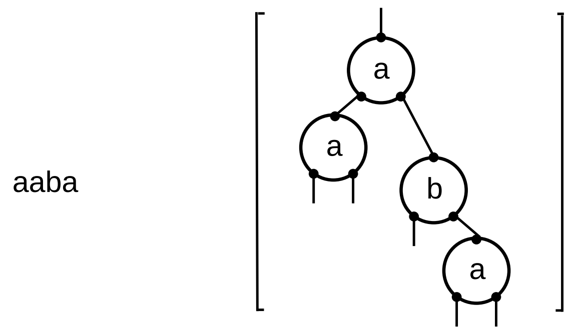

Figure 2.15: Struct corresponding to string “aaba”.

An example of a struct corresponding to a string is shown in Fig. 2.15.

Whenever it does not create ambiguity, we will use the notation $[\![\gamma]\!]$ for the *union* of all formations in $[\![\gamma]\!]$, i.e. the following set of *composites*:

$$\cup_{\bar{\gamma}\in\boldsymbol{\gamma}}\bar{\gamma}.$$

Whereas a non-empty formation usually contains infinitely many composites, the number of formations in a struct can be finite or infinite. This number stands for the number of constructive processes that could possibly have constructed a given representation. Denote the number of formations in a struct $\boldsymbol{\gamma}$ by $|\boldsymbol{\gamma}|$. The number of formations in a struct corresponding to a string of length n is $n!$.

2.2.6 Part/whole relation on structs

The part/whole relation on structs is obtained from that on formations. As it is the case with formations, we expect the part/whole relation on structs to be a partial ordering. We shall see that, in general, this is not the case, hence an additional restriction on the set of semantic identities will be imposed. Also, following Zeno, we assume that each object has finitely many parts (or finitely many ancestors, which turns out to be equivalent). This imposes yet another restriction on the set of semantic identities, which is proved to be equivalent to the finiteness of structs as sets of formations.

The above two restrictions also imply decidability of the semantic equivalence relation (see Section 2.3.7).

Assume that a set of primitives Π and a set of semantic identities $\mathcal{I}$ are fixed. All composites, formations, and structs in this section are assumed to belong to the sets Γ_Π, $\bar{\Gamma}_\Pi$, and $\Theta_\mathcal{I}$, respectively.

Definition 26. A struct $\boldsymbol{\alpha}$ is called a **part** (an **ancestor**) of a struct $\boldsymbol{\beta}$, denoted $\boldsymbol{\alpha} \preceq \boldsymbol{\beta}$, if there exist formations $\bar{\alpha} \in \boldsymbol{\alpha}$ and $\bar{\beta} \in \boldsymbol{\beta}$ such that $\bar{\alpha}$ is a part (an ancestor) of $\bar{\beta}$. We use the same notation, Anc($\boldsymbol{\alpha}$) and Parts($\boldsymbol{\alpha}$), for structs. ▶

The part/whole relation on structs is reflexive. To make sure that relation $\preceq$ on structs is antisymmetric, we have to impose the following restriction on the semantic equivalence relation:

Definition 27. Semantic equivalence relation $\sim_\mathcal{I}$ satisfies the **part/whole distinguishability** condition, if for all formations $\bar{\alpha}, \bar{\beta}$,

$$\bar{\alpha} \sim_\mathcal{I} \bar{\beta} \text{ and } \bar{\alpha} \preceq \bar{\beta} \;\Rightarrow\; \bar{\alpha} = \bar{\beta}.$$

▶

The above condition can be reformulated as follows: no formation is equivalent to any of its proper parts.

Lemma 20. If $\sim_\mathcal{I}$ satisfies the part/whole distinguishability condition, then the part/-whole relation on structs is antisymmetric.

Proof. Let $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ be two structs such that $\boldsymbol{\alpha} \preceq \boldsymbol{\beta}$ and $\boldsymbol{\beta} \preceq \boldsymbol{\alpha}$. Then, by definition, there exist formations $\bar{\alpha}_1, \bar{\alpha}_2 \in \boldsymbol{\alpha}$ and $\bar{\beta}_1, \bar{\beta}_2 \in \boldsymbol{\beta}$ such that

$$\bar{\alpha}_1 \preceq \bar{\beta}_1, \quad \bar{\beta}_2 \preceq \bar{\alpha}_2.$$

Since $\bar{\alpha}_2 \sim_{\mathcal{I}} \bar{\alpha}_1$, by compatibility of semantic equivalence with the part/whole relation on formations (Lemma 19), there exists formation $\bar{\beta}'_2 \in \boldsymbol{\beta}$ such that $\bar{\alpha}_2 \preceq \bar{\beta}'_2$. By transitivity of $\preceq$ on formations (Lemma 11), we obtain $\bar{\beta}_2 \preceq \bar{\beta}'_2$. Now, the part/whole distinguishability condition implies that $\bar{\beta}_2 = \bar{\beta}'_2$.

By antisymmetry of $\preceq$ on formations, we obtain $\bar{\beta}_2 = \bar{\alpha}_2 = \bar{\beta}'_2$, which implies $\boldsymbol{\alpha} = \boldsymbol{\beta}$. ■

Lemma 21. The part/whole relation on structs is transitive.

Proof. Let $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ be three structs such that $\boldsymbol{\alpha} \preceq \boldsymbol{\beta}$ and $\boldsymbol{\beta} \preceq \boldsymbol{\gamma}$. Then, there exist formations $\bar{\alpha} \in \boldsymbol{\alpha}$, $\bar{\beta}_1, \bar{\beta}_2 \in \boldsymbol{\beta}$, $\bar{\gamma} \in \boldsymbol{\gamma}$ such that $\bar{\alpha} \preceq \bar{\beta}_1$ and $\bar{\beta}_2 \preceq \bar{\gamma}$.

By Lemma 19, there exists formation $\bar{\gamma}'$ such that $\bar{\beta}_1 \preceq \bar{\gamma}'$ and $\bar{\gamma}' \sim_{\mathcal{I}} \bar{\gamma}$. By transitivity of $\preceq$ on formations, we have $\bar{\alpha} \preceq \bar{\gamma}'$, which implies $\boldsymbol{\alpha} \preceq \boldsymbol{\gamma}$. ■

The following theorem justifies, at least to some extent, the introduction of the struct finiteness condition.

Theorem 2. *Let* Π *be a set of primitives specified by a finite set of primitive types* $\bar{\Pi}$. *Let* $\mathcal{I}$ *be a set of semantic identities such that part/whole distinguishability condition holds for the semantic equivalence relation* $\sim_{\mathcal{I}}$. *Then for any struct* $\boldsymbol{\gamma}$, *the following are equivalent:*

- *the number of formations* $|\boldsymbol{\gamma}|$ *is finite*
- *the set* $\mathrm{Parts}(\boldsymbol{\gamma})$ *is finite*

- *the set* $\mathrm{Anc}(\boldsymbol{\gamma})$ *is finite.*

Proof.

1. Suppose, $|\boldsymbol{\gamma}|$ is finite. If $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \mathrm{Parts}(\boldsymbol{\gamma})$, $\boldsymbol{\alpha}_1 \neq \boldsymbol{\alpha}_2$, then for all formations $\bar{\alpha}_1 \in \boldsymbol{\alpha}_1, \bar{\alpha}_2 \in \boldsymbol{\alpha}_2$, we have $\bar{\alpha}_1 \neq \bar{\alpha}_2$. It follows from the definition of the part/whole relation on structs that for every struct $\boldsymbol{\alpha} \in \mathrm{Parts}(\boldsymbol{\gamma})$ and every formation $\bar{\alpha} \in \boldsymbol{\alpha}$, there exists a formation $\bar{\gamma} \in \boldsymbol{\gamma}$ such that $\bar{\alpha} \preceq \bar{\gamma}$. Since $|\boldsymbol{\gamma}|$ is finite and for each $\bar{\gamma} \in \boldsymbol{\gamma}$, $\mathrm{Parts}(\bar{\gamma})$ is finite (by Lemma 12), the set
$$\mathrm{Parts}(\boldsymbol{\gamma}) = \bigcup_{\bar{\gamma} \in \boldsymbol{\gamma}} \{[\bar{\alpha}] \mid \bar{\alpha} \in \mathrm{Parts}(\bar{\gamma})\}.$$
is also finite, and so is its subset $\mathrm{Anc}(\boldsymbol{\gamma})$.

2. Suppose, $|\boldsymbol{\gamma}|$ is infinite. Since the set of primitive types $\bar{\Pi}$ is finite, there exists a primitive type $\bar{\pi}^1$ and an infinite set subset $\{\bar{\gamma}_i^1\}_{i \in \mathbb{N}} \subset \boldsymbol{\gamma}$ such that $\bar{\pi}^1 \in \mathrm{Anc}(\bar{\gamma}_i^1)$ for all $i \in \mathbb{N}$. Hence, every $\bar{\gamma}_i^1$ contains a composite γ_i of the form $\gamma_i = \pi^1 \lhd \delta_i^1$, where π^1 is a fixed primitive from $\bar{\pi}^1$ and δ_i^1's are some composites. In the infinite set of formations $\{\bar{\gamma}_i^1\}_{i \in \mathbb{N}}$, we can find an infinite subset $\{\bar{\gamma}_i^2\}_{i \in \mathbb{N}}$ such that each formation in this subset contains a composite $\gamma_i^2 = \pi^{(1)} \lhd \pi^2 \lhd \delta_i^2$. Continuing this process, we obtain an infinite sequence of formations
$$[\pi^{(1)}], [\pi^{(1)} \lhd \pi^{(2)}], \ldots,$$
each of which is an ancestor of a formation from $\boldsymbol{\gamma}$. Moreover, neither two of them belong to the same struct. Indeed, any two formations from the above sequence have the form $[\alpha]$ and $[\alpha \lhd \beta]$. If they belong to the same struct, then formations $[\alpha]$ and $[\alpha \lhd \beta]$ are semantically equivalent. Since $[\alpha]$ is a part of

$[\alpha \lhd \beta]$, the part/whole distinguishability condition implies that $[\alpha] = [\alpha \lhd \beta]$, hence β is an empty composite, which is not the case for any two distinct composites from the above sequence.

■

Consider an example of a set of semantic identities, for which the conditions of the above lemma do not hold and, as a consequence, obtain a struct with infinitely many ancestors and formations. The specifying set of primitives is shown in Fig. 2.16a. Using this set of primitives, one can build formations that correspond to the strings over the alphabet $\{a, b, c, d, e\}$. For example, the formation shown in Fig. 2.16b corresponds to the string *abd*.[6] For simplicity, we denote composites by the corresponding strings in this example, assuming that the sites are chosen appropriately. Choose the

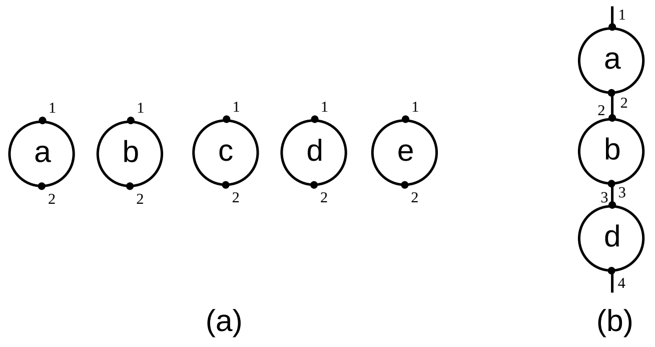

Figure 2.16: A set of specifying primitives and an example of a composite.

following formation pairs for semantic identities:

$$\mathcal{I} = \{[baa, bc],\ [ca, ac],\ [cd, ed],\ [ea, ae],\ [be, ba]\}.$$

[6]This representation of strings is different from the one constructed before in our running example.

Then for any $k \geq 1$, we have $ba^kd \sim_{\mathcal{I}} bad$, since

$$ba^{k+1}d = baaa^{k-1}d \overset{[baa,bc]}{\leftrightarrow} bca^{k-1}d \overset{[ca,ac]}{\leftrightarrow} \ldots \overset{[ca,ac]}{\leftrightarrow} ba^{k-1}cd \overset{[cd,ed]}{\leftrightarrow}$$
$$ba^{k-1}ed \overset{[ea,ae]}{\leftrightarrow} \ldots \overset{[ea,ae]}{\leftrightarrow} bea^{k-1}d \overset{[be,ba]}{\leftrightarrow} ba^kd.$$

Therefore, struct $[bad]$ has infinitely many formations. It has infinitely many ancestors as well:

$$\{[ba^k] \mid k \geq 1\} \subset \mathrm{Anc}([bad]).$$

Note that the elements of the above set are also ancestors of each other. Therefore, if we show that the semantic equivalence relation $\sim_{\mathcal{I}}$ satisfies the part/whole distinguishability condition, this will guarantee that all elements of this set are distinct.

Indeed, assume that there exist strings u and u' such that $u \sim_{\mathcal{I}} u'$ and $u \preceq u'$, which means that $u' = xuy$ for some strings x and y. Note that $\sim_{\mathcal{I}}$ preserves b's and d's, since none of the semantic identities affects them. Hence, x and y are free of b's and d's. Now observe that relation $\overset{\mathcal{I}}{\leftrightarrow}$ has the following invariants: given $u \overset{\mathcal{I}}{\leftrightarrow} u'$,

1. if $u = u_1u_2$ and $u' = u_1'u_2'$, where u_1,u_1' are free of b, and u_2,u_2' begin with b or are empty, then $|u_1| = |u_1'|$.

2. denote by $\#_l(u)$ the number of entries of a letter $l \in \{a, b, c, d, e\}$ in u. If $u = u_3u_4$ and $u' = u_3'u_4'$, where u_3,u_3' end in d or are empty, and u_4,u_4' are free of d, then

$$\#_a(u_4) + 2(\#_c(u_4) + \#_e(u_4)) = \#_a(u_4') + 2(\#_c(u_4') + \#_e(u_4')).$$

These invariants also hold for the reflexive and transitive closure of $\overset{\mathcal{I}}{\leftrightarrow}$, which is $\sim_{\mathcal{I}}$. Given $u \sim_{\mathcal{I}} u'$ and $u' = xuy$, we obtain that

1. $u_1' = xu_1$ and, due to the first invariant, x is empty.

2. $u'_4 = u_4 y$ and, due to the second invariant, y is empty.

Hence, $u \sim_{\mathcal{I}} u'$ and $u \preceq u'$ imply $u = u'$, and the part/whole distinguishability condition holds.

As mentioned above, we assume (and postulate) that each physical object and, correspondingly, each representation have finitely many parts. Perhaps, the statement that every object has finitely many ancestors can be justified even better based on evolutionary considerations. The above theorem asserts that both of these assumptions are equivalent to the assumption that there are finitely many processes that can construct an object. Of course, the equivalence of the assumptions does not justify them fully, but it raises our level of confidence in them. After all, the Church-Turing thesis was accepted on similar grounds.

Definition 28. Semantic equivalence $\sim_{\mathcal{I}}$ satisfies the **struct finiteness condition**, if for every struct $\boldsymbol{\gamma}$, the number of formations $|\boldsymbol{\gamma}|$ is finite. ▶

In Section 2.4.1, we will prove that struct finiteness also implies part/whole distinguishability.

To summarize this section:

- a semantic identity is a formation pair corresponding to a pair of equivalent constructive processes, which construct the same representation;

- a set of semantic identities induces a semantic equivalence relation on the set of formations; this relation is a congruence with respect to composition and is compatible with the part/whole relation;

- a struct is an equivalence class of formations with respect to semantic equivalence; a struct corresponds to a single object representation;
- the part/whole relation is extended from formations to structs and is a partial ordering relation, if the part/whole distinguishability condition is imposed;
- the struct finiteness condition means that the number of formations in a struct is finite; struct finiteness is equivalent to the finiteness of the sets of parts and ancestors of the struct.

2.3 Inductive structures

An inductive structure is specified by a set of primitives and a set of semantic identities satisfying the struct finiteness condition:

Let Π be set of primitives and let $\mathcal{I}$ be a set of semantic identities. Suppose that the semantic equivalence relation induced by $\mathcal{I}$ satisfies the struct finiteness conditions. An **inductive structure** induced by $\mathcal{I}$ and Π is loosely defined as the collection of all sets and relations that can be derived from Π and $\mathcal{I}$ and denoted by $\langle \Pi, \mathcal{I} \rangle$ The above sets and relations include the set of composites Γ_Π, the set of formations $\bar{\Gamma}_\Pi$, the semantic equivalence relation $\sim_\mathcal{I}$, the set of structs $\Theta_\mathcal{I}$, and the part/whole relations on formations and structs which are all denoted by $\preceq$.[7]

Consider several examples of inductive structures, corresponding to the conventional representations, such as natural numbers, binary sequences, strings, trees, and graphs. These examples illustrate the concept of inductive structure and its components: primitives, composites, formations, semantic identities, and structs. It will also become clear, which *constructive processes* are hidden in the conventional representations.

Conventional representations such as strings or graphs are considered as ambiguous from the ETS point of view, since they do not specify explicitly the constructive processes that generate them. In fact, very different processes can be assumed. For example, two different ways to generate strings are considered in this thesis, but by no means are these two exhaustive.

Take the string data structure. The constructive processes that, I claim, are

[7]The rest of the concepts that constitute an inductive structure will be introduced in Section 3.1.

always implicitly assumed, whenever a string is used to represent an object, are different for different kinds of objects:

(a) if a string of 1's represents a heap of stones, which can be thought of as constructed by adding stones one by one, the corresponding process that constructs the string appends 1's at the end of it;

(b) if a string represents a natural language sentence, it can be thought of as generated by a formal grammar;

(c) if a string represents an utterance, or sometimes a gene, it can be thought of as produced by a Hidden Markov model.

It is important to note that the difference between these constructive processes is not reflected in the string representation itself. In other words, the string representation *hides* the constructive processes behind it, and in this sense is ambiguous. Even more ambiguous in this respect are graphs, since they can be associated with an even greater variety of constructive processes.

Once the constructive processes for strings (or graphs) are made explicit, it often turns out that there are exponentially many processes that can construct a given string (or graph). We shall argue that the exponential number of constructive processes results in the exponential complexity of the problems that involve reconstruction of these processes. One important case of such problems is the grammatical inference problem, which is known to be intractable [Gold78].

Due to the above reasons, conventional data structures (or the corresponding inductive structures) do not provide a good starting point for representing real-world objects of any kind. Rather, it is more feasible to start with a simpler data structure, with fewer constructive histories, and then gradually increase the complexity and representational power by constructing higher representational levels (see Chapter 3) with the help of the inductive learning process (see Section 5.4).

Every particular area of application requires a specific inductive structure designed according to the existing scientific knowledge about the actual processes that construct the objects. I will give some suggestions on how to select an inductive structure, although I realize that a single successful application of the model would have explained it much better. Anyway, this initial step, the choice of the inductive structure, is inevitable. Any other approach, including those based on numeric representations, requires a similar initial step, with the only difference that it may be hidden behind the choice of "good representational features" or even the design of measurement devices.

2.3.1 Inductive structure of natural numbers

For this and the following inductive structures, assume that an infinite set $\mathcal{S}$ of sites, is fixed. For simplicity of notation, also assume that $\mathcal{S}$ contains the set $\mathbb{N}$ of natural numbers.[8]

The only specifying primitive from $\Pi_{\mathbb{N}}$ with one input and one output site is shown in Fig. 2.17. The set of semantic identities $\mathcal{I}_{\mathbb{N}}$ is empty. The concept of a

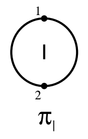

Figure 2.17: Specifying primitive $\pi_|$ for the inductive structure of natural numbers.

[8]The latter assumption is only used in the examples, but not in the definitions of inductive structures.

natural number, conventionally defined by Peano axioms [La51], implies that for each natural number n, there is a construction process consisting of a sequence of $n-1$ applications of the successor mapping

$$\mathrm{succ}(\mathrm{succ}(\ldots \mathrm{succ}(1)\ldots))$$

to the initial element $1 \in \mathbb{N}$. In the inductive structure of natural numbers, the following formation corresponds to this process:

$$\bar{\gamma}_n = [\pi_| \langle 1, 2\rangle, \pi_| \langle 2, 3\rangle, \ldots, \pi_| \langle n, n+1\rangle].$$

For every $n \in \mathbb{N}$, the struct $\boldsymbol{\gamma}_n = [\bar{\gamma}_n]$ consists of only one formation, $\bar{\gamma}_n$, since the set of semantic identities is empty.

Technical note. Alternatively to the Peano axiomatics, the concept of a natural number can be defined as a struct in the above inductive structure. This definition is not circular, since we have never relied on natural numbers or any other depending concepts in our definitions. In fact, all we have used are sets, although the existence of infinite sets is necessary. Let us compare this definition with the axiomatic definition based on Peano axioms and the Von Neumann's construction of finite ordinals based on the axiomatic set theory.

On the one hand, both of these definitions are preferable to ours, because they do not assume the existence of infinite sets. On the other, it seems possible to come up with an *axiomatic* definition of an inductive structure and avoid postulation of existence of infinite sets. This definition will, of course, be a generalization of the Peano axioms.

Regarding the von Neumann's construction of natural numbers as finite ordinals

$$1 = \{0\},\ 2 = \{0, \{0\}\},\ 3 = \{0, \{0\}, \{0, \{0\}\}\}, \ldots,$$

This construction is quite unnatural in the representational sense, i.e., it does not suggest the right way of thinking about the constructive processes for natural numbers. For this reason, I do not think that an axiomatic definition of inductive structures should be generalized from this construction.

The inductive structure $\langle \Pi_{\mathbb{N}}, \mathcal{I}_{\mathbb{N}} \rangle$ is only one of the possible ways to represent natural numbers. Others may be based on the decimal encoding or decomposition into prime factors and will result in completely different inductive structures. Representing constructive histories of the same number differently may give interesting illustrations to known properties of numbers or even suggest new theorems about them.[9]

2.3.2 Inductive structure of binary sequences

Consider an inductive structure, whose structs correspond to finite binary sequences. We would like to note that by introducing a topology on the set of structs and considering its compactification, one can obtain infinite binary sequences as well, which results in an ETS structure corresponding to the real segment $[0, 1]$. The interested reader is referred to [Golub03].

The set of specifying primitives for the inductive structure of binary sequences, $\Pi_{\mathbb{B}}$, is shown in Fig. 2.18. Primitives π_0 and π_1 correspond to the binary digits. The

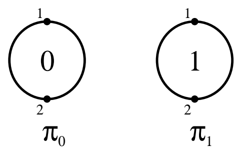

Figure 2.18: Specifying primitives for the inductive structure of binary sequences.

[9]I cannot justify this claim particularly for natural numbers, but it is already the case for strings (see [Golub03]).

only semantic identity from $\mathcal{I}_{\mathbb{B}}$ is shown in Fig. 2.19.

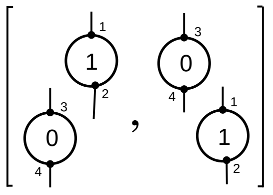

Figure 2.19: Semantic identity for the inductive structure of binary sequences.

An example of a binary sequence and of the corresponding struct is shown in Fig. 2.20. Note that the struct contains only one formation (shown inside the brackets that denote the equivalence class), since the only identity from $\mathcal{I}_{\mathbf{B}}$ does not apply to it. If we consider this binary sequence as a representation for binary fraction

$$0.01001 = 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 0 \cdot 2^{-4} + 1 \cdot 2^{-5},$$

where the sum, computed from left to right, describes the *process of construction* of the binary fraction. The formation in Fig. 2.20 corresponds to this process as well.

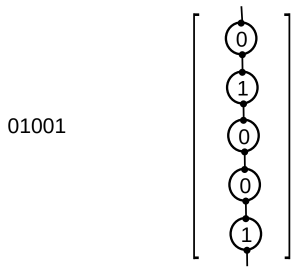

Figure 2.20: A binary sequence and the corresponding formation.

A process that constructs a binary sequence can also be thought of as a process that constructs a natural number or, more precisely, its binary encoding:

$$9 = 01001_2 = (((0 \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 0) \cdot 2 + 1.$$

This is the second example of a construction process for natural numbers (the first one was introduced in Section 2.3.1).

2.3.3 Inductive structure of trees

Consider an inductive structure, whose structs correspond to k-ary labeled trees. Let $\mathbb{T}(k)$ be the set of rooted k-ary trees, whose nodes are labeled by characters from the alphabet $\{a, b\}$. The set of specifying primitives for the corresponding inductive structure, $\Pi_{\mathbb{T}(k)}$, is shown in Fig. 2.21.

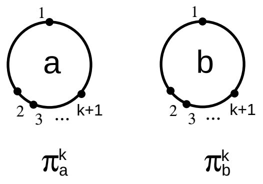

Figure 2.21: Primitives for the inductive structure of k-ary trees.

The set of semantic identities $\mathcal{I}_{\mathbb{T}(k)}$ is similar to $\mathcal{I}_{\mathbf{B}}$ (see Fig. 2.22). Every struct

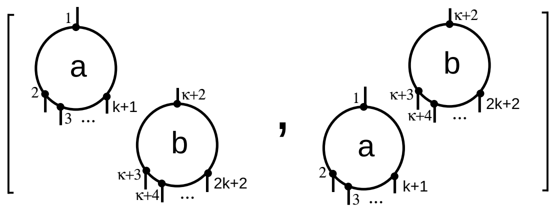

Figure 2.22: Semantic identity for the inductive structure of trees.

corresponding to a tree contains one formation. An example of a tree and the corresponding struct is shown in Fig. 2.23.

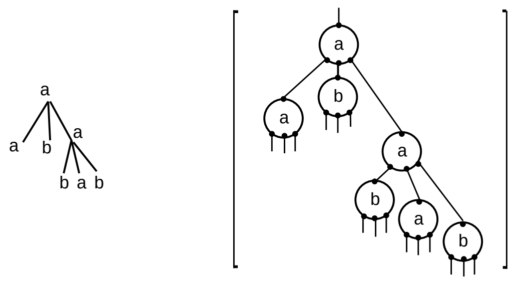

Figure 2.23: A tree and the corresponding struct.

2.3.4 Inductive structure of insertion strings

The inductive structure of insertion strings has been defined in our running example throughout Sections 2.1 and 2.2. Alternatively, it can be obtained from the the inductive structure of binary trees (restrict the construction in the previous section to $k = 2$) by adding the semantic identities shown on the left of Fig. 2.14. This implies that the number of constructive processes for a tree is less than or equal to that for the corresponding insertion string (although, both numbers are exponential in general).

So far, representation of deletion and substitution operations has been deliberately avoided in the definitions of inductive structures. Moreover, this representation is not be possible, since the introduction of the corresponding primitives and identities would violate the part/whole distinguishability condition and result in an infinite number of constructive histories. Given the relationship between the number of constructive histories and the complexity of inductive inference discussed above, it becomes clear while modeling of deletions and substitutions in the same way as constructive operations is undesirable.

On the other hand, without deletions and substitutions, the descriptive power of generating mechanisms on strings is clearly too low to be practical for most applications. Moreover, one may even claim that these operations are observed practice, e.g., as point mutations in genes. Thus, it is worthwhile to make a digression here and show how the processes modeled by deletions and substitutions on strings are described in the ETS formalism.

First of all, we model not all of these processes, but only those that are *irreversible*. The importance of irreversible processes (and of the models that account for irreversibility) has been emphasized by many scientists (see, for example, [Sch93]). In particular, thermodynamic processes are irreversible according to the second law of thermodynamics [Pr80]. Developmental processes in biological organisms are also irreversible (it is really inconceivable how one could possibly reverse, say, the process of development of a chicken from an egg). Finally, the process of evolution of species is (or has been so far) irreversible, since the complexity of species has been, generally, rising with time. It is these evolutionary and developmental processes that were kept in mind as a metaphor for the design of the ETS model.[10]

To simplify the concept of irreversibility, I reformulate it as follows: a process is irreversible, if it cannot, in principle, be in the same state twice. If an irreversible process generates objects, i.e., if objects are its states, then it is not allowed to produce a part of an object from the whole object, since otherwise it could continue and rebuild the object from the part and return to the same state. Thus, for processes that generate objects, irreversibility is equivalent to the property that we call *strong acyclicity*: an object cannot transform into one of its proper parts.

Strong acyclicity does not imply the absence of substitutions (although, deletions

[10]The thermodynamic irreversibility is of a very different nature (see [Pr80]).

are excluded, since a deletion immediately transforms an object into its proper part). A composition of a deletion with a substitution, considered as one operation (e.g., $abc \to bd$), also does not violate the strong acyclicity requirement. In particular, the class of strongly acyclic string-rewriting systems is quite broad. And, it turns out, all strongly acyclic string-rewriting systems can be simulated in the ETS model (see Section 4.1) and, quite clearly, this construction can be extended to graphs. The idea of the construction is simple: one can look at a substitution operation as a composition of a constructive transformation with a semantic identity (see Fig. 4.1). The roles of them is very different: the constructive transformation changes the object and the semantic identity just reflects the fact that an object has several possible constructive histories.

2.3.5 Inductive structure of graphs

The inductive structure of graphs completes our brief overview of the relationship between conventional data structures and inductive structures. The inductive structure of graphs is also interesting for us in that its structs turn out to "grow fast", which results in the fact that the generating processes (see Chapter 5) in the inductive structure of graphs , as opposed to all previously considered inductive structures, can construct infinite structs in a finite time (see [Golub02] for the details).

Consider the set $\mathbb{G}$ of directed multigraphs without loops and isolated vertices, and with edges labeled by characters a, b.[11]

The set of specifying primitives, $\Pi_{\mathbb{G}}$, is shown in Fig. 2.24. Primitives σ_a and

[11]This particular type of graphs has been chosen for no particular reason, mostly because the corresponding inductive structure is simple. Disconnected vertices, loops, undirected edges, and simple edges can be described by the corresponding inductive structures as well, but this is not our purpose now.

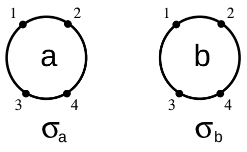

Figure 2.24: Primitives for the inductive structure of graphs.

σ_b correspond to the edges with labels a and b. The semantic identities from $\mathcal{I}_\mathcal{G}$ are shown in Fig. 2.25. Example of graphs and corresponding structs are shown in

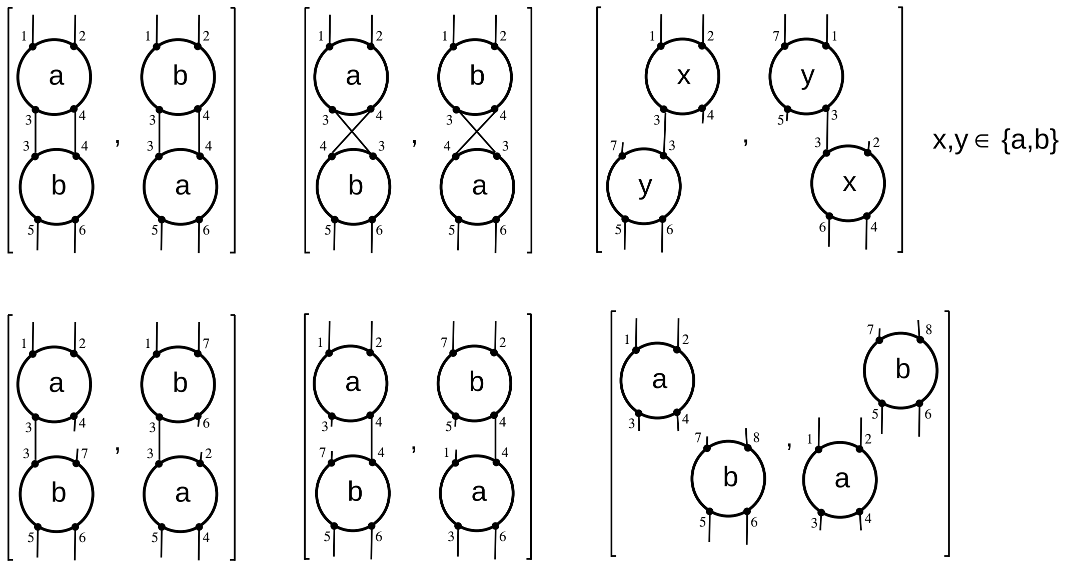

Figure 2.25: Semantic identities for the inductive structure of graphs.

Fig. 2.26.

There is no direct relationship between the number of vertices or edges in the graph and the number of formations in the corresponding struct, since symmetries of the graph may reduce the number of formations. For a general "asymmetric" graph with m edges, the number of formations is exactly $m!$. The question of how the number of formations depends on graph symmetries is yet another direction of study, which may reveal an further connection between the ETS model and graph theory.

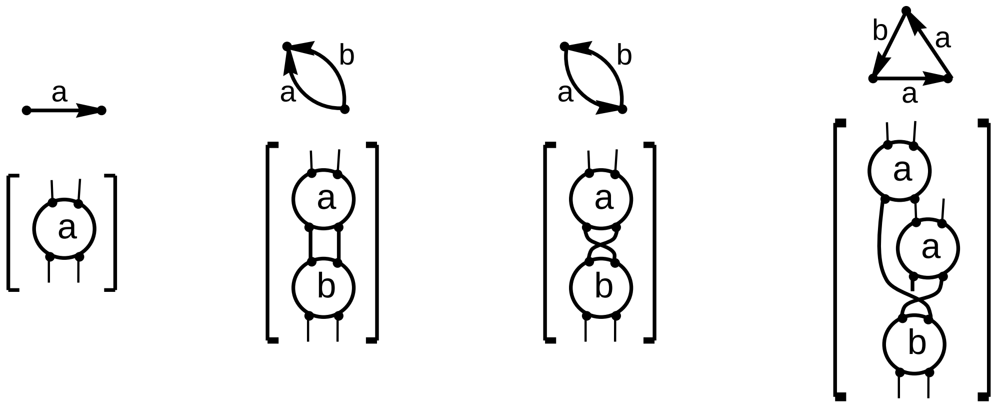

Figure 2.26: Examples of graphs and corresponding structs.

