HIGHER-LEVEL MODELLING LANGUAGES AND (ANTI)REDUCTIONIST PERSPECTIVES WITHIN PHILOSOPHY

Catholijn M. Jonker[1] and Jan Treur[2]

[1]Radboud Universiteit Nijmegen, Nijmegen Institute of Cognition and Information,
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
Email: C.Jonker@nici.ru.nl
URL: http://www.nici.ru.nl/~catholj
[2]Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Email: treur@cs.vu.nl
URL: http://www.cs.vu.nl/~treur

KEYWORDS

Modelling languages, philosophy of science, reduction, antireductionist perspective, philosophy of mind

ABSTRACT

In the philosophical literature, it is implicitly assumed that the advantages of reduction of are based on the elimination of a higher-level theory. Antireductionist philosophers cast doubt on the existence of reduction relations between higher-level and lower-level theories. This is a debatable strategy in view of the scientific progress made in areas such as Biochemistry and Neuroscience. An alternative strategy accepts the existence of reduction relations, but claims that this can support an antireductionist view on development and use of higher-level theories. To support this position, as a case study, the historical development of modelling languages at different levels of abstraction is considered. It shows how a practitioner can concentrate work at a higher-level of abstraction, while hidden for the human, within a computer automated translations to lower levels are made based on reduction relations. The higher modelling levels dramatically increased the complexity of applications that came within reach.

1 INTRODUCTION

Within philosophical literature, and especially in literature in the area of Cognitive Science and Philosophy of Mind the issue of reduction is a point of interest; e.g., Kim (1996, 1998, 2005), Bickle (1998, 2003). Within the recent literature one of the main perspectives advocated is physicalism: the idea that all processes (among which mental and biological processes), in one way or the other have a physical basis. Within Biology the strong development of Biochemistry supports this perspective. For Cognitive Science, the strong development of Neuroscience and its underlying Biochemistry plays a similar role.

In the philosophical literature on reduction of scientific theories, the advantages for scientific practice of having a reduction relation between two theories are not always addressed explicitly. In such cases it is implicitly assumed that these advantages are based on the elimination of the higher-level theory. For example, Kim (1996, p. 214-216) emphasizes ontological simplification and having to deal with fewer assumptions about the world as advantages of reduction. Since in his perspective the lower-level theory is retained, ontological simplification entails giving up (eliminating) the ontology of the higher-level theory. Such a eliminative reductionist perspective provokes resistance from those researchers and philosophers who defend an autonomous status for higher-level theories in the special sciences such as Biology and Cognitive Science. Historically, a main strategy to attack the eliminative tendency in the literature on reduction has been to cast doubt on the existence of reduction relations between higher-level and lower-level theories; e.g., Davidson (1993). The idea is that when such relations do not exist, elimination of the higher-level theory is impossible; thus it is protected. From our point of view this is a debatable strategy. It can be questioned whether it is desirable to protect scientific theories that have no connection to other, more down-to-earth theories. Moreover, scientific progress made in areas such as Biochemistry and Neuroscience makes it harder and harder to persist on the view that reduction relations do not exist.

The alternative strategy put forward in this paper accepts the existence of reduction relations, but claims that this can perfectly well go hand in hand with an antireductionist view on the development of higher-level theories. To clarify the different possible positions more explicitly, in this paper we distinguish between:

- reduction in a *structural sense*: as a *reduction relation*, already established or being established, between two theories and their ontologies and laws, and
- the *pragmatics* related to reduction: the *use* of an existing or to be achieved reduction relation in scientific practice.

Our position is that an actual or envisioned structural reduction is compatible with the use and (further) development of the higher-level theory.

To support our position, as a case study, we consider the historical development within Computer Science and Artificial Intelligence of representation and programming languages. Briefly, the historical pattern is as follows; cf. Tanenbaum (1976), Knuth (1981), Booch (1991), Russell and Norvig, (2003), Brachman and Levesque (2004), Sowa (2004). In the early days of Computer Science, languages were used that describe, for example, the world by specifying step by step the (physical) transitions. For simple processes this may suffice. However, with a broadening of the scope of applications, these step-by-step descriptions became more and more complex and lacked transparency; therefore higher-level languages that abstract from many of the details of the world became indispensable. Elements were introduced in the higher-level languages by which a specification can be structured in terms of increasingly abstract functional units that cover larger parts of the processes (for example, procedures, modules, objects, components, agents, organisations). The result is an increase in the degree of complexity of the phenomena for which transparent specifications are feasible.

