Bottom-Up Construction of Ontologies

Paul E. van der Vet and Nicolaas J.I. Mars, *Senior Member*, *IEEE*

Abstract—We present a particular way of building ontologies that proceeds in a bottom-up fashion. Concepts are defined in a way that mirrors the way their instances are composed out of smaller objects. The smaller objects themselves may also be modeled as being composed. Bottom-up ontologies are flexible through the use of implicit and, hence, parsimonious part-whole and subconcept-superconcept relations. The bottom-up method complements current practice, where, as a rule, ontologies are built top-down. The design method is illustrated by an example involving ontologies of pure substances at several levels of detail. It is not claimed that bottom-up construction is a generally valid recipe; indeed, such recipes are deemed uninformative or impossible. Rather, the approach is intended to enrich the ontology developer's toolkit.

Index Terms—Knowledge engineering, knowledge base management, ontology, knowledge integration, domain model, hierarchical reasoning.

1 Introduction: Setting and Scope

Ontology development and use constitute an important research area for AI. In our work in ontology development, we have found it beneficial to deviate from the usual top-down approach by using a bottom-up approach instead. In this paper, we present the ideas underlying the bottom-up approach. We will illustrate the ideas by an example taken from our own work, namely an ontology for the chemical domain of pure substances. The ontology supports unambiguous definitions of concepts for pure substances and hierarchical reasoning (both separately and combined) along two orthogonal dimensions, namely subconcept-superconcept and part-whole. We provide some background in the present section.

1.1 Ontologies

The term 'ontology' is ambiguous. In the literature [1], no less than seven different interpretations have been identified. We will have to outline our own view. In our work [2], an ontology serves as a partial specification of the knowledge representation to be built in a later stage. The specification is partial because it supplies concepts in which states of affairs can be expressed but does not actually specify states of affairs. With respect to the knowledge representation, the ontology supplies the meaning of every nonlogical constant that occurs in the representation. To allow this, an ontology is a concept system in which all concepts are defined. Concepts are interpreted in a declarative way, as standing for the sets of their instances. The concept system is limitative: concepts can only be used if they are defined in the ontology. Definitions are formal where possible and informal otherwise. In our group, a predefined ontology has been used as specification for representations in various languages in order to compare different representation systems [3].

In the development of a knowledge-based system, the use of an ontology is beneficial for two reasons [4]. It allows for a more disciplined design of the knowledge base; and it facilitates sharing and reuse. Both advantages are particularly important when the knowledge base becomes large. Where it already proves difficult for developers of small knowledge bases to gain insight into contents and structure, this is humanly impossible for their large counterparts. Also, developing large knowledge bases takes a large investment which can be earned back more readily if the knowledge base is used in more than a single application. These functions of ontologies are akin to (but not identical with) the functions played by the so-called reference models IBM started to develop in the 1960s, to guide software development for specific groups of customers (see, e.g., [5]).

1.2 Ontology Development

Currently, ontology development is a craft rather than a science. Discussions at recent workshops, e.g., those at ECAI '94, IJCAI '95, and ECAI '96, and in the **srkb** mailing-list, have tried to pinpoint the difficulties and find remedies. We believe that proposals for a general method of ontology building are premature or misguided. Experiences in other branches of engineering have shown that general recipes tend to be over-general and thus of little practical use. Instead, we should strive for practical experience and try to identify (even if only in retrospect) the principles upon which our ontologies are based. Working this way, we may end up with a set of sufficiently precise recipes, each with its own intended use.

In our view, it is profitable to gain practical experience with ontologies for nontrivial domains like subfields of science and engineering. Within the domains themselves, structuring of information and often also standardization have proceeded to a certain extent. This gives the ontology builder a start. As a side-benefit, these ontologies have the advantage of being potentially useful for practical applica-

• *The authors are with the Department of Computer Science, University of Twente, 7500 AE Enschede, the Netherlands. N.J.I. Mars is also affiliated with the Netherlands Institute for Scientific Information Services (NIWI), Royal Netherlands Academy of Sciences and Arts, Amsterdam, the Netherlands. E-mail: vet@cs.utwente.nl.*

Manuscript received 6 Feb. 1996; revised 27 Dec. 1996.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 104357.

1041-4347/98/$10.00 © 1998 IEEE

tions. We have found that developing such ontologies takes more than a passing knowledge of the domain.

1.3 Context

The ontology research reported here has been conducted in the course of the Sapiens and Plinius projects, aimed at semiautomatic indexing and semiautomatic knowledge extraction, respectively. In both projects, the source material is taken from document descriptions provided by producers of bibliographic databases. The domain for Sapiens is medicine, that for Plinius ceramic materials. Detailed information about the projects [6], [7], [8], [9] and about particular ontologies [4], [10], [11], [12], can be obtained elsewhere.

In a system like Plinius, it is impossible to exhaustively predict in advance which concepts will be needed to express the knowledge found in the texts. The list of chemical substances, for example, is in principle open-ended. The Plinius ontology is designed to serve as an anchor point for the development of the processes, so we did not like the idea of having to add nodes to a top-down taxonomic ontology each time the system encounters a new substance. In knowledge representation systems designed with the top-down approach in mind, like implementations of description logics, the addition of a node has to be done as a separate step because the system only deduces where the new node has to be placed in the already existing taxonomy. Still, this would be a minor problem if a single node for each new pure substance would have sufficed.

But chemical reasoning employs superconcepts at various levels of detail, like 'phosphate,' 'calcium compound,' and 'calcium compound that also contains phosphate.' Listing all these concepts in advance is impossible. If no phosphates and calcium compounds are encountered before, the occurrence of calcium phosphate requires the addition of four nodes to the taxonomy: one for the substance itself and three for superconcepts. Worse, the situation can repeat for the next new substance. Keeping track may require a considerable effort and is error-prone. There are also negative computational consequences. In implementations of description logics, the addition of a node is an expensive operation.

The (inorganic) substances mentioned in the Plinius source texts can be uniquely characterized by their chemical composition specified in terms of constituents and proportions. We have used this to design a bottom-up ontology for Plinius. Concepts for each and every substance that may possibly exist are made available implicitly by laying down which constituents there are and in which ways they can combine to form substances. Superconcepts are made available by allowing constituents to be unspecified. For instance, a calcium compound is a compound that consists of calcium and other, unspecified constituents. The result supports reasoning along two, orthogonal hierarchies: the partonomy formed by substances and their constituents and the taxonomy formed by concepts and superconcepts. Reasoning that combines the two hierarchies is also possible. We are not aware of other designs that combine these advantages in a similar, parsimonious and inexpensive manner. The method is not specific for chemical applications, although (like any other method) it is not applicable to each and every domain.

Lack of space prevents discussion of the full ontologies developed for the projects mentioned. Instead, the ideas are illustrated by a simplified version of the Plinius ontology involving pure substances. Again, a fair treatment of the necessary background knowledge is not possible and a summary will have to do. A full treatment of the example of pure substances, including an elaborate discussion of background knowledge and a formalization in Prolog, is available as a technical report [13].

2 Atomism and the Bottom-Up Approach

2.1 Atomism

Most ontologies published in the literature (for instance those discussed at the workshops cited above) are constructed by means of top-down, iterative differentiation. Description logics [14] are designed to support such ontologies. A declarative semantics is used. Concepts are interpreted as sets of their instances, and the important subconcept relation is interpreted as a subset relation between sets of instances [15]. A concept can be defined by telling in which respects its instances differ from the instances of its superconcept and from the instances of other subconcepts of the same superconcept. This approach underlies Linnaeus's biological taxonomies and can be traced back to Aristotle's ideas on classification involving *genus*, *species*, and *differentiae* [16].

The bottom-up approach is inspired by atomism, a tradition that dates back to antiquity. According to atomism, the objects we perceive are composed of indivisible units called atoms. The specification of an object in terms of indivisible units and their interactions constitutes the fullest possible description of the object (descriptive aspect), and allows derivation of all other properties of the object (explanatory aspect). The explanatory aspect is obviously more ambitious than the descriptive aspect. Both aspects are part and parcel of the received view in modern science, even though the explanatory aspect is only an 'in principle' possibility in most cases. We will be concerned here with the descriptive aspect.

Chemistry and particle physics are the paradigm cases. A complicated picture involving several levels of composition has been developed [17], [18]. The systems we study in the laboratory are composed of elementary particles; elementary particles interact to form atoms; atoms interact to form molecules; atoms and molecules interact to form the substances investigated by chemists. Extensions in both directions are possible. Already in antiquity, the paradigm has also been extended to engineering. Engineered artifacts can be uniquely described by specifying their parts and the way the parts are put together.