2.3.6 How to select and evaluate an inductive structure for a given real world problem?

The process of selection of the appropriate inductive structure for a given problem reduces to the selection of the appropriate primitives and semantic identities. This selection is guided by our intuitive understanding of the part/whole relation between the physical objects in question and of the admissible transformations for them. For example, in chemistry, atoms and bonds can be considered as primitive parts of molecules, and chemical reactions can indicate the admissible transformations. The first attempt to describe the corresponding inductive structure has been made in [Ko03].

Another criterion for the evaluation of an inductive structure is the number of formations in its structs, which, as it follows from the above discussions, should be reduced as much as possible.

Still, it is quite evident that the above two criteria give only a vague mechanism of design of the inductive structures. The main advantage of the ETS model is yet to be described in Chapter 3. In short, it is its hierarchy of representational levels, of which only the basic level has been defined so far, that allows to see the far-reaching consequences of the chosen primitives and identities at the higher representational levels and evaluate them according to the existing scientific knowledge of the corresponding objects and their classes.

2.3.7 Decidability of the semantic equivalence relation

Here, we prove that semantic equivalence relation is undecidable in general. Moreover, we conjecture that the part/whole distinguishability condition does not imply decidability. However, if the struct finiteness condition is imposed, semantic equivalence becomes decidable.

The first result will be obtained by reducing the word problem to the problem of semantic equivalence checking, which we shall call the semantic equivalence problem.

Consider the set of strings over a finite alphabet Σ, and let $I = \{(l_1, r_1), \ldots, (l_k, r_k)\}$ be a finite set of string pairs, which are called *rewriting rules*. A string u *can be rewritten in one step* into a string v, denoted $u \to v$, if there exists a rewriting rule $(l_i, r_i) \in I$ and strings x, y such that

$$u = xl_iy \quad \text{and} \quad v = xr_iy.$$

Let $\leftrightarrow_I$ be the reflexive symmetric transitive closure of the relation $\to$. The word problem for the set I is formulated as follows: given two strings u, v, determine whether $u \leftrightarrow_I v$. There exists a set I, for which the word problem is undecidable [RS97, Theorem 2.2, p. 444].

To reduce the word problem to the semantic equivalence problem, consider the inductive structure of sequences over Σ. This inductive structure is a straightforward generalization of the inductive structure of binary sequences introduced in Section 2.3.2 above.[12] Add to it the following set of semantic identities:

$$\mathcal{I} = \{[\tilde{l_i}, \tilde{r}_i] \mid (l_i, r_i) \in I\},$$

where $\tilde{u}$ denotes the composite corresponding to the string u in the inductive structure of sequences over Σ. Then, $u \leftrightarrow_I v$ if and only if $[\tilde{u}] \sim_{\mathcal{I}} [\tilde{v}]$, where $\sim_{\mathcal{I}}$ is the semantic equivalence relation induced by $\mathcal{I}$. This concludes the proof of the following result:

Theorem 3. *There exists an undecidable semantic equivalence relation induced by a finite set of semantic identities.*

We conjecture that there exists an undecidable semantic equivalence relation satisfying the part/whole distinguishability condition as well and leave the verification of this conjecture for future research.

However, if the struct finiteness condition is imposed on the set of semantic identities, then the semantic equivalence relation becomes decidable. Indeed, let $\mathcal{I}$ be a finite set of semantic identities. Given a formation $\bar{\gamma}$, one can compute the following set of formations:

$$N(\gamma) = \{\bar{\gamma}' \mid \bar{\gamma} \overset{\mathcal{I}}{\leftrightarrow} \bar{\gamma}'\}.$$

Now, one can decide whether two formations $\bar{\alpha}$ and $\bar{\beta}$ are semantically equivalent by constructing a chain of nested sets

$$B_1 \subset B_2 \subset B_3 \subset \dots \qquad (*)$$

as follows: set $B_1 = \{\bar{\alpha}\}$, and let

$$B_{i+1} = B_i \cup \bigcup_{\bar{\gamma} \in B_i} N(\bar{\gamma}).$$

[12]This generalization is not even necessary, since the word problem for strings over a two-letter alphabet is undecidable.

Since each B_i is a subset of the struct $[\bar{\alpha}]$, and the latter, according to the struct finiteness condition, is a finite set of formations, the chain $(*)$ is also finite. Let the last set in it be B_m, also denote it $B(\bar{\alpha})$. We will show that $B(\bar{\alpha}) = [\bar{\alpha}]$.

In fact, consider the relation

$$R = \{(\bar{\alpha}, \bar{\alpha}')) \mid \alpha' \in B(\bar{\alpha})\}$$

on the set of formations. By definition of $B(\alpha)$, we have that for all composites α, β,

$$\alpha \mapsto_C \beta \implies (\bar{\alpha}, \bar{\beta}) \in R.$$

Since the semantic equivalence relation $\sim_{\mathcal{I}}$ is the minimal equivalence relation satisfying the above property, we have $R \supseteq \sim_{\mathcal{I}}$. Hence, $B(\bar{\alpha}) \supseteq [\bar{\alpha}]$.

Finally, to check whether $\bar{\alpha} \sim_{\mathcal{I}} \bar{\beta}$, it is sufficient to check whether $\bar{\beta} \in B(\bar{\alpha})$. Thus, the following result is proved.

Theorem 4. *There exists an algorithm that, given a set of semantic identities $\mathcal{I}$ that induces a semantic equivalence relation satisfying the struct finiteness condition and two formations $\bar{\alpha}$ and $\bar{\beta}$, determines whether $\bar{\alpha} \sim_{\mathcal{I}} \bar{\beta}$.*

It is important to note that the above algorithm may be very inefficient. The reason for this becomes clear, once we notice that many of the difficult problems in computer science reduce to the semantic equivalence problem in an appropriate inductive structure. For example, the semantic equivalence problem in the inductive structure of graphs (Section 2.3.5) is equivalent to the graph isomorphism problem. In what follows, we will not only need to check semantic equivalence of formations, i.e. equality of structs, but also solve a more general problem of whether one struct is a part of another. In the inductive structure of graphs, this is equivalent to the *subgraph isomorphism problem*, which is known to be NP-complete [Ga79].

One could try to remedy this situation by imposing further restrictions on semantic identities. In general, the *nature* of semantic identities is not yet completely clear, since we do not have a good explanation so far of how the identities are constructed (it has been said that identities reflect the fact that certain constructive histories produce the same object, by no account has been given to why and how this happens). Once this will be clarified, further useful restrictions on the form of semantic identities should appear.

2.3.8 Undecidability of the part/whole distinguishability and struct finiteness conditions

Consider the following problems: given a set of semantic identities, determine whether the corresponding semantic equivalence relation satisfies the part/whole distinguishability condition or the struct finiteness condition. Both of these problems are undecidable in general.

Consider strings over a finite alphabet Σ. As above, a finite set of rewriting rules $I = \{(l_1, r_1), \ldots, (l_k, r_k)\}$ corresponds to the set of identities $\mathcal{I} = \{[\tilde{l}_i, \tilde{r}_i] \mid (l_i, r_i) \in I\}$, which, together with the primitives for sequences over Σ, induces an inductive structure. We shall prove that the above questions about these inductive structure are undecidable in general.

A finite set of string pairs I over a finite alphabet Σ specifies a *finitely presented monoid* $\mathcal{M} = \Sigma^*/_{\leftrightarrow_I}$. The pair $\langle \Sigma, I \rangle$ is called a *presentation* of $\mathcal{M}$. We will say that presentation $\langle \Sigma, I \rangle$ of $\mathcal{M}$ satisfies the part/whole distinguishability (or struct finiteness) condition, if so does the corresponding inductive structure. We will first show that the part/whole distinguishability and the struct finiteness properties are *invariant* [Bo93, §7.3] monoid properties, i.e. they are independent of the presentation.

Lemma 22. If $\langle \Sigma_1, I_1 \rangle$ and $\langle \Sigma_2, I_2 \rangle$ are two presentations of the same monoid $\mathcal{M}$, then

1. $\langle \Sigma_1, I_1 \rangle$ satisfies the part/whole distinguishability condition if and only if $\langle \Sigma_2, I_2 \rangle$ does.

2. $\langle \Sigma_1, I_1 \rangle$ satisfies the struct finiteness condition if and only if $\langle \Sigma_2, I_2 \rangle$ does.

Proof.

1. The part/whole distinguishability condition can be equivalently rewritten as a property of monoid $\mathcal{M}$ as follows:

$$\forall u, v, w \in \mathcal{M} \;\; (v \neq 1 \;\text{ or }\; w \neq 1) \implies vuw \neq u.$$

   Clearly, the fact whether this property holds is independent of the presentation of $\mathcal{M}$.

2. The struct finiteness condition can be rewritten as an invariant property as follows:

$$\forall u \in \mathcal{M} \;\; |\{v \in \mathcal{M} \mid \exists x, y \in \mathcal{M}\; u = xvy\}| < \infty.$$

■

Next, we show that these two properties are *hereditary* [Bo93, §7.3], i.e., if any of them holds for the monoid $\mathcal{M}$, it will also hold for every submonoid of $\mathcal{M}$.

Lemma 23.

1. If $\mathcal{M}$ is a monoid satisfying the part/whole distinguishability condition, then this condition holds for every submonoid $\mathcal{M}'$ of $\mathcal{M}$.

2. If $\mathcal{M}$ is a monoid satisfying the struct finiteness condition, then this condition holds for every submonoid $\mathcal{M}'$ of $\mathcal{M}$.

Proof.

1. Suppose that

$$\forall u, v, w \in \mathcal{M} \ \ (v \neq 1 \ \text{ or } \ w \neq 1) \implies vuw \neq u.$$

Then, clearly,

$$\forall u, v, w \in \mathcal{M}' \ \ (v \neq 1 \ \text{ or } \ w \neq 1) \implies vuw \neq u.$$

2. Suppose that

$$\forall u \in \mathcal{M} \ \ |\{v \in \mathcal{M} \mid \exists x, y \in \mathcal{M} \ u = xvy\}| < \infty.$$

Then for all $u \in \mathcal{M}'$,

$$\{v \in \mathcal{M}' \mid \exists x, y \in \mathcal{M}' \ u = xvy\} \subseteq \{v \in \mathcal{M} \mid \exists x, y \in \mathcal{M} \ u = xvy\},$$

hence the former set is finite.

■

Finally, for each of the two conditions, there exists at least one monoid that satisfies it, and also there exist at least one monoid that does not. Properties of monoids that are invariant, hereditary, and for which there exist monoids that satisfy them, as well as monoids that do not satisfy them, are called *Markov properties*.

According to [Bo93, Theorem 7.3.7], the question whether a Markov property holds for a monoid specified by a finite presentation $\langle \Sigma, I \rangle$ is undecidable in general. Thus, we have proved the following result:

Theorem 5. *The questions whether a semantic equivalence relation satisfies*

(a) the part/whole distinguishability condition

(b) the struct finiteness condition

are undecidable in general.

2.4 Struct tuples

The concept of a struct tuple plays a technical role in the definition of the next representational level (see Section 3.1.2). Here, we define struct tuples and prove several lemmas about them, most of which are *level-invariant*, i.e., their statements and proofs can be carried over to the higher representational levels without any changes. Throughout this section, we assume that an inductive structure $\langle \Pi, \mathcal{I} \rangle$ is fixed.

Definition 29. An equivalence class of formation n-tuples w.r.t. $\sim_{\mathcal{I}}$ is called a **struct n-tuple**. A struct n-tuple containing formation tuple $D = [\alpha_1, \ldots, \alpha_n]$ will be denoted $[D]$ or $[\![\alpha_1, \ldots, \alpha_n]\!]$. ▶

By definition, when $n = 1$, the concept of a struct n-tuple coincides with that of a struct. When $n > 1$, a struct n-tuple is not the same thing as an n-tuple of structs, since a struct n-tuple consists of formation n-tuples, each of which is different from a tuple of n formations, as it was pointed out in Section 2.2.1.

2.4.1 Projections of struct tuples

Next, we introduce the concept of projection for struct n-tuples, similarly to the case of formation tuples:

Definition 30. Let $S = [\![\gamma_1, \ldots, \gamma_n]\!]$ be a struct n-tuple, and let $\mathbf{i} = \langle i_1, \ldots, i_k \rangle$ be a subsequence of $\langle 1, 2, \ldots, n \rangle$.

The **projection of struct tuple** S onto $\mathbf{i}$ is defined as the set of projections of formation n-tuples from S onto $\mathbf{i}$:

$$S_{\mathbf{i}} \overset{\text{def}}{=} \{D_{\mathbf{i}} \mid D \in S\}.$$

▶

Lemma 24. **(Struct Projection)** For any struct n-tuple S and any non-empty subsequence $\mathbf{i} = \langle i_1, \ldots, i_k \rangle$ of $\langle 1, 2, \ldots, n \rangle$, the projection of S onto $\mathbf{i}$, $S_{\mathbf{i}}$, is a struct k-tuple.

Proof.

- Consider relation $\sim_{\mathbf{i}}$ on the set of formation n-tuples defined as

  $$D \sim_{\mathbf{i}} D' \iff D_{\mathbf{i}} \sim_{\mathcal{I}} D'_{\mathbf{i}}.$$

  Relation $\sim_{\mathbf{i}}$ is an equivalence relation and satisfies the following property:

  $$D \overset{\mathcal{I}}{\leftrightarrow} D' \implies D \sim_{\mathbf{i}} D'.$$

  Since $\sim_{\mathcal{I}}$ is the minimal equivalence relation satisfying the above property, we have $\sim_{\mathcal{I}} \subseteq \sim_{\mathbf{i}}$.

  Let $D, D' \in S_{\mathbf{i}}$. Then, by definition of struct projection, there exist formation tuples $\tilde{D}, \tilde{D}' \in S$ such that $\tilde{D}_{\mathbf{i}} = D$, $\tilde{D}'_{\mathbf{i}} = D'$. Since $\tilde{D}, \tilde{D}' \in S$, $\tilde{D} \sim_{\mathcal{I}} \tilde{D}'$, which implies, according to the fact that $\sim_{\mathcal{I}} \subseteq \sim_{\mathbf{i}}$, that $\tilde{D} \sim_{\mathbf{i}} \tilde{D}'$. By definition of $\sim_{\mathbf{i}}$, this is equivalent to $D_{\mathbf{i}} \sim_{\mathcal{I}} D'_{\mathbf{i}}$. Hence, $S_{\mathbf{i}} \subseteq [D]$.

- Vice versa, let $D \in S_{\mathbf{i}}$ and $D' \sim_{\mathcal{I}} D$. Let $\mathbf{j} = \langle 1, 2, \ldots, n \rangle \setminus \mathbf{i}$. By definition of struct projection, there exists formation $\tilde{D} \in S$ such that $\tilde{D}_{\mathbf{i}} = D$. Take any composite tuple $\tilde{C} = \langle \alpha_1, \ldots, \alpha_n \rangle \in \tilde{D}$, and let $C = \tilde{C}_{\mathbf{i}}$ be its projection onto $\mathbf{i}$. For any k-tuple of composites $C' = \langle \alpha'_1, \ldots, \alpha'_k \rangle$, denote by $\tilde{C}/C'$ the n-tuple of composites such that

  $$(\tilde{C}/C')_{\mathbf{i}} = C' \text{ and } (\tilde{C}/C')_{\mathbf{j}} = \tilde{C}_{\mathbf{j}},$$

i.e. $\tilde{C}/C'$ is obtained from $\tilde{C}$ by replacing all $\mathbf{i}$-components in $\tilde{C}$ with the corresponding composites from C'; obviously, since $C = \tilde{C}_{\mathbf{i}}$, we have $\tilde{C}/C = \tilde{C}$.

Consider the following relation R on the set of formation k-tuples:

$$\begin{aligned} R \overset{\text{def}}{=} \{(E, E') \quad | \quad & \forall\, C \in E \;\; \exists\, C' \in E' \, [\tilde{C}/C] \sim_{\mathcal{I}} [\tilde{C}/C'] \quad \text{and} \\ & \forall\, C' \in E' \;\; \exists\, C \in E \, [\tilde{C}/C] \sim_{\mathcal{I}} [\tilde{C}/C']\}. \end{aligned}$$

Relation R is an equivalence relation. Moreover, if $E \overset{\mathcal{I}}{\leftrightarrow} E'$, then, according to Lemma 15, $(E, E') \in R$. Since, by definition of semantic equivalence, relation $\sim_{\mathcal{I}}$ on formation k-tuples is the minimal equivalence relation containing $\overset{\mathcal{I}}{\leftrightarrow}$, we obtain $\sim_{\mathcal{I}} \subseteq R$, which implies that $(D, D') \in R$. Now, apply the part

$$\forall\, C \in E \;\; \exists\, C' \in E' \, [\tilde{C}/C] \sim_{\mathcal{I}} [\tilde{C}/C']$$

of the definition of R to $E = D$, $E' = D'$ and obtain that there exists C' such that $[\tilde{C}/C'] \in S = [\![\tilde{C}/C]\!]$ and $[\tilde{C}/C']_{\mathbf{i}} = D'$. Which implies that $D' \in S_{\mathbf{i}}$ and $[D] \subseteq S_{\mathbf{i}}$.

Altogether, $S_{\mathbf{i}} = [D]$. ■

In particular, the above lemma implies that projection of a struct n-tuple onto a single coordinate $i \in [1, n]$ is a struct. The following lemma establishes a connection between a struct n-tuple S and the n-tuple of its projections onto coordinates $\langle S_1, \dots, S_n \rangle$.

Lemma 25. For any struct n-tuple S,

$$S_1 \times \cdots \times S_n = \{\langle D_1, \dots, D_n \rangle \mid D \in S\}.[13]$$

[13]Note that a similar lemma for formation n-tuples would not hold.

Proof. The inclusion

$$S_1 \times \cdots \times S_n \supseteq \{\langle D_1, \ldots, D_n \rangle \mid D \in S\}$$

follows from the definition of projection, according to which for all $i \in [1, n]$, $S_i = \{D_i \mid D \in S\}$.

The inverse inclusion can be reformulated as follows: for any formations $\bar{\alpha}_1 \in S_1, \ldots, \bar{\alpha}_n \in S_n$, there exist composites $\alpha_i \in \bar{\alpha}_i$ $(i \in [1, n])$ such that $[\alpha_1, \ldots, \alpha_n] \in S$.

Since $\bar{\alpha}_1 \in S_1$, by definition of projection, there exists formation n-tuple $[C] \in S$, where $C = \langle \beta_1, \ldots, \beta_n \rangle$ is an n-tuple of composites, such that $\bar{\beta}_1 = \bar{\alpha}_1$. Set $\alpha_1 = \beta_1$, then

$$C = \langle \beta_1, \ldots, \beta_n \rangle = \langle \alpha_1, \beta_2, \beta_3, \ldots, \beta_n \rangle.$$

Consider projection of C onto the second component, and for any composite α_2, let $C/\alpha_2 = \langle \alpha_1, \alpha_2, \beta_3, \ldots, \beta_n \rangle$ as in the proof of the previous lemma, in which we have also shown that there exists $\alpha_2 \in \bar{\alpha}_2$ such that $[C/\alpha_2] \sim_{\mathcal{I}} [C]$.

Continuing this process, consider projections of C onto the third, …, n-th components and construct a formation n-tuple $[\alpha_1, \ldots, \alpha_n]$ such that $\alpha_i \in \bar{\alpha}_i$ for all $i \in [1, n]$ and $[\alpha_1, \ldots, \alpha_n] \sim_{\mathcal{I}} [C]$, which implies that $[\alpha_1, \ldots, \alpha_n] \in S$. ■

The above lemma admits the following generalization: if $\mathbf{i}_1 \sqcup \ldots \sqcup \mathbf{i}_k = [1, n]$, then

$$S_{\mathbf{i}_1} \times \ldots \times S_{\mathbf{i}_k} = \{\langle D_{\mathbf{i}_1}, \ldots, D_{\mathbf{i}_k} \rangle \mid D \in S\}.$$

Lemma 26. If $\sim_{\mathcal{I}}$ satisfies the struct finiteness condition, then it also satisfies the part/whole distinguishability condition.

Proof. Suppose, the part/whole distinguishability does not hold, i.e., there exist formations $\bar{\alpha} \neq \bar{\beta}$ such that $\bar{\alpha} \sim_{\mathcal{I}} \bar{\beta}$ and $\bar{\alpha} \preceq \bar{\beta}$.

By definition of $\preceq$, there exist $\alpha \in \bar{\alpha}$, $\beta \in \bar{\beta}$ and γ_1, γ_2, not both empty, such that $\beta = \gamma_1 \lhd \alpha \lhd \gamma_2$. Since $\bar{\alpha} \sim_{\mathcal{I}} \bar{\beta}$, we obtain $[\![\alpha]\!] = [\![\gamma_1 \lhd \alpha \lhd \gamma_2]\!]$.

According to the Struct Projection lemma, there exists γ'_1 such that

$$[\![\gamma_1, \alpha]\!] = [\![\gamma'_1, \gamma_1 \lhd \alpha \lhd \gamma_2]\!].$$

Since $\langle \gamma_1, \alpha \rangle$ satisfies the composition condition, by Lemma 18, $\langle \gamma'_1, \gamma_1 \lhd \alpha \lhd \gamma_2 \rangle$ must satisfy the weak composition condition, which implies that there exists a site replacement $h_1 : \text{sites}(\gamma'_1) \to \mathcal{S}$ such that $\langle \gamma'_1\langle h_1 \rangle, \gamma_1 \lhd \alpha \lhd \gamma_2 \rangle$ satisfies the composition condition and

$$[\![\gamma_1 \lhd \alpha]\!] = [\![\gamma'_1\langle h_1 \rangle \lhd \gamma_1 \lhd \alpha \lhd \gamma_2]\!].$$

Similarly, there exists composite γ'_2 and a site replacement $h_2 : \text{sites}(\gamma'_2) \to \mathcal{S}$ such that

$$[\![\gamma_1 \lhd \alpha \lhd \gamma_2]\!] = [\![\gamma'_1\langle h_1 \rangle \lhd \gamma_1 \lhd \alpha \lhd \gamma_2 \lhd \gamma'_2\langle h_2 \rangle]\!].$$

Note that γ'_1 and γ'_2 cannot be both empty, therefore,

$$[\gamma'_1\langle h_1 \rangle \lhd \gamma_1 \lhd \alpha \lhd \gamma_2 \lhd \gamma'_2\langle h_2 \rangle] \neq [\gamma_1 \lhd \alpha \lhd \gamma_2].$$

The process can be continued and, as a consequence, we have an infinite sequence of distinct formations

$$[\alpha],\ [\gamma_1 \lhd \alpha \lhd \gamma_2],\ [\gamma'_1\langle h_1 \rangle \lhd \gamma_1 \lhd \alpha \lhd \gamma_2 \lhd \gamma'_2\langle h_2 \rangle], \ldots,$$

each of which belongs to the struct $[\![\alpha]\!]$, which contradicts the struct finiteness condition. ■

2.4.2 Finiteness of struct tuples

The next theorem shows that the struct finiteness condition can be automatically extended to struct n-tuples.

Theorem 6. *If the struct finiteness condition holds, then the number of formation n-tuples in any struct n-tuple is finite.*

Proof. According to Lemma 25,

$$\{\langle D_1, \ldots, D_n \rangle \mid D \in S\} = S_1 \times \cdots \times S_n,$$

and the Cartesian product on the right is finite due to the struct finiteness condition.

According to Theorem 1, for any set of formations $\bar{\alpha}_1, \ldots, \bar{\alpha}_n$, the following set

$$\{D \mid D_1 = \bar{\alpha}_1, \ldots, D_n = \bar{\alpha}_n\}$$

is also finite, which concludes the proof. ■

Theorem 7. *For any structs $\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_n$, the following set of struct n-tuples*

$$\{S \mid S_1 = \boldsymbol{\gamma}_1, \ldots, S_n = \boldsymbol{\gamma}_n\}$$

is finite.

Proof. Fix any formations $\bar{\gamma}_1 \in \boldsymbol{\gamma}_1, \ldots, \bar{\gamma}_n \in \boldsymbol{\gamma}_n$. For every struct tuple S such that $S_1 = \boldsymbol{\gamma}_1, \ldots, S_n = \boldsymbol{\gamma}_n$, according to Lemma 25, there exists a formation tuple $D \in S$ such that $D_1 = \bar{\gamma}_1, \ldots, D_n = \bar{\gamma}_n$.

Since distinct struct tuples, as equivalence classes of formation tuples, do not intersect, and since, by Theorem 1, the set of projection tuples D having given projections $\bar{\gamma}_1, \ldots, \bar{\gamma}_n$ is finite, we obtain that the set of struct tuples S such that $S_1 = \boldsymbol{\gamma}_1, \ldots, S_n = \boldsymbol{\gamma}_n$ is also finite. ■

Note that the above theorem remains true even if the struct finiteness condition is not imposed.

2.4.3 Part/whole relation on struct tuples

The part/whole relation can be extended to composite, formation, and struct tuples as follows:

Definition 31. For two composite n-tuples $\langle \alpha_1, \ldots, \alpha_n \rangle$ and $\langle \alpha'_1, \ldots, \alpha'_n \rangle$, we shall write

$$\langle \alpha_1, \ldots, \alpha_n \rangle \preceq \langle \alpha'_1, \ldots, \alpha'_n \rangle,$$

if for all $i \in [1, n]$, $\alpha_i \preceq \alpha'_i$.

For two formation n-tuples D and D', we shall write $D \preceq D'$, if there exist composite n-tuples $C \in D$, $C' \in D'$ such that $C \preceq C'$.

Finally, for two struct n-tuples S and S', we shall write $S \preceq S'$, if there exist formation n-tuples $D \in S$, $D' \in S'$ such that $D \preceq D'$. ▶

Theorem 8. *If the struct finiteness condition holds, then for any struct n-tuple S, the set of its parts*

$$\text{Parts}(S) \overset{\text{def}}{=} \{S' \mid S' \preceq S\}$$

is finite.

Proof. If $S' \preceq S$, then for all $i \in [1, n]$, $S'_i \preceq S_i$. According to Theorem 2, for every $i \in [1, n]$, the set $\text{Parts}(S_i)$ is finite. Hence, by Theorem 7, $\text{Parts}(S)$ is finite as well. ■

2.4.4 Composable formation tuples and struct tuples

Consider *composable* struct n-tuples, i.e., the ones whose components can be *composed* into a single struct. For composable tuples, we define the concept of *result* of composition and prove its basic properties. Later on, composable struct n-tuples will serve as a basis for the definition of first level composites.

We begin our definitions with the auxiliary concepts of 1-composable composite and formation n-tuples.

Definition 32. A composite n-tuple $C = \langle \alpha_1, \ldots, \alpha_n \rangle$ will be called **1-composable**, if α_1 and α_2 satisfy the composition condition (see Def. 4). The 1-composition of C is defined as the following composite $(n-1)$-tuple:

$$K(C) \overset{\text{def}}{=} \langle \alpha_1 \lhd \alpha_2, \alpha_3, \ldots, \alpha_n \rangle.$$

A formation n-tuple D will be called **1-composable**, if there exists a 1-composable composite n-tuple $C \in D$. Then, the 1-composition of D, $K(D)$ is the formation $(n-1)$-tuple $[K(C)]$. ▶

The correctness of the above definition for formation n-tuples is justified by the following lemma:

Lemma 27. If D is 1-composable, then every composite n-tuple $C \in D$ is 1-composable and for all $C, C' \in D$, $[K(C)] = [K(C')]$.

Proof. The statement of the lemma directly follows from Lemma 9. ■

Next, define 1-composable struct n-tuples.

Definition 33. A struct n-tuple S will be called **1-composable**, if there exists a 1-composable formation n-tuple $D \in S$. Then, the **1-composition** of S, $K(S)$, is defined as the struct $(n-1)$-tuple $[K(D)]$. ▶

For the correctness of the above definition, we have to show that $K(S)$ is uniquely specified by S:

Lemma 28. If D, D' are 1-composable formation n-tuples and $D \sim_{\mathcal{I}} D'$, then $K(D) \sim_{\mathcal{I}} K(D')$.

Proof. This lemma is a slight extension of Lemma 18, which consists in the replacement of formation pairs by arbitrary formation n-tuples. So, we will only highlight the consequent changes in the proof.

First, we extend the definition of weak composition to arbitrary formation tuples: a formation tuple $D = [\alpha_1, \alpha_2, \dots, \alpha_n]$ is **weakly 1-composable**, if the pair $[\alpha_1, \alpha_2]$ is weakly composable. The **weak 1-composition** of D, denoted $\tilde{K}(D)$ is defined as the following formation n-1 tuple:

$$\tilde{K}(D) \overset{\text{def}}{=} [\alpha_1\langle f\rangle, \alpha_2\langle g\rangle, \alpha_3, \dots, \alpha_n],$$

where f and g are site replacements such that $\langle \alpha_1\langle f\rangle, \alpha_2\langle g\rangle\rangle$ is composable,

$$f\big|_{\text{ext}(\alpha_1)} = \text{id}, \quad g\big|_{\text{ext}(\alpha_2)} = \text{id},$$

and

$$[\text{int}(\alpha_1\langle f\rangle) \cup \text{int}(\alpha_2\langle g\rangle)] \cap [\text{sites}(\alpha_3) \cup \dots \cup \text{sites}(\alpha_n)] = \varnothing.$$

The existence of such site replacements follows from the definition of weak composition, and the last condition ensures that formation tuple

$$[\alpha_1\langle f\rangle, \alpha_2\langle g\rangle, \alpha_3, \dots, \alpha_n]$$

does not depend on the choice of f and g.

Now, Lemma 17 can be reformulated for arbitrary formation tuples: Let $\bar{c}$ be a semantic identity. If $D \overset{\bar{c}}{\leftrightarrow} D'$ and D satisfies the weak composition condition, then so does D' and $D \overset{\bar{c}}{\leftrightarrow} D'$. The proof is similar and for this reason omitted.

Now, consider a relation R on the set of formation n-tuples defined as follows: $\langle D, D' \rangle \in R$ if and only if either both D and D' are weakly 1-composable and $\tilde{K}(D) \sim_{\mathcal{I}} \tilde{K}(D)$, or none of them is weakly 1-composable.

Relation R is an equivalence relation and contains $\overset{\mathcal{I}}{\leftrightarrow}$, hence it must also contain $\sim_{\mathcal{I}}$, which implies the statement of our lemma. ■

Composable struct n-tuples are introduced via the following recurrent definition:

Definition 34. A struct n-tuple S will be called **composable**, if either $n = 1$ (i.e. S is a struct) or S is 1-composable and $K(S)$ is composable.

The ***i*-composition** of S, $K_i(S)$, is defined recurrently as follows:

$$K_0(S) \overset{\text{def}}{=} S, \;\; K_i(S) \overset{\text{def}}{=} K_{i-1}(K(S)).$$

►

Note that a composable struct tuple does not necessarily contain any composable formation tuples (see Fig. 2.27).

Theorem 9. *If the struct finiteness condition holds, then for any struct $\boldsymbol{\gamma}$, the set*

$$\{[\![\alpha_1, \ldots, \alpha_n]\!] \mid K_{n-1}([\![\alpha_1, \ldots, \alpha_n]\!]) = \boldsymbol{\gamma}, \; \alpha_i \neq \lambda\}$$

is finite.

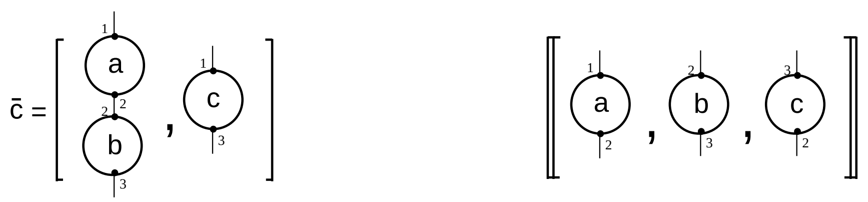

Figure 2.27: A composable struct tuple (right) in the inductive structure of sequences over $\Sigma = \{a, b, c, d\}$ with an additional identity (left). The struct tuple consists of only one formation tuple, which 1-composable but not composable.

Proof. First, show that the set

$$\{[\![\alpha_1, \alpha_2]\!] \mid K([\![\alpha_1, \alpha_2]\!]) = \boldsymbol{\gamma}\} \tag{$*$}$$

is finite. Indeed, by definition, $[\![\alpha_1, \alpha_2]\!]$ contains a composable formation pair $[\alpha_1, \alpha_2]$ such that $\boldsymbol{\gamma} = [\![\alpha_1 \lhd \alpha_2]\!]$. This implies that $[\![\alpha_1]\!], [\![\alpha_2]\!] \in \text{Parts}(\boldsymbol{\gamma})$, and, due to Theorem 2, the latter set is finite. Now, according to Theorem 7, set $(*)$ is finite as well.

It follows from Theorem 2 that the part/whole relation $\preceq$ on structs is a well-ordering, hence we can apply the principle of mathematical induction to it.