Each specification in a higher-level language can be translated into lower-level specifications, and ultimately into physical processes that are simulated within a computer. Translation is automated in a generic manner and hidden from the users of the higher-level languages. In order to perform simulations in the computer, higher-level specifications are automatically replaced to lower-level ones. The benefit for modellers is that by working at the more abstract level of the higher-level language, they can keep complexity within the scope of human capabilities; whereas with the lower-level specifications, the task would become too complex. Thus, the area of Computer Science and Artificial Intelligence shows on the one hand an *anti-reductionist pragmatics* exploited by a human practitioner, but on the other hand is based on explicitly defined *reduction relations* exploited by the computer, but hidden for the human practitioner.

The paper is structured as follows. First, Section 2 presents a brief overview of the development of different levels of language within Computer Science and Artificial Intelligence. Next, in Section 3 some of the concepts and issues from the literature on reduction are briefly summarized and related to the modelling languages case. Section 4 discusses three different contributions to the literature on scientific explanation and reduction: Jackson and Pettit (1988, 1990), Dennett (1987), and Bickle (1998). A common aspect in these three different positions is the inherently anti-reductionist perspective. In comparison, the pragmatics of the anti-reductionist perspective on explanation used in modelling languages is discussed. The different pragmatic uses of a reduction relation are discussed and compared in Section 5. Section 6 concludes the paper.

2 MODELLING LANGUAGES AT DIFFERENT LEVELS

In the earliest modelling languages[1] developed in Computer Science (machine languages, assembly languages) world state descriptions are based on *bit strings:* sequences of binary alternatives for the considered world state aspects, usually indicated by 0 and 1 for example: 10010110 00110101 11101011. For a computer these bits can be related to electrical signals within the hardware. Depending on which electrical circuits are causally affected by these physical bit representations, the processor successively modifies these physical states into other ones. Machine language can be used to describe such state transitions; this type of language is taken as the lowest computer language (Tanenbaum, 1976). Assembly languages are one step to come to a more abstract way of representing states and processes. Assembly and Machine Languages both are closely tuned to the brand and type of processor that is used in a computer, and modelling more complex processes in these languages is difficult and time consuming.

To overcome this, higher-level modelling languages, the so-called *structured programming* languages have been developed in which specifications of algorithms on a conceptual level are possible that do not refer to any specific computer (hardware) structure. Such languages are called *third generation languages*, after the two types of languages discussed above (machine and assembly language). A third generation modelling language, on the one hand, provides independent specifications of states that can be related to states in the world (giving the language independent semantics), independent of a computer type, at a conceptually higher level. On the other hand, well-defined relationships between such higher-level state and process specifications and state and process specifications defined at the level of bits are assumed, but these relationships are kept hidden; cf. Knuth (1981). At this language level, instead of states based on bits as in machine language, the states of a process specification (or program) are defined in a more abstract mathematical manner by values that are assigned to the variables chosen in the specification.

Within a specification in this modelling language (a program), assignment steps can be specified, describing the change of a state specification by assigning a new value to a variable as indicated (replacing the old value). Moreover, for example, a for-loop defines in one expression a sequence of successive steps, each of which involves a further series of steps, which finally are related to transitions of the actual states. This is an example of a higher-level concept in the language,

[1] Notice that for simplicity within this paper the languages (modelling languages, programming languages, specification languages, representation languages) developed within Computer Science and Artificial Intelligence will all be called modelling languages.

referring to a more complex process as a whole. Such concepts entail a more sophisticated and structured higher-level relationship over time in the trace of states than direct transitions.

Descriptions in a structured language such as the language C can be mapped onto specifications in assembly or machine language, which can result in quite complex specifications in general. Programs called *compilers* translate from a structured language to machine language. Such a translation is not unique: different compilers can be used, that map the same structured language specification onto different lower-level specifications.

A basis of a next level of modelling languages (the fourth generation) is formed by conceptual modelling languages, originating from the database area, and knowledge modelling languages originating from the area of Artificial Intelligence. An example of an early modelling language at this level for a specific type of application is the query language SQL. This modelling language allows for declarative expression of queries for a database, abstracting from the way in which the actual check of the query within a given database environment is performed. Usually such higher-level modelling languages are translated into a lower-level program (interpreted or compiled). Other examples of such higher level modelling languages can be found in the knowledge representation area within Artificial Intelligence going back to the 1970s. Examples of types of such modelling languages are rule-based, frame-based and semantic network-based languages, description languages, logic programming languages; cf. Minsky (1975), Russell and Norvig (2003), Brachman and Levesque (2004). These modelling languages have had a crucial impact on the development of the object-oriented modelling paradigm within Software Engineering; cf. Booch (1991). A standard recent example of a modelling language at this level is UML; cf. .Booch et al., (1999), D'Souza and Wills (1998), Fowler et al. (2000). Besides, currently in the development of areas such as Knowledge Engineering, Agent Systems, and Semantic Web this type of high-level modelling languages play a central role; cf. Guarino (1998), Brachman and Levesque (2004), Sowa (2000). Process steps for such modelling languages are often described by inference techniques, specified in terms of a lower level language.