The descriptions sought both in science and in engineering aim at generalization and therefore at a level above that of individual objects. Descriptive atomism classifies complex objects by their composition. Complex objects are instances of the same concept if they are composed of the

same kinds of constituents in the same numbers and connected in the same way. The concept of a helium atom is interpreted as the set of complex objects that consist of two protons, an unspecified number of neutrons, and two electrons, interacting to form an atom. The parts are themselves specified as concepts, because it does not matter which individual particles are present in a particular helium atom as long as they are instances of the specified concepts. We will consistently distinguish between *parts* (objects) and *constituents* (concepts for parts).

The concepts of descriptive atomism are given a declarative semantics here, that is, they are interpreted as sets of their instances. We will use the familiar symbol ⟦c⟧ to denote the set of instances of concept *c*. Declarative semantics is instructive for our account, but it is not entirely correct. Below, we will discuss the concept of a helium atom with fifteen neutrons in its nucleus. The concept is allowed in a bottom-up ontology, although no instances are known to exist in the real world. Part of the power of atomism is that we can nevertheless establish the truth of assertions like the one stating that instances of the concepts are not known to exist (which takes knowledge that enables experimental identification of instances). Where description logics seem to concentrate on (Fregean) reference [19], our atomist ontologies concentrate on sense.

The ability to construct concepts whose instances are not known to exist is an advantage in many cases. We may want to express the fact that no instances of a particular concept are known to exist; it takes the concept to say so. Also, knowledge of existence is highly time-dependent in several domains, that of pure substances among them. It would be unwise to burden an ontology with such knowledge because it would necessitate frequent updates.

2.2 The Principle of Bottom-Up Ontologies

Drawing on the results obtained in the atomist tradition, we can design ontologies for scientific and engineering domains in a bottom-up fashion. A bottom-up ontology is specified as a list of primitive concepts and a list of construction rules that implicitly define so-called complex concepts.

Formally, a bottom-up ontology is specified as a tuple $\langle A, C\rangle$, with A the set of primitive concepts and C the set of construction rules. When $\langle A, C\rangle$ is viewed as a calculus, its deductive closure gives the ontology proper, i.e., the complete list of primitive and complex concepts. It is intended, however, that the complete list is never actually constructed. Indeed, for the ontologies to be introduced below this would be impossible because the number of concepts is (denumerably) infinite. Instead of constructing the full list, complex concepts are made 'on the fly' as needed.

2.3 Primitive Concepts

Primitive concepts are interpreted as sets of indivisible objects. (We avoid the obvious but potentially confusing term 'atomic concepts.') In the present context, 'indivisibility' is a relative rather than an absolute notion. The choice of indivisible objects is a design decision. Primitive concepts are introduced extensionally or intensionally, but either way they are not fully defined in a formal sense. Instead, an informal explication anchors them into consensus domain knowledge. For instance, the concept of electron can be explained by citing physics textbooks.

In addition, we will have to be able to count. The obvious way is to introduce primitive concepts for natural numbers (i.e., integers > 0).

In any ontology, concepts have to be given unique names. For reasons that will become clear in a moment, we introduce a function `label`(*c*) that returns a label for concept *c*. For any primitive concept introduced by extensional specification, the label is a unique name chosen in the course of design. For primitive concepts introduced by means of intensional specification, we implement `label`(*c*) as a scheme that reflects the intensional specification. For an example of the latter, see the definition of labels for concepts of type *chemical element* in Section 6.2.

It will prove advantageous to be able to refer to classes of primitive concepts. We postulate a function `type` that assigns a type to every primitive concept, so that `type`(*c*) = *t* assigns type *t* to concept *c*. We also introduce the set $S(t)$ of concepts of the type *t*:

$$S(t) =_{\mathrm{Df}} \{c \mid \texttt{type}(c) = t\} \quad (1)$$

The choice of types of primitive concepts is a design decision. An elaborate type system can be designed involving subtypes at several levels. The result is an ontology that combines a bottom-up approach for the concepts with a top-down approach for the types. We prefer simpler type systems.

2.4 Complex Concepts

Complex concepts are interpreted as sets of complex objects. Complex concepts are not listed explicitly, but rather defined implicitly by means of construction rules. A construction rule defines a whole class of complex concepts by specifying primitive concepts and, where appropriate, the interactions that bind together parts to form a complex object. Thus, a construction rule for concepts for helium atoms specifies electrons, protons, and neutrons in particular numbers, and the interactions between these elementary particles that bind them together to form atoms.

Construction rules can be applied iteratively, so that complex concepts can be defined in terms of other complex concepts.

The `type` function is also defined for complex concepts. It is chosen such that there is a one-to-one correspondence between construction rules and types of complex concepts. All complex concepts constructed according to a particular construction rule are of the same type.

Since complex concepts are not listed explicitly, we need a scheme to assign unique names to them. The `label` function is used for this purpose. For any complex concept *c*, `label`(*c*) is chosen such that the label reflects the application of the construction rule used to generate the concept. To accomplish this, we write labels using the language of sets and tuples. For an example, take the concept of helium-4 atom that we will write as "He-4" for short. He-4

is one of the concepts of type *helium atom*. Leaving out the interaction for clarity, we would write:

$$\texttt{label}(\text{He-4}) = \{\langle \textit{proton}, 2\rangle, \langle \textit{neutron}, 2\rangle, \langle \textit{electron}, 2\rangle\} \quad (2)$$

Although formally a set, the expression at the right-hand side is a label for a single concept. Sets used as labels for concepts must be distinguished from sets of instances of those concepts. We will refer to an expression like the one at the right-hand side of (2) as a label or a label expression.

In any particular ontology, concepts are always identified by their labels. Still, concepts and labels identifying concepts are different things and it pays for now to make the distinction explicitly.

It is quite another matter to call concepts by names or symbols that are recognized by humans, like "helium-4 atom." We can construct a lexicon that maps label expressions onto such names and vice versa. A lexicon may be useful for interfacing, as in the Plinius process where the system is required to translate natural-language terms and chemical formulae into ontology concepts. The lexicon, however, is *not* part of the ontology. The mapping of natural-language terms and symbols onto concepts is many-to-many.

Label expression (2) can be viewed as a list of constituents. Every instance of the concept identified by the label consists of instances of the concepts of proton, neutron, and electron, and two of each. Put simply, in order to assemble a helium-4 atom, one needs two protons, two neutrons, and two electrons. In this way, a concept's label coincides with its definition (at least within the context of the ontology). This opens the possibility to design inferences that draw information from label expressions.

3 Design Considerations

The designer of a bottom-up ontology can vary along two dimensions: choice of indivisible objects, and choice of level of detail in the specification of interactions. Along both dimensions we can design a series of ever more detailed ontologies. We will give an example of an extension along both dimensions in the context of the pure substances case (Section 6.6 and Section 6.7).

3.1 Choice of Indivisible Objects

It is often unnecessary and undesirable to go down to the lowest level of composition known. Most chemistry is expressed in terms of chemical elements, although science recognizes several levels of composition below that of elements. In many biological applications, the level of cell organelles is a sensible choice of bottom-line. On the other hand, it is desirable to design an ontology such that it can be extended to embrace a deeper level without having to redesign the whole ontology.

For bottom-up ontologies, there is a historical parallel: the development of atomism. One of the characteristics of atomism in the past century is the introduction of ever deeper levels of composition [17], [18]. Post [20] argues that these transformations are driven by the *explanatory* ideal embodied in atomism. As the properties we have to ascribe to our ultimate constituents multiply, it becomes more attractive to regard those constituents as being composed of even smaller constituents. At the same time, the *descriptive* aspect of atomism holds up unscathed.

In the context of the He-4 example, it is easy to see how the deeper level of quarks can be introduced. We remove the concepts of proton and neutron from the list of primitive concepts. We add the concepts of 'up' quark and 'down' quark and construction rules for the concepts of proton and neutron. The occurrences of 'proton' and 'neutron' in the label expression of (2) are replaced by labels for the now complex concepts of proton and neutron. The specification of interactions will have to be changed accordingly.

3.2 Choice of Level of Detail in the Specification of Interactions

Until now, we have been silent about the specification of interactions. For explanatory purposes, a detailed account of interactions between objects is indispensable. For descriptive purposes, on the other hand, we are only interested in lack of ambiguity. Under those conditions, interactions may often be specified in a minimal way or left out entirely. One reason for doing so is that interactions are often difficult to model and may result in a large ontology. Consider modeling the interactions between elementary particles forming an atom in a quantum-mechanical way. The extra effort has to be justified by gains, something which can only be decided in the context of a particular application.

Leaving out interactions entirely effectively identifies complex concepts with lists of constituents, a kind of *partes pro toto* strategy. That instances of constituents interact to form complex objects is only expressed by collecting the constituents in the same list. We stipulate that concepts for noninteracting parts cannot occur in the same list. Thus, even if interactions are left out we can still distinguish between, say, a car and a building kit to make that car. The utility of complex concepts defined as constituent lists depends on whether reasoning about interactions between parts is needed for definitional purposes. If there are several ways to put the parts together, each resulting in a different artifact, and we want to distinguish each of those artifacts from the others, constituent lists obviously are insufficient.