Assume that for all $\boldsymbol{\alpha} \prec \boldsymbol{\gamma}$, the set

$$\{[\![\alpha_1, \ldots, \alpha_n]\!] \mid K_{n-1}([\![\alpha_1, \ldots, \alpha_n]\!]) = \boldsymbol{\alpha},\ \alpha_i \neq \lambda\}$$

is finite. For each struct pair $[\![\alpha_1, \alpha_2]\!]$ such that $K([\![\alpha_1, \alpha_2]\!]) = \boldsymbol{\gamma}$ and $\alpha_2 \neq \lambda$, we have, due to part/whole distinguishability, that $[\![\alpha_1]\!], [\![\alpha_2]\!] \prec \boldsymbol{\gamma}$. We obtain that

$$\begin{aligned}
&\{[\![\alpha_1, \ldots, \alpha_n]\!] \mid K_{n-1}([\![\alpha_1, \ldots, \alpha_n]\!]) = \boldsymbol{\gamma},\ \alpha_i \neq \lambda\} \subseteq \\
&\quad \{\boldsymbol{\gamma}\} \cup \{[\![\alpha_1, \alpha_2]\!] \mid K([\![\alpha_1, \alpha_2]\!]) = \boldsymbol{\gamma}\} \cup \\
&\quad \{[\![\alpha_1, \ldots, \alpha_n]\!] \mid K_{n-2}([\![\alpha_1, \ldots, \alpha_{n-1}]\!]) \prec \boldsymbol{\gamma},\ \alpha_i \neq \lambda \text{ and } [\![\alpha_\mathrm{n}]\!] \prec \boldsymbol{\gamma}\},
\end{aligned}$$

which is finite by inductive assumption and according to the fact that the set $\{\boldsymbol{\alpha} \mid \boldsymbol{\alpha} \prec \boldsymbol{\gamma}\} \subset \text{Parts}(\boldsymbol{\gamma})$ is finite by Theorem 2. ■

2.4.5 Result of composition

For a composite tuple from a composable struct tuple, we also define the *result of composition.* First, we do it for a composite pair, and then extend the definition to arbitrary composite tuples.

Definition 35. For a composite pair $\langle \alpha, \beta \rangle$ such that the struct pair $[\![\alpha, \beta]\!]$ is composable, define the **result of composition** as any composite γ such that for all composites δ,

$$K([\![\alpha, \beta, \delta]\!]) = [\![\gamma, \delta]\!].$$

The set of all results for a composite pair $\langle \alpha, \beta \rangle$ will be denoted by $\mathrm{res}(\langle \alpha, \beta \rangle)$. ▶

Theorem 10. (Result existence) *If $[\![\alpha, \beta]\!]$ is a composable struct pair, then the set of results $\mathrm{res}(\langle \alpha, \beta \rangle)$ is non-empty.*

Proof. Since $[\![\alpha, \beta]\!]$ is composable, it contains a composite pair $\langle \alpha', \beta' \rangle$ satisfying the composition condition.

Applying the Struct Projection lemma (L. 24), we can find a composite γ such that

$$[\![\alpha, \beta, \gamma]\!] = [\![\alpha', \beta', \alpha' \lhd \beta']\!].$$

We will prove that for all composites δ,

$$K([\![\alpha, \beta, \delta]\!]) = [\![\gamma, \delta]\!].$$

Indeed, for any composite δ, due to the Struct Projection lemma, there exists δ' such that

$$[\![\alpha, \beta, \gamma, \delta]\!] = [\![\alpha', \beta', \alpha' \lhd \beta', \delta']\!],$$

which implies that

$$[\![\alpha, \beta, \delta]\!] = [\![\alpha', \beta', \delta']\!] \text{ and } [\![\alpha' \lhd \beta', \delta']\!] = [\![\gamma, \delta]\!].$$

Hence,

$$K([\![\alpha, \beta, \delta]\!]) = K([\![\alpha', \beta', \delta']\!]) = [\![\alpha' \lhd \beta', \delta']\!] = [\![\gamma, \delta]\!].$$

■

Definition 36. For a composite n-tuple $\langle \alpha_1, \ldots, \alpha_n \rangle$ such that the corresponding struct tuple $[\![\alpha_1, \ldots, \alpha_n]\!]$ is composable, the set of results is defined recurrently as

$$\operatorname{res}(\langle \alpha_1, \ldots, \alpha_n \rangle) \overset{\text{def}}{=} \bigcup_{\gamma \in \operatorname{res}(\langle \alpha_1, \alpha_2 \rangle)} \operatorname{res}(\langle \gamma, \alpha_3, \ldots, \alpha_n \rangle).$$

►

Lemma 29. **(Result Criterion)** If $[\![\alpha_1, \ldots, \alpha_n]\!]$ is composable, then

$$\operatorname{res}(\langle \alpha_1, \ldots, \alpha_n \rangle) = \{\gamma \mid \forall \delta \;\; K_{n-1}([\![\alpha_1, \ldots, \alpha_n, \delta]\!]) = [\![\gamma, \delta]\!]\}.$$

Proof. The proof is by induction on n. If $n = 2$, then the statement holds by definition of $\operatorname{res}(\langle \alpha_1, \alpha_2 \rangle)$. Suppose the statement holds for n, prove it for $n + 1$:

$$\begin{aligned}
&\operatorname{res}(\langle \alpha_1, \ldots, \alpha_{n+1} \rangle) = \\
&\quad \bigcup_{\beta \in \operatorname{res}(\langle \alpha_1, \alpha_2 \rangle)} \operatorname{res}(\langle \beta, \alpha_3, \ldots, \alpha_{n+1} \rangle) && = \\
&\quad \bigcup_{\beta \in \operatorname{res}(\langle \alpha_1, \alpha_2 \rangle)} \{\gamma \mid \forall \delta \;\; K_{n-1}([\![\beta, \alpha_3 \ldots, \alpha_{n+1}, \delta]\!]) = [\![\gamma, \delta]\!]\} && = \\
&\quad \bigcup_{\beta \in \operatorname{res}(\langle \alpha_1, \alpha_2 \rangle)} \{\gamma \mid \forall \delta \;\; K_{n-1}(K([\![\alpha_1, \alpha_2, \alpha_3 \ldots, \alpha_{n+1}, \delta]\!])) = [\![\gamma, \delta]\!]\} && = \\
&\quad \{\gamma \mid \forall \delta \;\; K_n([\![\alpha_1, \alpha_2, \alpha_3 \ldots, \alpha_{n+1}, \delta]\!] = [\![\gamma, \delta]\!]\}\,,
\end{aligned}$$

since, as an easy consequence of Def. 36, we have that if $\beta \in \operatorname{res}(\langle \alpha_1, \alpha_2 \rangle)$, then for any $\delta_1, \ldots, \delta_k$,

$$K([\![\langle \alpha_1, \alpha_2, \delta_1, \ldots, \delta_k \rangle]\!] = [\![\beta, \delta_1, \ldots, \delta_k]\!]).$$

■

It is straightforward that

$$\mathrm{res}(\langle \alpha_1, \ldots, \alpha_n \rangle) \subset K_{n-1}([\![\alpha_1, \ldots, \alpha_n]\!]).$$

Chapter 3

Representational hierarchy

This chapter consists of two sections. In Section 3.1, the first representational level is introduced, and a number of lemmas that ensure the consistency of this level with the basic one are proved (some of them have been prepared in Section 2.4). The statements and proofs of these lemmas are level-invariant, which allows to generalize the construction of the first representational level to an infinite representational hierarchy in Section 3.2.

Particularly important are the formation projection lemma (L. 33 in Section 3.1.5), struct projection lemma (L. 24 in Section 2.4.1), composite reconstruction lemma (L. 32 in Section 3.1.4), result existence theorem (Th. 10 in Section 2.4.5), and congruence lemmas (L. 18 in Section 2.2.4 and L. 37 in Section 3.1.6). These statements are meant to be generalized into a formal axiomatic system in future research. In order to illustrate this body of formal definitions and statements, we also give an example based on strings—which, however, will clearly go beyond the string data structure at higher levels.

3.1 First representational level

So far, we have introduced the concept of an inductive structure, which has two main constituents: formations and structs. Formations correspond to the processes that construct object representations and structs correspond to the representations themselves. The constructive processes are understood as sequences of constructive steps; so far, the steps have been *primitive* (indivisible) and *context-free*.

In nature, primitive constructive steps tend to *group together*, forming larger, *composite*, constructive steps. Due to their compositional structure, these constructive steps also become *context-dependent*. These composite constructive steps are called *transformations*.

I see the essence of evolution in nature in the creation of new composite constructive steps out of existing primitives. Examples of such creation are ubiquitous: atoms group into small molecules and ions, which then group into proteins, which in turn group into cells, followed by tissues, organs, organisms, families, species.[1] Similarly, groups of letters form syllables, which, in turn, form words, followed by sentences and paragraphs. The same kind of construction is very likely to occur with mental representations, which is the reason why, in my opinion, the term ‘language of thought” [Fo88] is very appropriate.

Each level of grouping is called a *representational level*; let us also agree to name a representational level after its primitives. For example, the representational level, whose primitives correspond to atoms, is called *atomic*.

Transformations at a particular representational level are composed from primi-

[1]This list by no means is exhaustive.

tives at this level. At the same time, these transformations play the role of primitives at the next representational level. For example, transformations of the atomic level are small chemical compounds, which, at the same time, play the role of primitives at the molecular level.

The basic representational level ("level 0") has been defined in Sections 2.1–2.3. The concepts of a primitive, composite, formation, and struct, from now on, will be accompanied by the attribute "basic-level". Here, we define the *first* representational level, i.e., introduce *first level primitives* (which are also basic-level transformations), followed by first-level composites, formations, and structs. In Section 3.2, this construction is generalized by induction to an infinite hierarchy of representational levels. The definitions are illustrated by an example based on the inductive structure of insertion strings.

Throughout this section we assume that an inductive structure $\langle \Pi, \mathcal{I} \rangle$ is fixed and will omit Π and $\mathcal{I}$ in the indices of the sets Γ_Π, $\bar{\Gamma}_\Pi$, Θ_Π, and $\sim_\mathcal{I}$.

3.1.1 Transformations

Definition 37. A pair of composites $\tau = \langle \alpha, \beta \rangle$ satisfying the composition condition and having $\beta \neq \lambda$ is called a **transformation** with the **context** α and **body** β, denoted

$$\text{context}(\tau) \overset{\text{def}}{=} \alpha, \qquad \text{body}(\tau) \overset{\text{def}}{=} \beta.$$

If $\alpha = \lambda$, the transformation is called **context free**. ▶

As mentioned above, transformations serve the role of primitives at the next level and, consequently, form composites. For a sequence of basic level primitives, the existence of their composition is determined by the attachment condition expressed

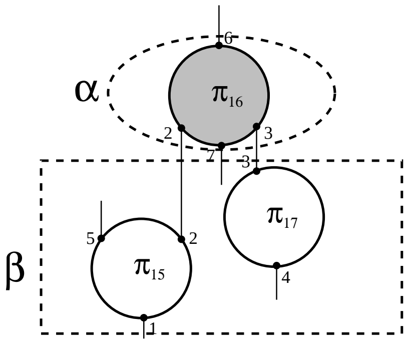

Figure 3.1: Pictorial representation of a transformation $\tau = \langle\langle\pi_{16}\rangle, \langle\pi_{15}, \pi_{17}\rangle\rangle$.

in terms of *sets* of sites. At the first level, things become more complicated, since first level primitives (i.e., basic level transformations) are context-dependent.

Definition 38. A transformation $\tau = \langle\alpha, \beta\rangle$ is **applicable** to a composite γ, if $\alpha \preceq \gamma$ and the pair $\langle\gamma, \beta\rangle$ satisfies the composition condition. If τ is applicable to γ, the **result of application** of τ to γ (Fig. 3.2) is defined as the composite

$$\gamma \lhd \tau \overset{\text{def}}{=} \gamma \lhd \beta.^{2}$$

►

According to the above definition, application of a transformation $\tau = (\alpha, \beta)$ to a composite γ can be viewed as composition of γ and β subject to α being a part of γ. The transformation does not *delete* any parts from the composite, and therefore will be called a *constructive* transformation. The reason of imposing such restriction is the requirement of strong acyclicity, which has been discussed in Section 2.3.4 in relation with evolutionary processes, and has already resulted in imposing the

[2] In this formula, the same notation ($\lhd$) is used for two different operations: application of a transformation to a composite (left) and composition of two composites (right). This should not lead to confusion, because composites can be thought of as context-free transformations.

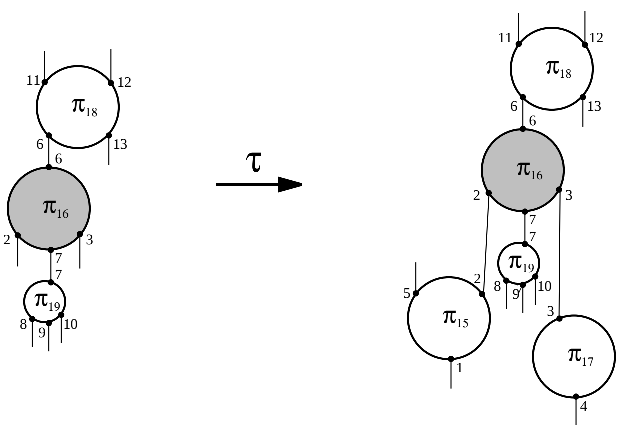

Figure 3.2: Application of transformation τ from Fig. 3.1 to a composite $\gamma = \langle \pi_{18}, \pi_{16}, \pi_{19} \rangle$. The shaded part of γ is the context of τ.

part/whole distinguishability condition on the semantic equivalence. Another reason is that non-deleting transformations are easier to learn, since the context and body of one transformation cannot be erased by other transformations and, therefore, remains explicitly present in the training set (the learning problem is discussed in Section 5.5). I am not saying that they are very easy to learn, given that the training set is represented by structs, which are quite complex, but what I *am* saying is that if additive transforms are not efficiently learnable, than it is hard to imagine which transforms are.

Let me point out some basic distinctions of the transformations from the Chomsky production rules. In a production rule (regardless of whether it is based on strings, graphs, or hypergraphs, and is context-free or context-sensitive), the context is a substring (subgraph) composed of *non-terminals*. Let us try to understand the meaning of non-terminals from the representational point of view. I claim that non-terminal symbols only *label* (name) certain constructive processes but do not *represent* them, because a non-terminal symbol on its own does not contain any information about the corresponding constructive process. For example, non-terminal symbol NOUN *labels* a constructive process that presumably takes place whenever a mental representation of a noun is generated, but this non-terminal does not *represent* this constructive process. As a consequence, when several constructive processes need to be described, each requires a separate non-terminal symbol to label it (one symbol will not enable us to distinguish between the processes). Hence, the number of non-terminals grows proportionally to the number of constructive processes, which results in excessively large grammars. This phenomenon has been observed, when context-free grammars were applied to the description of natural languages (see an example of such grammar at [Car01]). It is this inevitable growth of the number of non-terminals, which indicates that composite contexts are necessary.

Given this, the deficiency of conventional symbolic representations, such as strings, trees, and graphs, becomes even more apparent. Indeed, for context-sensitive grammars, even the parsing problem (to say nothing about grammatic inference) is intractable (see [RS97, Section 4.2.2.6] and [Kar72]).

I would like to conclude the argument for the necessity of preserving the constructive history (non-deleting transformations) and having composite contexts that address this history with the following quotation from Noam Chomsky's ground-setting paper [Cho65], in which formal grammars were first introduced:

"We cannot incorporate the rule (26) or anything like it in a grammar $[\Sigma, F]$ of phrase structure, because of certain fundamental limitations on such grammars. The essential property of rule (26) is that in order to apply it to sentences S_1 and S_2 to form the new sentence S_3 we must know not only the actual form of S_1 and S_2 but also their constituent structure—we must know not only the final shape of these sentences, but also their 'history of derivation'."

Noam Chomsky, "Syntactic Structures" [Cho65]

This quote can be considered as an indication that explicit representation constructive history is indeed important and that transformations need to refer to it via complex contexts. It is quite surprising to me that, even though "Syntactic Structures" were published half a century ago, this indication, so far, has not influenced much the formalisms that followed the line of phrase structure grammars.

3.1.2 First level primitives and composites

From now on, transformations will be called *first level primitives*; the set of all transformations is denoted $\dot{\Pi}$. Next, an attachment condition for a sequence of first level primitives is introduced, which leads to the definition of the concept of a *first level composite*. Note the difference between the *attachment* of two first level primitives, which results in a first level composite, and the *application* of a first level primitive to a basic level composite, which results in a basic level composite.

Definition 39. Let $S = \langle \dot{\pi}_1, \dot{\pi}_2, \ldots, \dot{\pi}_n \rangle$ be a sequence of first level primitives $\dot{\pi}_i =$

$\langle \alpha_i, \beta_i \rangle$ $(i \in [1, n])$.[3] Sequence S satisfies the **attachment condition**, if there exists composite γ, called an **initial composite**, such that the struct $(n+1)$-tuple $[\![\gamma, \beta_1, \ldots, \beta_n]\!]$ is composable and for all $i \in [1, n]$,

$$[\![\alpha_i, \beta_i]\!] \preceq K_{i-1}([\![\gamma, \beta_1, \ldots, \beta_i]\!]).^4$$

If S satisfies the attachment condition, it is called a **first level composite** and denoted by a dotted Greek letter, e.g., $\dot{\gamma}$.

The set of all initial composites for $\dot{\gamma}$ is denoted by $\text{init}(\dot{\gamma})$. Also, the following composite tuple

$$\text{body}(\dot{\gamma}) \stackrel{\text{def}}{=} \langle \text{body}(\dot{\pi}_1), \ldots, \text{body}(\dot{\pi}_n) \rangle$$

is called the **body** of first level composite $\dot{\gamma} = \langle \dot{\pi}_1, \ldots, \dot{\pi}_n \rangle$. ▶

Consider the inductive structure of insertion strings over the alphabet $\{a, b, c\}$ (see Section 2.3.4) and the running example in Sections 2.1–2.2. In Fig. 3.3, four sequences of first level primitives satisfying the attachment condition are shown.

In Fig. 3.4, the results (see Def. 36) of composite tuples $\langle \gamma \rangle, \langle \gamma, \beta_1 \rangle, \langle \gamma, \beta_1, \beta_2 \rangle$, corresponding to the four sequences in Fig. 3.3, are shown. The existence of the results means that the struct tuple $[\![\gamma, \beta_1, \ldots, \beta_n]\!]$ is composable and for all $i \in [1, n]$,

$$[\![\alpha_i, \beta_i]\!] \preceq K_{i-1}([\![\gamma, \beta_1, \ldots, \beta_i]\!]).$$

Note that in the example (d) $[\gamma_2] \sim [\gamma_1 \triangleleft \dot{\pi}_1]$.

Another example of a first level composite and its i-compositions is shown in Fig. 3.5. Other sequences of i-compositions can be obtained by changing the initial composite γ, e.g., by adding other basic level primitives to it (see Fig. 3.6).

[3] $[1, n] = \{1, 2, \ldots, n\}$.
[4] See Def. 34 for the definition of K_i.

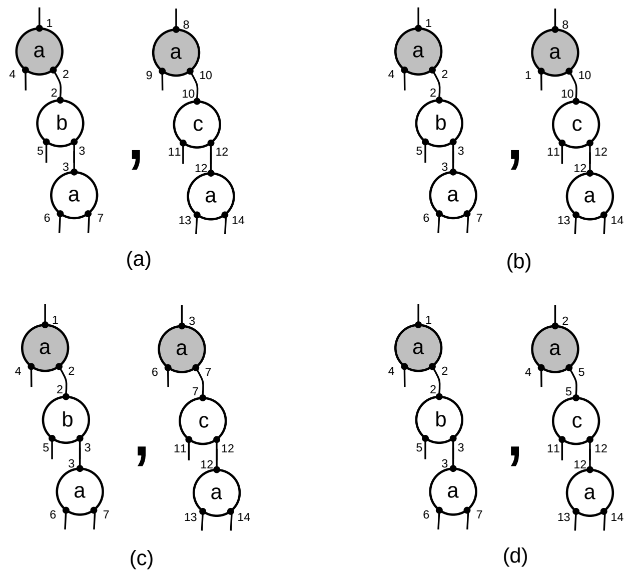

Figure 3.3: Four sequences (each of length 2) of first level primitives satisfying the attachment condition.

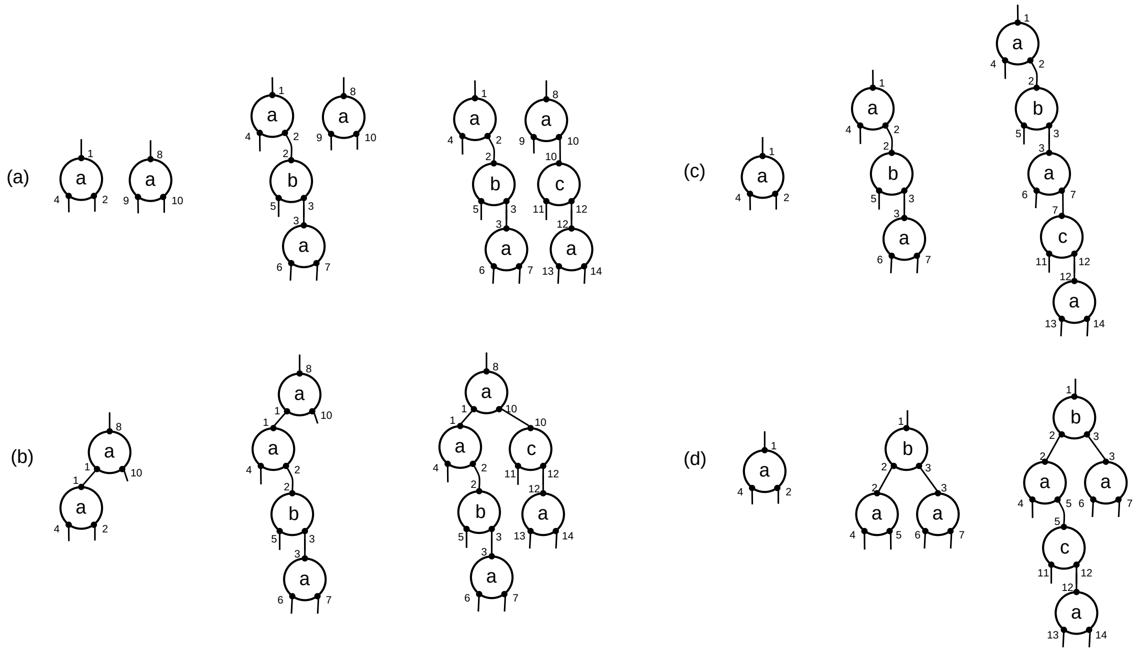

Figure 3.4: i-compositions of the first level composites from Fig. 3.3.

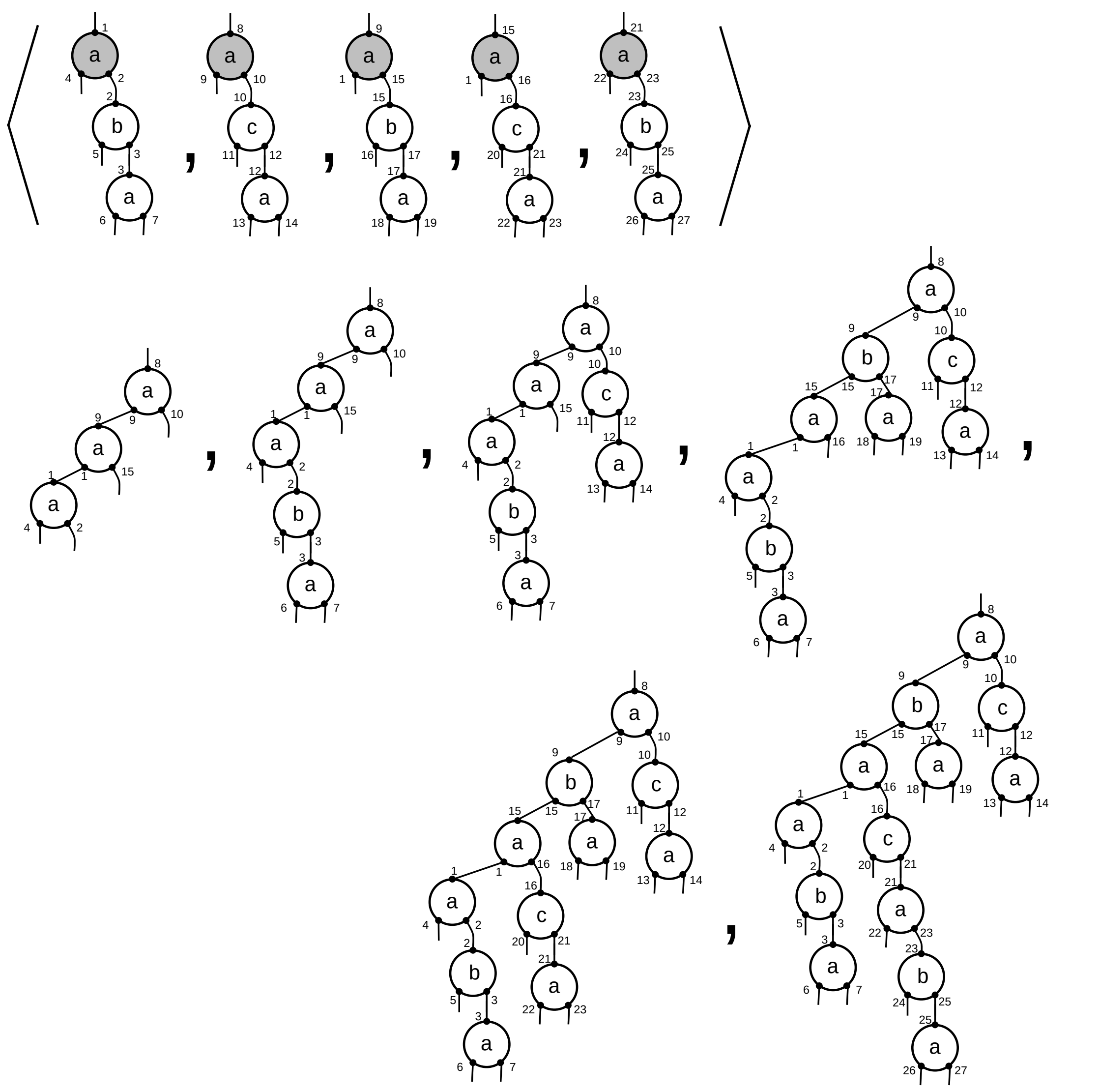

Figure 3.5: A first level composite and the corresponding sequence of i-compositions.

The definitions of composition and part/whole relation for first level composites are straightforward (compare to Lemma. 1 and Def. 8):

Definition 40. A pair of first level composites $\langle \dot{\alpha}, \dot{\beta} \rangle$ satisfies the **composition condition**, if the concatenation of sequences $\dot{\alpha}$ and $\dot{\beta}$ satisfies the attachment condition. In this case, the result of this concatenation is a first level composite, which will be

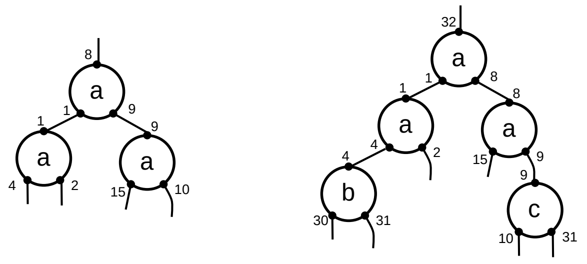

Figure 3.6: Some initial composites of the first level composite from Fig. 3.5.

called the **composition** of $\dot{\alpha}$ and $\dot{\beta}$ and denoted $\dot{\alpha} \lhd \dot{\beta}$. ▶

It is easy to show that first level composition operation is associative.

Definition 41. A first level composite $\dot{\alpha}$ is a **part** of a first level composite $\dot{\beta}$, denoted $\dot{\alpha} \preceq \dot{\beta}$, if $\dot{\beta} = \dot{\gamma}_1 \lhd \dot{\alpha} \lhd \dot{\gamma}_2$ for some composites $\dot{\gamma}_1, \dot{\gamma}_2$. The set of all parts of a composite $\dot{\beta}$ will be denoted Parts($\dot{\beta}$). ▶

Again, one can easily show that the part/whole relation is a partial ordering.

3.1.3 Sites of first level composites

In the ETS model, *sites*, in general, are responsible for the specification of whether and how two primitives are attachable (see Def. 2) and, also, whether two composites are composable (Lemma 6). At the basic level, this role is played by *sets* denoted init, term, and sites: according to the attachment and composition conditions, these sets contain all necessary information to specify whether and how primitives can be attached or composites can be composed. At the first level, a similar role is played by the contexts and bodies of transformations (these contexts and bodies are basic level composites). We have already introduced the set init($\dot{\gamma}$) of initial composites for a first level composite $\dot{\gamma}$, which now can be naturally called the set of **initial sites** of

$\dot{\gamma}$. Let us introduce the set of terminal sites for a first level composite as well:

Definition 42. For a first level composite $\dot{\gamma}$ and every $\gamma \in \text{init}(\dot{\gamma})$, the set

$$\text{term}_\gamma(\dot{\gamma}) \overset{\text{def}}{=} \text{res}(\langle \gamma, \text{body}(\dot{\gamma}) \rangle)$$

is called the **set of terminal composites** of $\dot{\gamma}$ corresponding to the initial composite γ.

The **set of site pairs** for $\dot{\gamma}$ is defined as the following set of composite pairs:

$$\text{sites}(\dot{\gamma}) \overset{\text{def}}{=} \{\langle \gamma, \gamma' \rangle \mid \gamma \in \text{init}(\dot{\gamma}),\ \gamma' \in \text{term}_\gamma(\dot{\gamma})\}.$$

For a site pair $\langle \gamma, \gamma' \rangle$, the first component is called an **initial composite** and the second a **terminal composite**. The sets of all initial and terminal composites for a first level composite $\dot{\gamma}$ will be denoted by $\text{init}(\dot{\gamma})$ and $\text{term}(\dot{\gamma})$, respectively. ▶

Lemma 30. If composites $\dot{\alpha}$ and $\dot{\beta}$ satisfy the composition condition, then

$$\text{sites}(\dot{\alpha} \lhd \dot{\beta}) = \{\langle \alpha, \gamma \rangle \mid \exists\, \beta \ \ \langle \alpha, \beta \rangle \in \text{sites}(\dot{\alpha}),\ \langle \beta, \gamma \rangle \in \text{sites}(\dot{\beta})\}.$$

Proof. Let $\langle \alpha, \gamma \rangle \in \text{sites}(\dot{\alpha} \lhd \dot{\beta})$. Take any $\beta \in \text{term}_\alpha(\dot{\alpha})$. Then, it follows from the definition of $\text{term}_\alpha(\dot{\alpha})$ and the Result criterion (L. 29) that $\beta \in \text{init}(\dot{\beta})$, hence $\langle \alpha, \beta \rangle \in \text{sites}(\dot{\alpha})$ and $\langle \beta, \gamma \rangle \in \text{sites}(\dot{\beta})$.

Vice versa, if $\langle \alpha, \beta \rangle \in \text{sites}(\dot{\alpha})$ and $\langle \beta, \gamma \rangle \in \text{sites}(\dot{\beta})$, then, obviously, $\langle \alpha, \gamma \rangle \in \text{sites}(\dot{\alpha} \lhd \dot{\beta})$. ■

Lemma 31. **(Composition Criterion)** A pair of first level composites $\langle \dot{\alpha}, \dot{\beta} \rangle$ satisfies the composition condition if and only if

$$\text{term}(\dot{\alpha}) \cap \text{init}(\dot{\beta}) \neq \varnothing.$$

Proof. Let $\dot\alpha = \langle \dot\pi_1, \dots, \dot\pi_m \rangle$, $\dot\beta = \langle \sigma_1, \dots, \sigma_n \rangle$ be a pair of first level composites satisfying the composition condition. Let $\gamma \in \mathrm{init}(\dot\alpha \lhd \dot\beta)$, then, by definition, the struct tuple $[\![\gamma, \mathrm{body}(\dot\alpha), \mathrm{body}(\dot\beta)]\!]$ is composable, hence so is $[\![\gamma, \mathrm{body}(\dot\alpha)]\!]$. Take any $\gamma' \in \mathrm{res}(\langle \gamma, \mathrm{body}(\dot\alpha) \rangle)$, then $\gamma' \in \mathrm{term}(\dot\alpha)$. By the definition of result, we have

$$[\![\gamma', \mathrm{body}(\dot\beta)]\!] = K_m([\![\gamma, \mathrm{body}(\dot\alpha), \mathrm{body}(\dot\beta)]\!]),$$

hence $\gamma' \in \mathrm{init}(\dot\beta)$. Therefore, $\gamma' \in \mathrm{term}(\dot\alpha) \cap \mathrm{init}(\dot\beta) \neq \varnothing$.

Vice versa, let $\gamma' \in \mathrm{term}(\dot\alpha) \cap \mathrm{init}(\dot\beta)$. Take any $\gamma \in \mathrm{init}(\dot\alpha)$ such that $\gamma' \in \mathrm{term}_\gamma(\dot\alpha)$. Since

$$K_m([\![\gamma, \mathrm{body}(\dot\alpha), \mathrm{body}(\dot\beta)]\!]) = [\![\gamma', \mathrm{body}(\dot\beta)]\!]),$$

the struct tuple $S = [\![\gamma, \mathrm{body}(\dot\alpha), \mathrm{body}(\dot\beta)]\!]$ is attachable and for all $i \in [1, m]$, $j \in [1, n]$,

$$[\![\mathrm{context}(\dot\pi_i), \mathrm{body}(\dot\pi_i)]\!] \preceq K_{i-1}(S), \qquad [\![\mathrm{context}(\dot\sigma_j), \mathrm{body}(\dot\sigma_j)]\!] \preceq K_{m+j-1}(S),$$

which means that $\dot\alpha$ and $\dot\beta$ satisfy the composition condition. ■

For a first level composite, there might exist several "minimal" initial composites, which is why the set $\mathrm{init}(\dot\gamma)$ cannot be replaced by a single composite, or even a single struct. For example, consider the inductive structure of insertion strings with an additional semantic identity shown in Fig. 3.7(a). The first level composite in Fig. 3.7(b) has at least two different initial composites (the corresponding attachment sequences are shown in Fig. 3.7(c)). Both of these initials are minimal, i.e., no primitives can be removed from them, and they also do not belong to the same struct.

