

REAP the fruits of REA:

An ontological evaluation of Firstness.

Wim Laurier, Faculty of Economics and Business Administration, Ghent university, wim.laurier@ugent.be
Geert Poels, Faculty of Economics and Business Administration, Ghent university, geert.poels@ugent.be
Frederik Gailly, Faculty of Economics and Business Administration, Ghent university, frederik.gailly@ugent.be

Abstract

In this article, we analyze Geerts' and McCarthy's mapping between REA and Sowa's ontology and refine their analysis by extending Sowa's ontology with parts of the Bunge-Wand-Weber ontology. We also provide a methodology for this extension, by determining where an ontology is too fine-grained or too coarse grained. Finally, we present the refined REA ontology Firstness categories, illustrate their structure and discuss the advantages of this refinement.

Keywords

Ontology, methodology, categorization, completeness, validity, REA, modeling grammar, Sowa, Bunge-Wand-Weber.

I. Introduction

In this article, we will discuss the choices that led to the REA ontology as presented in Geerts and McCarthy (2002). We will start with discussing the constraints of Sowa's ontology that were used to define Geerts' and McCarthy's view on the REA ontology. Next we will propose a refinement of the REA ontology definition. This is followed by a discussion and refinement of the method that Geerts and McCarthy used for their Sowa-REA mapping. We will argue that this mapping is actually a completeness check for REA and will also introduce validity issues in REA ontology development. Finally we will present some advantages we attribute to the refined REA ontology. These advantages should be sought in lucidity and homogeneity.

We started our analysis of REA because we think REA is an attractive ontology for the reason that its limited number of categories and its visual rendering (Poels et al. 2007) and UML compliance (Hruby 2006) make it an ontology that is fit for educational purposes. In addition, REA originates in empirical observations in accounting (McCarthy 1982) and has a theoretical basis in microeconomics (Geerts and McCarthy 2002). Finally, REA's main selling argument is that it does not limit its scope to the enterprise boundaries (Lampe 2002), facilitating interoperability modeling. Hence, inter-enterprise patterns like CMI (customer managed inventory) and supply chains can be modeled more easily.

REA is a promising ontology that is popular in education (Dunn et al. 2005) and is used in standards (ISO/IEC 2006) although it is still facing some challenges. Some of those challenges are listed in Lampe (2002). The majority of them might be categorized as ontological clarity issues. In this paper, we will focus on the ontological deficiencies of Firstness¹, as indicated by the red ovals in Figure 1. A red oval indicates that one construct in Sowa's ontology (Sowa 2000) maps to several REA ontological constructs. This yields the notion of construct overload for the constructs in Sowa's ontology. If this construct overload for Sowa's ontology does not mean construct redundancy for the REA ontology further specification of Sowa's ontology is needed. Our approach is to extend Sowa's ontology to eliminate this construct overload, as the ontological completeness of the REA ontology cannot be thoroughly evaluated when ontological clarity issues remain unsolved. This way, we advocate an extended Sowa's ontology as a meta-model for the REA language, as represented in Figure 2.

In the remainder of this paper, all denoted words start with an uppercase. Section 2 contains a discussion of the constraints of Sowa's ontology that Geerts and McCarthy (2002) used to define the REA ontology. In this section we also extend these constraints in order to solve a number of ontological clarity issues in the REA ontology. Section 3 discusses the method that Geerts and McCarthy (2002) used to check the ontological completeness of the REA ontology with

respect to Sowa's ontology. In this section we refine this method and show that it can also be used to evaluate the ontological validity of the REA ontology. Using the extended Sowa ontology (section 2) and the improved ontological completeness and validity evaluation method (section 3), we present a refined REA ontology in section 4, focusing mainly on the Firstness level, and discuss the advantages of this refinement. Finally, section 5 summarizes our contributions and outlines future research.

II. Constraints

We see constraints as things that determine a theory (Nickles 1980b, 1980a) and consider ontologies theories about the world. In this light, we consider elementary categories as constraints. An elementary category is a one-dimensional partition of the items provided. We discuss the constraints of Sowa's ontology, as applied in Geerts and McCarthy (2002), and extend this set in order to solve a number of ontological clarity issues in the REA ontology.

Firstness, Secondness and Thirdness.