If detail about interactions has to be added, we face a dilemma. There is a trade-off between parsimony and addition of detail. In principle, there is always a lot of detail that can be added. To take an extreme case, for distinguishing a badly constructed combustion engine from a well-constructed one we may have to specify the torque (as read on the dial of a torque wrench) with which certain bolts have been turned. In many applications, however, such details are not needed for making distinctions. Letting the inclusion of details depend on the application at hand may result in ontologies that cannot be shared or reused, an unattractive prospect. The alternative, addition of all details we can get to anticipate situations where details are indispensable for making necessary distinctions, is

equally unattractive because it would force the use of large and unwieldy ontologies where slimmer versions would have sufficed.

A way out of this dilemma is to mirror the atomist strategy of levels of composition. The levels-of-composition approach can be modeled in a domain-independent way; after all, parts are parts. An equally domain-independent levels-of-detail approach is not possible because patterns of interaction are highly domain-dependent. We conjecture, however, that there is a general way to add superficial interaction information. The result is known in chemistry as a configuration. It models complex objects as graphs $\langle N, E \rangle$ with N the set of nodes and E the set of edges. Constituents are labeled nodes; edges stand for interactions, such as atomic bonds in molecules or physical connections in cars. An attractive feature of this choice is that constituent lists can be generated automatically from graphs. An electronic circuit is typically specified as a configuration in this sense. In model-based diagnosis, such graphs are widely used [21]. The distinction between explanatory and descriptive aspects returns here. Fully quantitative prediction of behavior takes detailed interaction information of a kind not needed to unambiguously describe the artifact.

3.3 Relating Ontologies by Strictly Information-Discarding Transformations

Along both dimensions we may have a number of ontologies that differ with respect to the level of detail such that

1) every level in itself constitutes a legitimate and correct ontology, and
2) every ontology at some level other than that of most detail can be obtained from an ontology at a more detailed level by means of a *strictly information-discarding transformation*.

As the name implies, a strictly information-discarding transformation results in loss of information while no new information has to be added. The relation is irreflexive, asymmetric, and transitive. Strictly information-discarding transformations correspond to the class of computer programs known as filters.

The idea of relating ontologies by means of strictly information-discarding transformations in general is not new, but we believe that our particular application is. Ontolingua [22] defines `include-theory` to have the definitions laid down by the included ontology (Ontolingua says 'theory') taken for granted by the including ontology. Deleting one or more `include-theory`'s is equivalent to a strictly information-discarding transformation. In the Ontolingua approach, the underlying idea is to promote modular ontology development. The ontology designer can reuse an already developed module such as that for engineering quantities [23] instead of having to write it from scratch. The strictly information-discarding transformations we have been discussing, by contrast, serve the purpose of obtaining entire but less detailed ontologies from more detailed ontologies for the same domain.

A layered system of ontologies connected by strictly information-discarding transformations has also been proposed by us [4] for an ontology of measurement units. The transformations in this case affect information on the conversion of arbitrary measurement units into SI units. Layer 0, a default layer, specifies a straightforward numerical conversion that is sufficiently precise for most practical purposes. At layer 1, conversion factors depend on time and location. For instance, since 1958 the inch is by definition equal to 2.54 cm. In the U.K. between 1923 and 1958, the inch was 2.5399956 cm; the U.S. inch was slightly different. At layer 2, relativistic effects are also taken into account.

A *graph of ontologies* is obtained when the ontologies are connected by directed edges standing for strictly information-discarding transformations. Since for bottom-up ontologies such transformations can be performed along two degrees of freedom, graphs of ontologies can become quite complex. They are instrumental in picking the ontology with the appropriate level of detail from among the whole set and in selecting the transformations needed to convert an existing but too detailed ontology into a version fit for the job at hand.

4 The Language $\mathcal{L}$

4.1 Lists of Constituents

We have favorable experience with ontologies in which definitions of complex concepts omit a specification of interactions. It is satisfactory for large parts of inorganic chemistry (including the Plinius domain). It is also satisfactory in the context of intelligent component catalogs [24], [25]. A component manufacturer may want to include part-whole information in the catalog. Constituent lists are sufficient if every assembly is uniquely characterized by its set of components in the context of the catalog.

The label expressions employed for constituent list concepts can be characterized as sets of tuples, see the expression on the right-hand side of (2) for `label`(He-4). Each tuple has two members: one is the concept that identifies the constituent, the other the concept that identifies the amount. Since the order in which the tuples are listed is conceptually unimportant, they are collected in a set:

$$\{\langle c_i, n_i \rangle, \langle c_j, n_j \rangle, \ldots, \langle c_p, n_p \rangle\} \quad (3)$$

where the c_x are constituents and the n_x specify amounts. Expressions like these can be nested because the c_x can be constituent lists themselves. How many nesting levels there are depends on the particular ontology.

When working with constituent lists, primitive concepts fall into one of two categories: one collects the concepts for indivisible objects, the other the concepts for natural numbers. Calling the former kind of concepts *atoms* for now, the BNF specification of label expressions for constituent list concepts is given in Table 1. It emphasizes appearance rather than formal properties, because the braces that indicate sets and angle brackets that indicate tuples are specified as terminals.

TABLE 1
BNF SPECIFICATION OF THE LABEL LANGUAGE $\mathcal{L}$

```
concept   ::=   "{"tuples"}" | atom
 tuples   ::=   tuple | tuple tuples
  tuple   ::=   "⟨"concept "," natural_number"⟩"
```

To obtain unique label expressions, we impose the so-called canonicity constraint. It demands that no constituent occurs in more than one tuple in the same set. The constraint cannot be expressed in a context-free specification. It can be formalized by returning to the view of labels as sets of tuples. For any label l_1 written as a set of tuples (list of constituents), we demand that for all l_2 and n_1 (where the l_i stand for constituents and the n_j for natural numbers):

$$\langle l_2, n_1\rangle \in l_1 \Rightarrow \neg\exists n_2[\langle l_2, n_2\rangle \in l_1 \wedge n_1 \neq n_2] \quad (4)$$

The label language defined this way will be called $\mathcal{L}$. It is not an ontology because the 'atoms' are not specified. A BNF specification of an ontology is obtained by adding one or more rules to specify the atoms.

4.2 The Constituent Relation

The language $\mathcal{L}$ supports a particular part-of relation that we will call the constituent relation. The occurrence of $\mathbf{label}(c_1)$ in $\mathbf{label}(c_2)$ entails, that a specified number of instances of c_1 are a physical part of each instance of c_2. We will generalize this to the condition that, for each instance of c_2, at least one instance of c_1 is a physical part of it. Compare, again, the expression at the right-hand side of (2) for $\mathbf{label}$(He-4). The occurrence of the tuple $\langle$proton, 2$\rangle$ in $\mathbf{label}$(He-4) entails that at least one (instance of the concept of) proton is a physical part of each (instance of the concept of) helium-4 atom. The name 'constituent relation' is chosen to distinguish this part-of relation from other part-of relations identified in the literature [26].

In order to interpret the constituent relation between concepts, we have to assume in the universe of discourse a relation P between the instances of concepts. $\langle o_1, o_2\rangle \in P$ is true whenever object o_1 is a physical part of object o_2. P is irreflexive, asymmetric, and transitive. The constituent relation corresponds to a set $\mathbf{constituent}$ of concept pairs $\langle c_1, c_2\rangle$ defined as:

$$\mathbf{constituent} =_{\mathrm{Df}}$$
$$\{\langle c_1, c_2\rangle \mid \forall o_2 \in [\![c_2]\!]\, \exists o_1 \in [\![c_1]\!]\, [\langle o_1, o_2\rangle \in P]\} \quad (5)$$

The assertion $\mathbf{constituent}(c_1, c_2)$ is true if and only if $\langle c_1, c_2\rangle \in \mathbf{constituent}$. This can be verified by inspecting labels $\in \mathcal{L}$:

$$\mathbf{constituent}(c_1, c_2) \Leftrightarrow$$
$$\exists n[\langle \mathbf{label}(c_1), n\rangle \in \mathbf{label}(c_2)] \vee$$
$$\exists c_3 \exists n[\langle \mathbf{label}(c_1), n\rangle \in \mathbf{label}(c_3) \wedge$$
$$\mathbf{constituent}(c_3, c_2)] \quad (6)$$

where labels for natural numbers are written as n. The first condition on the right-hand side applies to direct occurrence of $\mathbf{label}(c_1)$ in the constituent list and the second clause to the nested case.

Another way to model part-whole relations makes use of mereology. Roughly, in mereology there is no distinction between the membership and subset relations (for a recent account and application, see [27]). Mereology seems suited to model continuum theories, where, for instance, counting parts is impossible. We, however, are working in the atomist tradition where parts are discrete and can always be counted.