Within Philosophy two different types of reduction relations are distinguished: definitional or empirical, see Kim (1996). The concepts and relations representable in the higher level modelling languages directly pertain to the states and processes in the domain of application at hand, thereby allowing for immediate empirical validation of the higher level theory provided in the higher level language. The reduction relation to the lower level language can therefore be seen as empirical correlation laws. Empirical validation is of course also possible at the lower level modelling language, since the concepts of that language can also be mapped to objects and processes in the domain of application.

In summary, to increase the scope of applicability, the strategy in the scientific area of Computer Science and Artificial Intelligence has been to increase the distance between the relevant complex physical processes and the descriptions of them, by the development of high-level modelling languages and supporting software environments (relating the high level languages to the lower-level ones in a hidden manner). These enabled practitioners to work at a high level of description without the need of technical expertise of the lower-level languages: these details are hidden from them by the software environments used.

3 MULTIPLE REALISATION RELATIONS BETWEEN DIFFERENT LEVELS

As discussed in Section 2, modelling languages have been developed to describe the world at different levels. Each of these languages has its own way of referring to instances or traces in the world, which allows empirical validation of a specification. Moreover, such a relationship can be formalised in the semantics of such a modelling language. The different language levels have an independent and autonomous status; they do not depend on each other. Yet, relationships between specifications at the different levels exist. When relating a higher-level specification to a lower-level specification, different choices can be made: every higher-level specification is *multiply realizable* in lower levels. Both in Computer Science and Artificial Intelligence, and in Cognitive Science multiple realisation occurs. An example in the latter case is the cognitive state of being prepared for a certain action, which can be physically realised in different manners for different organisms.

An example from Computer Science: when a high level specification of a sorting process, is translated, this translation could take the form of any of the known sorting algorithms, many of which would be represented by a completely different execution pattern by the hardware, but the final result, when translated back, is the same. If a choice for a particular implementation environment is made, specifications in higher-level modelling languages can be translated into lower-level ones in an automated manner. For example by using a compiler, the lower-level code can be automatically obtained, thus choosing one of the possible realisation relations. To analyse the case study in more detail a number of concepts from philosophy are briefly introduced.

3.1 The Classical Perspective

Nagel's classical definition (1961) of reduction of a theory T_2 (the higher-level theory) to a theory T_1 (the base theory or reducing theory) is as follows:

a) A *bridge principle* is a definitional or empirical principle connecting an expression of T_2 to an expression of T_1. A bridge principle is *biconditional* if it has the form $a \leftrightarrow b$ where a is an expression of T_2 and b an expression of T_1.
b) A theory T_2 is *Nagel-reducible* to T_1 if and only if all laws of T_2 are logically derivable from the laws of T_1 augmented with appropriate bridge principles connecting the expressions of T_2 to expressions of T_1.

The key concept here is the existence of a reduction relation in the form of bridge principles. In practice, these bridge principles have to be biconditional to permit the possibility of deriving T_2-laws from T_1 laws, thereby satisfying b). In the case of mental-physical reduction relations the base property b in a biconditional bridge principle $a \leftrightarrow b$ is also called a *realizer* of a.

How are these concepts related to modelling languages? Suppose two specifications are given, a higher-level specification and a lower-level specification. For the higher-level theory, T_2, the higher-level specification is taken; for the base theory T_1 the lower-level specification. Bridge principles relate expressions that are part of the higher-level specification to expressions in the lower-level specification, and do so in a semantics-preserving manner, i.e., the properties in the world to which the two expressions refer are the same. At first sight it may seem that such principles indeed exist. They are well-defined in a mathematically precise manner and even automated in compilers that translate any specification in terms of the higher-level language into a specification in terms of the lower-level language. However, one higher-level specification can be translated into many lower-level specifications: each specification in one of the language levels is multiply realizable in the lower levels. This situation is quite similar to the situation in Cognitive Science, where it has also been argued that multiple realizability occurs. For example, suppose a mental state property Q in one organism is related to physical state property P_1, and in another to P_2, which is not equal or equivalent to P_1. An argument against the existence of a reduction relation then is as follows. Suppose P_1 is a realizer of Q, then $P_1 \leftrightarrow Q$. If also P_2 would be a realizer, then $P_1 \leftrightarrow Q \leftrightarrow P_2$ would show that P_1 and P_2 are equivalent, a contradiction.