The minimal terminal composite may also not be unique, as the example in Fig. 3.8 shows. Again, take the inductive structure of insertion strings plus two

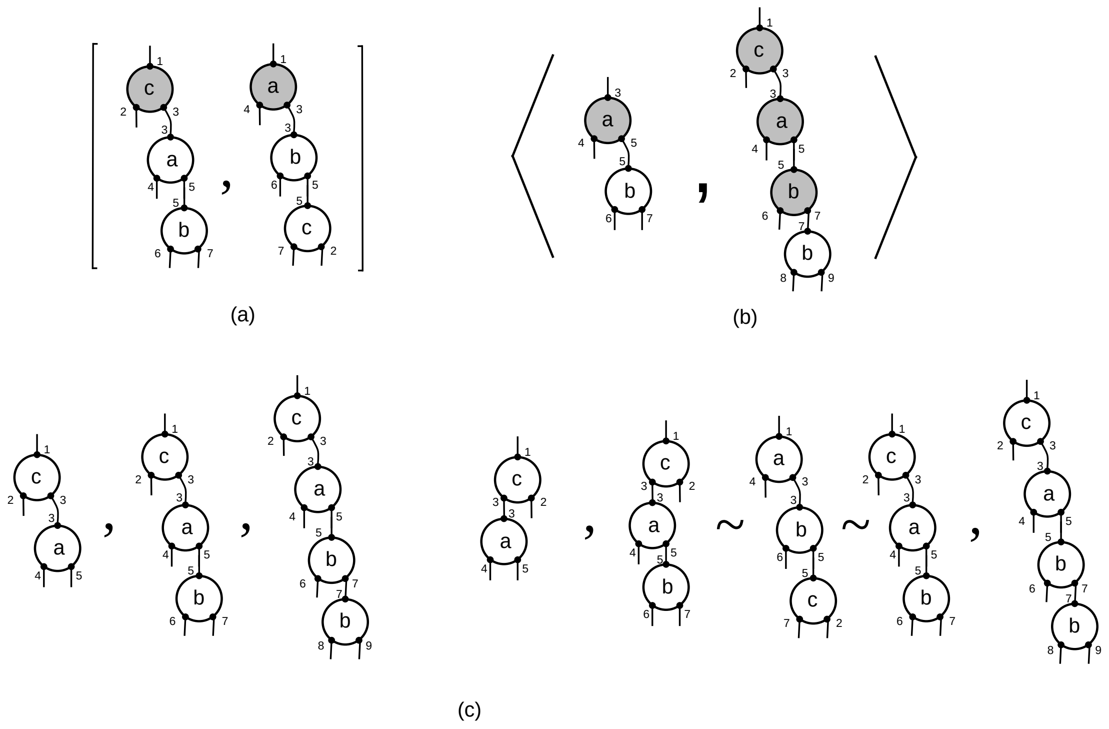

Figure 3.7: Different minimal initial composites of a first level composite.

semantic identities shown in Fig. 3.8(a); the first level composite in Fig. 3.8(b) has at least two different minimal terminal composites, shown in Fig. 3.8(c).

Non-uniqueness of initial and terminal composites, perhaps, reflects quite a natural phenomenon, which can be explained using the following chemical metaphor. Suppose that contexts “c” and “d” in Fig. 3.8 represent different catalysts of a chemical reaction, which is represented by the entire first level composite. Suppose that the reaction occurs, if either catalyst “c” or catalyst “d” is present. For the first representational level, which in this case is the *molecular* level, it only matters whether the reaction occurs, but the particular catalyst that facilitates it is not important, hence there is only one representation of the reaction described by the corresponding first level composite. At the basic, atomic, level, both initial and terminal composites

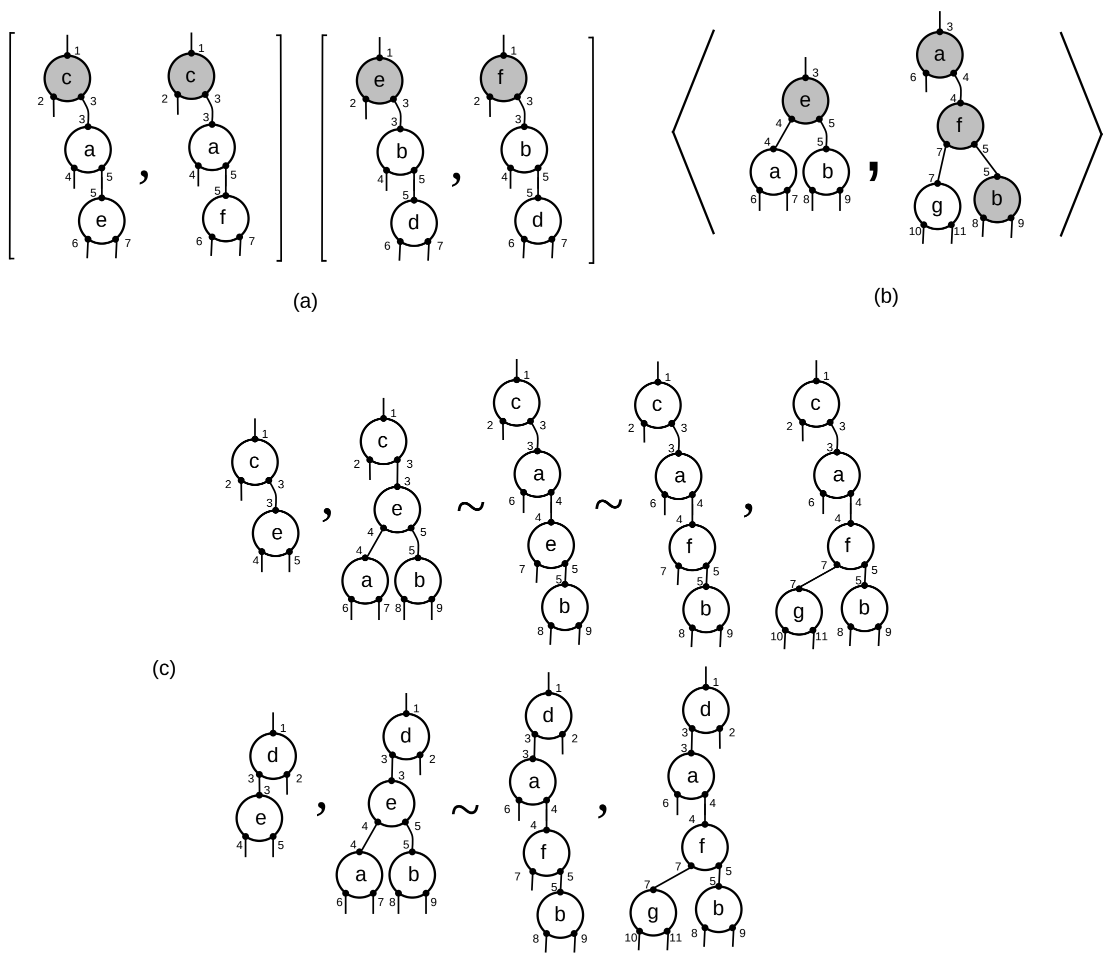

Figure 3.8: Different minimal terminal composites for a first level composite.

contain the information about the catalyst, hence they can vary.

3.1.4 First level formations

In the ETS model, a *formation* corresponds *uniquely* to a single constructive process. Thus, first level formations will correspond to the processes, whose steps are described by first level primitives. Note that multiple first level composites may correspond to a single constructive process. The reason for this is similar to the one at the basic level, where, by definition, composites transformable into each other via a site replacement corresponded to the same constructive processes.

The role of sites at the first level is played by basic level composites, which form the contexts and bodies of transformations. Hence, the first level composites corresponding to the same constructive processes, should be described via some sort of replacement of these basic level composites. Since the latter have structure (as opposed to basic level sites), they cannot be replaced arbitrarily; we suggest that they are replaced by semantically equivalent ones, which implies the following definition.

Definition 43. For first level composites $\dot{\alpha}$ and $\dot{\beta}$, of length n each, we shall write $\dot{\alpha} \approx \dot{\beta}$, if $[\![\mathrm{body}(\dot{\alpha})]\!] = [\![\mathrm{body}(\dot{\beta})]\!]$ and $[\![\dot{\alpha}_i]\!] = [\![\dot{\beta}_i]\!]$ $(i = 1, \dots, n)$.

A **first level formation** is an equivalence class of first level composites modulo $\approx$. The first level formation containing $\dot{\gamma}$ is denoted by $[\dot{\gamma}]$. ▶

Thus, the semantic equivalence relation plays the role of site replacement mappings at the first representational level.

In Fig. 3.9, two first level composites from a first level formation are shown.[5] This formation contains the composite from Fig. 3.5 as well.

The following lemma asserts that the set of all body-tuples of first level composites from a first level formation forms a struct n-tuple, as well as, for every $i \in [1, n]$, the set of all i-th first level primitives of these composites forms a struct pair.

Lemma 32. **(Composite Reconstruction)** For a first level composite $\dot{\gamma}_0$ of length n,

$$\{\mathrm{body}(\dot{\gamma}) \mid \dot{\gamma} \in [\dot{\gamma}_0]\} = [\![\mathrm{body}(\dot{\gamma}_0)]\!]$$

[5]Note that the only difference between these two composites is in the contexts of their last transformations.

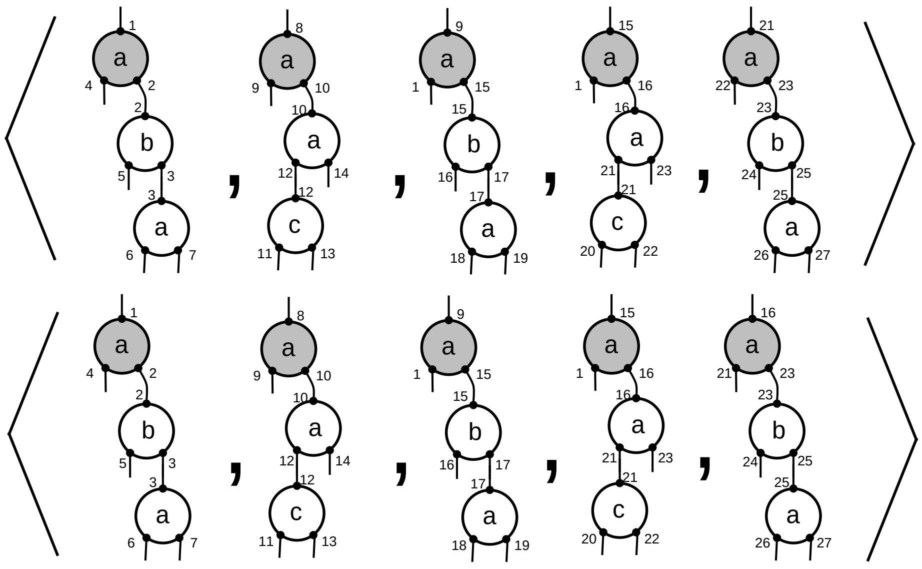

Figure 3.9: First level composites from the same first level formation.

and

$$\{\dot{\gamma}_i \mid \dot{\gamma} \in [\dot{\gamma}_0]\} = [\![\dot{\gamma}_{0_i}]\!], \qquad i = 1, \ldots, n.$$

Proof. By definition of the relation $\approx$ on first level composites, for all $\dot{\gamma} \in [\dot{\gamma}_0]$,

$$[\![\mathrm{body}(\dot{\gamma})]\!] \subseteq [\![\mathrm{body}(\dot{\gamma}_0)]\!],$$

which implies that

$$\{\mathrm{body}(\dot{\gamma}) \mid \dot{\gamma} \in [\dot{\gamma}_0]\} \subseteq [\![\mathrm{body}(\dot{\gamma}_0)]\!].$$

To prove the inverse inclusion, take any composite tuple

$$\langle \beta_1, \ldots, \beta_n \rangle \in [\![\mathrm{body}(\dot{\gamma}_0)]\!].$$

For each $i = 1, \ldots, n$, the i-th projection of the struct tuple $[\![\mathrm{body}(\dot{\gamma}_0)]\!]$ is a struct, due to the Struct Projection lemma (L. 24), therefore $\beta_i \in [\![\mathrm{body}(\dot{\gamma}_{0_i})]\!]$. Also, the second projection of the struct pair $[\![\dot{\gamma}_{0_i}]\!]$ is equal to $[\![\beta_i]\!]$, hence there exists composite

α_i such that $\langle \alpha_i, \beta_i \rangle \in [\![\dot{\gamma}_{0_i}]\!]$. Then the following sequence of first level primitives

$$S = \langle \langle \alpha_1, \beta_1 \rangle, \ldots, \langle \alpha_n, \beta_n \rangle \rangle$$

satisfies the attachment condition. Indeed, take any $\gamma_0 \in \mathrm{init}(\dot{\gamma}_0)$. There exists composite γ such that

$$[\![\gamma, \beta_1, \ldots, \beta_n]\!] = [\![\gamma_0, \mathrm{body}(\dot{\gamma}_0)]\!],$$

since, by Struct Projection lemma, the projection of the $(n{+}1)$-struct tuple $[\![\gamma_0, \mathrm{body}(\dot{\gamma}_0)]\!]$ onto coordinates

$$\mathbf{i} = \langle 2, \ldots, n+1 \rangle$$

is a struct n-tuple.

Since $\dot{\gamma}_0$ is a first level composite, struct tuple $[\![\gamma_0, \mathrm{body}(\dot{\gamma}_0)]\!]$ is composable, and so is $[\![\gamma, \beta_1, \ldots, \beta_n]\!]$, because it is the same struct $(n+1)$-tuple. Moreover, since i-composition of a struct n-tuple is uniquely defined (Lemma 28), we have that

$$[\![\alpha_i, \beta_i]\!] = [\![\dot{\gamma}_{0_i}]\!] \preceq K_{i-1}([\![\gamma_0, \mathrm{body}(\dot{\gamma}_0)]\!]_{[1,i+1]}) = K_{i-1}([\![\gamma, \beta_1, \ldots, \beta_i]\!]).$$

Thus, $\langle \langle \alpha_1, \beta_1 \rangle, \ldots, \langle \alpha_n, \beta_n \rangle \rangle$ is a first level composite, hence

$$\{\mathrm{body}(\dot{\gamma}) \mid \dot{\gamma} \in [\dot{\gamma}_0]\} \supseteq [\![\mathrm{body}(\dot{\gamma}_0)]\!].$$

To show that for all $i \in [1, n]$,

$$\{\dot{\gamma}_i \mid \dot{\gamma} \in [\dot{\gamma}]\} = [\![\dot{\gamma}_{0_i}]\!],$$

one can observe that, by Struct Projection lemma, for all $i \in [1, n]$, β_i can be any composite from the struct $[\![\mathrm{body}(\dot{\gamma}_{0_i})]\!]$, and then α_i can then be chosen arbitrarily so that $[\![\alpha_i, \beta_i]\!] = [\![\dot{\gamma}_{0_i}]\!]$. ■

One can show that if $\dot{\alpha} \approx \dot{\beta}$, then $\dot{\beta}$ can be obtained from $\dot{\alpha}$ through a finite sequence of direct conversions w.r.t. semantic identities, which are applied to the contexts and bodies of the first level primitives from $\dot{\alpha}$ and/or site replacements consistent across body($\dot{\alpha}$) and within each primitive of $\dot{\alpha}$. That is, relation $\approx$ at the first level is decidable.

3.1.5 First level formation tuples

The concept of formation tuple is extended to the first representational level. The meaning and the technical purpose of this concept are exactly the same as they were at the basic level, so we refer the reader to Section 2.2.1. This similarity also will be supported by a first level analogue of the Formation Projection lemma (see Lemma 13).

Definition 44. For first level composite tuples $\dot{A} = \langle \dot{\alpha}_1, \dots, \dot{\alpha}_n \rangle$ and $\dot{B} = \langle \dot{\beta}_1, \dots, \dot{\beta}_n \rangle$, we shall write $\dot{A} \approx \dot{B}$, if

1. $[\![\text{body}(\dot{\alpha}_1), \dots, \text{body}(\dot{\alpha}_n)]\!] = [\![\text{body}(\dot{\beta}_1), \dots, \text{body}(\dot{\beta}_n)]\!]$;
2. $[\![\dot{\alpha}_{k_i}]\!] = [\![\dot{\beta}_{k_i}]\!]$, $k \in [1, n]$, $i \in [1, |\dot{\alpha}_k|]$.

An equivalence class of first level composite tuples w.r.t. $\approx$ is called a **first level formation tuple**. The first level formation tuple containing $\dot{A}$ is denoted by $[\dot{A}]$. ▶

The definition of projection for first level formation tuples repeats the corresponding basic level definition (see Defs. 13,14) literally, so it is omitted here.

Lemma 33. For any first level formation tuple $\dot{D} = [\dot{\gamma}_1, \dots, \dot{\gamma}_n]$ and any non-empty subsequence $\mathbf{i} = \langle i_1, \dots, i_k \rangle$ of $\langle 1, 2, \dots, n \rangle$, the projection of $\dot{D}$ onto $\mathbf{i}$, $\dot{D}_{\mathbf{i}}$, is a formation k-tuple.

Proof. Take a composite tuple

$$\dot{A} = \langle \dot{\alpha}_1, \ldots, \dot{\alpha}_n \rangle \in \dot{D}$$

and let $\dot{A}' = \dot{A}_{\mathbf{i}}$.

1. First, prove that $\dot{D}_{\mathbf{i}} \subseteq [\dot{A}']$. Take any composite tuple $\dot{B}' \in \dot{D}_{\mathbf{i}}$, then, by definition of projection, there exists $\dot{B} \in \dot{D}$ such that $\dot{B}_{\mathbf{i}} = \dot{B}'$. Since both $\dot{A}$ and $\dot{B}$ belong to $\dot{D}$, $\dot{A} \approx \dot{B}$. Let

   $$\dot{B} = \langle \dot{\beta}_1, \ldots, \dot{\beta}_n \rangle,$$

   then by definition of the relation $\approx$ on composite tuples (Def. 44),

   $$[\![\text{body}(\dot{\alpha}_1), \ldots, \text{body}(\dot{\alpha}_n)]\!] = [\![\text{body}(\dot{\beta}_1), \ldots, \text{body}(\dot{\beta}_n)]\!]$$

   and

   $$[\![\dot{\alpha}_{p_j}]\!] = [\![\dot{\beta}_{p_j}]\!], \;\; p \in [1, n], \;\; j \in [1, |\dot{\alpha}_p|].$$

   According to the Struct Projection lemma (L. 24),

   $$[\![\text{body}(\dot{\alpha}_{i_1}), \ldots, \text{body}(\dot{\alpha}_{i_k})]\!] = [\![\text{body}(\dot{\beta}_{i_1}), \ldots, \text{body}(\dot{\beta}_{i_k})]\!],$$

   hence, by definition of relation $\approx$,

   $$\langle \dot{\alpha}_{i_1}, \ldots, \dot{\alpha}_{i_k} \rangle \approx \langle \dot{\beta}_{i_1}, \ldots, \dot{\beta}_{i_k} \rangle.$$

   It remains to observe that

   $$\begin{aligned} \dot{A}' &= \langle \dot{\alpha}_{i_1}, \ldots, \dot{\alpha}_{i_k} \rangle \\ \dot{B}' &= \langle \dot{\beta}_{i_1}, \ldots, \dot{\beta}_{i_k} \rangle, \end{aligned}$$

   hence $\dot{B}' \in [\dot{A}']$.

2. Second, prove that $[\dot{A}'] \subseteq \dot{D}_{\mathbf{i}}$. Take any $\dot{B}' \in [\dot{A}']$, i.e., $\dot{B}' \approx \dot{A}'$. Let

$$\dot{B}' = \langle \dot{\beta}'_1, \ldots, \dot{\beta}'_k \rangle.$$

Then, by definition of $\approx$ on composite tuples,

$$[\![\text{body}(\dot{\alpha}_{i_1}), \ldots, \text{body}(\dot{\alpha}_{i_k})]\!] = [\![\text{body}(\dot{\beta}_1), \ldots, \text{body}(\dot{\beta}_k)]\!]$$

and

$$[\![\dot{\alpha}_{i_{p_j}}]\!] = [\![\dot{\beta}_{p_j}]\!], \quad p \in [1, k], \quad j \in [1, |\dot{\alpha}_{i_p}|].$$

Note that

$$[\![\text{body}(\dot{\alpha}_{i_1}), \ldots, \text{body}(\dot{\alpha}_{i_k})]\!] = [\![\text{body}(\dot{\alpha}_1), \ldots, \text{body}(\dot{\alpha}_n)]\!]_{\mathbf{j}},$$

where $\mathbf{j}$ is the set of indices of elements in the composite tuple

$$\text{body}(\dot{\alpha}_1), \ldots, \text{body}(\dot{\alpha}_n)$$

that are occupied by

$$\text{body}(\dot{\alpha}_{i_1}), \ldots, \text{body}(\dot{\alpha}_{i_k}),$$

i.e.,

$$\mathbf{j} = \bigcup_{j=1}^{k} [q(i_j), q(i_j + 1) - 1], \quad q(i) = \sum_{j=1}^{i-1} |\dot{\alpha}_j|, \quad i \in [1, n].$$

Let

$$q = \sum_{j=1}^{n} |\dot{\alpha}_j|.$$

According to the Struct Projection lemma, there exists a basic level composite tuple

$$\langle \beta_1, \ldots, \beta_q \rangle \in [\![\text{body}(\dot{\alpha}_1), \ldots, \text{body}(\dot{\alpha}_n)]\!]$$

such that

$$\langle \beta_1, \ldots, \beta_q \rangle_{\mathbf{j}} = \langle \mathrm{body}(\dot\beta_1), \ldots, \mathrm{body}(\dot\beta_k) \rangle.$$

For $i \in [1, n] \setminus \{i_1, \ldots, i_k\}$, applying the Struct Projection lemma again, we obtain that

$$\langle \beta_{q(i)}, \ldots, \beta_{q(i+1)-1} \rangle \in [\![\mathrm{body}(\dot\alpha_i)]\!].$$

Hence, due to the Composite Reconstruction lemma (L. 32), there exists a first level composite $\dot\beta_i \in [\dot\alpha_i]$ such that

$$\mathrm{body}(\dot\beta_i) = \langle \beta_{q(i)}, \ldots, \beta_{q(i+1)-1} \rangle.$$

For $j \in \{1, \ldots, k\}$, let $\dot\beta_{i_j} \overset{\mathrm{def}}{=} \dot\beta'_j$. For the constructed composite tuple $\dot B = \langle \dot\beta_1, \ldots, \dot\beta_n \rangle$, we have $\dot B \approx \dot A$, therefore $\dot B \in \dot D$, as well as $\dot B_{\mathbf{i}} = \dot B'$. Hence, $\dot B' \in \dot D_{\mathbf{i}}$.

■

Lemma 34. If $\langle \dot\alpha_0, \dot\beta_0 \rangle \approx \langle \dot\alpha, \dot\beta \rangle$ and $\mathrm{sites}(\dot\alpha_0) = \mathrm{sites}(\dot\beta_0)$, then $\mathrm{sites}(\dot\alpha) = \mathrm{sites}(\dot\beta)$.

Proof. By definition of relation $\approx$,

$$[\![\mathrm{body}(\dot\alpha_0), \mathrm{body}(\dot\beta_0)]\!] = [\![\mathrm{body}(\dot\alpha), \mathrm{body}(\dot\beta)]\!].$$

Let $\langle \gamma, \gamma' \rangle \in \mathrm{sites}(\alpha)$. According to the Struct Projection lemma, there exist composites γ_0, γ'_0 such that

$$[\![\gamma_0, \gamma'_0, \mathrm{body}(\dot\alpha_0), \mathrm{body}(\dot\beta_0)]\!] = [\![\gamma, \gamma', \mathrm{body}(\dot\alpha), \mathrm{body}(\dot\beta)]\!],$$

which implies that

$$[\![\gamma_0, \gamma'_0, \mathrm{body}(\dot\alpha_0)]\!] = [\![\gamma, \gamma', \mathrm{body}(\dot\alpha)]\!]$$

and, hence, $\langle \gamma_0, \gamma_0' \rangle \in \text{sites}(\dot{\alpha}_0)$. Since $\text{sites}(\dot{\alpha}_0) = \text{sites}(\dot{\beta}_0)$, we obtain that $\langle \gamma_0, \gamma_0' \rangle \in \text{sites}(\dot{\beta}_0)$. Now the equality

$$[\![\gamma_0, \gamma_0', \text{body}(\dot{\beta}_0)]\!] = [\![\gamma, \gamma', \text{body}(\dot{\beta})]\!]$$

implies that $\langle \gamma, \gamma' \rangle \in \text{sites}(\dot{\beta})$.

Hence, $\text{sites}(\dot{\alpha}) \subseteq \text{sites}(\dot{\beta})$. The opposite inclusion can be shown symmetrically, hence $\text{sites}(\dot{\alpha}) = \text{sites}(\dot{\beta})$. ■

The following lemma expresses the fact that relation $\approx$ is a congruence w.r.t. composition.

Lemma 35. If $\langle \dot{\alpha}, \dot{\beta} \rangle$ satisfies the composition condition and $\langle \dot{\alpha}, \dot{\beta} \rangle \approx \langle \dot{\alpha}', \dot{\beta}' \rangle$, then $\langle \dot{\alpha}', \dot{\beta}' \rangle$ satisfies the composition condition as well and $\dot{\alpha} \lhd \dot{\beta} \approx \dot{\alpha}' \lhd \dot{\beta}'$.

Proof. Take any $\gamma \in \text{init}(\dot{\alpha} \lhd \dot{\beta})$. According to the definition of $\langle \dot{\alpha}, \dot{\beta} \rangle \approx \langle \dot{\alpha}', \dot{\beta}' \rangle$,

$$[\![\text{body}(\dot{\alpha}), \text{body}(\dot{\beta})]\!] = [\![\text{body}(\dot{\alpha}'), \text{body}(\dot{\beta}')]\!]. \qquad (*)$$

Hence, by Struct Projection lemma, there exists γ' such that

$$[\![\gamma, \text{body}(\dot{\alpha}), \text{body}(\dot{\beta})]\!] = [\![\gamma', \text{body}(\dot{\alpha}'), \text{body}(\dot{\beta}')]\!],$$

which implies that concatenation of sequences $\dot{\alpha}'$ and $\dot{\beta}'$ is a first level composite, i.e., $\dot{\alpha}$ and $\dot{\beta}$ satisfy the composition condition. The relation $\dot{\alpha} \lhd \dot{\beta} \approx \dot{\alpha}' \lhd \dot{\beta}'$ also follows from the equality $(*)$. ■

3.1.6 First level semantic identities and equivalence

The concepts of a semantic identity and semantic equivalence relation are also generalized to the first representational level. The basic logic of the definitions remains, unchanged, however, the few minor differences that occur are made explicit here.

For the basic level, we have defined a semantic identity as a formation pair $[\alpha, \beta]$ satisfying the following condition: for every composite pair $\langle \alpha, \beta \rangle \in [\alpha, \beta]$,

$$\mathrm{init}(\alpha) = \mathrm{init}(\beta), \;\; \mathrm{term}(\alpha) = \mathrm{term}(\beta).$$

For the same reasons (see Section 2.2.4), the same condition is imposed on the first level semantic identities, which now acquires the following form:

Definition 45. A first level formation pair $[\dot\alpha, \dot\beta]$ will be called a **first level semantic identity**, if for every composite pair $\langle \dot\alpha, \dot\beta \rangle \in [\dot\alpha, \dot\beta]$, $\mathrm{sites}(\dot\alpha) = \mathrm{sites}(\dot\beta)$. ▶

Note that, according to Lemma 34, the above condition is invariant w.r.t. relation $\approx$, hence, if $[\dot\alpha, \dot\beta]$ contains at least one composite pair with the equal sets of site pairs, then it is a semantic identity.

In order to define semantic equivalence, fix a set of semantic identities $\dot{\mathcal{I}}$, define the corresponding direct convertibility relation $\overset{\dot{\mathcal{I}}}{\leftrightarrow}$ and take its reflexive transitive closure to obtain the semantic equivalence relation $\sim_{\dot{\mathcal{I}}}$. These definitions repeat Defs. 15,16,17 literally, so I refer the reader to Sections 2.2.2–2.2.4. Here I only give an example and reprove the key lemmas.

In Fig. 3.10, five semantically equivalent first level formations are shown. The corresponding set of first level semantic identities $\dot{\mathcal{I}}$ contains *all* possible identities (this will be the case for all subsequent examples at the first representational level). The formations differ in the orders of attachments of first level primitives B_1, C, and B_2 (schematically shown on the right of each formation). These formations look exactly as basic level primitives of the insertion string inductive structure. There exist no other semantically equivalent formations composed of the same five first level primitive types.[6]. Although, semantically equivalent first level formations composed of *other* first level primitive types do exist, for example, the one in Fig. 3.11.

Lemma 36. If $\dot{\alpha} \overset{\dot{c}}{\leftrightarrow} \dot{\beta}$, where $[\dot{c}]$ is a semantic identity, then $[\dot{\alpha}, \dot{\beta}]$ is a semantic identity as well, i.e., $\text{sites}(\dot{\alpha}) = \text{sites}(\dot{\beta})$.

Proof. The proof directly follows from Lemma 30 and the definition of the relation $\overset{\dot{c}}{\leftrightarrow}$. ■

Lemma 37. First level semantic equivalence is a congruence w.r.t. composition, i.e., if $\langle \dot{\alpha}, \dot{\beta} \rangle$ satisfies the composition condition and $[\dot{\alpha}, \dot{\beta}] \sim_{\dot{\mathcal{I}}} [\dot{\alpha}', \dot{\beta}']$, then $\langle \dot{\alpha}', \dot{\beta}' \rangle$ satisfies the composition condition as well and

$$[\dot{\alpha} \lhd \dot{\beta}] \sim_{\dot{\mathcal{I}}} [\dot{\alpha}' \lhd \dot{\beta}'].$$

Proof. First, prove that if $[\dot{\alpha}, \dot{\beta}] \overset{[\dot{c}]}{\leftrightarrow} [\dot{\alpha}', \dot{\beta}']$, where $\dot{c}$ is a semantic identity, and $\langle \dot{\alpha}, \dot{\beta} \rangle$ satisfies the composition condition, then so does $\langle \dot{\alpha}', \dot{\beta}' \rangle$.

By definition, there exist composite pairs $\langle \dot{\alpha}_0, \dot{\beta}_0 \rangle \in [\dot{\alpha}, \dot{\beta}]$, $\langle \dot{\alpha}'_0, \dot{\beta}'_0 \rangle \in [\dot{\alpha}', \dot{\beta}']$, and $\dot{c}_0 \in [\dot{c}]$ such that $\langle \dot{\alpha}_0, \dot{\beta}_0 \rangle \overset{\dot{c}}{\leftrightarrow} \langle \dot{\alpha}'_0, \dot{\beta}'_0 \rangle$. According to Lemma 35, $\langle \dot{\alpha}_0, \dot{\beta}_0 \rangle$ satisfies the composition condition. It follows from Lemma 30 and the definition of the relation $\overset{\dot{c}_0}{\leftrightarrow}$ that so does $\langle \dot{\alpha}'_0, \dot{\beta}'_0 \rangle$, hence, according to Lemma 35, $\langle \dot{\alpha}', \dot{\beta}' \rangle$ satisfies the composition condition as well.

Also, immediately from the definition of $\overset{\dot{c}_0}{\leftrightarrow}$, we have that $\dot{\alpha}_0 \lhd \dot{\beta}_0 \overset{\dot{c}_0}{\leftrightarrow} \dot{\alpha}'_0 \lhd \dot{\beta}'_0$, hence, by definition of $\overset{[\dot{c}]}{\leftrightarrow}$, $[\dot{\alpha} \lhd \dot{\beta}] \overset{[\dot{c}]}{\leftrightarrow} [\dot{\alpha}' \lhd \dot{\beta}']$.

Consider a relation R on the set of formation pairs defined as follows:

$$\langle [\dot{\alpha}, \dot{\beta}], [\dot{\alpha}', \dot{\beta}'] \rangle \in R$$

[6]By a first level primitive type we mean a formation of the form $[\dot{\pi}]$, where $\dot{\pi}$ is a first level primitive.