5 SUPERCONCEPTS

In top-down ontologies, the kind-of relation is the main structuring device. (We adhere to the custom of using 'kind-of' for subconcept-superconcept relations, to be distinguished from 'is-a' for instantiation relations.) In our treatment so far, there is no place for the kind-of relation because there are no superconcepts. We first introduce the kind-of relation and then show how superconcepts can be defined implicitly. The formal account will concentrate on concepts defined as constituent lists.

Our kind-of relation is identical to the kind-of relation familiar from description logics, and we will use the same symbol, $\sqsubseteq$. Recalling that we write the set of instances of concept c as $[\![c]\!]$, $c_1 \sqsubseteq c_2$ is true if and only if $[\![c_1]\!] \sqsubseteq [\![c_2]\!]$. The $\sqsubseteq$ relation is reflexive, antisymmetric, and transitive.

5.1 The Language $\mathcal{L}_{\mathrm{ext}}$

Superconcepts are introduced implicitly by means of the function $\mathbf{superconcept}$. The function takes a concept type as its single argument and returns a concept that is a superconcept of every concept of the type, or, for all c:

$$c \sqsubseteq \mathbf{superconcept}(t) \Leftrightarrow \mathbf{type}(c) = t \quad (7)$$

We have to extend the language $\mathcal{L}$ to allow for the occurrence of superconcepts, obtaining the language $\mathcal{L}_{\mathrm{ext}}$. Its BNF specification is given in Table 2. Actual ontologies specify the atoms (as before) and a class of types from which t has to be chosen. The constraint of canonicity, (4), applies. Note that $\mathcal{L} \subset \mathcal{L}_{\mathrm{ext}}$.

The universe of discourse does not change as a result of our decision to introduce superconcepts. Thus, for every actual ontology, the set of instances of all concepts identified by $\mathcal{L}$ is identical to the set of instances of all concepts identified by $\mathcal{L}_{\mathrm{ext}}$. As a consequence, the constituent relation introduced in Section 4.2 and the verification condition given by (6) are the same for $\mathcal{L}$ and $\mathcal{L}_{\mathrm{ext}}$.

TABLE 2
BFN SPECIFICATION OF THE LABEL LANGUAGE $\mathcal{L}_{ext}$

concept	**::=**	**"{"tuples"}" | atom|superconcept(*t*)**
tuples	**::=**	**tuple | tuple tuples**
tuple	**::=**	**"⟨" concept "," natural_number "⟩"|**
		"⟨" concept "," superconcept(*N*) "⟩"

Note: t is a type other than N.

As follows from Table 2, there are two kinds of superconcepts: 'simple' superconcepts identified by a label that just consists of a single occurrence of **superconcept**(*t*) (with a specified *t*), and more complex superconcepts that arise by embedding superconcepts in label expressions. This opens the possibility to construct a great variety of superconcepts. An example of a superconcept obtained by embedding a superconcept in a label expression is:

$$\{\langle \text{proton}, 2\rangle, \langle \text{neutron}, \textbf{superconcept}(N)\rangle, \langle \text{electron}, 2\rangle\} \quad (8)$$

This is a label for the concept of helium atom in general. Note that it covers more than the kinds of helium atom we know. It also covers an atom whose nucleus consists of two protons and fifteen neutrons. Such atoms are not known to exist, but the presence of exactly two protons in their nucleus makes them helium atoms by definition.

5.2 Semantics and Verification

Intuitively, the set of instances of a superconcept is the union of the sets of instances of all its subconcepts. We can immediately lay down, for any concept $s \in \mathcal{L}_{ext}$:

$$[\![s]\!] =_{Df} \bigcup \{[\![c]\!] \mid c \in \mathcal{L} \mid c \sqsubseteq s\}. \quad (9)$$

The definition relies on a set of necessary and sufficient conditions for $c \sqsubseteq s$ to be true. We will specify these conditions in the form of a procedure that verifies whether $c \sqsubseteq s$ by inspecting **label**(*s*) and **label**(*c*). Since there is a one-to-one mapping between concepts and their labels, there is no harm in writing **label**(*c*) $\sqsubseteq$ **label**(*s*). Three cases, numbered (a) through (c) below, can be distinguished. In all other cases, $c \sqsubseteq s$ is false.

Case a. Equality: **label**(*c*) = **label**(*s*) is a sufficient condition for $c \sqsubseteq s$ to be true.

Case b. If **label**(*s*) = **superconcept**(*t*) and **type**(*c*) = *t*, then $c \sqsubseteq s$ is true. As a result, for simple superconcepts Definition 9 reduces to:

$$[\![\textbf{superconcept}(t)]\!] =_{Df} \bigcup \{[\![c]\!] \mid c \in \mathcal{L} \mid \textbf{type}(c) = t\} \quad (10)$$

Note that natural numbers are treated as concepts, so that case a and case b may also obtain for natural numbers. In particular, the following two relations always hold for any natural number *n*: $n \sqsubseteq n$ and $n \sqsubseteq$ **superconcept**(*N*).

Case c. *c* and *s* are both identified by label expressions in set form. To find out whether $c \sqsubseteq s$, we have to recursively 'peel off' **label**(*c*) and **label**(*s*) level for level until we arrive at a level where we can perform an immediate verification as in case a or case b. The specification of $\mathcal{L}_{ext}$ guarantees, that we will eventually arrive at one of these two base cases. We distinguish two subcases.

case c1. **label**(*c*) and **label**(*s*) both consist of a single tuple, that we will write as $\langle v_c, n_c\rangle$ and $\langle v_s, n_s\rangle$, respectively. Then:

$$\{\langle v_c, n_c\rangle\} \sqsubseteq \{\langle v_s, n_s\rangle\} \Leftrightarrow$$
$$(v_c \sqsubseteq v_s \wedge n_c \sqsubseteq n_s) \quad (11)$$

Whether this is a base case depends on whether v_c and v_s are primitive or complex concepts. If the latter, recursion takes place.

case c2. **label**(*c*) and **label**(s) consist of the same number of tuples, where that number is greater than one. We now try to find a 'matching pair,' i.e., a tuple ∈ label(*c*) and a tuple ∈ label(s) such that the right-hand condition of (11) holds. If this succeeds, the tuples are removed from the sets and the procedure is repeated. Upon continued success, we eventually obtain sets that each consist of a single tuple, which is case c1. Failure entails that there is no kind-of relation. Formally,

$$c \sqsubseteq s \Leftrightarrow$$
$$\exists v_c \exists n_c \exists v_s \exists n_s [\langle v_c, n_c\rangle \in c \wedge \langle v_s, n_s\rangle \in s \wedge v_c \sqsubseteq v_s \wedge n_c \sqsubseteq n_s \wedge c - \{\langle v_c, n_c\rangle\} \sqsubseteq s - \{\langle v_s, n_s\rangle\}] \quad (12)$$

5.3 Combining the Two Hierarchies

Finally, in the atomist tradition we often want to introduce a superconcept in such a way that all its subconcepts denote complex objects sharing a common constituent. The chemical concept of calcium compound is a good example. It takes a combination of the constituent and the kind-of hierarchies to establish such relations. It is an advantage of the bottom-up approach that both hierarchies are available.

We define a **superconcept** function with two arguments, the first standing for the type of the superconcept and the second for the shared constituent:

$$[\![\mathbf{superconcept}(t, c_1)]\!] =_{\mathrm{Df}} \bigcup \{[\![c_2]\!] \mid \mathbf{type}(c_2) = t \mid \mathbf{constituent}\}(c_1, c_2)\} \quad (13)$$

Verification whether such a superconcept relation holds between two concepts proceeds, as before, by inspecting the labels:

$$c_2 \sqsubseteq \mathbf{superconcept}\ (t, c_1) \Leftrightarrow [\mathbf{type}(c_2) = t \wedge \mathbf{constituent}\}(c_1, c_2)] \quad (14)$$

Since the two hierarchies are orthogonal, additional kind-of relations may hold between such superconcepts. In particular, the following are true by definition for any t, c, c_1, and c_2:

$$\mathbf{superconcept}(t, c) \sqsubseteq \mathbf{superconcept}(t) \quad (15)$$

$$\mathbf{superconcept}(t, c_1) \sqsubseteq \mathbf{superconcept}(t, c_2) \Leftrightarrow \mathbf{constituent}(c_1, c_2) \quad (16)$$

6 Example: Pure Substances

6.1 Introduction

The concept is pure substance is one of the most important in chemistry. It is introduced by imposing a dichotomy on the set of (what we will call) *samples*, concrete pieces of stuff used for experimentation. Put summarily, a sample is said to consist of a mixture if it consists of two or more different substances, and of a pure substance otherwise. (See any inorganic chemistry textbook like [28] for details.) The properties of isolated atoms and molecules differ significantly from those of samples composed of very many atoms and molecules. We concentrate on the latter. They are the instances of our concepts.