Kim (1996, Ch. 9) outlines two alternative approaches for coping with multiple realizability in Cognitive Science: reduction using bridge principles based on *disjunctions* of the lower-level properties specified in the different realisations, or *local reduction*, based on multiple sets of context-specific bridge principles. The first option posits bridge principles of the form $M \leftrightarrow P^d$ where P^d is a disjunction $P_1 \vee P_2 \vee \ldots$, with $P_1, P_2, \ldots$ the different candidate realizers (indicated by expressions in T_1-ontology) of higher-level property M (in T_2-ontology). The problem is that there may be an indeterminate (possibly infinite) number of such candidate realizers. One cannot, using the vocabulary of the lower level, specify the set of candidate realizers. A similar problem arises with respect to modelling languages, since there is an endless variation of lower-level programs that essentially do the same, viewed from a higher-level perspective. Within the lower level language this cannot be expressed in an adequate manner. So, the approach based on disjunctions of lower-level properties has serious problems. The other alternative, the perspective of local or context-specific reduction is considered a more promising approach.

3.2 Local or Context-Specific Reduction

In context-specific reduction (Kim, 1996, pp. 233-236) the aim is not to find *one set* of bridge principles, but *multiple sets* of context-specific bridge principles. In this case at each instance of time, each higher-level description can be related to a lower-level description based on an *appropriately chosen* context-specific set of bridge principles. The contexts are chosen in such a manner that all situations in which a specific type of realisation occurs are grouped together, and are jointly described by one set of bridge principles. In Cognitive Science such a grouping could be based on species, i.e., groups of organisms with (more or less) the same architecture, although objections may be put forward against this granularity of grouping; it might well be the case that certain mental properties have different realisations over organisms of the same species, or even different realisations within one organism over time.

In the context of an organism or system with structure or architecture description S, biconditional bridge principles can be stated in a conditional manner as follows; cf. Kim (1996), p. 233: $S \rightarrow (M \leftrightarrow P)$. This means that for all systems with structure S the bridge principle $M \leftrightarrow P$ applies. For systems with another structure, other bridge principles apply. This also generates a disjunctive form, but logically different from the one above; namely instead of a set of disjunctive bridge principles (as discussed in Section 3.1) this time in the form of a disjunction of sets (labelled by contexts) of non-disjunctive bridge principles:

(case of S_1)	(case of S_2)	(case of S_3)	…..
[$M \leftrightarrow P_1$	[$M \leftrightarrow P_2$	[$M \leftrightarrow P_3$	[…
$N \leftrightarrow Q_1$	$N \leftrightarrow Q_2$	$N \leftrightarrow Q_3$	
….]	….]	….]	]

For the case of modelling languages, an implementation environment (including a compiler) of the modeller's choice can play the role of this architecture S, and its context-specific set of bridge principles. A different choice of implementation environment will come up with a different set of bridge principles and, hence, with a different lower-level specification for the same higher-level specification. To be more precise, in the case of modelling languages the role of S above is played by the specification of the compiler that is used to relate high level specifications to low level specifications; such a compiler is the core of an implementation environment. Indeed, committing oneself to one specific compiler can be described as the situation where only one set of bridge principles is used. The situation for modelling languages is, in a structural sense, similar to the situation that one psychological theory of pain T_2 can be related to different realisations in the form of physiological theories T_1 in different species (or even within different animals within one species), as put forward by Kim (1996), Ch. 9, pp. 233-236. A choice for a particular implementation environment for a specification in a higher-level modelling language can be compared to the choice of a particular species for a higher-level cognitive theory.

4 HIGHER-LEVEL EXPLANATIONS

This section discusses why explanations at different levels are considered useful in practice, from the angle of the pragmatics of explanation. First the perspective developed by Jackson and Pettit (1988, 1990) is discussed: their 'program explanation' as opposed to causal explanation. They argue that a program explanation is more useful because it has a wider scope of applicability (in possible worlds) than a causal explanation at the physical level (in the actual world). Subsequently, Dennett (1987)'s view on the use of intentional stance versus physical stance explanations is addressed. Dennett emphasizes that intentional stance explanations are tractable in cases where physical stance explanations are not. Next, Bickle (1998)'s perspective on folkpsychological explanations versus neuro-biological explanations is discussed. He emphasizes the value it can have to keep and improve the higher-level theory as a basis for explanation, in addition to a neurobiological theory. In Section 5, the use of explanations in modelling languages is discussed and compared.