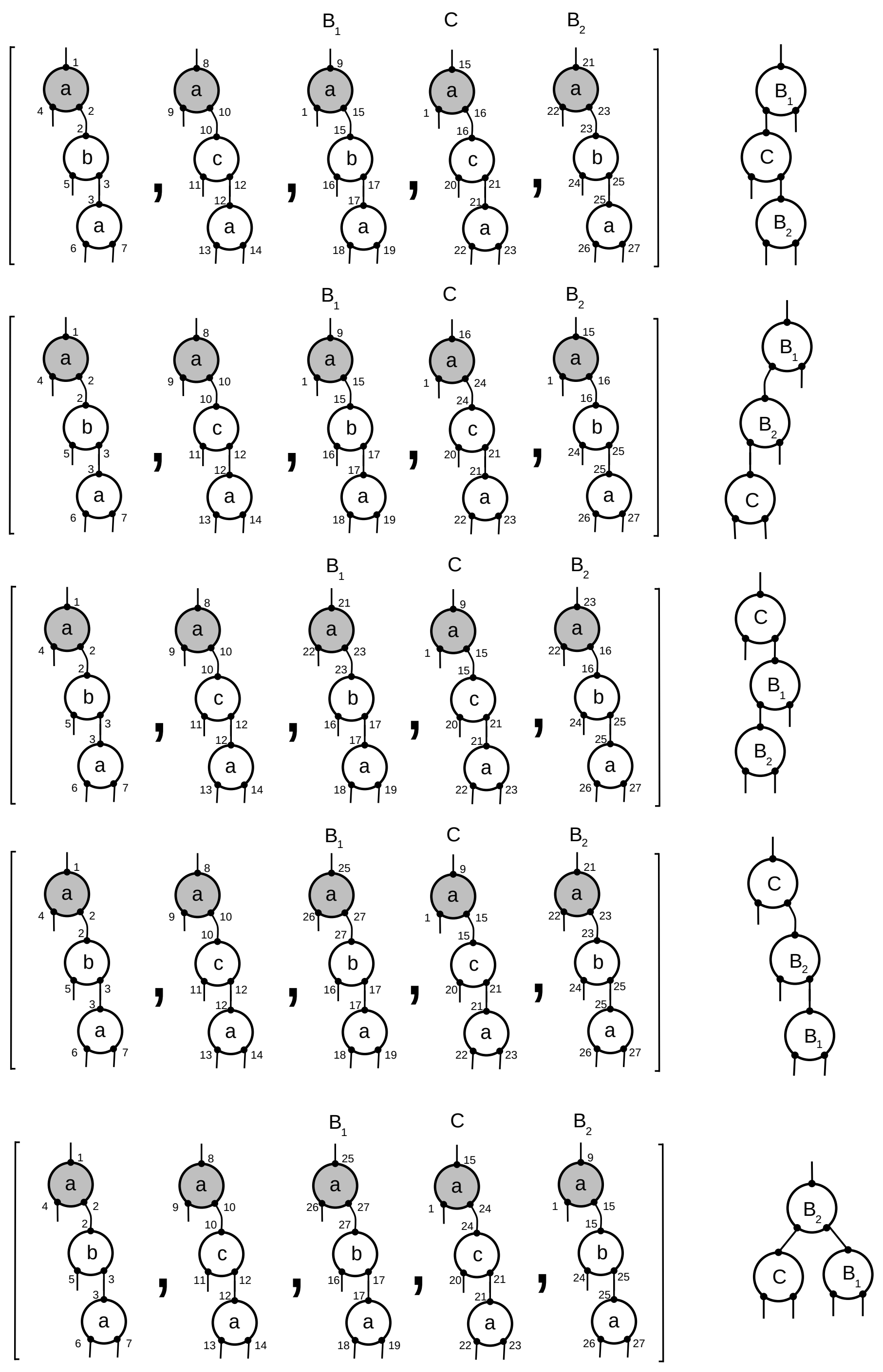

Figure 3.10: Semantically equivalent first level formations.

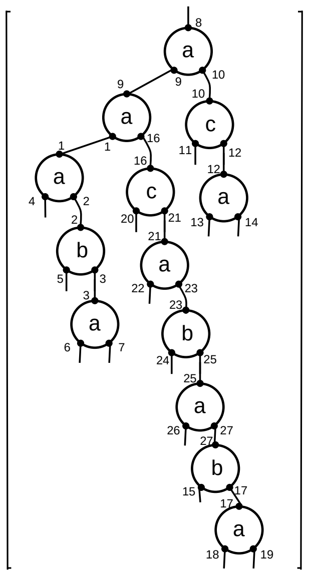

Figure 3.11: Another first level formation equivalent to the ones in Fig. 3.10 but composed of different primitive types.

if and only if either both $[\dot{\alpha}, \dot{\beta}]$ and $[\dot{\alpha}', \dot{\beta}']$ satisfy the composition condition and $[\dot{\alpha} \lhd \dot{\beta}] \sim_{\dot{\mathcal{I}}} [\dot{\alpha}' \lhd \dot{\beta}']$, or none of them does so.

Relation R is reflexive, symmetric, and transitive, hence an equivalence relation. Moreover, as shown above, R contains all pairs $\langle [\dot{\alpha}, \dot{\beta}], [\dot{\alpha}', \dot{\beta}'] \rangle$ such that $[\dot{\alpha}, \dot{\beta}] \overset{\dot{\mathcal{I}}}{\leftrightarrow} [\dot{\alpha}', \dot{\beta}']$. Since $\sim_{\dot{\mathcal{I}}}$ is, by definition, the minimal equivalence relation containing these pairs, $\sim_{\dot{\mathcal{I}}}$ is contained in R, which implies the statement of the lemma. ■

Note that the situation with congruence of the semantic equivalence relation w.r.t. composition is somewhat simpler at the first level than at the basic level: there is no need to introduce the auxiliary concept of weak composition, since the congruence already holds for the regular operation of composition.

Lemma 38. If $[\dot{\alpha}] \overset{[\dot{c}]}{\leftrightarrow} [\dot{\beta}]$, then

1. For all $\dot{c} \in [\dot{c}]$, there exist $\dot{\alpha} \in [\dot{\alpha}]$, $\dot{\beta} \in [\dot{\beta}]$ such that $\dot{\alpha} \overset{\dot{c}}{\leftrightarrow} \dot{\beta}$.

2. For all $\dot{\alpha} \in [\dot{\alpha}]$, there exist $\dot{\beta} \in [\dot{\beta}]$, $\dot{c} \in [\dot{c}]$ such that $\dot{\alpha} \overset{\dot{c}}{\leftrightarrow} \dot{\beta}$.

Proof. We shall only prove the second statement, since the proof of the first one is similar.

By definition of $[\dot{\alpha}] \overset{[\dot{c}]}{\leftrightarrow} [\dot{\beta}]$, there exist $\dot{\alpha}_0 \in [\dot{\alpha}]$, $\dot{\beta}_0 \in [\dot{\beta}]$, $\dot{c}_0 \in [\dot{c}]$ such that $\dot{\alpha}_0 \overset{\dot{c}_0}{\leftrightarrow} \dot{\beta}_0$.

According to the definition of $\overset{\dot{c}_0}{\leftrightarrow}$, this means that there exist composites $\dot{\gamma}_1, \dot{\gamma}_2$ such that

$$\begin{aligned}\dot{\alpha}_0 &= \dot{\gamma}_1 \lhd \dot{\gamma}_0 \lhd \dot{\gamma}_2 \\ \dot{\beta}_0 &= \dot{\gamma}_1 \lhd \dot{\gamma}'_0 \lhd \dot{\gamma}_2,\end{aligned}$$

where $\dot{c}_0 = \langle \dot{\gamma}_0, \dot{\gamma}'_0 \rangle$ or $\dot{c}_0 = \langle \dot{\gamma}'_0, \dot{\gamma}_0 \rangle$.

Let

$$\begin{aligned}\dot{\gamma}_1 &= \langle \dot{\pi}_1, \ldots, \dot{\pi}_i \rangle \\ \dot{\gamma}_0 &= \langle \dot{\pi}_{i+1}, \ldots, \dot{\pi}_j \rangle \\ \dot{\gamma}_2 &= \langle \dot{\pi}_{j+1}, \ldots, \dot{\pi}_n \rangle,\end{aligned}$$

then $\dot{\alpha}_0 = \langle \dot{\pi}_1, \ldots, \dot{\pi}_n \rangle$. Since $\dot{\alpha}_0 \approx \dot{\alpha}$, the lengths of sequences $\dot{\alpha}_0$ and $\dot{\alpha}$ have to be equal, hence $\dot{\alpha} = \langle \dot{\pi}'_1, \ldots, \dot{\pi}'_n \rangle$ for some primitives $\dot{\pi}'_1, \ldots, \dot{\pi}'_n$.

Take

$$\begin{aligned}\dot{\gamma}'_1 &= \langle \dot{\pi}'_1, \ldots, \dot{\pi}'_i \rangle \\ \dot{\gamma} &= \langle \dot{\pi}'_{i+1}, \ldots, \dot{\pi}'_j \rangle \\ \dot{\gamma}'_2 &= \langle \dot{\pi}'_{j+1}, \ldots, \dot{\pi}'_n \rangle,\end{aligned}$$

then $\dot{\alpha} = \dot{\gamma}'_1 \lhd \dot{\gamma} \lhd \dot{\gamma}'_2$ and $[\dot{\gamma}_1, \dot{\gamma}_2, \dot{\gamma}_0] = [\dot{\gamma}'_1, \dot{\gamma}'_2, \dot{\gamma}]$.

Applying the Formation Projection lemma, find $\dot{\gamma}'$ such that

$$[\dot{\gamma}_1, \dot{\gamma}_2, \dot{\gamma}_0, \dot{\gamma}'_0] = [\dot{\gamma}'_1, \dot{\gamma}'_2, \dot{\gamma}, \dot{\gamma}'],$$

which, also by Formation Projection lemma, implies that $[\dot{\gamma}, \dot{\gamma}'] = [\dot{\gamma}_0, \dot{\gamma}'_0]$.

From congruence of $\approx$ w.r.t. composition, we obtain that

$$[\dot\gamma_1' \lhd \dot\gamma' \lhd \dot\gamma_2'] = [\dot\gamma_1 \lhd \dot\gamma_0' \lhd \dot\gamma_2'] = [\dot\beta].$$

Therefore, for $\dot c = \langle \dot\gamma, \dot\gamma' \rangle$ and $\dot\beta = \dot\gamma_1' \lhd \dot\gamma' \lhd \dot\gamma_2'$, we have $\dot\alpha \overset{\dot c}{\leftrightarrow} \dot\beta$. ∎

Lemma 39. If $[\dot\alpha] \sim_{\dot{\mathcal{I}}} [\dot\beta]$, then for all $\dot\alpha \in [\dot\alpha]$, there exists $\dot\beta \in [\dot\beta]$ such that $\mathrm{sites}(\dot\alpha) = \mathrm{sites}(\dot\beta)$.

Proof. Consider relation R on the set of formations defined as follows:

$$\begin{aligned} R \overset{\text{def}}{=} \{\langle [\dot\alpha], [\dot\beta] \rangle \mid \quad & \forall\, \dot\alpha \in [\dot\alpha]\ \exists\, \dot\beta \in [\dot\beta]\ \ \mathrm{sites}(\dot\alpha) = \mathrm{sites}(\dot\beta) \ \text{ and} \\ & \forall\, \dot\beta \in [\dot\beta]\ \exists\, \dot\alpha \in [\dot\alpha]\ \ \mathrm{sites}(\dot\alpha) = \mathrm{sites}(\dot\beta)\}. \end{aligned}$$

Relation R is an equivalence relation and, according to Lemma 36 and the definition of $\overset{[\dot c]}{\leftrightarrow}$, contains relation $\overset{[\dot c]}{\leftrightarrow}$, which implies, by definition of $\sim_{\dot{\mathcal{I}}}$, that R contains $\sim_{\dot{\mathcal{I}}}$. ∎

A **first level struct** is defined as an equivalence class of first level formations w.r.t. $\sim_{\dot{\mathcal{I}}}$. The following theorem states that, if the struct finiteness condition is imposed at the basic level, then, regardless of the set $\dot{\mathcal{I}}$ of first level semantic identities, all first level structs are finite. In particular, it implies that the semantic equivalence relation at the first level is decidable.

Theorem 11. *If the struct finiteness condition holds, then for every first level struct* $\dot{\boldsymbol{\alpha}}$*, the number of first level formations in it,* $|\dot{\boldsymbol{\alpha}}|$*, is finite.*

Proof. Fix a composite $\dot\alpha_0 \in \dot{\boldsymbol{\alpha}}$, and let $\langle \gamma_0, \gamma_0' \rangle \in \mathrm{sites}(\dot\alpha_0)$. According to Lemma 39, for any formation $[\dot\alpha] \in \dot{\boldsymbol{\alpha}}$, there exists composite $\dot\alpha \in [\dot\alpha]$ such that $\mathrm{sites}(\dot\alpha) = \mathrm{sites}(\dot\alpha_0)$. This implies that

$$\mathrm{res}(\langle \gamma_0, \mathrm{body}(\dot\alpha) \rangle) \subset [\![\gamma_0']\!].$$

According to Theorem 9, the set of all possible struct tuples $[\![\gamma_0, \mathrm{body}(\dot{\alpha})]\!]$, such that the above inclusion holds, is finite. Moreover, if $\dot{\alpha} = \langle \dot{\pi}_1, \ldots, \dot{\pi}_n \rangle$, then for all $i \in [1, n]$,

$$[\![\dot{\pi}_i]\!] \preceq K_{i-1}([\![\gamma_0, \mathrm{body}(\dot{\alpha})]\!]_{[1,i+1]}).$$

Hence, due to the struct finiteness condition and Theorem 6, there are finitely many possibilities for each $[\![\dot{\pi}_i]\!]$. According to the definition of relation $\approx$ on first level composites, struct tuples $[\![\mathrm{body}(\dot{\alpha})]\!]$ and $[\![\dot{\pi}_i]\!]$ ($i \in [1, n]$) uniquely specify a first level formation $[\dot{\alpha}]$. ■

3.1.7 First level inductive structure

The introduction of the first representational level is concluded with the concept of a first level inductive structure, accompanied by an example on insertion strings.

Similarly to Section 2.1.7, two first level primitives $\dot{\pi} = \langle \alpha, \beta \rangle$ and $\dot{\sigma} = \langle \alpha', \beta' \rangle$ will be called **equivalent**, if for the composites $\langle \dot{\pi} \rangle$ and $\langle \dot{\sigma} \rangle$ we have $\langle \pi \rangle \approx \langle \sigma \rangle$. According to the definition of relation $\approx$ on first level composites, this is equivalent to $[\![\alpha, \beta]\!] = [\![\alpha', \beta']\!]$. Thus, struct tuple $[\![\alpha, \beta]\!]$ will be called a **first level primitive type** and denoted $[\dot{\pi}]$.

A set of first level primitives $\dot{\Pi}$ is called **closed under site replacements**, if for all $\dot{\pi} \in \dot{\Pi}$, $[\dot{\pi}] \subset \dot{\Pi}$. Clearly, to specify a set of primitives closed under site replacements, it is sufficient to fix a set of primitive types $\ddot{\Pi}$ and one primitive, called a **specifying primitive**, for every $[\dot{\pi}] \in \ddot{\Pi}$.

Similarly to Section 2.3, we define a **first level inductive structure** $\langle \dot{\Pi}, \dot{\mathcal{I}} \rangle$ (where $\dot{\mathcal{I}}$ is a set of first level semantic identities) as the collection of all first level concepts that can be derived from $\dot{\Pi}$ and $\dot{\mathcal{I}}$, including the set of first level composites build out of primitives from $\dot{\Pi}$, $\dot{\Gamma}_{\dot{\Pi}}$, the set of formations $\ddot{\Gamma}_{\dot{\Pi}}$, the semantic equivalence relation $\sim_{\dot{\mathcal{I}}}$, the set of structs $\Theta_{\dot{\mathcal{I}}}$, and the part/whole relations $\preceq$ on each of these sets. Note that there is no need to mention the restriction that $\sim_{\dot{\mathcal{I}}}$ satisfies the struct finiteness condition, since this requirement automatically follows from Theorem 11.

3.1.8 An example of first level inductive structure based on insertion strings

Let us extend the inductive structure of insertion strings (Section 2.3.4) to a first level inductive structure $\langle \dot{\Pi}, \dot{\mathcal{I}} \rangle$. For the specifying primitives, take transformations $\dot{\pi}_1$ and $\dot{\pi}_2$ shown in Fig. 3.12, and let $\dot{\Pi} = [\dot{\pi}_1] \cup [\dot{\pi}_2]$.

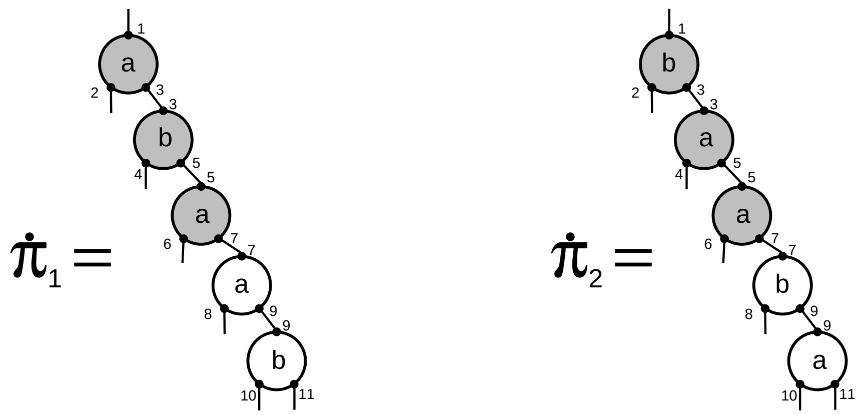

Figure 3.12: Specifying first level primitives.

A typical first level composite is shown in Fig. 3.13a, together with its pictorial representation in Fig. 3.13b.

Note that the pictorial representation uniquely specifies the composite, however, not all pictorial representations are admissible. For example, there exists no first level composite with the pictorial representation shown in Fig. 3.14.

The pictorial representation can be used for two purposes: to visualize a first level composite and to obtain a compact encoding for it. It should be noted here

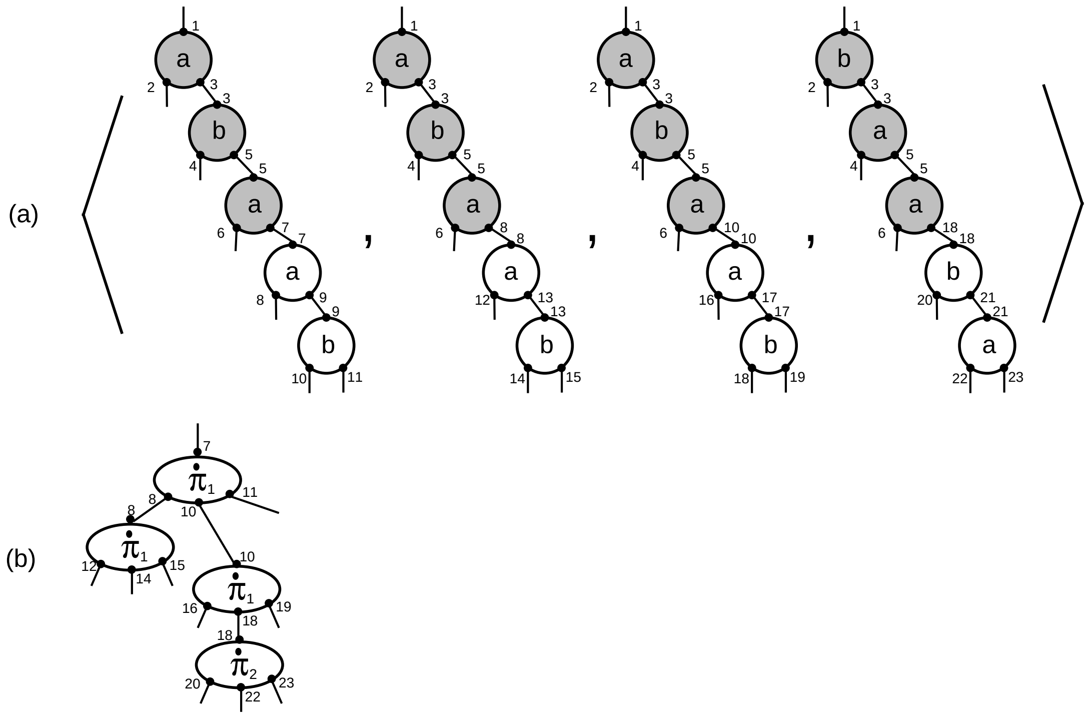

Figure 3.13: A first level composite.

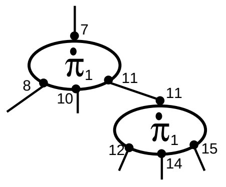

Figure 3.14: An inadmissible pictorial representation for a first level composite.

that the standard encoding shown in Fig. 3.13a is not feasible for computational purposes because of its obvious redundancy and implied growth in size. On the other hand, one has to be careful with the encoding based on the pictorial representation, since it does not reflect the complexity of verification of the attachment condition. This verification, as it is clear from Def. 39, involves context matching for each first level primitive in the sequence. Thus, our ability to work efficiently with first level structures depends on the efficiency of the context matching algorithm, which, to a great extent, depends on the basic level semantic identities. This algorithm, in general, can be quite complex, although it always exists due to the struct finiteness condition, as shown in Section 2.3.7). In the inductive structure of insertion strings, the context matching operation requires finding a subsequence in a string, which is easy. In the inductive structure of graphs, the problem is equivalent to the subgraph isomorphism problem, which is NP-complete [Ga79]). That is why an additional informal assumption is made, that the contexts of first level primitives are *small*, or, in other words, the contexts are always matched *locally*. This locality requirement has not been reflected in our definitions so far, but it should be kept in mind in view of its direct relation to the complexity of the context matching algorithm. A detailed analysis of computational complexity of verification of the attachment condition, as well as an efficient implementation of first (and higher) level inductive structures, is one of the primary goals for future research.

An example of a first level semantic identity is shown in Fig. 3.15a. The formation pair shown in this figure satisfies the semantic identity condition, since the two composites in it have the same sets of first level site pairs, the smallest of which is shown in Fig. 3.15b.[7] This identity (call it $[\dot{c}]$) induces all other semantic identities, i.e., if $\dot{\mathcal{I}}$ is the set of all possible first level semantic identities, then $\sim_{\dot{\mathcal{I}}}=\sim_{[\dot{c}]}$.

Since every struct in the inductive structure $\langle\dot{\Pi},\dot{\mathcal{I}}\rangle$ has a unique *minimal* site pair, there is a one-to-one correspondence between first level structs of $\langle\dot{\Pi},\dot{\mathcal{I}}\rangle$ and their *terminal structs* (which are basic level structs). It will be shown in Section 5.3 that the set of terminal composites for a first level struct can always be represented

[7] In this case, as opposed to the general case, the smallest site pair does exist.

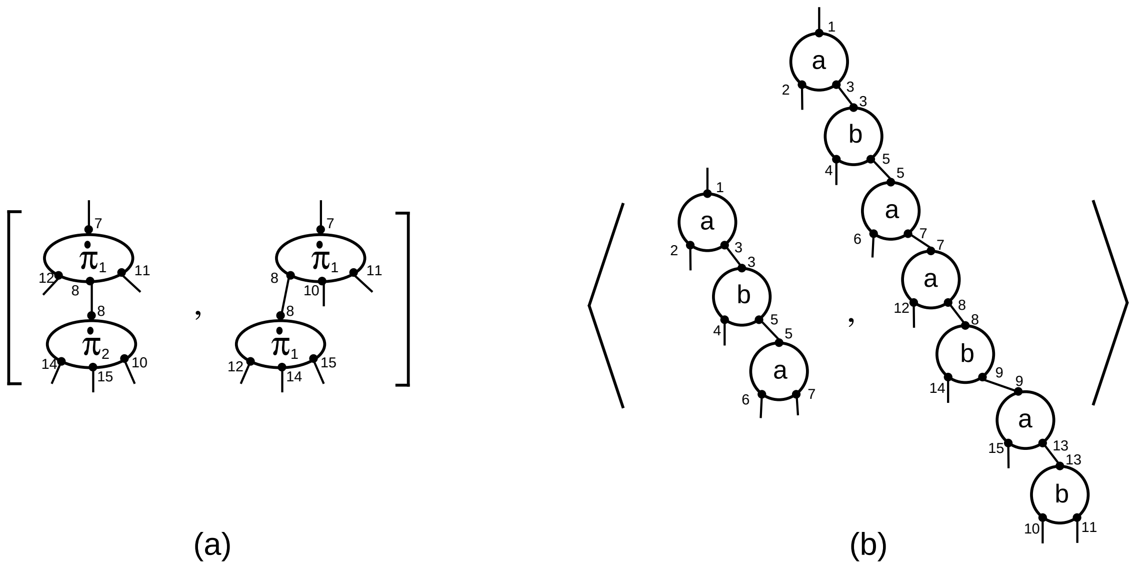

Figure 3.15: A first level semantic identity and a site pair for composites in it.

as a disjoined union of structs. For example, the first level struct shown in Fig. 3.16a corresponds to its minimal terminal struct, which, in turn, correspond to the string

$$aba\,\underbrace{abab\ldots ab}_{n+1}.$$

Similarly, the minimal struct that corresponds to the first level struct shown in Fig. 3.16b, corresponds to the string

$$abaaba\,\underbrace{ab\ldots ab}_{n}\,b.$$

Note that the edit distances between the two first level structs and the corresponding strings are different. Indeed, to convert the first struct into the second, one needs to

(a) remove all primitives except for the first two, and

(b) attach n primitives to the middle site of the first primitive.

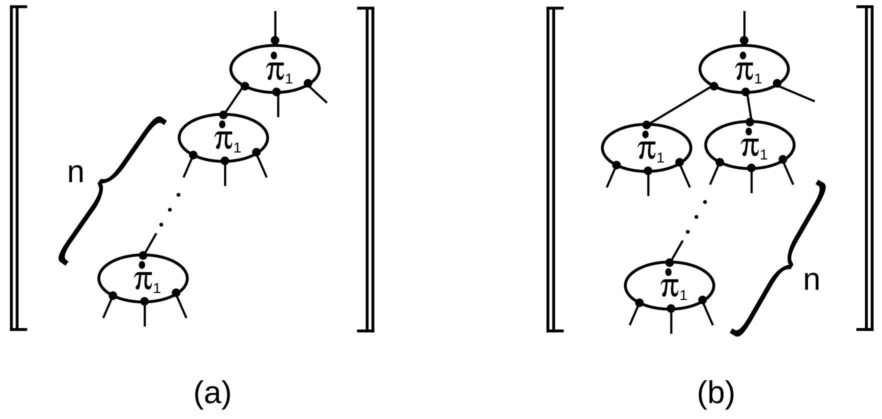

Figure 3.16: Edit distance between first level structs.

This takes $n - 1 + n = 2n - 1$ editing operations. On the other hand, it is easy to see that the edit distance between the corresponding strings is equal to 2.

Thus, editing operations at the first level are more specific, compared to the basic level. This observation remains true for higher level structs (which will be introduced in Section 3.2) as well.

To summarize this section:

- a transformation (=first level primitive) is a pair of composites $\tau = \langle \alpha, \beta \rangle$; τ applies to a composite γ, if the latter contains α as a part, the result of application being the composition of γ with β.

- a first level composite is a sequence of first level primitives satisfying the attachment condition;

- the sites of a first level composite are basic level composites;

- first level formations are defined as first level composites "up to a site replacement", which allows to replace sites with semantically equivalent ones;

- first level semantic equivalence is induced by a set of first level semantic identities, similarly to the basic level case, the equivalence classes are called first level structs; the struct finiteness condition at the basic level implies that first level structs are finite as well.

3.2 Representational hierarchy

The construction of the first representational level (see Sections 2.4 and 3.1) can be generalized by induction to an infinite hierarchy of representational levels.

The base case is the first level. Assume that the k-th representational level ($k \geq 1$) has been defined. That is, assume that $(k-1)$-st level struct tuples have been defined and all the statements of Section 2.4 hold for them, as well as k-th level primitives, composites, formations, structs, and other k-th level concepts have been defined as in Section 3.1 and all the statements of this section hold for them. In what follows, to simplify notation, we will call the k-th level *current*, and the $(k+1)$-level (the one that is being introduced) the *next* level. Concepts, corresponding to the current level, are denoted by dotted letters (e.g. $\dot{\pi}$, $\dot{\alpha}$, $[\dot{\alpha}]$, $\dot{\boldsymbol{\alpha}}$), and the next level concepts are denoted by double-dotted letters (e.g. $\ddot{\alpha}$).

Following the same outline, we begin our definitions with the current level struct tuples (as in Section 2.4), which will be followed by current level transformations (=next level primitives), next level composites, formations, and structs. Most of the definitions and statements remain unchanged ("level-invariant"), in which case we just indicate it and refer the reader to the corresponding location in the above sections. In a few cases, where the proofs of lemmas change slightly, we will redo them.

Assume that a current level inductive structure $\langle \dot{\Pi}, \dot{\mathcal{I}} \rangle$ is fixed.

3.2.1 Current level struct tuples

The current level semantic equivalence relation is extended to current level struct tuples exactly as in Def. 29. Then, the concept of a struct tuple projection (Def. 30) is defined and the Struct Projection lemma (L. 24) is restated, whose formulation and proof are already level-invariant, hence remain unchanged. Lemma 25 is also restated, and its proof is invariant as well. The proof of the next lemma, which states that the part/whole distinguishability condition holds, involves site replacement mappings, which is particular to the basic level and absent at all higher levels (since structural site replacements cannot be expressed by just a mapping), therefore we restate and reprove it here, even though the changes in the proof are minor.

Lemma 40. The current level semantic equivalence relation $\sim_{\dot{\mathcal{I}}}$ satisfies the part/whole distinguishability condition.

Proof. Suppose, the part/whole distinguishability does not hold, i.e., there exist formations $[\dot{\alpha}] \neq [\dot{\beta}]$ such that $[\dot{\alpha}] \sim_{\dot{\mathcal{I}}} [\dot{\beta}]$ and $[\dot{\alpha}] \preceq [\dot{\beta}]$.

By definition of $\preceq$, there exist $\dot{\alpha} \in [\dot{\alpha}]$, $\dot{\beta} \in [\dot{\beta}]$ and $\dot{\gamma}_1, \dot{\gamma}_2$, not both empty, such that $\dot{\beta} = \dot{\gamma}_1 \lhd \dot{\alpha} \lhd \dot{\gamma}_2$. Since $[\dot{\alpha}] \sim_{\dot{\mathcal{I}}} [\dot{\beta}]$, we obtain $[\![\dot{\alpha}]\!] = [\![\dot{\gamma}_1 \lhd \dot{\alpha} \lhd \dot{\gamma}_2]\!]$.

According to the Struct Projection lemma, there exists $\dot{\gamma}'_1$ such that

$$[\![\dot{\gamma}_1, \dot{\alpha}]\!] = [\![\dot{\gamma}'_1, \dot{\gamma}_1 \lhd \dot{\alpha} \lhd \dot{\gamma}_2]\!].$$

Since $\langle \dot{\gamma}_1, \dot{\alpha} \rangle$ satisfies the composition condition, by Lemma 37, $\langle \dot{\gamma}'_1, \dot{\gamma}_1 \lhd \dot{\alpha} \lhd \dot{\gamma}_2 \rangle$ must satisfy the composition condition as well, and

$$[\![\dot{\gamma}_1 \lhd \dot{\alpha}]\!] = [\![\dot{\gamma}'_1 \lhd \dot{\gamma}_1 \lhd \dot{\alpha} \lhd \dot{\gamma}_2]\!].$$

Similarly, there exists composite $\dot{\gamma}'_2$ such that

$$[\![\dot{\gamma}_1 \lhd \dot{\alpha} \lhd \dot{\gamma}_2]\!] = [\![\dot{\gamma}'_1 \lhd \dot{\gamma}_1 \lhd \dot{\alpha} \lhd \dot{\gamma}_2 \lhd \dot{\gamma}'_2]\!].$$

Note that $\dot{\gamma}'_1$ and $\dot{\gamma}'_2$ cannot be both empty, therefore,

$$[\dot{\gamma}_1 \lhd \dot{\gamma}_1 \lhd \dot{\alpha} \lhd \dot{\gamma}_2 \lhd \dot{\gamma}'_2] \neq [\dot{\gamma}_1 \lhd \dot{\alpha} \lhd \dot{\gamma}_2].$$

The process can be continued and, as a consequence, we have an infinite sequence of distinct formations

$$[\dot{\alpha}],\ [\dot{\gamma}_1 \dot{\lhd} \dot{\alpha} \lhd \dot{\gamma}_2],\ [\dot{\gamma}_1' \lhd \dot{\gamma}_1 \lhd \dot{\alpha} \lhd \dot{\gamma}_2 \lhd \dot{\gamma}_2'], \ldots,$$

each of which belongs to the struct $[\![\dot{\alpha}]\!]$, which contradicts Theorem 11 stating that the struct finiteness condition holds for the first level structs. ■

The theorems that are related to the finiteness of struct tuples (Th. 6 and 7) remain unchanged. The same is true for the definition of the part/whole relation on struct tuples (Def. 31) and Theorem 8. The struct finiteness condition can now be omitted in the statement of the theorem (due to Theorem 11), but the proof remains the same:

Theorem 12. *For any current level struct tuple $\dot{S}$, the set of its parts*

$$\mathrm{Parts}(\dot{S}) \overset{\mathrm{def}}{=} \{\dot{S}' \mid \dot{S}' \preceq \dot{S}\}$$

is finite.

Note, however, that the proof of the above theorem refers to the following lemma and theorem, which need to be restated and reproved here.

Lemma 41. For every formation $[\dot{\gamma}]$, the set $\mathrm{Parts}([\dot{\gamma}])$ is finite.