Ontology development can profit from standardization efforts in the application domain. The International Union of Pure and Applied Chemistry (IUPAC) maintains an elaborate set of recommendations for chemistry. The recommendations function as *de facto* standards, and many of them have been turned into *de jure* ISO standards. The recommendations for inorganic chemistry are listed in what chemists fondly call the "Red Book." The latest edition has been published in 1990 [29]. The Red Book has been instrumental in designing the ontology discussed here.

A pure substance consists of chemical elements in natural-number proportions. Examples are table salt (NaCl) and ethanol (CH_3CH_2OH, called alcohol in daily life). In these formulae, the subscripts denote amounts. A subscript is omitted if it is '1.' A substance's *empirical formula* is a bare listing of the constituent elements and their proportions. The empirical formula of ethanol is C_2H_6O.

Two complications block the simple expedient of modeling complex concepts of type *pure substance* after empirical formulae.

First, empirical formulae are ambiguous due to a phenomenon known as isomerism. The empirical formula for ethanol is also the empirical formula for the quite different pure substance methoxymethane. To distinguish between ethanol and methoxymethane, we need knowledge of interactions between the atoms forming the molecule. Isomerism is not always a problem. In the Plinius domain, it does not occur.

Second, the subscripts denoting amounts of constituents can be absolute or relative. This is a consequence of particular features of the bonds between atoms. Chemistry recognizes three kinds of bonds: atomic, ionic, and metallic. Only atomic bonds give rise to molecules. As a result of the nature of atomic bonds, the numbers denoting amounts are absolute for molecules. (Thus, NO_2 and N_2O_4 are different pure substances. Both exist.) For ionic and metallic compounds, the numbers are relative. Chemists often signal cases involving both absolute numbers and proportions by writing a formula that groups molecular subunits, as in the formula commonly given for hydroxylapatite, $Ca_5(PO_4)_3OH$. Here, PO_4 and OH (note that the number "1" is omitted at three places) represent molecular subunits and the numbers are absolute. By contrast, the numbers "5" for Ca, "3" for PO_4, and the omitted "1" for OH are relative and their greatest common divisor is required to be one.

In the present section, we will first specify an ontology of pure substances followed by examples (Section 6.2 through Section 6.4). Then we give an outline of the full Plinius ontology (Section 6.5). Extensions of the ontology of pure substances along the two degrees of freedom discussed in Section 3 follow (Section 6.6 and Section 6.7). The example is rounded off by briefly looking at representation and implementation (Section 6.8) and taking stock (Section 6.9).

6.2 Complex Concepts for Pure Substances

We first lay down an ontology of pure substances for the Plinius domain. We do not anticipate isomerism, so the concepts can be modeled as constituent lists. We will have to deal with the second complication, however.

The primitive concepts needed to construct complex concepts for pure substances are concepts for chemical elements and concepts for natural numbers (type *N*, as before). Concepts of type *E* (for *chemical element*) can be introduced extensionally, by listing the chemical elements currently known. This would result in a time-dependent ontology, however, because new chemical elements are synthesized in particle accelerators at a rate of about one per year. It is unwise to burden an ontology with time-dependent knowledge, so it is a better choice to define concepts of type *E* intensionally. A chemical element is uniquely identified by its atomic number. We introduce a function **atom_kind** that, when applied to an atomic number, yields a concept of type *E*. For example, **atom_kind**(6) is the label for the concept of the chemical element carbon.

To distinguish between absolute and relative amount specifications, we have introduced complex concepts of

type *GR* (for *group*). A group is defined as consisting of chemical elements. Groups are chosen such that, for any given pure substance, the instances of any concept of type group are either molecules or atoms not connected to other atoms by atomic bonds. There is no way to decide automatically what the groups are. This is not a disadvantage of our approach but rather a consequence of the state of the art in the domain. Next, complex concepts of type *PS* (for *pure substance*) are defined in terms of concepts of type *GR*. The BNF specification is given in Table 3. The type of any concept is unambiguously fixed by its syntax. The subscripts at the closing braces of labels in set form, *GR* and *PS*, are inserted for convenience only.

The canonicity constraint, (4), applies to all labels written as sets of tuples. Another constraint has to be added to ensure that the greatest common divisor of the numbers specifying relative amounts in labels expressions for pure substances be one. The latter constraint cannot be expressed in a context-free language. First-order expressions that define the present label language and incorporate the two additional constraints can be found elsewhere [13].

6.3 Example and Counterexample

We provide two examples to make matters concrete. The first constructs a concept for hydroxylapatite, $Ca_5(PO_4)_3OH$. For convenience, we will specify concepts by their chemical formulae in addition to the label expressions. It is understood, however, that only the labels form part of the ontology.

Hydroxylapatite consists of three groups. The atomic numbers of Ca, P, O, and H are 20, 15, 8, and 1, respectively.

$$\mathbf{label}(\mathrm{Ca}) = \{\langle \mathbf{atom\_kind}(20), 1\rangle\}_{GR}$$
$$\mathbf{label}(\mathrm{PO_4}) = \{\langle \mathbf{atom\_kind}(15), 1\rangle, \langle \mathbf{atom\_kind}(8), 4\rangle\}_{GR}$$
$$\mathbf{label}(\mathrm{OH}) = \{\langle \mathbf{atom\_kind}(8), 1\rangle, \langle \mathbf{atom\_kind}(1), 1\rangle\}_{GR}$$

so that the label for hydroxylapatite becomes

$$\mathbf{label}(\mathrm{Ca_5(PO_4)_3OH}) =$$
$$\{\langle \mathbf{label}(\mathrm{Ca}), 5\rangle, \langle \mathbf{label}(\mathrm{PO_4}), 3\rangle, \langle \mathbf{label}(\mathrm{OH}), 1\rangle\}_{PS}$$

As stated before, the present ontology is unable to discriminate isomers. Ethanol and methoxy-methane have identical labels. They are both described by the same empirical formula C_2H_6O. They are wholly molecular and thus consist of a single group.

6.4 Relations

The definitions of concepts of type group and pure substance are of the kind identified by $\mathcal{L}_{\mathrm{ext}}$ (Section 5.1). Therefore the constituent and kind-of relations identified in Section 4.2 and Section 5 can be applied here directly. Writing the concept for table salt (not explicitly introduced) as NaCl we can, for instance, verify the following:

$$\mathbf{constituent}(\mathrm{PO_4}, \mathrm{Ca_5(PO_4)_3OH})$$
$$\neg\mathbf{constituent}(\mathrm{PO_4}, \mathrm{NaCl})$$
$$\mathbf{constituent}(\mathbf{atom\_kind}(15), \mathrm{PO_4})$$
$$\mathbf{constituent}(\mathbf{atom\_kind}(15), \mathrm{Ca_5(PO_4)_3OH})$$
$$\neg\mathbf{constituent}(\mathbf{atom\_kind}(15), \mathrm{NaCl})$$

Superconcepts are easily introduced. An example is the concept for pure substance in general, **superconcept**(*PS*). In most cases, superconcepts involving both hierarchies are particularly interesting from a chemical point of view. It is an advantage of the bottom-up approach that such superconcepts are easily available. For example, a phosphate is defined in chemistry as a pure substance that contains at least one PO_4 group. The concept of a phosphate, abbreviated here as 'phosphate,' can be introduced by way of its label:

$$\mathbf{label}(\mathrm{phosphate}) =_{\mathrm{Df}} \mathbf{superconcept}(PS, \mathbf{label}(\mathrm{PO_4}))$$

We can again immediately verify a host of subconcept-superconcept relations, among them $Ca_5(PO_4)_3OH \sqsubseteq$ phosphate and NaCl $\not\sqsubseteq$ phosphate.

Note that the verification of all relations mentioned only needs information packed into the label expressions.

6.5 The Plinius Ontology

The actual ontology employed in the Plinius project (mentioned in Section 1.3) defines materials in a bottom-up way.

TABLE 3
BNF SPECIFICATION OF THE LABEL LANGUAGE THAT DEFINES AN ONTOLOGY OF PURE SUBSTANCES

```
   element_label   ::=  atom_kind(natural_number) | superconcept(E)
    number_label   ::=  natural_number | superconcept(N)
   element_tuple   ::=  "⟨" element_label "," number_label"⟩"
  element_tuples   ::=  element_tuple | element_tuple "," element_tuples
     group_label   ::=  "{" element_tuples "}GR" | superconcept(GR) |
                        superconcept(GR, element_label)
     group_tuple   ::=  "⟨" group_label "," number_label "⟩"
    group_tuples   ::=  group_tuple | group_tuple "," group_tuples
 pure_subs_tuple   ::=  "{" group_tuples "}PS" | superconcept(PS) |
                        superconcept(PS, group_label) |
                        superconcept(PS, element_label)
```

Chemists speak about systems; materials are a special kind. Systems either have one or more discontinuities that divide the space occupied by the system into separate parts, as when a layer of oil floats on water, or discontinuities are absent, as in an ethanol-water mixture. The former kind of systems is called heterogeneous, the latter homogeneous. The discontinuities in heterogeneous systems separate parts that themselves are homogeneous.