Jackson and Pettit (1988, 1990) develop a notion of higher-level explanation, meant to be suitable for special sciences such as Biology, Cognitive Science and Social Sciences: *program explanation*. According to this type of explanation, 'G occurred because F occurred' for higher-level properties F and G, can be an adequate explanation in the following way: F ensures ('programs for') some lower-level property P, which causes G. Or: F ensures ('programs for') some lower-level property P, which causes a lower-level property Q for which G is a higher-level description. For example, 'Why was the vase breaking?' can be explained by: 'Because it was fragile'. Here the higher-level property of being fragile ensures the lower-level property of having a specific molecular structure, and similarly for the broken state of the vase. They discuss the value of such a higher-level explanation, using the example of adiation from the decay of atoms, as follows:

'According to (Lewis, 1988), to explain something is to provide information on its causal history . . . A program explanation provides a different sort of information . . . A program account tells us what the history might have been. It gives modal information about the history, telling us for example that in any relevantly similar situation, as in the original situation itself, the fact that some atoms are decaying means that there will be a property realized - that involving the decay of such and such particular atoms - which is sufficient in the circumstances to produce radiation. In the actual world it was this, that and the other atom which decayed and led to radiation, but in possible worlds where their place is taken by other atoms, the radiation still occurs. ' (Jackson and Pettit, 1990), p. 117.

Jackson and Pettit emphasize that such a form of higher-level explanation, based on some theory T_2 has advantages over causal explanation, based on a basic theory T_1, in the sense that other information is provided, which implies *increased genericity*: it not only applies to the actual world, but also to other possible worlds.

As opposed to explanations from a direct physical perspective (the *physical stance*), Dennett, (1987, 1991) advocates use of the *intentional stance*; cf. Dennett (1987), pp. 37-39; Dennett (1991), pp. 37-42. He uses different description levels in a computer system (actually of a chess computer), as a metaphor to explain the advantage of intentional stance explanations for mental phenomena over physical stance explanations.

'Predicting that someone will duck if you throw a brick at him is easy from the folk-psychological stance; it is and will always be intractable if you have to trace the protons from brick to eyeball, the neurotransmitters from optic nerve to motor nerve, and so forth. ' (Dennett, 1991), p. 42

Dennett puts the emphasis on tractability. To explain more complex phenomena in our real world, higher-level explanations are tractable, whereas lower-level explanations are not.

Bickle (1998, pp. 199-211) discusses an approach to reduction, called *revisionary reduction*. Bickle (1998, pp. 205-208), illustrates this account for the higher-level (e.g., folk psychological) and lower-level (e.g., neurobiological) explanation in the context of Hawkin and Kandel' s (1984a,b) case:

'Of course, the functional profiles assigned to cognitive states on Hawkin and Kandel' s neurobiological account are much more fine grained and detailed, for that account recognizes distinctions and connections that folk psychology either lumps together or leaves extremely vague (…) we can expect that injection of some neurobiological details back into folk psychology would fruitfully enrich the latter, and thus allow development of a more fine-grained folk-psychological account that better matches the detailed functional profiles that neurobiology assigns to its representational states. There is no principled reason against such enrichment.' (Bickle, 1998), p. 207-208

Here Bickle proposes that by relating a folk psychological explanation to a neurobiological account, a decision can be made to enrich the former by introducing some new intermediary states, based on the more detailed path provided by the latter. He puts forward the possibility to use a reduction relation in scientific practice not to eliminate the theory T_2 in favour of a theory T_1, but to extend or improve the theory T_2 on the basis of theory T_1. Therefore, abandoning explanation on the basis of T_2 and replacing such explanation by explanation on the basis of T_1 is not at issue; on the contrary, the explanatory value of T_2 is strengthened by the process of co-evolution of T_2 and T_1.[2]

For the case of modelling languages, providing an explanation is typically observed in the context of the search for bugs. A bug is an error in a specification, resulting in a deviation from the behaviour that the modeller aimed for. A bug will thus display behaviour that deviates from the expectations of the modeller. This behaviour, however, is still in accordance with the specification of the behaviour in the language that the modeller used, and all subsequent lower languages that it has been translated into. The modeller will want an explanation of why the computer system shows this unexpected behaviour: he or she needs to know why the bug happened, so as to fix all possible occurrences of the bug. In principle, several types of explanations are possible for a bug.

The lowest level explanation is that the wrong electrical currents went through the circuitry. The electrons inside the hardware went the wrong way. In order to fix this, the modeller would need to change the circuitry. However, this would only stop the bug from happening on that machine. Using the same not corrected higher-level specification will result in similar deviating behaviour in all other cases that the higher-level specification is used. Next, an explanation can be given in machine code, for example that a certain machine register contained the wrong value. Fixing the machine code would fix the particular occurrence of the bug, but only for that particular type of processor. The bug could still manifest on other types of processors.