Proof. Suppose, $\mathrm{Parts}([\dot{\gamma}])$ is infinite. Fix a composite $\dot{\gamma}_0 \in [\dot{\gamma}]$. By definition of $\preceq$ on formations, for each $[\dot{\alpha}] \in \mathrm{Parts}([\dot{\gamma}])$, there exist composites $\dot{\alpha} \in [\dot{\alpha}]$ and $\dot{\gamma} \in [\dot{\gamma}]$ such that $\dot{\alpha} \in \mathrm{Parts}(\dot{\gamma})$. Since both $\dot{\gamma}$ and $\dot{\gamma}_0$ belong to the same formation,

$$[\![\mathrm{body}(\dot{\gamma})]\!] = [\![\mathrm{body}(\dot{\gamma}_0)]\!],$$

and

$$[\![\dot\gamma_i]\!] = [\![\dot\gamma_{0_i}]\!], \quad (i \in [1, |\dot\gamma|]),$$

which, according to the Struct Projection lemma, implies that

$$[\![\text{body}(\dot\alpha)]\!] = [\![\text{body}(\dot\alpha_0)]\!],$$

and

$$[\![\dot\alpha_i]\!] = [\![\dot\alpha_{0_i}]\!], \quad (i \in [1, |\dot\alpha|]),$$

hence $\dot\alpha \approx \dot\alpha_0$. Since formations are equivalence classes of composites, all $\dot\alpha_0$ corresponding to distinct $[\dot\alpha] \in \text{Parts}([\dot\gamma])$ are distinct. However, the set of parts for any composite is finite, contradiction! ■

Theorem 13. *For any struct* $\dot{\boldsymbol{\gamma}}$*, the set* $\text{Parts}(\dot{\boldsymbol{\gamma}})$ *is finite.*

The proof is exactly the same as the one for case 1 of Theorem 2, where the reference to Lemma 12 should be replaced by the above Lemma 41.

Now we proceed to composable tuples, thus preparing for the definition of a next level composite. The *1-composable composite and formation tuples* are introduced exactly as in Def. 32. Correctness follows from the fact that relation $\approx$ on current level composites is a congruence w.r.t. concatenation (Lemma 35). A 1-composable struct tuple is defined as in Def. 33, whose correctness is justified by an analogue of Lemma 28, which is formulated in exactly the same way. We reprove it here:

Lemma 42. If $\dot D, \dot D'$ are 1-composable formation tuples and $\dot D \sim_{\dot{\mathcal{I}}} \dot D'$, then $K(\dot D) \sim_{\mathcal{I}} K(\dot D')$.

Proof. First, reformulate Lemma 37 for arbitrary formation tuples: Let $[\dot c]$ be a semantic identity. If $\dot D \overset{[\dot c]}{\leftrightarrow} \dot D'$ and $\dot D$ is 1-composable, then so is $\dot D'$ and $\dot D \overset{[\dot c]}{\leftrightarrow} \dot D'$. The proof is similar and for this reason omitted.

Now, consider a relation R on the set of formation n-tuples defined as follows: $\langle \dot{D}, \dot{D}' \rangle \in R$ if and only if either both $\dot{D}$ and $\dot{D}'$ are 1-composable and $K(\dot{D}) \sim_{\dot{\mathcal{I}}} K(\dot{D})$, or none of them is 1-composable.

Relation R is an equivalence relation and contains $\overset{\dot{\mathcal{I}}}{\leftrightarrow}$, hence it must also contain $\sim_{\dot{\mathcal{I}}}$, which implies the statement of our lemma. ■

Composable struct tuples and their *i-compositions* are defined as in Def. 34. The theorem about the finiteness of struct tuples corresponding to a given i-composition (Th. 9) also remains unchanged.

The definition of a result for a composite tuple is identical to the one given in Defs. 35,36, and the proofs of existence of a result (Theorem 10), as well as the Result criterion (Lemma 29) are level-invariant.

3.2.2 Next representational level

A *current level transformation* is, as in Def. 37, defined as a pair of current level composites $\langle \dot{\alpha}, \dot{\beta} \rangle$ satisfying the composition condition and having $\dot{\beta} \neq \lambda$. The definition of *application* of a current level transformation to a current level composite copies Def. 38 literally. As mentioned above, "current level transformation" will be considered as a synonym for "next level primitive".

The attachment condition for a sequence of next level primitives $S = \langle \ddot{\pi}_1, \ldots, \ddot{\pi}_n \rangle$ remains unchanged, and so does the definition of a *next level composite* $\ddot{\gamma}$ (Def. 39), its body-tuple body($\ddot{\gamma}$) and sites (Def. 42). Note that the sites of $\ddot{\gamma}$ are current level composites.

The following definitions and statements are level-invariant, i.e., can be carried over to the next level without any changes: composition operation on composites

(Def. 40), part/whole relation (Def. 41), the formula for sites($\ddot{\alpha} \lhd \ddot{\beta}$) (Lemma 30), Composition criterion (Lemma 31), next level formation (Def. 43), Composite Reconstruction lemma (L. 32), next level formation tuple (Def. 44), Formation Projection lemma (L. 33), invariance of site equality w.r.t. $\approx$ (Lemma 34) and congruence of $\approx$ w.r.t. composition (Lemma 35), next level semantic identity (Def. 45), direct convertibility relation $\overset{\ddot{\mathcal{I}}}{\leftrightarrow}$, next level semantic equivalence $\sim_{\ddot{\mathcal{I}}}$, invariance of site equality w.r.t. $\overset{\ddot{c}}{\leftrightarrow}$ (Lemma 36) and congruence of $\sim_{\ddot{\mathcal{I}}}$ w.r.t. composition (Lemma 37), Lemmas 38 and 39, next level struct finiteness theorem (Th. 11), concluded by the concept of a next level inductive structure $\ddot{\mathbb{I}} = \langle \ddot{\Pi}, \ddot{\mathcal{I}} \rangle$ (Section 3.1.7).

The definition of the next representational level is complete and, by induction, we obtain an infinite hierarchy of representational levels. The above list of definitions and lemmas can be considered as an *axiomatic definition* of a representational hierarchy. Every representational level, starting from the second, satisfies all the axioms. The first level has the same definitions of concepts, but the proofs may slightly vary when they refer to the basic-level concepts. The basic level is somewhat exceptional, which is clear why: it was not possible to construct level 0 following the same scheme, since the scheme, being inductive by its nature, relies on the concepts of the $(k-1)$-st level when the k-th level is introduced.

Consider the concept of an *abstract representational hierarchy*, which is a sequence of inductive structures

$$\ldots\ \mathbb{I}^{(-2)},\ \mathbb{I}^{(-1)},\ \mathbb{I},\ \dot{\mathbb{I}},\ \ddot{\mathbb{I}},\ \mathbb{I}^{(3)}, \ldots,$$

where each structure is related to the previous one according to our scheme. I conjecture that the sequence cannot extend to $-\infty$, otherwise the struct finiteness condition will not hold at any level (although, it seems possible to construct such bidirectional infinite hierarchies of levels, if the struct finiteness condition is disregarded). I would like to leave the conjecture open until a formal axiomatic system will replace the inductive construction presented in this section.

To summarize this section, the infinite hierarchy of representational levels has been introduced. This hierarchy, as was mentioned in the Introduction (see Section 1.8), corresponds to the natural hierarchy of objects such as atoms, molecules, cells, tissues, organs, organisms, etc, and is supposed to be constructed by the process of inductive inference. Thus, the representational hierarchy offers a systematic way of constructing representations for complex objects, provided that an inductive inference algorithm is designed (see the next chapter for the formulation of the optimization criterion for the inductive inference problem). It also allows to classify objects, since a description of a representational level can be thought of as class description.

Chapter 4

Generative power of the representational hierarchy

In order to justify the restriction that transformations are context-dependent attachments, the generative power of the ETS transformations is compared to that of other generative mechanisms. In particular, I single out a class of string-rewriting systems that can be simulated by transformations in an appropriate basic level inductive structure and give three examples of graph languages (all simple graphs, cubic graphs, and graphs without triangle subgraphs) that can be generated by first level transformations. Based on intuition gained from the latter examples, I formulate a conjecture that any recursive language can be similarly generated in the ETS model, thus making it Turing-complete.

4.1 Simulation of acyclic string-rewriting systems with inductive structures

In this section we will address the problem of simulation of rewriting systems with inductive structures and transformations. We have briefly touched this issue in Section 3.1.1, when we were justifying our definition of transformations as context-dependent attachments, as opposed to deletions or substitutions. The reason for restricting ourselves to attachments only was that transformations correspond to *evolutionary steps*, and evolutionary processes in nature are, in general, irreversible (or acyclic).

Rewriting systems, including all kinds of string and graph grammars, are probably the most developed, if not the only, formalism, in which classes of objects are described in a generative form. Most of rewriting systems, however, include deletions and substitutions, and it is quite clear that without them the theory of rewriting systems would have lost much of its power and many of its applications.

This is not the case with inductive structures. Indeed, we will show that, by incorporating proper semantic identities into the inductive structure, one can design an inductive structure isomorphic (in the sense explained below) to any given string-rewriting system, provided that its derivations are acyclic.

A string-rewriting system is defined as a pair (Σ, I), where Σ is a finite alphabet and $I = \{(l_i, r_i)\}$, $i = 1, \ldots, k$ is a finite set of rewriting rules, each of which is a pair of strings over Σ [Bo93]. Let $\rightarrow_i$ be the single-step rewriting relation induced by the i-th rewriting rule, i.e.,

$$u \rightarrow_i v \quad \Longleftrightarrow \quad u = x l_i y,\ v = x r_i y.$$

Let $\longrightarrow_I^+$ be the transitive closure of the single-step rewriting relation

$$\rightarrow_I = \cup_{i=1}^k \rightarrow_i .$$

A string-rewriting system (Σ, I) will be called *strongly acyclic*, if $u \longrightarrow_I^+ v$ implies that string v is not a subsequence in u, i.e., u cannot be obtained from v through a sequence of insertions. In other words, (Σ, I) is strongly acyclic, if for $I_e = I \cup \{(\lambda, a) \mid a \in \Sigma\}$, where λ is the empty string, the relation $\longrightarrow_{I_e}^{+}$ is irreflexive.

Clearly, strongly acyclic rewriting system may contain substitution rules. For example, if $I = \{(abc, bd)\}$, then no string can be rewritten into any of its subsequences, thus the string-rewriting system is strongly acyclic. Deletions, however, are not allowed.

The following theorem asserts that for any strongly acyclic string-rewriting system, there exists an "isomorphic" set of transformations over a basic level certain inductive structure. Thus, provided that the condition of strong acyclicity holds (which, as mentioned above, often happens to be the case in applications), it justifies why we do not lose any generative power by restricting our transformations to context-dependent attachments only.

Theorem 14. *For every strongly acyclic string-rewriting system (Σ, I), there exists a basic level inductive structure $\langle \Pi, \mathcal{I} \rangle$, an injective mapping $\phi : \Sigma^* \to \Theta$ that maps strings to the structs in this inductive structure, and a set of first level primitive types $T = \{\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_k\}$, $\boldsymbol{\tau}_i = [\![\alpha_i, \beta_i]\!]$, such that for all $u, v \in \Sigma^*$ and $i = 1, \ldots, k$*

$$u \rightarrow_i v \quad \Longleftrightarrow \quad \exists\, \alpha \in \phi(u),\ \beta \in \phi(v),\ \tau \in \boldsymbol{\tau}_i\ \ \beta = \alpha \lhd \tau.$$

Proof. Consider the inductive structure $\langle \Pi, \mathcal{I} \rangle$ of insertion strings over Σ. Let mapping $\phi : \Sigma^* \to \Theta$ be the mapping that corresponds strings to structs as it was defined in Section 2.3.4.

For each rewriting rule $(l_i, r_i) \in I$, construct a transformation $\tau_i = \langle \alpha_i, \beta_i \rangle$ as follows (see an example in Fig. 4.1(a)):

1. let the context of τ_i, α_i be any composite from $\phi(l_i)$;

2. let the body of τ_i be a special primitive π_i with the label i, $|l_i| + 1$ initial sites and $|r_i| + 1$ terminal sites.

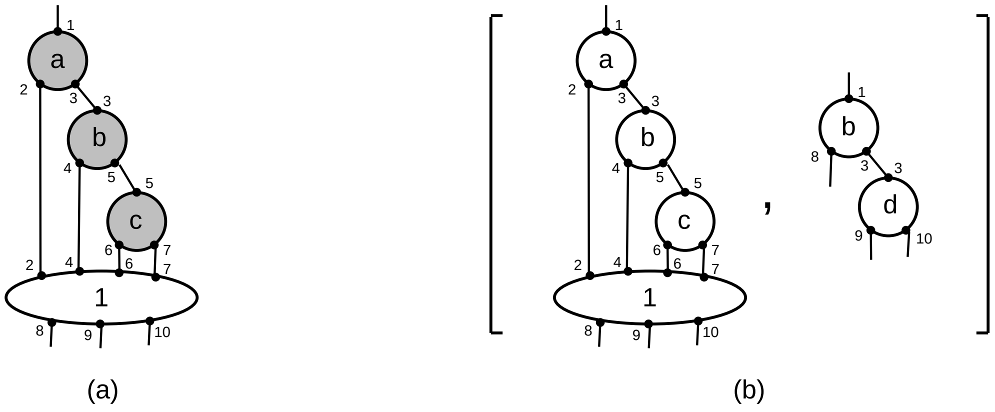

Figure 4.1: Transformation and semantic identity corresponding to rewriting rule (abc, bd).

Let Π' be the set of primitives, whose specifying primitives are the above π_i, i.e., Π' is the minimal closed under site replacements set of primitives containing all π_i.

Also, for each constructed transformation $\langle \alpha_i, \langle \pi_i \rangle \rangle$, add a new semantic identity $[\alpha_i \lhd \langle \pi_i \rangle, \beta_i]$ to $\mathcal{I}$ (see Fig. 4.1(b)), where β_i is a composite from $\phi(r_i)$ such that its external sites match those of α_i. Let $\mathcal{I}'$ be the set of thus constructed semantic identities.

The fact that for all $u, v \in \Sigma^*$ and $i = 1, \ldots, k$

$$u \rightarrow_i v \quad \Longleftrightarrow \quad \exists\, \alpha \in \phi(u),\ \beta \in \phi(v),\ \tau \in \boldsymbol{\tau}_i \ \ \beta = \alpha \lhd \tau$$

follows directly from our construction.

It remains to show that the semantic equivalence relation induced by the constructed set of identities satisfies the struct finiteness condition.

Define the one-way rewriting relation $\mapsto$ on composites as follows: for two composites α, β and a composite pair $c = \langle \gamma, \gamma' \rangle$ from one of the above constructed semantic identities, let $\alpha \mapsto_c \beta$ if and only if there exist composites γ_1, γ_2 such that

$$\begin{aligned} \alpha &= \gamma_1 \lhd \gamma \lhd \gamma_2 \\ \beta &= \gamma_1 \lhd \gamma' \lhd \gamma_2. \end{aligned}$$

Let $\mapsto$ be the union of all relations $\mapsto_c$, where $[c] \in \mathcal{I}'$. Extend relation $\mapsto$ to the set of structs of the inductive structure $\langle \Pi \cup \Pi', \mathcal{I} \rangle$:

$$\boldsymbol{\alpha} \mapsto \boldsymbol{\beta} \iff \exists\, \alpha \in \boldsymbol{\alpha},\, \beta \in \boldsymbol{\beta}\;\; \alpha \mapsto \beta.$$

If $\boldsymbol{\alpha} \mapsto \boldsymbol{\beta}$, $\boldsymbol{\alpha}$ will be called a *direct descendant* of $\boldsymbol{\beta}$, and $\boldsymbol{\beta}$ a *direct ancestor* of $\boldsymbol{\alpha}$[1] Also, struct $\boldsymbol{\alpha}$ will be called an *ancestor* of struct $\boldsymbol{\gamma}$ (and $\boldsymbol{\gamma}$ is a *descendant* of $\boldsymbol{\alpha}$), if $\boldsymbol{\gamma} \mapsto^* \boldsymbol{\alpha}$, where $\mapsto^*$ is the reflexive transitive closure of relation $\mapsto$.

Let $\equiv$ be the reflexive symmetric transitive closure of $\mapsto$. Then

$$\boldsymbol{\alpha} \equiv \boldsymbol{\beta} \iff [\alpha] \sim_{\mathcal{I} \cup \mathcal{I}'} [\beta]$$

for all formations $[\alpha] \in \boldsymbol{\alpha}$, $[\beta] \in \boldsymbol{\beta}$.

Since structs of the inductive structure $\langle \Pi \cup \Pi', \mathcal{I} \rangle$ are finite, it remains to show that every equivalence class w.r.t. $\equiv$ is also finite. We shall denote an equivalence class w.r.t. $\equiv$ containing a struct $\boldsymbol{\alpha}$ by $[\boldsymbol{\alpha}]$.

Relation $\mapsto$ on the set of structs from $\langle \Pi \cup \Pi', \mathcal{I} \rangle$ satisfies the following two properties:

1. $\mapsto$ is noetherian, i.e., there exist no infinite chains

$$\boldsymbol{\alpha}_1 \mapsto \boldsymbol{\alpha}_2 \mapsto \ldots$$

[1]Here we would like to be consistent with the ETS terminology, which is opposite to the one conventional in the rewriting to theory and, according to which, ancestors are "smaller" than their descendants.

This is because if $\boldsymbol{\alpha}_1 \mapsto \boldsymbol{\alpha}_2$, then the number of primitives from Π' in $\boldsymbol{\alpha}_2$ is less by one than this number for $\boldsymbol{\alpha}_1$ (see Fig. 4.1).

2. $\mapsto$ is locally confluent, i.e., if $\boldsymbol{\alpha} \mapsto \boldsymbol{\beta}$ and $\boldsymbol{\alpha} \mapsto \boldsymbol{\gamma}$, where $\boldsymbol{\beta} \neq \boldsymbol{\gamma}$, then there exists $\boldsymbol{\alpha}'$ such that $\boldsymbol{\beta} \mapsto \boldsymbol{\alpha}'$ and $\boldsymbol{\gamma} \mapsto \boldsymbol{\alpha}'$. This is because different left-hand side parts of the semantic identities from $\mathcal{I}'$ do not overlap, which is guaranteed by the presence of primitives from Π' in them.

Then, according to [Bo93, Theorem 1.1.12], every equivalence class $[\boldsymbol{\alpha}]$ has unique normal form, i.e., a struct $\boldsymbol{\alpha}_0$ that is a common ancestor of all $\boldsymbol{\gamma} \in [\boldsymbol{\alpha}]$.

Assume that $[\boldsymbol{\alpha}]$ is infinite. Then $\boldsymbol{\alpha}_0$ has infinitely many descendants (which are all structs from $[\boldsymbol{\alpha}]$). Since the number of direct descendants of $\boldsymbol{\alpha}_0$ is finite, one of them has to have infinitely many descendants, call it $\boldsymbol{\alpha}_1$. For the same reason, $\boldsymbol{\alpha}_1$ has a direct descendant $\boldsymbol{\alpha}_2$ with infinitely many descendants. Continuing this process, obtain an infinite chain

$$\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots \qquad (*),$$

where for all $j > 0$, $\boldsymbol{\alpha}_j \mapsto \boldsymbol{\alpha}_{j-1}$.

Now, observe that if we remove all primitives that belong to Π' from a struct $\boldsymbol{\alpha}$, we obtain a disjoined struct composed of primitives from Π, which thus corresponds to a finite set of strings. Denote this set of strings by $S(\boldsymbol{\alpha})$ Then, if $\boldsymbol{\beta} \mapsto \boldsymbol{\alpha}$, we obtain that $S(\boldsymbol{\beta}) = S(\boldsymbol{\alpha}) \setminus \{u\} \cup \{v\}$, for some strings $u \in S(\boldsymbol{\alpha})$, $v \in S(\boldsymbol{\beta})$ such that $v \to_I u$.

Thus, we have obtained an infinite sequence of strings corresponding to the above sequence of structs $(*)$: $u_0, u_1, u_2, \ldots$, where for all $j > 0$, $u_j \to_I u_{j-1}$.

According to Higman's Lemma [Hi52, Der90], for any infinite sequence of strings, there exist indices $i < j$ such that u_i is a subsequence of u_j, which contradicts our assumption that relation $\rightarrow_I$ is strongly acyclic. ■

4.2 Inductive structures for graphs and graph languages

As has been shown before, the set of all labeled directed multigraphs without disconnected vertices and loops can be represented as a particular (basic level) inductive structure (see Section 2.3.5). It turns out that at the first level one can represent more interesting graphs languages. Here, the examples of first level inductive structures corresponding to the simple graphs, cubic graphs, and graphs without triangle subgraphs are given. In the end, a conjecture that the generative power of first level inductive structures is exactly the family of all recursive languages is formulated.

4.2.1 Simple graphs

Consider a simple undirected graph as a result of a constructive process which creates vertices and connects some of them by directed edges. In order to prohibit multiple vertices and edges at the same place, represent them by the *first level* primitives shown in Fig. 4.3. These first level primitives are assumed to be constructed over a basic level inductive structure, whose primitives and semantic identities are shown in Fig. 4.2.[2] The struct finiteness condition holds, since all identities preserve the number of primitives, and the set of all formations with a fixed number of primitives is finite.

Assume that all semantic identities are present at the first level. Then for each simple undirected graph, there is a unique first level struct corresponding to it. An example of such correspondence is shown in Fig. 4.4. Vice versa, every struct $\dot{\boldsymbol{\alpha}}$, such that for every composite $\dot{\alpha} \in \dot{\boldsymbol{\alpha}}$ there exists $\gamma \in \text{init}(\dot{\alpha})$ free of primitive "v",

[2]In this and the following example, identities of the form $[\pi_1 \lhd \pi_2, \pi_2 \lhd \pi_1]$ will always be implicitly assumed to be present.

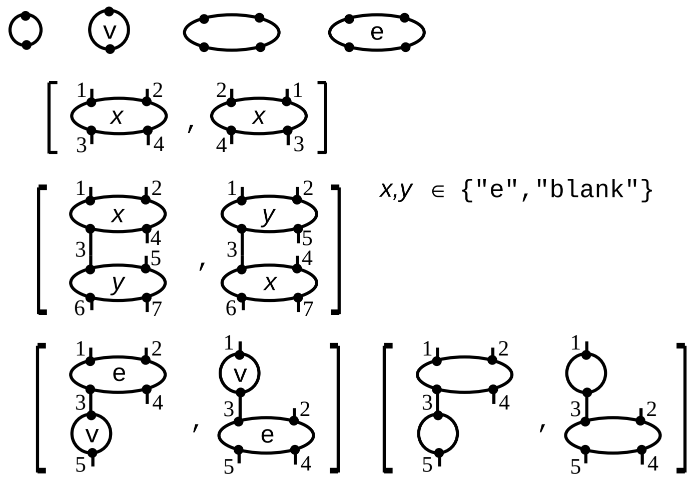

Figure 4.2: Specifying basic level primitives and semantic identities for the inductive structure of simple graphs.

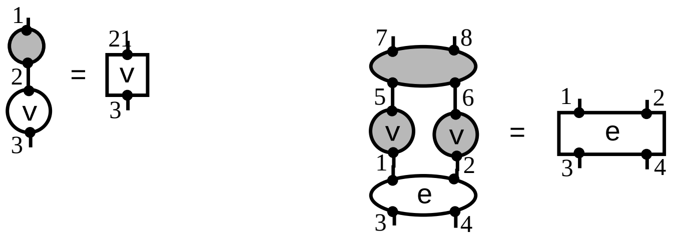

Figure 4.3: Specifying first level primitives for the inductive structure of simple graphs.

corresponds to a simple graph. All other structs contain edges that connect vertices not created within these structs and may be thought of as "transformations" of graphs, rather than just "graphs"—for example, a first level struct consisting of a single edge primitive is such a transformation, which can also be interpreted as "an edge without vertices", or "an edge, which can be applied to a pair of vertices, thus connecting them".

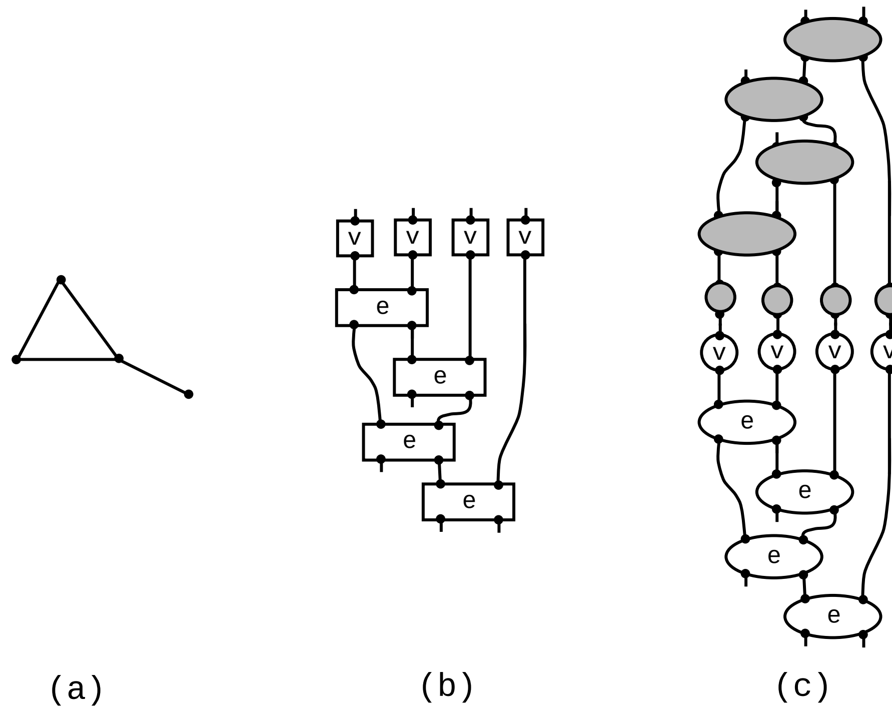

Figure 4.4: Example of a simple graph (a), the corresponding first level struct (a particular formation from it is shown) (b), and a site pair (initial composite shaded) for a composite from this formation (c).

4.2.2 Cubic graphs

The first level inductive structure of cubic graphs (simple undirected unlabeled graphs in which every vertex has degree 3) is defined in Fig. 4.6 (the first level specifying primitives) and Fig. 4.5 (the basic level specifying primitives and semantic identities). The struct finiteness condition holds, because one can assign weights to primitives so that all identities preserve the total weight of formations (for example, one can assign 2 to each primitive except "a" and "b", and 1 to "a" and "b"); the set of formations with a fixed weight is finite.

Now, consider the *union* of the inductive structure of cubic graphs and that of

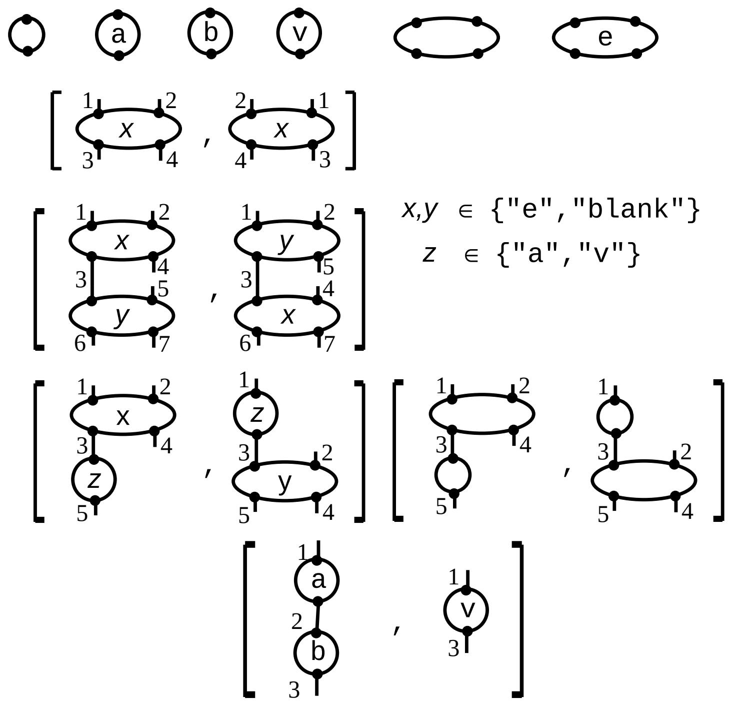

Figure 4.5: Specifying basic level primitives and semantic identities for the inductive structure of cubic graphs.

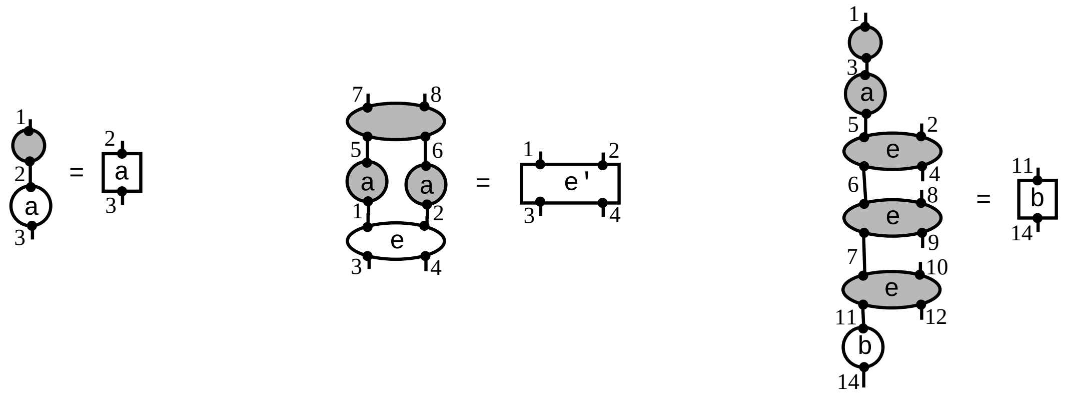

Figure 4.6: Specifying first level primitives for the inductive structure of cubic graphs.

simple graphs (i.e., put all primitives and semantic identities together). The first level formations whose composites are composed of the primitives from Fig. 4.6 will be called "cubic formations"; similarly, formations whose composites are composed of the primitives from Fig. 4.3 will be called "simple graph formations". Now, the structs that contain both kinds of formations, cubic *and* simple graph, correspond to cubic graphs. An example of such struct is shown in Fig. 4.7.

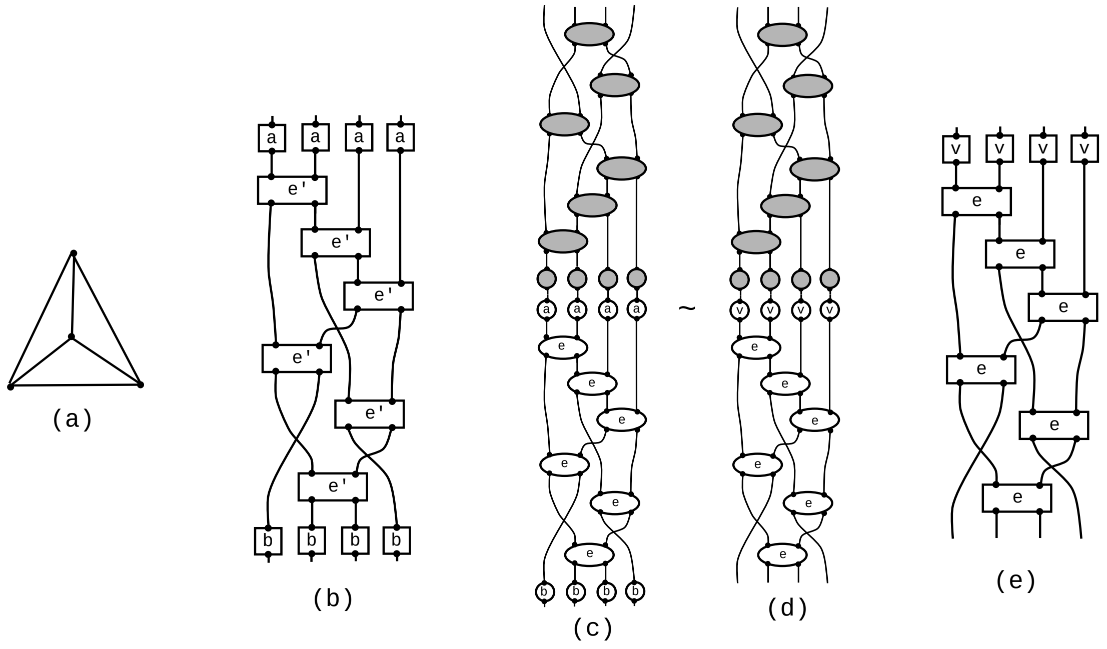

Figure 4.7: Example of a cubic graph (a), the corresponding first level cubic struct (a particular formation from it is shown) (b), a site pair for a composite from this formation (c), a pair of composites semantically equivalent to this site pair (d), and a formation from the corresponding simple graph struct with the same site pair (e).

4.2.3 Graphs without triangle subgraphs

Consider the set of simple undirected unlabeled graphs without triangle subgraphs. The specifying first level primitives for the corresponding inductive structure are shown in Fig. 4.11; the underlying basic level primitives and semantic identities are shown in Figs. 4.8, 4.9, 4.10.

As in the above example of cubic graphs, consider the union of the inductive

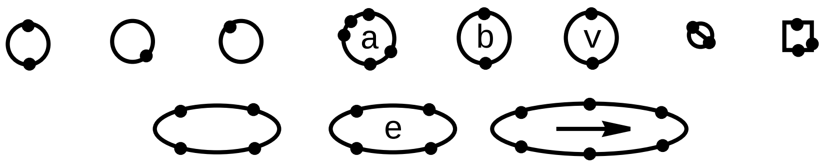

Figure 4.8: Specifying basic level primitives for the inductive structure of graphs without triangles.