Under a microscope, materials like ceramics are seen to be heterogeneous systems. The separate parts are called grains. In between the grains there may be substances whose composition differs from that of the grains themselves.

The basis of the Plinius ontology is the ontology of pure substances just discussed. Then there is a construction rule for homogeneous systems that defines those systems in terms of pure substances and their proportions. Finally, there is a construction rule for heterogeneous systems, which are defined in terms of homogeneous systems and their proportions. All complex concepts are defined as constituent lists. The design, including the use of superconcepts, proved to be very fruitful for the system's purposes.

6.6 Varying the Choice of Indivisible Objects

Above (Section 3.1) we have discussed the theme of replacing indivisible objects by indivisible objects on a lower level of composition. A variant on this theme replaces a set of primitive concepts of the same type by a set of concepts at the *same* level of composition to account for finer distinctions among instances. This situation is not contrived for the purpose of the present discussion. It arose in chemistry 1914–1920 when isotopes were discovered [30]; in thermodynamics, it arises in the context of the entropy of mixing [31], [32].

To take account of isotopes in the ontology of pure substances, we have to replace the class of primitive concepts of type *E* by a class of primitive concepts of type *I* (for *isotope*). Isotopes are characterized by atomic number (common symbol *Z*) and mass number (common symbol *A*); the mass number is the sum of the numbers of protons and neutrons in the atomic nucleus. If two isotopes have the same atomic number, they are isotopes of the same chemical element. The replacement can proceed by means of a redefinition of the function `atom_kind` to take two arguments rather than one, to obtain `atom_kind`(*Z*, *A*). (For the second argument we could also have chosen the number of neutrons, but the present choice conforms more closely to chemical practice.) The value of the function for any two arguments is a concept of type *I*. For chemical elements, the value of *A* does not matter. Therefore, in the new version concepts for elements can be written as `atom_kind`(*Z*, `superconcept`(*N*)). The concepts for isotopes of any given element are then subconcepts of the concept for the element, which is the way chemists construe the relationship, too.

The only other change required is to redefine concepts of type *group*, by substituting *I* for *E* in the BNF specification of Table 3. The definition of concepts of type *pure substance* and definitions in terms of pure substances remain unaffected, because these concepts do not 'see' elements or isotopes. They only 'see' groups, and those we have appropriately redefined.

6.7 Introducing Interactions

Section 3.2 argues that the introduction of graphs is a generally applicable way to obtain concepts that incorporate at least a first approximation of the interactions. There is abundant experience with this use of graphs in chemistry. The reason is clear: It takes graphs to distinguish, for instance, ethanol and methoxymethane (compare the counterexample of Section 6.3).

In a chemical graph, the nodes stand for atoms and the edges for bonds. Chemical graphs are undirected and very often cyclic. In the literature there is ample discussion of chemical graphs. Research has addressed issues at the conceptual, representational, and implementational levels. AI applications are described in [33], [34], [35], [36] among others. An interesting application outside AI is the derivation of topological characteristics of molecules on the basis of graphs. The characteristics prove to be remarkably effective in predicting certain properties of the substances [37]. The DENDRAL project [38] has been immensely fruitful for chemical graph research [39], [40], [41], [42] [43], [44].

To accommodate isomerism in an ontology of pure substances, we have to redefine concepts of type *group* such that they are expressed as graphs. Concepts of type *pure substance* and higher up remain unaffected.

Chemical graphs are not always attractive. Graphs may be ambiguous because they cannot distinguish between stereo-isomers. Graphs can also be over-specific. A graph is over-specific if the costs of its inclusion are not compensated by gains. For example, defining a concept of type group as a chemical graph entails that we have to specify each and every group as a graph. This in turn means that we have to know the structure. Often, but not always, we can take such information from chemical databases most of which are quite expensive to access. If only a few structural isomers are expected in the domain, it may be advantageous to seek a more modest way of specifying configuration information. Lack of canonicity may then become a problem. When graphs are used only to distinguish the isomers that occur in the domain under study, we run into trouble when the domain has to be enlarged.

6.8 Formalization and Implementation

A number of versions of the ontologies of pure substances discussed here can, and actually have been, formalized in various description logics and implemented in the corresponding systems [3]. There is no doubt that the same can be said for other bottom-up ontologies. Yet a description logic is not a 'natural' choice for representation language when bottom-up ontologies are involved. A description logic stores conceptual information in its predicates, whereas we have been storing this information in arguments. In fact, so much knowledge is stored in arguments that assertions like `constituent`(c_1, c_2) and $c_1 \sqsubseteq c_2$ need not be stated explicitly. Their truth can be determined by in-

specting **label**(c_1) and **label**(c_2). The worst-case computational complexity of these verification operations is of the order l^n, where l is the average number of tuples in a set and n is the number of nested levels. Ontologies with large n will be rare (in the Plinius ontology, $n \leq 5$), so performance is not a problem.

The underlying logic is a simple first-order logic that expresses knowledge as assertions of the form **predicate** (**argument**$_1$, **argument**$_2$, ...), where the only thing to note is that the **argument**$_i$'s are label expressions. From this perspective, Prolog (as a subset of first-order predicate logic) and feature logics [45] appear a more natural choice of representation language. Accompanying implementations can use Prolog (as a programming language) or ALE [46].

We have in fact implemented the Plinius ontology in Prolog. We do not have the space to go into details (see [13]). The effort was mainly directed at implementing the inferences to automatically verify whether a constituent or kind-of relation holds between two concepts. The expressions for label inspection presented in Section 4 and Section 5 can be turned into Prolog rules more or less directly.

6.9 Taking Stock

The different versions of the ontology of pure substances discussed here are put together in a graph of ontologies, Fig. 1. The four information-discarding transformations shown in the figure have been implemented in the Prolog version mentioned in Section 6.8.

It has been observed by Gruber [47] that a good ontology consists of two mutually supplementary parts: a natural-language part for explanatory purposes and a formal part for ambiguity reduction. The two parts together define the concepts. The important difference is that automated reasoning is supported by the formal part only. Therefore, we prefer formal definitions whenever possible.

It is only natural to inquire how much of the definitions of concepts given above is formal. For the atomic concepts the answer is simple. Concepts of type natural number are only posited. Concepts of type chemical element are defined by means of the function **atom_kind** that is also only posited. In these cases, meaning is acquired through the natural-language account that anchors the definitions into consensus domain knowledge. Formally, we just have names and automatic reasoning about their meanings is impossible.

For the complex concepts, the answer is less obvious. The construction rules define concepts such that they coincide with their definitions. All the information available to distinguish any concept from other concepts is stored in the label expression for the concept itself. The expression involves partial rather than full information. This is particularly evident for constituent lists.

Although the information present in label expressed as constituent lists is partial, it is still sufficient to unambiguously identify the type of the concept. It also suffices for verifying constituent and kind-of relations. This does not entail that types are fully defined with respect to each other for each and every application. Full definition is far off: in the case of chemistry it would involve a full conceptualization of bonding. Moreover, for most practical purposes the definitions given here provide just what we need.

The bottom-up design of the Plinius ontology solves the problem (mentioned in Section 1.3) of defining all chemical substances in advance. A parsimonious set-up involving two construction rules of the parts list variety is sufficient. In a description logic we would have had to add an explicit concept each time a new substance is encountered. Ontolingua [22] allows construction rules, but they have to be written in raw KIF [48]. The automatic translators currently available in the Ontolingua system are unable to handle raw KIF.