The deviating behaviour can also be explained in the higher-level language that the modeller used to specify the behaviour. There, a certain expression can be deemed responsible: execution of the specification results in the deviating behaviour, because the expression is not adequate. This is the type of explanation the modeller wants, since (1) it is much easier to understand for the human at this level what is wrong, and (2) by correcting this higher-level specification, the behaviours for all (automatically generated) realizations of the specification are corrected.[3]

[2] For further development of Bickle's ideas, see Bickle (2003).

[3] One minor comment can be made here. A modeller assumes that the implementation environments used are adequate. In other words, the context-specific bridge principles are assumed correct. An observed deviation in behaviour is not attributed to this environment and these bridge principles, but rather to the higher-level specification. In exceptional cases, however, it is possible that a bug is present in the compiler software, especially when features are used that were not extensively tested or verified earlier. A modeller may consider this possibility if a bug is displayed whereas the higher-level specification seems adequate. This situation can also occur in the case of scientic theories, by the way; then also a need can arise to reconsider the reduction relation. For the sake of the argument we exclude this situation from our considerations.

In conclusion, modelling languages at different levels can be considered for explaining the behaviour of processes in the world or in computer systems. In practice, explanations are considered more valuable when expressed in terms of a higher level of description. Due to the fact that translations into lower level languages can be completely automated, the modeller need not have any knowledge of the lower levels.

5 PRAGMATIC STRATEGIES

If reduction relations between theories have been established, still choices can be made on how to exploit these relationships in scientific practice.
(1) One possible choice is to *eliminate* theory T_2 including its ontology and its laws. For example, scientific texts will only include terms of T_1, not of T_2.
(2) Another possible choice is to *relate* theory T_2's ontology to expressions of theory T_1, without actually eliminating theory T_2, but still using its ontology and its laws, knowing that they are related to certain expressions of T_1. In this case, two types of scientific texts are possible for two types of readers: (a) those including terms of both of T_1 and T_2 and their relationships (for specialists), and (b) those only including terms of T_2, (for less specialised readers), with reference to the former texts for specialised readers. The availability of a reduction relation can be considered as additional knowledge for some that provides a better foundation and embedding for T_2, thus increasing the value of T_2. Therefore motivation and justification to use T_2 has increased.

Kim (1996, pp. 214-216), proposes *ontological simplification* as an advantage in the case where theory T_2 is eliminated: choice (1) above. Having two theories T_1 and T_2 with their own ontologies indeed means having to learn and being aware of more terms than if only the terms of T_1 are sufficient. In choice (2), however, ontological simplification may also take place, in the sense that in scientific practice T_1's ontology is used less extensively. As discussed, this actually has happened in modelling languages, where developers can work with a much more restricted high level ontology. A second aspect of reduction Kim (1996, pp. 214-216) puts forward is that the number of *assumptions* about the world is less. By relating T_2-laws to T_1-laws, logical dependencies are identified. This can be considered a substantial advantage, since if some assumptions are logically implied by other assumptions, it is good to know about these relationships. This advantage arises both if T_2 is eliminated (choice (1)), and if T_2 is not eliminated (choice (2)).

A third aspect of reduction of a theory T_2 to a theory T_1 as considered by Kim (1996, pp. 214-216) is that reduction gives more *insight in underlying lower-level mechanisms*. As an example, the discovery of the structure of DNA and of how segments on DNA play a causal role in the inheritance of properties, gave an extended insight in how things work in the world. The biological notion of a gene, which can be defined in a functional manner within biology, now could be related to a realisation in the form of chemical structures and mechanisms. It is a question whether this advantage is only an advantage of eliminating a (biological) theory (choice (1)), or that the advantage is that as yet unknown relationships will be found between two theories that both had and will have their independent value (choice (2)). It is the question whether advantages of reduction are introduced by eliminating one of the theories, or by obtaining additional knowledge about relationships between two theories which both will remain to have their - maybe even increased - value. A fourth aspect of reduction is that reduction provides new *empirical possibilities*. Knowing, for example, that a specific type of perceptual processing relates to activation within a certain brain area makes it possible for brain imaging to test theories about perceptual processes. This advantage counts if T_2 is not eliminated, since the empirical possibilities for T_2 are increased, not for T_1. In a case of elimination (choice (1)), this advantage does not count, as T_1's empirical possibilities remain the same. For choice (2) this may count as a substantial advantage.