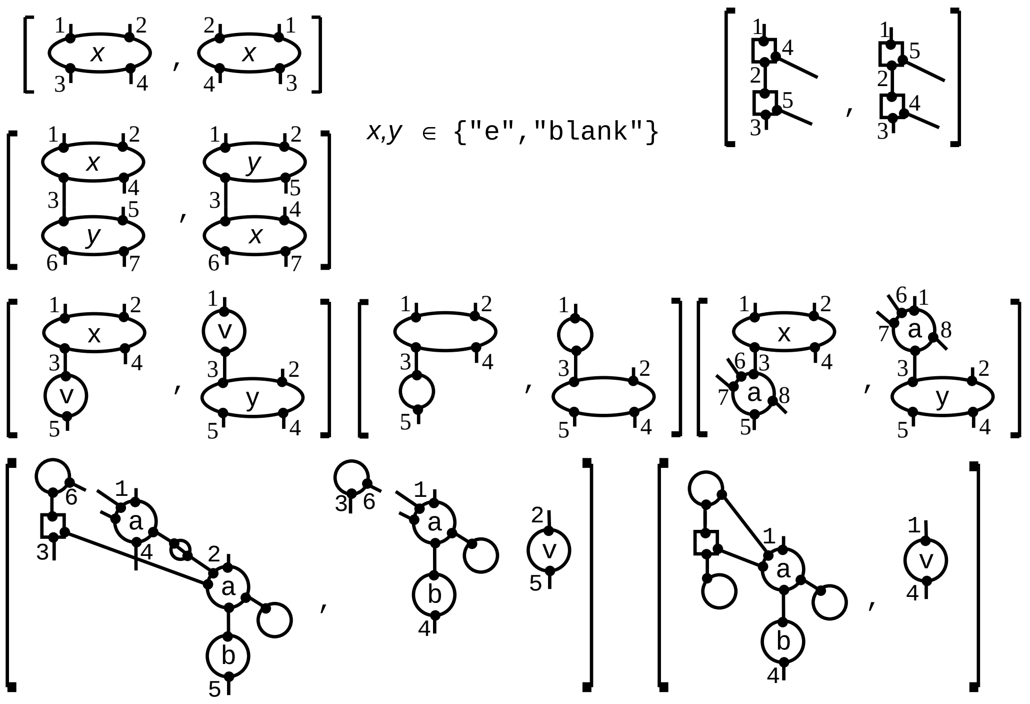

Figure 4.9: Semantic identities for the inductive structure of graphs without triangles (part 1).

structures of graphs without triangle subgraphs and simple graphs. The structs that contain formations from both inductive structures correspond to graphs without triangles. An example of such graph and the corresponding first level struct is shown in Fig. 4.12. A chain of equivalences that proves that the struct in Fig. 4.12 contains a simple graph formation is shown in Fig. 4.13.

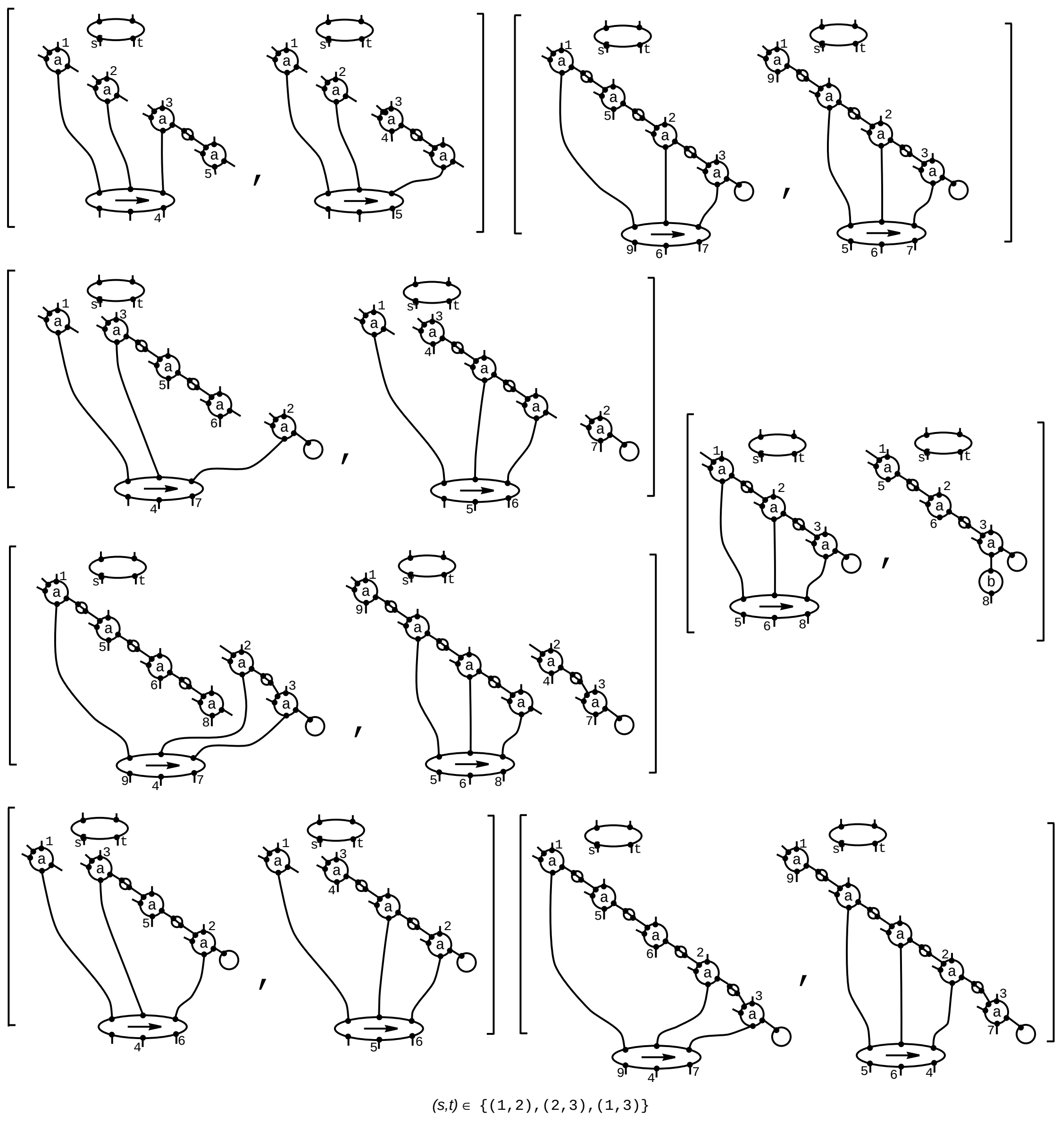

Figure 4.10: Semantic identities for the inductive structure of graphs without triangles (part 2). These identities allow to shift the arrow primitive to the right, provided that its three initial sites are not connected to a triangle. Starting from a leftmost position, the arrow primitive can traverse all $n(n-1)(n-2)/6$ positions to the rightmost position if and only if in all these positions the arrow primitive is not connected to a triangle.

The example of graphs without triangle subgraphs can be easily generalized to the case of graphs without any particular fixed subgraph (or several fixed subgraphs).

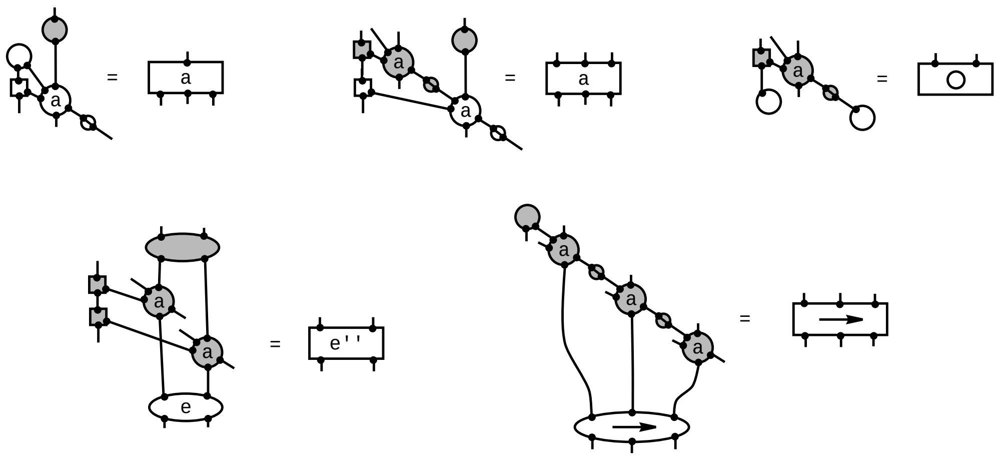

Figure 4.11: Specifying first level primitives for the inductive structure of graphs without triangles.

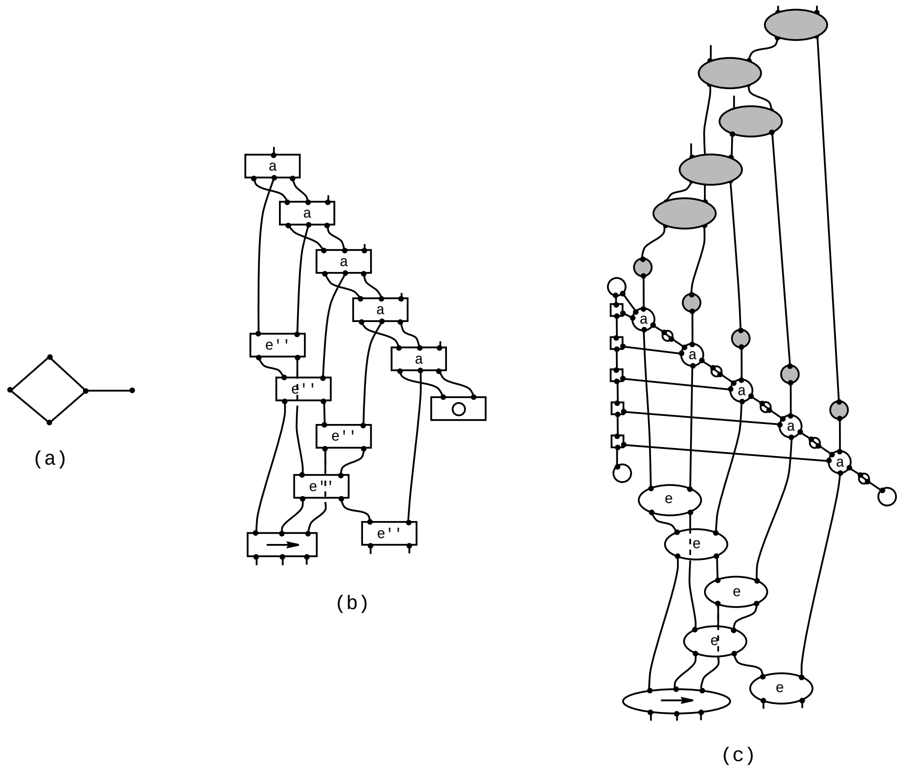

Figure 4.12: Example of a graph without triangle subgraphs (a), the corresponding first level struct (b), and a site pair for a composite from this struct (c).

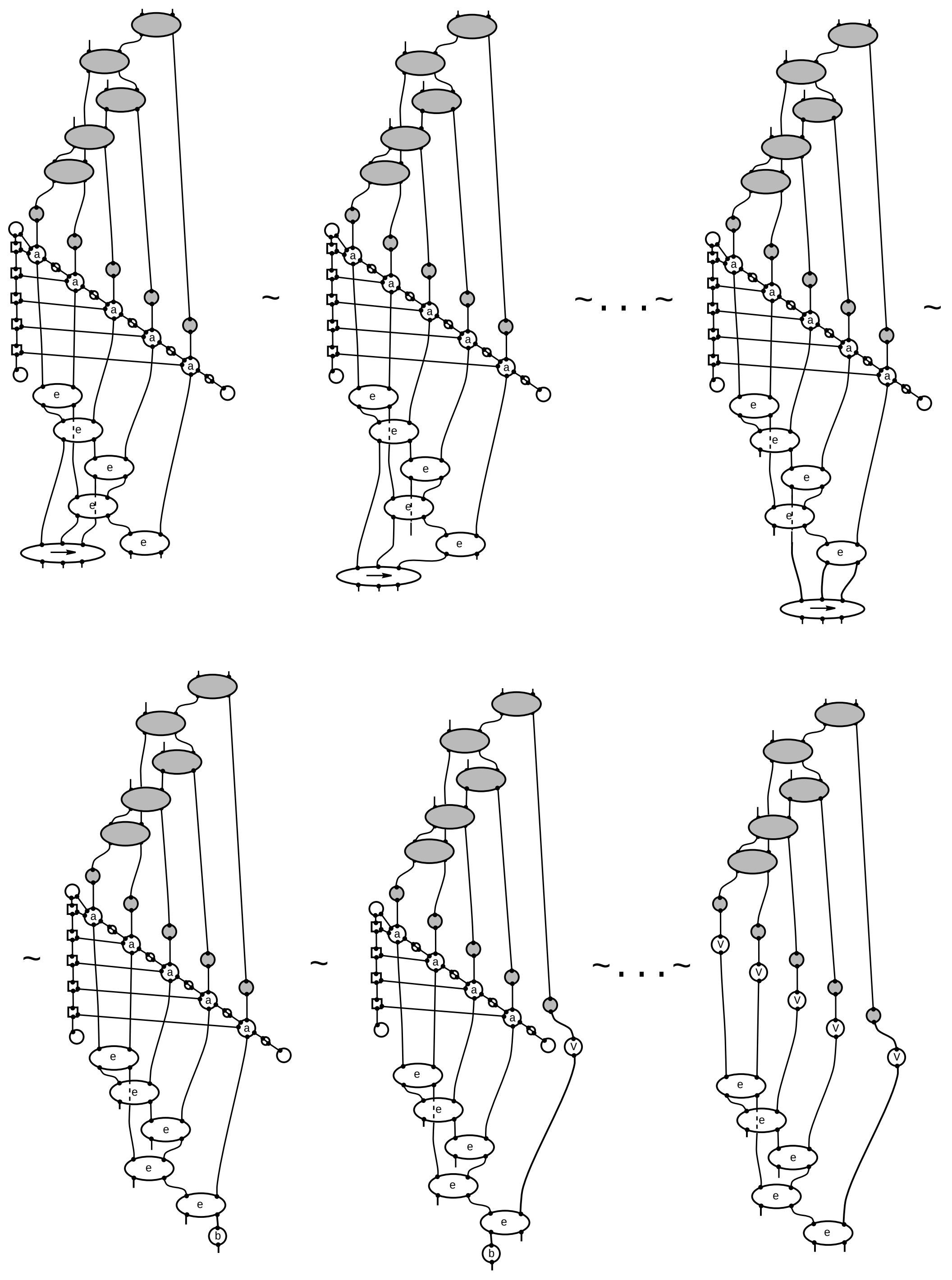

Figure 4.13: Chain of equivalences that proves that the struct in Fig. 4.12 contains a simple graph formation.

In order to do this, one should modify the "arrow identities" in Fig. 4.10.

The above examples with graph languages show that in ETS model one can choose certain first level primitives, basic level primitives, and semantic identities so that the construction of first level structs, essentially, incorporates the parsing algorithm.[3] It turns out that one can, in principle, incorporate *any* parsing algorithm in this manner—not only those for cubic graphs or graphs without triangles. To see this, think about the parsing algorithm as a Turing machine that starts with an input written on the tape and finishes in state q_Y or q_N, depending on whether the input belongs to the language or not; in addition, one can always force the Turing machine to clean up the tape before entering states q_Y or q_N. In the corresponding first level inductive structure, there are transformations that add primitives corresponding to the tape, head, and state q_Y of the Turing machine[4]; then, other transformations simulate the *reverse* computation of the Turing machine; finally, whenever a formation containing the primitive corresponding to the initial state q_0 of the Turing machine is obtained, it is guaranteed to be equivalent to a string-like formation (the identities replace the head and current state of the machine by a single string initialization primitive, and the tape becomes the string itself). In such a construction, the struct finiteness condition should follow from the fact that a Turing machine cannot enter the same situation twice during a particular computation (otherwise, it will never stop on certain inputs). I would like to leave the exact formulation and proof of this conjecture about the Turing-completeness of first level inductive structures beyond the scope of this thesis. If the conjecture is true, the first level inductive structures

[3]The parsing algorithm determines whether a given graph belongs to the language.

[4]These three things are called a *situation* of the Turing machine.

(FLIS) fill the gap between the context-sensitive (type 1) and unrestricted (type 0) grammars in the Chomsky hierarchy (in the following table, all inclusions are proper):

Languages:	Reg	$\subset$	CF	$\subset$	CS	$\subset$	Recursive	$\subset$	RecEnum
Grammars:	type 3		type 2		type 1		FLIS		type 0

Chapter 5

Generating process and inductive learning

For an inductive structure $\langle \Pi, \mathcal{I} \rangle$ of any representational level, we define a stochastic generating process associated with it. The version of the generating process presented here is preliminary, for intuitively the generating process should encapsulate all representational levels simultaneously. Some kind of interaction between the processes at different levels should be present in the model, and the precise formalization of this interaction is one of my primary research directions.

As mentioned above, the concept of formation is the one which corresponds to a single constructive process. However, formations only describe the "spatial" component of the process, i.e., which constructive transformations and "where" are applied. The *stochastic* generating process adds *temporal* information to the one already stored in a formation. Again, since both concepts—formation and stochastic generating process—essentially describe the same thing—a constructive process—they should better be unified and studied as a single data structure, which is also left for future research.

Because my formal understanding of the concept of generating process is now at such an early stage, compared to the rest of the model, I have decided to give only a semi-formal exposition of its theory here. The reader interested in some formal results about the first level generating process is referred to [Golub02]. The main reason why the generating process is being discussed here is that this is the central concept, around which the ETS model currently continues to develop. Once formulated, it will allow to complete the formalization of the model with the formulation of the inductive inference (learning) problem as an optimization problem and proceed toward the design of the inductive learning algorithm and applications.

5.1 Definition of the generating process

To account for the continuous temporal component of constructive processes (the structural component is already encoded in composites and formations), assume that each primitive in a composite takes a certain time to be applied, depending on the primitive's type. To reflect this assumption, we add to the description of an inductive structure $\langle \Pi, \mathcal{I} \rangle$ a mapping $t : \bar{\Pi} \to \mathbb{R}_+$. For a primitive type $\bar{\pi} \in \bar{\Pi}$, $t(\bar{\pi})$ signifies the *mean* time it takes to apply any primitive $\pi \in \bar{\pi}$ (which implies that this time is a random variable). The triple $\langle \Pi, \mathcal{I}, t \rangle$ is called a **weighted inductive structure**.

For a struct $\boldsymbol{\gamma}$, let

$$A(\boldsymbol{\gamma}) \overset{\text{def}}{=} \{ [\![\gamma, \langle \pi \rangle]\!] \mid [\![\gamma, \langle \pi \rangle]\!] \text{ is composable}, \gamma \in \boldsymbol{\gamma}, \quad \pi \in \Pi \}$$

be the **set of applications** (of primitives from Π) to $\boldsymbol{\gamma}$. The **result of application** $[\![\gamma, \langle \pi \rangle]\!] \in A(\boldsymbol{\gamma})$ is the struct $\boldsymbol{\gamma}' \overset{\text{def}}{=} K([\![\gamma, \langle \pi \rangle]\!])$.

Definition 46. For a weighted inductive structure $\langle \Pi, \mathcal{I}, t \rangle$, a **generating process** is a continuous parameter Markov chain $G = x_t(\omega)$ (or, equivalently, a stochastic

Markov process with a countable state space [Chu60]) defined as follows:

1. The states of G are structs
2. The initial state of G is the empty struct $[\![\lambda]\!]$.
3. The amount of time G spends in a state $\boldsymbol{\gamma}$ is a random variable distributed exponentially with mean
$$T = \frac{1}{\sum\limits_{[\![\gamma,\langle\pi\rangle]\!]\in A(\boldsymbol{\gamma})} 1/t(\bar{\pi})} \tag{$*$}$$
4. When G leaves state $\boldsymbol{\gamma}$, it chooses randomly an application $[\![\gamma,\langle\pi\rangle]\!] \in A(\boldsymbol{\gamma})$ with probability
$$\frac{T}{t(\bar{\pi})} \tag{$**$}$$
and enters the result of application $\boldsymbol{\gamma}' = K([\![\gamma,\langle\pi\rangle]\!])$.
5. All random variables in 3 and 4 are mutually independent.

►

I would like to state explicitly the assumptions on the generating process which have lead to the above definition.

First, since no generating process can be fully isolated from the surrounding environment, its behavior cannot be completely and deterministically specified by its own description. At the same time, we cannot take all external factors into account explicitly, since there are too many of them. Hence, we have to use a *probabilistic* model.

Second, the assumption that the corresponding stochastic process is a Markov process, i.e., its behavior depends only on the current state and not on any of its previous states, is based on the fact that the current state of the process is a struct which already contains all necessary information about its constructive history, thus there is no need to refer to the previous states.

Third, the time the process spends in a state, until a primitive is applied and the state changes, is distributed exponentially, because it is assumed that the primitives, from the perspective of the current level generating process, are *indivisible.* I.e., a primitive is either completely applied, which causes a change in the state of the process, or not applied at all, meaning that no changes are made whatsoever. This implies that if ξ is the random amount of time the process spends in a state, then for all $T, t > 0$,

$$\mathcal{P}(\xi < T + t \,|\, \xi > T) = \mathcal{P}(\xi < t).$$

The above property means that the system is memoryless and implies that the random variable ξ is distributed exponentially [Fe66, Chapter I].

Fourth, transformations that are applicable to a fixed state $\boldsymbol{\gamma}$ are assumed to be *independent* of each other. Thus, random variables $\xi_1, \ldots, \xi_k$, which denote the waiting times for different applications to $\boldsymbol{\gamma}$, are mutually independent. Since the process remains in state $\boldsymbol{\gamma}$ until one of the applications occurs, the time the process spends in $\boldsymbol{\gamma}$ is expressed by the random variable $\xi = \min(\xi_1, \ldots, \xi_k)$. The minimum of mutually independent exponentially distributed random variables is an exponentially distributed random variable with mean $(*)$ [Fe66, Chapter 1]. Also, the probability that a particular random variable ξ_i is minimal among $\{\xi_1, \ldots, \xi_k\}$ is proportional to the inverse of the expectation of ξ_i, hence we obtain formula $(**)$.

Finally, we can assume that all random variables are mutually independent, if we postulate that the choice of a particular application to a struct is independent of both the constructive historic process of the struct and the time this process has taken. This postulate arises naturally from the fact that a struct represents a set of constructive processes *indistinguishable* (or equivalent) from the point of view of a higher level primitive.

5.2 Typicality measure

A stochastic generating process induces a probability measure on the set of structs (of a particular representational level) as follows. Assume that the process is being *observed* by an external observer, who "approaches" the process at a random time moment and "records" its state. The probability that a particular struct is recorded is called the **typicality** of this struct for the observer.

In order to define the typicality measure formally, we first need to provide a formal model of the observer. Take the simplest possible model: assume that the expected waiting time for the observer to come is always the same, which implies that the moment of observation is an exponentially distributed random variable. Call it ξ_u:

$$\mathcal{P}\{\xi_u(\omega) < t\} = 1 - e^{-ut}, \qquad t \geq 0,$$

where u is the parameter of exponential distribution. Also, assume that this random variable is independent of the generating process $x_t(\omega)$, i.e., independent of each random variable $x_t(\omega)$ for all $t \geq 0$.

Given this model of an observer, the typicality measure on the set of structs is defined as follows:

$$\mathbf{g}_u(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \mathcal{P}\{x_{\xi_u(\omega)}(\omega) = \bar{\boldsymbol{\alpha}}\}.$$

The typicality measure can be computed in $O(|\operatorname{Anc}(\boldsymbol{\alpha})|^2)$ time (see [Golub02, Section 7]), where $\operatorname{Anc}(\boldsymbol{\alpha})$ denotes the set of ancestors of the struct $\boldsymbol{\alpha}$. The reader can also find some examples of typicality measures for the inductive structures of natural numbers, binary sequences, insertion strings, and graphs in [Golub02].

5.3 Global typicality

From now on, the observer's parameter u will be included into the description of the weighted inductive structure, resulting in an *observed* weighted inductive structure $\langle \Pi, \mathcal{I}, t, u \rangle$. This inductive structure is endowed with a typicality measure $\mathbf{g} \overset{\text{def}}{=} \mathbf{g}_u$ on the set of structs.

As it is clear from the definition of a first (or higher) level composite (Def. 39), it requires an initial composite from the previous level, in order to be constructed. Therefore, if for a particular composite $\dot{\gamma}$, the composites from $\operatorname{init}(\dot{\gamma})$ have a low typicality w.r.t. the previous level typicality measure, it should naturally follow that the typicality of $\dot{\gamma}$ is also low.

Two minor technical problems need be addressed here, in order to account for the dependence of the typicality of a composite $\dot{\gamma}$ on the typicality of its initials: first, we clearly need a combination of the typicality measures from different levels, and second, each one of them is defined not on composites but on structs, so we shall extend the concepts of initials and terminals to structs.

In what follows, the current level composites are denoted by dotted letters (e.g. $\dot{\alpha}$), and the ones from the previous level by letters without dots (e.g. α).

First, let us introduce for a current level formation $[\dot{\gamma}]$ an auxiliary set, which is the union of all site pairs of composites in $[\dot{\gamma}]$: let

$$s([\dot{\gamma}]) \overset{\text{def}}{=} \bigcup_{\dot{\alpha}\in[\dot{\gamma}]} \text{sites}(\dot{\alpha}).$$

It turns out that $s([\dot{\gamma}])$ can be represented as a disjoined union of previous level struct pairs:

Lemma 43. If $\langle\alpha,\beta\rangle \in s([\dot{\gamma}])$, then $[\![\alpha,\beta]\!] \subset s([\dot{\gamma}])$.

Proof. Let $\langle\alpha',\beta'\rangle \in [\![\alpha,\beta]\!]$. Let $\dot{\gamma}$ be an attachment sequence for $\dot{\gamma}$ such that $\alpha \in \text{init}(\dot{\gamma})$ and $\beta \in \text{res}(\langle\alpha, \text{body}(\dot{\gamma})\rangle)$. According to the Struct Projection lemma (L. 24), there exists a composite tuple B' such that

$$[\![\alpha, \text{body}(\dot{\gamma}), \beta]\!] = [\![\alpha', B, \beta']\!].$$

Due to the Composite Reconstruction lemma (L. 32), there exists first level composite $\dot{\gamma}' \in [\dot{\gamma}]$ such that $\text{body}(\dot{\gamma}') = B$. Then, $\alpha' \in \text{init}(\dot{\gamma}')$, since struct tuple

$$[\![\alpha', \text{body}(\dot{\gamma}')]\!] = [\![\alpha, \text{body}(\dot{\gamma})]\!]$$

is composable.

To show that $\beta' \in \text{res}(\langle\alpha', \text{body}(\dot{\gamma}')\rangle)$, we will prove that for any composite δ',

$$K_n([\![\alpha', \text{body}(\dot{\gamma}'), \delta']\!]) = [\![\beta', \delta']\!],$$

where $n = |\dot{\gamma}|$, and then apply the Result Criterion (L. 29). According to the Struct Projection lemma, there exists δ such that

$$[\![\alpha', \text{body}(\dot{\gamma}'), \delta', \beta']\!] = [\![\alpha, \text{body}(\dot{\gamma}), \delta, \beta]\!].$$

Then, since $\beta \in \text{res}(\langle\alpha, \text{body}(\dot{\gamma})\rangle)$ and, due to the Result Criterion,

$$K_n([\![\alpha', \text{body}(\dot{\gamma}'), \delta', \beta']\!]) = K_n([\![\alpha, \text{body}(\dot{\gamma}), \delta, \beta]\!]) = [\![\beta, \delta, \beta]\!] = [\![\beta', \delta', \beta']\!].$$

Hence,

$$K_n(\llbracket \alpha', \mathrm{body}(\dot{\gamma}'), \delta' \rrbracket) = \llbracket \beta', \delta' \rrbracket.$$

■

The above lemma justifies the following definition of the set of site pairs for a current level formation:

Definition 47. For a current level formation $[\dot{\gamma}]$,

$$\mathrm{sites}([\dot{\gamma}]) \overset{\mathrm{def}}{=} \{\llbracket \alpha, \beta \rrbracket \mid \langle \alpha, \beta \rangle \in s([\dot{\gamma}])\}.$$

►

The **initial and terminal structs** of $[\dot{\gamma}]$ are defined as projections of $\mathrm{sites}([\dot{\gamma}])$ onto the first and second components, respectively.

As it follows from Lemma 36, the set $\mathrm{sites}([\dot{\gamma}])$ is invariant w.r.t. the semantic equivalence relation $\sim_{\dot{\mathcal{I}}}$, which implies correctness of the following definition:

Definition 48. For a current level struct $\dot{\boldsymbol{\gamma}}$,

$$\mathrm{sites}(\dot{\boldsymbol{\gamma}}) \overset{\mathrm{def}}{=} \mathrm{sites}([\dot{\gamma}]),$$

where $[\dot{\gamma}]$ is any formation from $\dot{\boldsymbol{\gamma}}$. ►

Now we are ready to introduce the global typicality measure:

Definition 49. Let

$$\mathbb{I}, \dot{\mathbb{I}}, \ddot{\mathbb{I}}, \ldots, \mathbb{I}^{(k)}, \ldots$$

be an infinite hierarchy of observed weighted inductive structures, where each inductive structure $\mathbb{I}^{(k)}$ is assumed to be endowed with a typicality measure $\mathbf{g}^{(k)}$.

For a basic level struct, the global typicality measure is, by definition, equal to the typicality measure of the basic level:

$$\mu(\boldsymbol{\gamma}) \overset{\text{def}}{=} \mathbf{g}^{(0)}(\boldsymbol{\gamma}).$$

For all $k > 0$, the **global typicality measure** of a k-th level struct $\boldsymbol{\gamma}^{(k)}$ is defined by the following recurrence relation:

$$\mu(\boldsymbol{\gamma}^{(k)}) \overset{\text{def}}{=} \mathbf{g}^{(k)}(\boldsymbol{\gamma}^{(k)}) \cdot \sum_{\boldsymbol{\alpha}^{(k-1)} \in \text{init}(\boldsymbol{\gamma}^{(k)})} \mu(\boldsymbol{\alpha}^{(k-1)}). \tag{5.1}$$

▶

The first factor in (5.1) is the typicality of $\boldsymbol{\gamma}^{(k)}$ w.r.t. the k-th level generating process, and the second factor represents the typicality of the "prerequisites" for this process, i.e., the sum of typicalities of all initial structs for the struct $\boldsymbol{\gamma}^{(k)}$.

5.4 Structural measurement and inductive learning

From the ETS point of view, the goal of inductive learning is construction of new object representations, which are in some sense advantageous to the old ones and provide a better adaptation of the cognitive agent to the environment. The construction of new representations is inevitably based on the agent's previous knowledge, i.e., on its previous representational capabilities.

Within the conventional classification of machine learning paradigms [Br96], the model suggested here falls into the category of *symbolic empirical learning*, i.e. given examples from a class, it attempts to learn the class description, which can then be used to recognize other objects as instances of the class, as well as generate representations of new objects from the class. The learning optimization criterion is based not on a similarity measure (which is the most common choice in symbolic empirical learning models) but on the class typicality measure introduced in the previous section. It is also useful to note that in ETS the *instance space* and the *hypothesis space* are essentially the same spaces, because both structs and transformations are constructed from primitives of the same inductive structure (see also the discussion of implications below), as well as the typicality measures in these spaces are similar. Below we formulate the supervised version of the learning problem; more specifically, it is the problem of learning of a single class description from positive examples.

In terms of the ETS model, the representational capabilities of a cognitive agent A can be described as a finite hierarchy of representational levels

$$\mathcal{H}_A = \mathbb{I},\, \dot{\mathbb{I}},\, \ddot{\mathbb{I}}, \ldots, \mathbb{I}^{(k)}.$$

Then, the goal of inductive learning is the construction of the next, $(k+1)$-th, level, yielding a new hierarchy

$$\mathcal{H}'_A = \mathbb{I},\, \dot{\mathbb{I}},\, \ddot{\mathbb{I}}, \ldots, \mathbb{I}^{(k)}, \mathbb{I}^{(k+1)}.$$

The existing levels may also change, but the most essential part of learning is still the construction of a new level, hence I omit here the discussion of possible changes of the existing levels.

Since learning always involves some kind of interaction of the agent with the environment, we also need to describe this process of interaction. It will be called the **structural measurement process** (see also [GG01]). Note that both structural and conventional numeric measurement processes interact with real objects and produce their representations—but there is also an important difference in the form of the resulting representations. Even though most of the existing measurement devices are numeric[1], the structural measurement devices are expected to be much more general. For instance, one can think of organs of perception and sensation, which all animals have, as examples of structural measurement devices. The fact that human sensation is, to a large extent, inseparable from further processing and classification, is also reflected in the ETS model (but certainly not in numeric models).

To describe interaction with environment, assume the global environmental process, which creates and modifies all objects in the environment, is described by a generating process in a certain "huge" high-level representational hierarchy $\mathcal{H}_E$. Also, assume that representational hierarchies of the cognitive agent ($\mathcal{H}_A$) and of the environment ($\mathcal{H}_E$) have some common primitives. In other words, assume that the perceived object and the perceiving device are, perhaps at a very low level of the representational hierarchy, made of the same primitives (this does not have to be level 0, though). Then $\mathcal{H}_E$ can provide contexts for primitives from $\mathcal{H}_A$ and, vice versa, $\mathcal{H}_A$ can influence $\mathcal{H}_E$ in the same way.