The representations of our ontologies in description logics [3] are complex and unwieldy compared to the slim Prolog version. The taxonomic, top-down approach that underlies description logics is simply not suited to conceptualize this kind of knowledge. With respect to kind-of hierarchies, our bottom-up ontology is as expressive as a

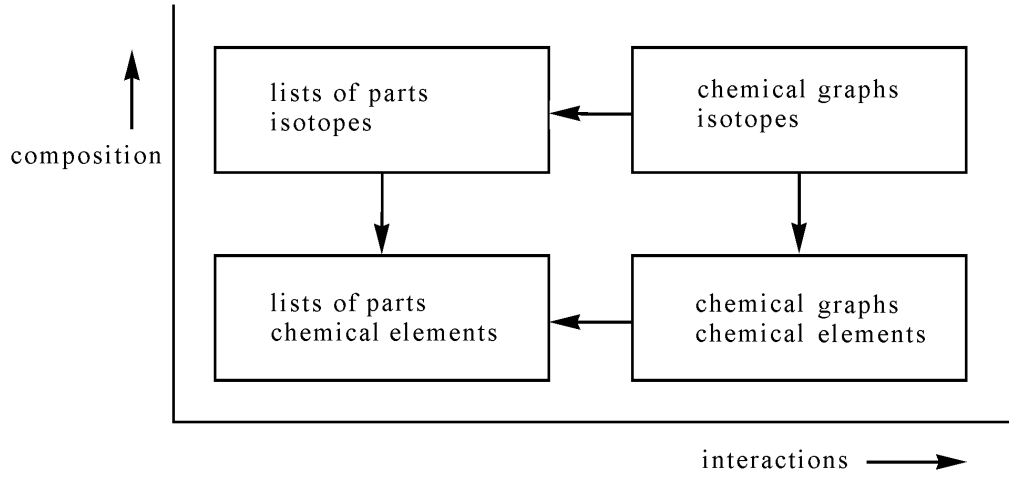


Fig. 1. Graph of ontologies of pure substances.

top-down ontology written in a description logic or pure Ontolingua, but at less cost. A bottom-up ontology additionally has the constituent relation and combinations of kind-of and constituent relations built in from the start.

7 Attribute Combinations

The bottom-up approach can be extended to accommodate more abstract cases, where concepts are not modeled by specifying their parts but by specifying meaning constituents. Such complex concepts can be formed by 'gluing' together other concepts with the agreement that each of the other concepts in some unidentified way specifies a partial meaning of the complex concept. This may be called an abstract version of atomism. We are defining the meaning of complex concepts as assemblies of atomic meaning constituents, but we do not imply that the referents of meaning constituents can be regarded as parts of the referents of complex concepts.

The meaning constituents will be called attributes and their combination, a complex concept, an attribute combination. Since the parts-whole idea has been abandoned, meaning constituents are assembled in tuples rather than sets and the constituent relation is not defined between these complex concepts and their meaning constituents. For an example, consider the inclusion of ions in the ontology of pure substances. Ions are charged atoms or molecules. Charges are positive or negative and their magnitude is a whole multiple of the elementary charge. It is obvious to model ions as combinations of two attributes: a concept of type *group* to specify chemical composition and a concept for a whole number to specify charge.

In an ontology of diseases developed for the Sapiens project, we have modeled diseases as attribute combinations [7]. Sapiens aimed at automatic generation of index terms through analysis of title and abstract of bibliographic document descriptions. The bibliographic database in question was Elsevier's *Excerpta Medica*; the index terms had to be taken from the accompanying thesaurus EMTREE. We found the classification of diseases in EMTREE confusing. Some diseases are classified according to location in the body (macroscopic anatomy), others according to cause, yet others in still another way. We decided to merge the hierarchies and define any disease as a combination of four attributes. The attributes are taken from sets of concepts defined previously: location according to macroscopic anatomy, location according to functional anatomy, cause (etiology in official terminology), and organism affected. This differs from an approach reported in the literature [49], which also models diseases as attribute combinations. There, the approach is used to model relevant relations for existing diseases. In other words, that ontology pays attention to reference. The Sapiens ontology, by contrast, pays attention to Fregean sense because it is intended to *define* diseases in a parsimonious way that allows construction of concepts for hitherto unknown diseases when needed.

8 Concluding Remarks

The bottom-up approach to ontology design proposes to lay down the meaning of complex concepts by means of primitive meaning constituents. In concrete cases, the instances of complex concepts are complex objects and the instances of constituents are their indivisible parts. This interpretation leads immediately to an ontology of atomism that naturally accommodates the two degrees of freedom so characteristic of the atomist research tradition: choice of smallest objects and choice of the way to specify interaction between smallest objects. A concrete example of ontologies of pure substances has demonstrated the viability of the approach. Other applications have been outlined.

The bottom-up design is an attractive way to construct ontologies. In the applications described here, the bottom-up approach leads to parsimonious ontologies that naturally support hierarchical reasoning along two, orthogonal dimensions: constituent and kind-of.

We do not claim that the bottom-up approach constitutes a generally applicable recipe. In keeping with the spirit of our opening remarks, the bottom-up approach is presented as one out of many ways to construct ontologies. We have favorable experiences with the approach in two nontrivial cases, an ontology of the chemical composition of materials and an ontology of diseases. We can imagine other applications, but further research is needed to make those work. We have also encountered cases where the approach does not work. To sum, the bottom-up approach is intended to enrich rather than supplant the ontology designer's toolkit.

Acknowledgments

It is a pleasure to acknowledge the contributions of Jan Vis and Hidde de Jong (both of the University of Twente) to the present work. Jan Vis has suggested various improvements to the treatment of the semantics of the concepts. We are also indebted to the anonymous referees for their helpful comments.

References

[1] N. Guarino and P. Giaretta, "Ontologies and Knowledge Bases: Towards a Terminological Clarification," *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing 1995*, N.J.I. Mars, ed., pp. 25–32, Amsterdam: IOS Press, 1995.

[2] N.J.I. Mars, "What is an Ontology?" *The Impact of Ontologies on Reuse, Interoperability, and Distributed Processing*, A. Goodall, ed., pp. 9–19, Uxbridge, Middlesex, U.K.: Unicom, 1995.

[3] P.-H. Speel, "Selecting Knowledge Representation Systems," PhD thesis, Univ. of Twente, Enschede, the Netherlands, 1995.

[4] N.J.I. Mars, "The Role of Ontologies in Structuring Large Knowledge Bases," *Knowledge Building and Knowledge Sharing*, K. Fuchi and T. Yokoi, eds., pp. 240–248, Tokyo: Ohmsha, 1994.

[5] IBM, *Business System Development Method: Introducing BSDM*, second ed., London: IBM U.K., 1992.

[6] N.J.I. Mars and A. Schreiber, "Direct Access to Knowledge in Bibliographic Databases," *Proc. ARTINT Workshop Artificial Intelligence and Information Retrieval*, Luxembourg, pp. 83–86, Sept. 1985, Luxembourg, Commission of the European Communities, 1985.

[7] P.-H. Speel, N.J.I. Mars, and P.E. van der Vet, "A Knowledge-Based Approach to Semi-Automatic Indexing," *Proc. Workshop Language Information Processing*, Oct. 1991, Washington, D.C., held during the 54th ASIS Annual Meeting, A.T. McCray, ed., pp. 49–58, 1991.

[8] N.J.I. Mars, W.G. ter Stal, H. de Jong, P.E. van der Vet, and P.-H. Speel, "Semi-Automatic Knowledge Acquisition in Plinius: An Engineering Approach," *Proc. Eighth Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, Jan.–Feb. 1994, B. Gaines and M. Musen, eds., pp. 4.1–4.15, 1994.

[9] P.E. van der Vet, H. de Jong, N.J.I. Mars, P.-H. Speel, and W.G. ter Stal, "Plinius Intermediate Report," Memoranda Informatica 94-35, Univ. of Twente, Enschede, the Netherlands, 1994.

[10] P.E. van der Vet and N.J.I. Mars, "Structured System of Concepts for Storing, Retrieving, and Manipulating Chemical Information," *J. Chemical Information and Computer Sciences*, vol. 33, pp. 564–568, 1993.

[11] P.E. van der Vet and N.J.I. Mars, "Concept Systems as an Aid for Sharing and Reuse of Knowledge Bases in Materials Science," *Knowledge-Based Applications in Materials Science and Eng.*, J.K. McDowell and K.J. Meltsner, eds., pp. 43–55, Warrendale, Penn., Minerals, Metals, and Materials Soc., 1994.

[12] P.E. van der Vet, P.-H. Speel, and N.J.I. Mars, "Ontologies for Very Large Knowledge Bases in Materials Science: A Case Study," *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing 1995*, N.J.I. Mars, ed., pp. 73–83, Amsterdam: IOS Press, 1995.

[13] P.E. van der Vet and N.J.I. Mars, "Bottom-Up Construction of Ontologies: The Case of an Ontology of Pure Substances," Memoranda Informatica 95-35, Univ. of Twente, Enschede, the Netherlands, 1995.

[14] R. MacGregor, "The Evolving Technology of Classification-Based Knowledge Representation Systems," *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, J.F. Sowa, ed., pp. 385–400, San Mateo Calif.: Morgan Kaufmann, 1991.

[15] R.J. Brachman, "What's in a Concept: Structural Foundations for Semantic Networks," *Int'l J. Man-Machine Studies*, vol. 9, pp. 127–152, 1977.

[16] D. Knight, *Ordering the World: A History of Classifying Man*, London: Burnett, 1981.

[17] G. Toraldo di Francia, *The Investigation of the Physical World*, Cambridge, U.K.: Cambridge Univ. Press, 1981.