Another contribution of reduction to scientific practice is increased human understanding due to the tractability, or *transparency* of theories. On this criterion, choice (1) usually has the disadvantage that more complex descriptions are involved in an explanation. As a higher-level theory often is more transparent and understandable than a lower-level one, elimination of T_2 and its basic ontology typically decreases transparency. Dennett's explanation of why you dive when I throw a brick (discussed in Section 4) makes this vivid. Choice (2) has the advantage that transparency is kept, or even increased if a shift of attention to T_2 is made, thereby leaving behind T_1. Jackson and Pettit (1988, 1990), as discussed in Section 4, emphasize the issue of *genericity*. A higher-level explanation applies to more application contexts than a lower-level explanation. This was also emphasized in the context of modelling languages in Section 4. If choice (1) is made, this entails lower-level explanations with a more limited scope of application. For choice (2), also higher-level explanations are available.

6 CONCLUSION

In this paper scientific strategies related to reduction and its pragmatics were discussed. This conclusion summarises the lessons that can be learned from our case study of the development of modelling languages within Computer Science and Artificial Intelligence. The design of higher-level modelling languages enables modellers to describe the world on a high level and thus avoid addressing physical details directly. This means that more complex applications can be covered. Although the pragmatic strategy of a practitioner is anti-reductionist, well-established context-specific reduction relations, automated and hidden in specific implementation (software) environments, assure that a solid relation with a physical reality is maintained.

This situation has similarities to the perspective of local (species-specific) reduction within Cognitive Science, as put forward, for example, by Kim (1996). The case study is also compared to other positions on explanation that emphasize the advantages of taking some distance to physical reality, such as Jackson and Pettit (1988, 1990)'s program explanation, Dennett (1987)'s intentional stance explanation and folk-psychological explanation in the context of 'new wave reduction' as put forward by Bickle (1998). Some aspects of the discussion on reduction can be clarified by distinguishing between (1) the structural aspect: the

Table 1	*Why eliminate higher-level descriptions: choice (1)*	*Why use higher-level descriptions in addition to lower-level descriptions: choice (2)*
ontology	Ontological simplification: less terms to describe phenomena in the world.	Adding additional ontologies to improve understanding; simplification if restricting to higher-level ontologies. Knowledge of how higher-level terms relate to lower-level terms is valuable in addition.
assumptions	Less assumptions on the world.	Assumptions about the world made more understandable. Relations between assumptions at different levels of description.
insight	More insight in underlying lower-level mechanisms for higher-level phenomena.	More insight from a conceptually higher level perspective. By going back and forth between higher-level and lower-level descriptions, a lower-level theory T_1 can have impact on the development of the higher-level theory T_2
validity	Empirical test possibilities at the lower level.	Possibilities for verification and validation at different levels: both at the lower the higher level of description.
transparency	Modelling complex systems is (more) difficult	Modelling complex systems is less difficult.
genericity	Scope of application limited by the complexity of the specification.	Wider scope of application.

availability of reduction relations (possibly local) between two scientific theories, and (2) the pragmatic aspect: the actual use of these reduction relations by researchers in scientific practice. In some part of the literature, by leaving this distinction implicit, the suggestion is created that the availability of a reduction relation in the structural sense, in the pragmatic sense, is to be used to eliminate the higher-level theory. Indeed, the main arguments put forward aiming at protection of the autonomous status of the higher-level theory criticizes the existence of systematic reduction relations. Such an argument strongly suggests the silent assumption that if such systematic reduction relations structurally exist, in the sense of pragmatics the higher-level theory is to be given up. In contrast, the position put forward here is that, to protect the higher-level theory the availability of a reduction relation is not bad at all; it can go hand in hand with an antireductionist pragmatic strategy reinforcing the higher-level theory by giving more status to and putting more emphasis on the higher-level theory instead of less. The case study of the development of modelling languages within Computer Science and Artificial Intelligence supports this claim.

ACKNOWLEDGEMENTS

Discussions with Frank Bruggeman, Huib Looren de Jong, Hans Westerhoff and Wouter Wijngaards have improved our formulation of the ideas put forward.

REFERENCES

Bickle, J. (1998). *Psychoneural Reduction: The New Wave*. MIT Press, Cambridge, Massachusetts.

Bickle, J. (2003). *Philosophy and Neuroscience: A Ruthless Reductive Account*. Kluwer Academic Publishers.

Booch, G. (1991). *Object oriented design with applications*. Benjamins Cummins Publishing Company, Redwood City.

Booch, G., Rumbaugh, J., and Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Addison-Wesley, Reading MA, USA.

Borman, Bubenko, Johannesson & Wangler: *Conceptual modelling*. Prentice Hall, 1997.

Brachman, R.J., and Levesque, H.J., (2004). *Knowledge Representation and Reasoning*. The Morgan Kaufmann Series in Artificial Intelligence, 381 pp.