Creation of new contexts by the environmental process $\mathcal{H}_E$ opens new directions for the agent's generating process, since the transformations that did not have matching contexts before may now become applicable. This affects the agent's global typicality measure μ, resulting in the increase of typicality of those structs, whose primitives use the new contexts. As a result, these structs are *observed* (in the sense of Section 5.3) more and more often and eventually extracted into a separate collection called the **training set** (assume that the agent has a special extraction mechanisms that creates training sets from the structs with high typicality). Since a collection of

[1]See an example of a "not-quite-numeric" measurement in Section 1.4.

structs can always be represented by a single disjoined struct, we will speak about the **training struct**, which, in general, corresponds to a training set.

Extraction of a training struct induces a change in the agent's representational hierarchy $\mathcal{H}_A$, via an **inductive learning step**. The result of a single inductive learning step can be defined formally as a solution to the following optimization problem:

Definition 50. Given a k-th level training struct $\boldsymbol{\beta}^{(k)}$, construct a $(k+1)$-st representational level $\langle \Pi^{(k+1)}, \mathcal{I}^{(k+1)}, t^{(l+1)}, u^{(k+1)} \rangle$ such that

$$\mu(\bar{\Pi}^{(k+1)}) \cdot \sum_{\boldsymbol{\gamma}^{(k+1)}:\, \boldsymbol{\beta}^{(k)} \in \mathrm{term}(\boldsymbol{\gamma}^{(k+1)})} \mu(\boldsymbol{\gamma}^{(k+1)}) \tag{5.2}$$

is maximal. ▶

The first factor in (5.2) is the typicality of the new set of primitive types $\bar{\Pi}^{(k+1)}$. Since these primitive types have to be constructed by the agent's generating process, the higher their typicality is, the more likely they will appear as a result of the inductive step. As shown in Section 3.1.7, each $(k+1)$-level primitive type can be specified by a k-th level composable struct pair $[\![\alpha^{(k)}, \beta^{(k)}]\!]$. We assume that the typicality of this struct pair is computed as the typicality of its composition, i.e., equals

$$\mu([\![\alpha^{(k)} \lhd \beta^{(k)}]\!]).$$

Next, a set of primitive types $\bar{\Pi}^{(k+1)}$ can be thought of as one "disjoined" k-th level struct pair, whose typicality is denoted by $\mu(\bar{\Pi}^{(k+1)})$ in (5.2).

The second factor in (5.2) is the new typicality of the training struct $\boldsymbol{\beta}^{(k)}$ computed after its conversion to the $(k+1)$-st level. Indeed, after the inductive step is completed, $\boldsymbol{\beta}^{(k)}$ acquires a new representation, which can be any $(k+1)$-level struct $\boldsymbol{\gamma}^{(k+1)}$ whose minimal terminal struct equals $\boldsymbol{\beta}^{(k)}$. If several minimal terminal structs exist, any one will do for the $(k+1)$-level representation of $\boldsymbol{\beta}^{(k)}$, hence we obtain the sum of their typicalities for the new typicality of $\boldsymbol{\beta}^{(k)}$ as in (5.2).

Roughly speaking, the optimization criterion "works", because the first factor limits the size of the new $(k+1)$-level primitives from above (since very large primitives become untypical), and the second factor, on the contrary, forces to decrease the number of primitives in $\boldsymbol{\gamma}^{(k+1)}$, thus limiting the size of the primitives from below. Correspondingly, the maximum is achieved for some primitives of "intermediate size", so that the trivial extreme solutions, such as the one where each new primitive consists of a single k-th level primitive or, on the contrary, where a new primitive is equal to the entire training struct $\boldsymbol{\beta}^{(k)}$, are excluded. An illustration of the behavior of the optimization criterion based on a simple inductive structure of shapes, with computation of the corresponding typicalities, can be found in [GG01].

I do not know yet any efficient algorithm that would solve the above optimization problem exactly. However, I suggest that even an exponential algorithm can be fast enough. Indeed, the size of $(k+1)$-level primitives, which need be composed from known k-th level primitives, should normally be small. An indication for it has been noted by psychologists, who have observed that human capability to process several different representations at once (in short-term memory) is limited to a very small number of them, known as "the magic number seven" [So88, Chapter 6]. However, the *structural complexity* of each of these representations is not limited, in principle, which in the ETS model is reflected by the fact that they can correspond to structs of any level. Thus, if the cognitive agent has to combine significantly more than seven representations together, this is done via several inductive learning steps, involving construction of several intermediate representational levels. Such gradual construction algorithm can be based on a greedy strategy and therefore should have a smaller complexity, compared to an algorithm that searches for an optimal large transformations at a fixed representational level.

5.5 ETS learning criterion and MDL principle

One can consider the optimization criterion formulated in Def. 50 as an instance of the minimum description length (MDL) principle in its probabilistic interpretation (sometimes called the principle of stochastic complexity) [Ri89, Section 3.6]. Here we discuss the relationship between them.

The MDL principle is formulated as follows (see [Gr98]). Given a countable set A, whose elements are called *data items.* A *probabilistic model* (also called a *hypothesis*) is, in general, a probability distribution defined over arbitrary long data samples $x = x_1, \dots, x_n$, $x_i \in A$. Let $\mathcal{M}$ be a class of probabilistic models parametrized by a set Γ:

$$\mathcal{M} = \{P(\cdot|\theta) \mid \theta \in \Gamma\}.$$

Here $P(\cdot|\theta)$ means the probability of the data given that the model used is the model named θ. For a sequence of data samples x, a hypothesis $H \in \mathcal{M}$ is called the *best hypothesis* according to the MDL principle, if the product

$$P(\theta) \cdot P(x|\theta) \tag{5.3}$$

is maximal. The above expression implies that the MDL principle requires a distribution on the class of models $\mathcal{M}$ (or, equivalently, on its parameterizing set Γ), denoted

$P(\theta)$, which is known as a *prior.* Our discussion here will be centered around this concept, since it looks like here the ETS learning criterion (Def. 50) has a fundamental advantage over the general formulation of the MDL principle and many of its applications.

Clearly, the selection of the prior significantly affects the MDL criterion However, as mentioned in [Ri89], this selection is not guided by any formal principle whatsoever. On the contrary, it is suggested that the priors are determined by "the code lengths of the coding system defined by the ground language, the mixture of English and mathematics. [...] since we have not formalized the ground language, caution and judgment are needed in estimating the relevant code lengths. This ambiguity, we think, is here to stay; there is no way to reconcile the arbitrariness in the formalization of the language and the demands of intuition" ([Ri89, p.81]).

I cannot agree with such categoric claim that it is impossible in principle to formalize the informal ground language, since this is precisely what the ETS model attempts to do. In the ETS formalism, every hypothesis (in the MDL sense) is a certain high level inductive structure, whose description is constructed based on the known previous level inductive structure. Moreover, the k-th level inductive structure induces a measure on the descriptions of the $(k + 1)$-st level inductive structures, denoted by $\mu(\bar{\Pi})$ in Def. 50. This measure plays exactly the role of a prior, with the only difference that its selection is now based on the known k-th level inductive structure and the inductive inference algorithm (and not on any other informal ground language). It is only the choice of the basic level inductive structure, that lies outside the formal scope of our model, whereas all higher levels are supposed to be constructed *automatically*, as a result of several inductive learning steps.

This distinction between the ETS optimization criterion and the general formulation of the ETS principle exhibits itself also in a different form. The two factors in Def. 50,

$$\mu(\bar{\Pi}^{(k+1)}) \quad \text{and} \quad \sum_{\boldsymbol{\gamma}^{(k+1)}:\, \boldsymbol{\beta}^{(k)} \in \text{term}(\boldsymbol{\gamma}^{(k+1)})} \mu(\boldsymbol{\gamma}^{(k+1)}),$$

are, essentially, based on the *same* typicality measure μ. On the contrary, in various various applications of the MDL principle, due to a significant difference in the descriptions of the *hypotheses* and the *data*, the reasons behind the choice of the *prior* $P(\theta)$ (=distribution of the hypotheses) and *probabilistic models* $P(x|\theta)$ (=distributions of the data) are also completely different. For example, in [Ke97], the MDL principle is applied to the grammatic inference problem, in which the hypotheses are specified by probabilistic grammars. It is clear that the formulas for the prior, i.e., the probability of a grammar, and for the probabilistic models, i.e., the probability of derivation of a particular sentence by a given grammar, are chosen in [Ke97] on completely different grounds. As a result, the two probabilities in the MDL principle, $P(\theta)$ and $P(x|\theta)$, are, in principle, *incommensurable*, which makes their combination in a single optimization criterion not quite meaningful. The importance of providing such meaning is also emphasized in [Ri89]. In the ETS model, the meaning is provided by the fact that both hypothesis and data are expressed in the *same* language, since they both are represented in certain inductive structures.

Chapter 6

Conclusion and future directions

The formal framework proposed in this thesis (see also [GGK01]) defines the concept of structural object representation as a collection of semantically equivalent representations of object constructive histories. A constructive history is defined as a sequence of elementary constructive steps, called primitives. A primitive can be a basic level primitive, in which case it is defined as a triple consisting of a label and two finite ordered sets of sites. The sites specify whether and how several primitives can be sequentially attached to each other, thus forming a composite. A primitive can also be a higher (first, second, etc.) level primitive, in which case it consists of a context and a body, both being previous level composites. Context and body play the role of sites at higher levels, i.e., they determine whether several higher level primitives can be sequentially attached to each other, thus forming a higher level composite. When the sites are "erased" from a composite (formally, an equivalence class of composites with respect to site replacements is considered), the concept of formation is obtained, which corresponds to a representation of an object constructive history. The semantic equivalence relation is induced on the set of formations by a finite set of semantic identities; each identity represents a postulated equivalence between two (short) segments of constructive history. An equivalence class of formations with respect to semantic equivalence relation is called a struct and corresponds to a representation of a single object. Thus, each level of representation consists of primitives, composites, formations, and structs; it is completely specified by a set of primitives and a set of semantic identities. Together, all representational levels form an infinite representational hierarchy.

The inductive learning (or inference) problem is formulated as a problem of constructing new next level primitives, given a training struct at a certain level. In order to formulate this problem as an optimization problem, for each representational level, weights on its primitives are introduced. These weights and interpreted as the mean times of application of elementary constructive steps, to which the primitives correspond. A stochastic process (continuous parameter Markov chain), called the generating process, which applies primitives according to their weights and produces structs is defined. The generating process induces a typicality measure on structs, at each level. The typicality measures for all levels are unified into a global typicality measure. The optimization criterion for inductive learning has the form of a minimum description length criterion and requires to maximize the typicalities of both new next level primitives and the training struct represented at the next level via these primitives. The problem of defining the complexities of class description and of the training set within a class in a consistent manner, which is common for the applications of the MDL principle, is resolved by putting both class descriptions and training sets into a general framework of representational levels, in which both complexities are computed in essentially the same way, via the above typicality measure.

In applications of the framework, the basic representational level needs to be postulated. All higher levels can, in principle, be constructed by the inductive learning algorithm (assuming that the latter can be designed based on the above optimization criterion). Thus, the framework suggests a systematic approach for construction of representations of complex objects. These representations are automatically classified, where the classes are specified by the descriptions of representational levels. The lower the level, the broader the corresponding class is; the basic level corresponds to the universal class that contains all objects in discourse of a given scientific problem.

As most of the concepts are defined as certain equivalence classes, correctness of the corresponding definitions (i.e., invariance with respect to the choice of representatives) is proved (which takes a significant part of proofs in this thesis). The proofs are presented in a level-invariant form, so that the inductive construction of an infinite hierarchy of representational levels becomes possible. The question of finiteness of object representations as collections of representations of constructive histories is studied formally and a criterion of finiteness is proved. It is also proved that the equality relation is decidable for finite object representations but undecidable in general. In addition, it is shown that the question whether all structs in a given inductive structure are finite is undecidable in general. The generative power of first level primitives is compared with that of existing generative formalisms, string and graph grammars; it is proved that any strongly acyclic string-rewriting system (or, equivalently, unrestricted formal grammar) can be isomorphically simulated in a certain first level inductive structure.

Conventional data structures, including natural numbers, sequences, strings, trees, and graphs, are shown to be particular cases of basic level inductive structures. Once represented as inductive structures, strings, trees, and graphs reveal an exponential number of constructive histories behind them, which is argued to be the reason for the intractability of the inductive inference, and sometimes even parsing, problems for the corresponding grammars. It is suggested that the number of constructive histories should be decreased as much as possible (which, of course, requires explicit representation of them), in order to allow for efficient inductive inference. The proposed definition of transformations (i.e., next level primitives) as context-dependent attachments of composites (without deletions or substitutions) should facilitate the development of the inductive inference algorithm as well.

The "proof of concept" for the proposed model, of course, depends on the success of its applications. Out of a broad range of possible applied areas, I would like to select bioinformatics, computational linguistics, artificial intelligence, and pattern recognition as the most promising ones. These areas have nowadays attracted a greater interest from the scientific community than ever before because of a tremendous accumulation of new experimental data and challenging problems of classification and explanation of these data. In my opinion, our ability to explain them, essentially depends upon the proper understanding and formalization of the concept of structural representation. Conventional numeric models do not fulfill this purpose, as is pointed in the introduction to this thesis and referred sources in the literature. However, since most of the existing formalisms are based on numeric concepts, and structural concepts seem to be so much different, it should not be forgotten that the development of structural formalisms is a very complex and long-term task, and this thesis pretends to have made one of the first little steps toward its solution.

Speaking about the following steps, which I can foresee in the nearest future, one can single out the following two main directions. First, there are still concepts that need further formal clarification. I think that the core concept, the representational hierarchy, has currently acquired quite a reliable formalization. Yet, a full formal understanding of the multi-level generating process and the inductive inference problem is still to be developed. Only then one can start working on algorithms for inductive inference (for particular conventional data structures, such algorithms have been proposed and studied in [Dew91, Sa92, Ni93, GN94, Kam95, Ab02]). Second, in order to resolve computational problems successfully, one has to establish further connections with the existing computational models, or prove that they are inadequate and design a new one. In particular, I would like to investigate more the generating power of high-level inductive structures and compare it with the power of existing generative formalisms. Also, based on the definition of infinite representational hierarchy, one can try to obtain a new description of classes of good instances for computationally hard problems, e.g., NP-complete problems. From the purely theoretical perspective, it would be interesting to consider infinite structures corresponding to the finite formations and structs introduced in this thesis. Since many (if not all) conventional data structures can be expressed as particular cases of inductive structures, one will then immediately obtain a theory of infinite data structures. We have already started (see the forthcoming paper [Golub03]) to investigate this direction and obtained a topology on the set of insertion strings with a countable compactification, which turned out to be related with the the well-quasi-ordering property of the part/whole relation on strings (see [Der90]). This study can potentially produce new results about classification of countable topological spaces and related to stochastic processes with countable state spaces.

As it has always been the case in science, formal models that arise from the study of a completely new natural phenomenon, cannot fail to produce non-trivial mathematical results. It has often been the case that these results appeared long before the practical applications of the models were found. Since I do not doubt the novelty of the concept of structural representation, and also hope that the formal model presented in this thesis has grasped at least some of its important features properly, I expect that some non-trivial results will appear down the road of development of the model.

Appendix A

Primitive types

The definition of primitives (Def. 1) raises several questions, such as: Why are primitives defined in this way? What do they correspond to in real world? Why are the sets of initial and terminal sites ordered?

Let us introduce auxiliary concepts of a primtype, primtype realization, and connection, in order to answer these questions and provide a mental image for primitives and composites.

Definition 51. A **primitive type**, or **primtype**, is a triple $\langle \alpha, i, t \rangle$, where α is any individual called the **label of the primitive type** and i, t are non-negative integers. ▶

Primitive types serve as templates for objects called *primtype realizations*:

Definition 52. For a primitive type $\langle \alpha, i, t \rangle$, a triple of the form $\langle \alpha, I, T \rangle$, where I and T are finite disjoint sets of cardinalities i and t respectively, is called a **primtype realization**. The elements of I and T are called the initial and terminal **connectors of the primtype realization** $\langle \alpha, i, t \rangle$. ▶

Pictorially, a primtype realization can be represented similarly to a simplistic representation of an atom (see Fig. A.1). However, the connectors of a primtype realization are distinguishable, whereas, one might argue, electrons are not. Moreover, the connectors of a primtype realization are split into initial and terminal. This is because a primtype realization (as well as a primitive) also stands for a primitive constructive operation, which naturally has a direction: it is applied to a context, which comes first, and then produces a result. We hope that the reader will not be confused by this dual interpretation of primtype realizations, simultaneously as a particle and as an operation.

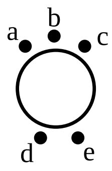

Figure A.1: A primtype realization.

Imagine an environment[1] consisting of objects, each of which can be represented by a primtype realization. For example, the primtype realizations could represent atoms, in which case the environment is called atomic. Other examples of environments include those of amino acids, proteins, cells, organs, organisms, societies. Aside from chemistry and biology, there exist virtual environments of phonemes, letters, words, sentences, visual primitives, etc.

Primtype realizations from one environment can be (and often are) composed of those from another environment (e.g., amino acids are composed of atoms and proteins are composed of amino acids).

Fix an environment and call the set of primtype realizations in it PR. Initially (i.e.

[1]This is an informal concept, which corresponds to the formal concept of inductive structure introduced in Section 2.3.

before any evolutionary or constructive processes commence), the primitive objects in the environment are independent, or disconnected. In terms of primtype realizations, this means that for any pair of distinct primtype realizations $\langle \alpha, I, T\rangle, \langle \alpha', I', T'\rangle \in$ PR,

$$(I \cup T) \cap (I' \cup T') = \varnothing.$$

Then, as a result of constructive processes, connections between primitive objects start to appear. A *connection* from a primtype realization $pr = \langle \alpha, I, T\rangle$ to a primtype realization $pr' = \langle \alpha', I', T'\rangle$ is specified by an injective partial mapping $f : I \to T'$ indicating which initial connectors of pr are connected to which terminal connectors of pr'.

A *pair of connected primtype realizations* is specified by a triple $\langle pr_1, pr_2, f\rangle$, where pr_1, pr_2 are distinct primtype realizations and f is a connection from pr_2 to pr_1. This pair corresponds to two interconnected primitive objects.

Consider a triple $pc \overset{\text{def}}{=} \langle pr, g, h\rangle$, where g is an injective mapping with domain I and h is an injective mapping with image T (the image of g and the domain of h can be any finite sets). This triple can be interpreted as a primtype realization $pr = \langle \alpha, I, T\rangle$ *potentially connected* with (in both directions) other primtype realizations.

A pair of potentially connected primtype realizations $pc_1 = \langle pr_1, g_1, h_1\rangle$ and $pc_2 = \langle pr_2, g_2, h_2\rangle$ corresponds to a pair of (actually) connected primtype realizations, $\langle pr_1, pr_2, f\rangle$, where $f = h_1 \cap g_2$.[2]

An injective mapping f with finite domain and image can be specified by fixing the set of pairs $B(f) = \{\langle x, f(x)\rangle\}$, known as the graph of f. Once $B(f)$ is fixed,

[2]In this formula, consider mappings h_1 and g_2 as relations, in order for their intersection to be meaningful.

the domain and image of f can be obtained as projections of $B(f)$ onto the first and second components, denoted $B_1(f)$ and $B_2(f)$. Any other injective mapping f' with the same domain and image can be specified by fixing two linear orderings $<_1, <_2$ on $B(f)$. Indeed, $<_1$ induces an ordering on the domain $B_1(f)$ and $<_2$ induces an ordering on the image $B_2(f)$. Since both are finite, f' can be specified as follows: if a is the i-th element of $B_1(f)$ with respect to $<_1$, then $f'(a)$ equals the i-th element of $B_2(f)$ with respect to $<_2$.

Now, a potentially connected primtype realization $pc = \langle pr, g, h \rangle$ can be equivalently specified by

- a pair of sets $B_I = B(g)$ and $B_T = B(h)$ and
- a pair of linear orderings, $<_1$ on B_I and $<_2$ on B_T, where $<_1$ is the first ordering for g and $<_2$ is the second ordering for h.

The 3-tuple $\langle \alpha, \langle B_I, <_1 \rangle, \langle B_T, <_2 \rangle \rangle$ is a primitive (see Def. 1). Thus, primitives are potentially connected primtype realizations.

Now it should be clear what the sites of a primitive are and why the sets of initial and terminal sites are ordered. Indeed, sites are points of the graph $B(f)$ of the injective partial mapping, which specifies a connection between two primtype realizations. For a pair of primitives, π_1 and π_2, the set $\mathrm{term}(\pi_1) \cap \mathrm{init}(\pi_2)$ and two orderings on it, which are induced from those on $\mathrm{term}(\pi_1)$ and $\mathrm{init}(\pi_2)$, specify a connection between the primtype realizations, that constitute the first components of primitives π_1 and π_2.

Now that connections are incorporated into primitives, we can define composites, which are quite complex graph-like objects, simply as *sequences* of primitives. All our subsequent definitions depend on this interpretation of composites.

References

[Ab02] J.M. Abela, ETS Learning of Kernel Languages, Ph.D. Thesis, Faculty of Computer Science, University of New Brunswick, 2002.

[Ai97] K. Aizawa, Explaining systematicity, *Mind and Language*, 12, 115–136, 1997.

[Bal01] W. W. Rouse Ball, *A Short Account of the History of Mathematics*, (originally published by Maximillan & Co., Ltd., London, 1912), Facsimile Edition, Sterling Publishing Company, New York, 2001.

[Bat85] V. Batagelj, Inductive classes of graphs, *Proc. Sixth Yugoslav Seminar on Graph Theory, Dubrovnik, 1985*, Novi Sad, pp. 43–56, 1986.

[Be01] S. Ben-David, N. Eiron, H. Simon, Limitations of learning via embeddings in Euclidean half-spaces, *Proc. COLT 2001*, Springer, Amsterdam, pp. 385–401, 2001.

[Bl95] D. Blostein, H. Fahmy, A. Grbavec, Practical use of graph rewriting, TR95-373, Department of Comp. and Inf. Sc., Queen's University, Kingston, Ontario, 1995.

[Bo93] R. Book, F. Otto, *String-Rewriting Systems*, Springer-Verlag, New York, Berlin, 1993.

[Br96] G. Briscoe, T. Caelli, *A Compendium of Machine Learning. Volume 1: Symbolic Machine Learning*, Ablex Publishing Corp., Norwood, New Jersey, 1996.

[Car01] Parser Comparison—Context-Free Grammar (CFG) Data, John A. Carroll's webpage,
`http://www.cogs.susx.ac.uk/lab/nlp/carroll/cfg-resources`

[Cam93] N. Campbell, *Biology*, Third edition, Benjamin/Cummings, Redwood City, CA, 1993.

[Cha92] T. Chan, Learning as Optimization, Ph.D. Thesis, U.N.B., 1992.

[Cho65] N. Chomsky, *Syntactic Structures*, Fifth printing, Mouton & Co., London, 1965.

[Chu60] K. Chung, *Markov Chains with Stationary Transition Probabilities*, Springer-Verlag, Berlin, 1960.

[Cu98] J. Culberson, On the futility of blind search: an algorithmic view of "No Free Lunch", *Evolutionary Computation Journal* 6(2), 109–128, 1998.

[Cu99] J. Culberson, I. Gent, Well out of reach: Why hard problems are hard, Research Report, APES-13, 1999.

[Der90] N. Dershowitz, J.-P. Jouannaud, Rewrite systems, Chapter 6 of *Handbook of Theoretical Computer Science, Volume B: Formal Methods and Semantics*, ed. J. van Leeuwen, North-Holland, Amsterdam, 243–320, 1990.

[Des96] S. Deshpande, On the Foundations of Vision, Master's Thesis, U.N.B., 1996.

[Dew91] S. Dewi, Dynamic Selection of Primitives in the Metric Model for Pattern Learning, Master's Thesis, U.N.B, 1991.

[Do78] J. Doob, *Stochastic Processes*, Wiley, New York, 1953.

[Fe66] W. Feller, *An Introduction to Probability Theory and Its Applications*, Volume II, Second Edition, John Wiley & Sons, New York, 1966.

[Fo88] J. Fodor, Z. Pylyshyn, Connectionism and cognitive architecture: a critical analysis, In S. Pinkerr & J. Mehler (Eds), *Connection and Symbols*, Cambridge, Mass., MIT Press, 1988.

[Fu74] K.S. Fu, *Syntactic Methods in Pattern Recognition*, volume 112 of Mathematics in Science and Engineering, Academic Press, New-York, 1974.

[Ga79] M. Garey, D. Johnson, *Computers and Intractability*, W.H. Freeman, 1979.

[Gr98] P. Grünwald, *The Minimum Description Length Principle and Reasoning under Uncertainty*, Ph.D. Thesis, ILLC Dissertation Series DS 1998-03, 1998.

[Go03] L. Goldfarb, Is there a different mathematics, "mathematics of the mind", that would explain the biological (structural) "measurement" processes? Lev Goldfarb's home page, `http://www.cs.unb.ca/profs/goldfarb`

[Go90] L. Goldfarb, On the foundations of intelligent processes I: An evolving model for pattern learning, *Pattern Recognition 23*, 595–616, 1990.

[Go96] L. Goldfarb, What is inductive learning? Construction of inductive class representation, *Proc. Workshop What Is Inductive Learning*, ed. L. Goldfarb, 9–21, Faculty of Computer Science, U.N.B., Fredericton, New Brunswick, 1996.

[GG01] L. Goldfarb, O. Golubitsky, What is a structural measurement process? Faculty of Computer Science, U.N.B., Technical Report TR01-147, 2001.

[GGK01] L. Goldfarb, O. Golubitsky, D. Korkin, What is a structural representation? Faculty of Computer Science, U.N.B., Technical Report TR01-137, 2001.

[GN94] L. Goldfarb, S. Nigam, The unified learning paradigm: A foundation for AI, in V. Honavar and L. Uhr, eds., *Artificial Intelligence and Neural Networks: Steps towards Principled Integration*, Academic Press, Boston, MA, 1994.

[Golub03] O. Golubitsky, Infinite strings generated by insertions, submitted to *Programming.*

[Golub02] O. Golubitsky, On the generating process and the class typicality measure, Faculty of Computer Science, U.N.B., Technical Report TR02-151, 2002.

[Gold78] E. Gold, Complexity of Automaton Identification from Given Data, *Information and Control 37*, 302–320, 1978.

[He89] W. Heisenberg, *Encounters with Einstein*, Princeton Univ. Press, Princeton, New Jersey, 1989.

[Hi52] G. Higman, Ordering by divisibility in abstract algebras, *Proc. London Mathematical Society (3)*, 2 (7), 326–336, 1952.

[Ho98] J. Hook, Are Artificial Neural Networks Learning Machines?, Master's Thesis, U.N.B, 1998.

[Kar72] R. Karp, Reducibility among combinatorial problems, *Complexity of Computer Computations*, Plenum Press, New York, 85–104, 1972.

[Kam95] V. Kamat, Inductive Learning with the Evolving Tree Transformation System, Ph.D. Thesis, U.N.B., 1995.

[Ke97] B. Keller, R. Lutz, Evolving stochastic context-free grammars from examples using a minimum description length principle, *Proc. Worksop on Automata Induction, Grammatical Inference and Language Acquisition*, Nashville, Tennessee, USA, 1997.

[Kn70] D. Knuth, P. Bendix, Simple word problems in universal algebras, *Proc. Conf. Computational Problems in Abstract Algebra*, Pergamon Press, Oxford, 1970.

[Ko03] D. Korkin, A new model for molecular representation and classification: formal approach based on the ETS framework, Ph.D. Thesis, Faculty of Computer Science, University of New Brunswick, 2003.

[Ku65] A.G. Kurosh, *Lectures in General Algebra*, Pergamon Press, Oxford, New York, 1965.

[La51] E. Landau, *Foundations of Analysis*, Chelsea, 1951.

[Le92] M. Leyton, *Symmetry, Causality, Mind*, A Bradford Book, The MIT Press, Cambridge, Massachusets, 1992.

[Li97] M. Li, P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*, Springer Verlag, New York, 1997.

[Ni93] S. Nigam, Metric Model Based on Generalization and Generalization Capabilities of Connectionist Models, Master's Thesis, U.N.B., 1993.

[Paul75] L. Pauling, P. Pauling, *Chemistry*, W. H. Freeman and Company, San Francisco, 1975.

[Paun98] G. Paun, G. Rozenberg, A. Salomaa, W. Brauer, *DNA Computing: New Computing Paradigms*, Springer Verlag, New York, 1998.

[Pr80] I. Prigogine, *From Being to Becoming*, W. H. Freeman and Company, San Francisco, 1980.

[Ri89] J. Rissanen, *Stochastic Complexity in Statistical Inquiry*, World Scientific Publishing, Singapore, New Jersey, 1989.

[Ro97] *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations*, ed. G. Rozenberg, World Scientific Publishing Co., Inc., River Edge, NJ, 1997.

[RS97] *Handbook of Formal Languages*, eds. G. Rozenberg, A. Salomaa, Springer-Verlag, Berlin, Heidelberg, 1997.

[Sch93] O. Schindewolf, *Basic Questions in Paleontology: Geologic Time, Organic Evolution, and Biological Systematics*, University of Chicago Press, 1993.

[Sa92] W. Santoso, Learning Algorithm for the Reconfigurable Learning Machine, Master's Thesis, U.N.B., 1992.

[Slo95] A. Sloman, Musings on the roles of logical and non-logical representations in intelligence. *Diagrammatic Reasoning: Computational and Cognitive Perspectives*, eds. J. Glasgow, H. Narayanan, Chandrasekaran, AIII Press, 1995.

[Slo00] A. Sloman, Interacting trajectories in design space and niche space: a philosopher speculates about evolution, *Proc. Sixth Intern. Conf. on Parallel Problem Solving from Nature*, Springer: Lecture Notes in Computer Science, eds. Schoenauer et al., 2000.

[Sle93] M. Sleep, M. Plasmeijer, M. van Eekelen (ed.) *Term Graph Rewriting*, John Wiley & Sons, Chichester, New York, 1993.

[So88] R. Solso, *Cognitive Psychology*, Second Edition, Allyn and Bacon, Inc., Boston, London, 1988.

Index

VITA

Candidate's full name: Oleg Dmitrievich Golubitsky

Universities attended:

PhD candidate in Computer Science	University of New Brunswick, Canada Dissertation: "On the Formalization of the Evolving Transformation System Model" (submitted, expected in March, 2004)
PhD in Mathematics	Lomonosov Moscow State University, Russia Dissertation: "Gröbner Walks" (2003)
Diploma in Mathematics	Lomonosov Moscow State University, Russia Thesis: "Involutive Bases and Gröbner Bases" (1999)

Journal publications:

O. Golubitsky, S. Falconer, Infinite strings generated by insertions (accepted for publication in Programming and Computer Software, 31, Mar-Apr 2004).

O. D. Golubitsky, Involutive Gröbner walk, Fundamental and Applied Mathematics, 7 (4): pp. 993–1001, 2001.

O. D. Golubitsky, Distance calculation on strings, Programming and Computer Software 26 (2): pp. 97–99, Mar–Apr 2000.

A. V. Astrelin, **O. D. Golubitsky**, and E. V. Pankratiev, Involutive bases of ideals in the ring of polynomials, Programming and Computer Software, 26 (1): pp. 31–35, Jan-Feb 2000.

Journal papers in preparation:

Lev Goldfarb, **Oleg Golubitsky**, What is a structural measurement process? (to be submitted to a special issue of Pattern Recognition).

Lev Goldfarb, **Oleg Golubitsky**, Dmitry Korkin, What is a structural representation in chemistry: Towards a unified framework for CADD, (to be submitted to a special issue of Pattern Recognition).

Lev Goldfarb, David Gay, **Oleg Golubitsky**, Dmitry Korkin, What is a structural representation? (to be submitted to a special issue of Pattern Recognition).

Technical report:

O. Golubitsky, On the generating process and the class typicality measure, Faculty of Computer Science, U.N.B., Technical Report TR02-151, 2002.

Refereed conference proceedings:

O. Golubitsky, Transformation of characteristic sets from one ranking to another. Proc. 8-th Rhine Workshop on Computer Algebra, pp. 225–236, Mannheim, Germany, 2002.

O. Golubitsky, Differential Gröbner Walk, Proceedings of International Workshop on Computer Algebra and its Application to Physics (CAAP), pp. 114–126, Dubna, Russia, 2001.

A.V. Astrelin, **O.D. Golubitsky**, E.V. Pankratiev, Comparison of the algorithms for computation of the Gröbner Bases and involutive bases, Proc. Int. Seminar Dedicated to the 70th Anniversary of the Chair of Higher Algebra of Moscow State University, Moscow University Press, pp. 9–10, 1999.

A.V. Astrelin, **O.D. Golubitsky**, E.V. Pankratiev, Gröbner bases and involutive bases, Proc. Int. Conf. Dedicated to the 90th Birthday of A.G. Kurosh, Walter de Gruyter, pp. 49–55, 1999.

Conference presentations and poster sessions:

L. Goldfarb, **O. Golubitsky**, Fundamental inadequacies of the traditional machine learning representations and a new formalism, Snowbird Learning Workshop (papers by invitation), Salt Lake City, April 2002.

O. Golubitsky, Classification using repetition-dependent transformation distance measure on strings, International Association for Mathematics and Computers in Simulation Conference of Applications of Computer Algebra (IMACS ACA), St.Petersburg, 2000.

O. Golubitsky, Uniqueness of the representation of a class of strings generated by insertions into the kernel, Computer Algebra Workshop, Dubna, Russia, 2000.

O. Golubitsky, The Involutive Gröbner Walk, International Association for Mathematics and Computers in Simulation Conference of Applications of Computer Algebra (IMACS ACA), Prague, 1998.