[18] E. Segrè, *From X-Rays to Quarks*, San Francisco: Freeman, 1980.

[19] R.J. Brachman, " 'I Lied about the Trees,' or Defaults and Definitions in Knowledge Representation," *AI Magazine*, vol. 6, no. 3, pp. 80–93, 1985.

[20] H.R. Post, "The Problem of Atomism," *British J. Philosophy of Science*, vol. 26, pp. 19–26, 1975.

[21] R.R. Bakker, P.C. van den Bempt, N.J.I. Mars, D.-J. Out, and D.C. van Soest, "Issues in Practical Model-Based Diagnosis," *Future Generation Computer Systems*, vol. 9, pp. 329–337, 1993.

[22] T.R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, pp. 199–220, 1993.

[23] T.R. Gruber and G.R. Olsen, "An Ontology for Engineering Mathematics," *Proc. KR, Fourth Int'l Conf. Principles of Knowledge Representation and Reasoning*, J. Doyle, E. Sandewall, and P. Torasso, eds., pp. 258–269, San Francisco, Morgan Kaufmann, 1994.

[24] J.H. Adams and M.A. Dahl, "Using Knowledge-Based Systems to Define Materials Technology in the Aircraft Design/Build Process," *Knowledge-Based Applications in Materials Science and Engineering*, J.K. McDowell and K.J. Meltsner, eds., pp. 67–74, Warrendale, Penn., Minerals, Metals, and Materials Soc., 1994.

[25] S. Bradley, A. Agogino, and W. Wood, "Intelligent Engineering Component Catalogs," *Artificial Intelligence in Design '94*, J. Gero and F. Sudweeks, eds., pp. 641–658, Dordrecht: Kluwer Academic, 1994.

[26] M.E. Winston, R. Chaffin, and D. Herrmann, "A Taxonomy of Part-Whole Relations," *Cognitive Science*, vol. 11, pp. 417–444, 1987.

[27] N. Asher and L. Vieu, "Toward a Geometry of Common Sense: A Semantics and Complete Axiomatization of Mereotopology," *Proc. IJCAI, 14th Int'l Joint Conf. Artificial Intelligence*, Montreal, Que., Canada, Aug. 1995, C.S. Mellish, ed., San Mateo Calif., pp. 846–852, IJCAII/Morgan Kaufmann, 1995.

[28] D. Shriver, P. Atkins, and C. Langford, *Inorganic Chemistry*. Oxford, U.K.: Oxford Univ. Press, second ed., 1994.

[29] *Nomenclature of Inorganic Chemistry*, G. Leigh, ed., *IUPAC Recommendations 1990*, Oxford, U.K.: Blackwell Scientific Publications, 1990.

[30] P.E. van der Vet, "The Aborted Takeover of Chemistry by Physics: A Study of the Relations between Chemistry and Physics in the Present Century," PhD thesis, Univ. of Amsterdam, Amsterdam, 1987.

[31] J. von Neumann, *Mathematische Grundlagen der Quantenmechanik*, Berlin: Springer, 1932.

[32] D. Dieks and V. van Dijk, "Another Look at the Quantum Mechanical Entropy of Mixing," *Am. J. Physics*, vol. 56, pp. 430–434, 1988.

[33] R. Levinson, "A Self-Organizing Retrieval System for Graphs," *Proc. AAAI '84, Nat'l Conf. Artificial Intelligence*, R. Brachman, ed., Los Altos, Calif., pp. 203–206, AAAI/William Kaufmann, 1984.

[34] H. Gelernter, J.R. Rose, and C. Chen, "Building and Refining a Knowledge Base for Synthetic Organic Chemistry via the Methodology of Inductive and Deductive Machine Learning," *J. Chemical Information and Computer Sciences*, vol. 30, pp. 492–504, 1990.

[35] A. Napoli, "Subsumption and Classification-Based Reasoning in Object-Based Representations," *Proc. 10th European Conf. Artificial Intelligence*, B. Neumann, ed., pp. 425–429, Chichester, U.K.: John Wiley, 1992.

[36] R.E. Valdés-Pérez, "Machine Discovery in Chemistry: New Results," *Artificial Intelligence*, vol. 74, pp. 191–201, 1995.

[37] *Computational Chemical Graph Theory*, D.H. Rouvray, ed., New York: Nova Science Publishers, 1990.

[38] R.K. Lindsay, B.G. Buchanan, E.A. Feigenbaum, and J. Lederberg, "DENDRAL: A Case Study of the First Expert System for Scientific Hypothesis Formation," *Artificial Intelligence*, vol. 61, pp. 209–261, 1993.

[39] R.E. Carhart, D.H. Smith, H. Brown, and C. Djerassi, "An Approach to Computer-Assisted Elucidation of Molecular Structure," *J. Am. Chemical Soc.*, vol. 97, pp. 5,755–5,762, 1975.

[40] H. Brown, L. Hjelmeland, and L. Masinter, "Constructive Graph Labeling Using Double Cosets," *Discrete Math.*, vol. 7, pp. 1–30, 1974.

[41] H. Brown and L. Masinter, "An Algorithm for the Construction of the Graphs of Organic Molecules," *Discrete Math.*, vol. 8, pp. 227–244, 1974.

[42] J.G. Nourse, "Generalized Stereoisomerization Modes," *J. Am. Chemical Soc.*, vol. 99, pp. 2,063–2,069, 1977.

[43] J.G. Nourse, R.E. Carhart, D.H. Smith, and C. Djerassi, "Exhaustive Generation of Stereoisomers for Structure Elucidation," *J. Am. Chemical Soc.*, vol. 101, pp. 1,216–1,223, 1979.

[44] J.G. Nourse, D.H. Smith, R.E. Carhart, and C. Djerassi, "Computer-Assisted Elucidation of Molecular Structure with Stereochemistry," *J. Am. Chemical Soc.*, vol. 102, pp. 6,289–6,295, 1980.

[45] B. Carpenter, *The Logic of Typed Feature Structures*, Cambridge, U.K.: Cambridge Univ. Press, 1992.

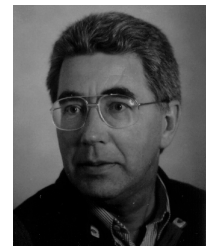
[46] B. Carpenter and G. Penn, "ALE: The Attribute Logic Engine, Version 2.0, User's Guide," technical report, Computational Linguistics Program, Philosophy Dept., Carnegie Mellon Univ., Pittsburgh, 1994.

[47] T.R. Gruber, "The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases," *Proc. KR '91, Second Int'l Conf. Principles of Knowledge Representation and Reasoning*, Cambridge, Mass., J. Allen, R. Fikes, and E. Sandewall, eds., pp. 601–602, 1991.

[48] M.J. Genesereth and R.E. Fikes, "Knowledge Interchange Format, Version 3.0, Reference Manual," Report Logic 92-1, Computer Science Dept., Stanford Univ., Stanford, Calif., 1992.

[49] O. Senyk, R.S. Patil, and F.A. Sonnenberg, "Systematic Knowledge Base Design for Medical Diagnosis," *Applied Artificial Intelligence*, vol. 3, no. 2, pp. 249–274, 1989.

Paul E. van der Vet obtained his MSc degree in chemistry and philosophy of science in 1977 from the University of Utrecht, the Netherlands, and his PhD degree in chemistry in 1987 from the University of Amsterdam. He has been a senior staff member of the Knowledge–Based Systems Group at the University of Twente, Enschede, the Netherlands, since 1989. Earlier, he worked at the Tiele Academy, a library polytechnic, and at a courseware firm. He has served on the boards of the Dutch Society for Philosophy of Science and the Dutch Artificial Intelligence Society. His current research interests are knowledge-intensive natural-language engineering, particularly of scientific texts and, with Nicolaas J.I. Mars, building ontologies for several scientific domains. He heads the Plinius project.



Nicolaas J.I. Mars obtained his BSc and MSc degrees in electrical engineering in 1972 and 1974, respectively, and his PhD degree in technical sciences in 1982, all from the University of Twente, Enschede, the Netherlands. He has been a professor of computer science at the University of Twente since 1986. Since 1996, he has also served as deputy director for electronic services at the Netherlands Institute for Scientific Information Services (NIWI), an institute of the Royal Netherlands Academy of Arts and Sciences in Amsterdam. From 1993 to 1995, he was also dean of the Department of Computer Science at the University of Twente. Previously, he worked at the University of Leiden and at Yale University. He has been on the boards of the Dutch Artificial Intelligence Society, the Dutch Information Science Society, and the European Coordinating Committee for AI. His current research interests are real-world applications of knowledge-based systems, especially for engineering and scientific tasks. With Paul E. van der Vet, he works on building and using ontologies for several scientific domains. He is a senior member of the IEEE and a member of the IEEE Computer Society.