Brazier, F.M.T., Jonker, C.M., and Treur, J. (2002). Principles of Component-Based Design of Intelligent Agents. *Data and Knowledge Engineering*, vol. 41, 2002, pp. 1-28.

Brown, A.W. (ed.) (1996). *Component-Based Software Engineering*. IEEE Computer Society Press, 1996.

Davidson, D. (1993), Thinking Causes. In: J. Heil & A. Mele (eds.), *Mental Causation*. Clarendon Press, Oxford.

Dennett, D.C. (1987). *The Intentional Stance*. MIT Press. Cambridge Mass, 1987.

Dennett, D.C. (1991). Real Patterns. *The Journal of Philosophy*, vol. 88, 1991, pp. 27-51.

D' Souza, D.F. and Wills, A.C. (1998). *Objects, Components, and Frameworks with UML: The Catalysis Approach*. Addison-Wesley.

Fowler, M., Scott, K. and Booch, G. (2000). *UML Distilled (2nd edition): A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley.

Gasper, P. Reduction and instrumentalism in genetics, *Philosophy of Science*, vol. 59, 1992, pp. 655-670.

Guarino, N., (ed.): *Formal Ontology in Information Systems*. IOS Press, 1998.

Hawkins, R.D., and Kandel, E.R. (1984a). Is There a Cell-Biological Alphabet for Simple Forms of Learning? *Psychological Review*, vol. 91, pp. 375-391

Hawkins, R.D., and Kandel, E.R. (1984b). Steps Toward a Cell-Biological Alphabet for Elementary Forms of Learning. In: G. Lynch, J.L. McGaugh, and N.M. Weinberger (eds.), *Neurobiology of Learning and Memory*, Guilford Press, New York, pp. 385-404, vol. 91, pp. 375-391

Jackson, F., and Pettit, P. (1988). Functionalism and Broad Content. *Mind*, vol. 97, pp. 381-400.

Jackson, F., and Pettit, P. (1990). Program Explanation: A General Perspective. *Analysis*, vol. 50, pp. 107-117.

Kernighan, B.W. & Ritchie, D.M. (1978, 1988), *The C Programming Language*, Prentice Hall, New Jersey.

Kim, J. (1996). *Philosophy of Mind*, Westview Press.

Kim, J. (1998). *Mind in a Physical world: an Essay on the Mind-Body Problem and Mental Causation*. MIT Press, Cambridge, Mass.

Kim, J. (2005). *Physicalism, or Something Near Enough*. Princeton University Press, Princeton.

Kotonya, G. and Sommerville, I. (1998). *Requirements Engineering*. Wiley.

Knuth, D.E. (1981). *The Art of Computer Programming*, Addison-Wesley.

Lewis, D. (1988). Causal Explanation. In: *Philosophical Papers*, vol. 2. Oxford: Basil Blackwell.

Minsky, Marvin (1975). A framework for representing knowledge. In Winston, Patrick Henry (ed.). *The psychology of computer vision*. New York et al.: McGraw-Hill, pp. 211-277.

Nagel, E. (1961). *The Structure of Science*. Harcourt, Brace & World, New York.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence. A Modern Approach*. New Jersey, NJ: Prentice Hall, 2nd ed.

Sowa, J.F., (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, CA, 594 pp.

Tanenbaum, A.S. (1976). *Structured Computer Organisation*. Prentice-Hall, London.

Catholijn M. Jonker is a full professor in Artificial Intelligence and Cognitive Science at the Nijmegen Institute for Cognition and Information of the Radboud Universiteit Nijmegen in the Netherlands. She studied computer science at Utrecht University. She completed her PhD on the topic of Negations and Constraints in Logic Programming at the same university. After completing a post-doc position on the same topic at the Universität Bern, she became an assistant professor at the Vrije Universiteit Amsterdam and switched her research topic to agent systems. From that time she worked in modelling of cognitive and multi-agent system organisations and the analysis of the dynamics of behaviour of complex systems.

Jan Treur is a full proefssor of Artificial Intelligence at the Vrije Universiteit Amsterdam. He received his Ph.D. in Mathematics and Logic in 1976 from Utrecht University. Since 1986 he works in Artificial Intelligence, from 1990 as a full professor and head of the Department of Artificial Intelligence at the Vrije Universiteit Amsterdam. In the 1990s he headed a research programme on component-based design of knowledge-based and agent systems. In the last five years the research programme focussed on modelling dynamics of agent systems in practical application areas, and related to other disciplines such as Biology, Cognitive Science, Organisation Theory, and Philosophy of Mind.