The concepts Firstness, Secondness and Thirdness that are used in Sowa's ontology are borrowed from Charles Sanders Peirce. (Sowa 2000). On his website² Sowa defines these concepts as follows. “An entity characterized by some inherent **Firstness**, independent of any relationships it may have to other entities. Formally, *Independent* is a primitive for which the has-test³ need not apply. If x is an independent entity, it is not necessary that there exists an entity y such that x has y or y has x .” **Secondness** is defined this way: “An entity in a relationship to some other entity. Formally, *Relative* is a primitive for which the has-test must apply.” “An entity characterized by some **Thirdness** that brings other entities into a relationship. An independent entity need not have any relationship to anything else, a relative entity must have some relationship to something else, and a mediating entity creates a relationship between two other entities. An example of a mediating entity is a marriage, which creates a relationship between a husband and a wife. According to Peirce, the defining aspect of Thirdness is “the conception of mediation, whereby a first and a second are brought into relation.” That property could be expressed in second-order logic.”

In his article, Lampe (2002) uses other names and definitions for the same concepts. These definitions come from Sowa (2000). Lampe calls items of Firstness Independent: “**Independent** items ... can be described as a monadic predicate – i.e. affirmed or denied without regard to anything external to the independent item. ... Another way of stating independence is that the item is defined alone and does not inherit any additional meaning by being combined with other items.” Items of Secondness are referred to as Relative: “**Relative** categories are representations of dyadic predicates – i.e. how two items are related when neither individual item describes the complete relationship ...” Items of Thirdness are described as Mediating: “**Mediating** categories in the ontological classification scheme are comprised of triadic combinations or groupings. In the same way that a Secondness level relative describes the dyadic relationship of a level

one Firstness category item to another compatible item, the Thirdness mediating categories explain why a Secondness level relative is grouped with a third item in an irreducible triadic relationship.”

We see no reason to challenge this partition, as we consider the definition of items of Firstness a pragmatic choice. A choice that should deliver the most convenient results, and not the most ‘correct’ theoretical grounds. As we consider Secondness and Thirdness as categories that describe the number of Independent items that are needed to reach full semantics, we can question whether Thirdness is the highest category. This question is not relevant for this paper as we limit ourselves to the items of Firstness.

Abstract vs. Physical

On his website, Sowa defines abstract and physical this way. **Physical** is “*An entity that has a location in space-time. Formally, Physical is a primitive that satisfies the following axiom: (1) Anything physical is located in some place. (2) Anything physical occurs at some point in time.*” **Abstract** is “*Pure information as distinguished from any particular encoding of the information in a physical medium. Formally, Abstract is a primitive that satisfies the following axioms: (1) No abstraction has a location in space. (2) No abstraction occurs at a point in time. As an example, the information you are now reading is encoded on a physical object in front of your eyes, but it is also encoded on paper, magnetic spots, and electrical currents at several other locations. Each physical encoding is said to represent the same abstract information.*”

This definition yields an ambiguity, as all information is deemed to be absolute and everlasting. In addition personal interpretations or erroneous interpretations are judged impossible. This means all information has always been there and will always be there in the future without ever being changed or revised. This definition for information is not convenient in the context of information systems as it entails that information about occurrent phenomena that was not observed at the time of the occurrence is still retrievable when one manages to grasp this abstract information in a physical representation, as information is absolute and everlasting. This definition could be considered supposing informational time machines, which we consider a speculative hypothesis. We advocate a more pragmatic approach where information ceases to exist when all its accessible physical representations have ceased to exist, and information is created by events, measurements or observations. This reduces information to a physical category in Sowa’s ontology and makes the abstract category void for the information systems context.

Geerts and McCarthy (2002) have an analysis that can be interpreted as far more valuable and interesting from the information systems point of view. We understand their Physical categories as categories for representations of objects and events in the real world, while their Abstract categories interpret these representations. This object, interpretation and

representation triangle is represented in Peirce's semantic triangle, also called meaning triangle (Sowa 2000). The original semantic triangle is represented in Figure 3. We see how an 'interpretamen', which is an atomic interpretation, 'refers to' an 'object' in the real world. We suppose this 'interpretamen' is deemed Abstract in Geerts and McCarthy (2002). 'Object' and 'interpretamen' are related. Figure 3 also depicts how a 'representamen' 'evokes' an 'interpretation' and 'stands for' an 'object'. In this analysis, the 'representamen' 'mediates' between a real world object, that cannot be transformed into bits and bytes, and its interpretation that has informational value for the information system. The 'representamen' is part of the information system as it replaces the real world 'object'. We suspect the 'representamen' to be Physical in Geerts and McCarthy (2002).

However, since this approach distinguishes clearly between operational information objects, called Physical, and conceptual information objects, called Abstract, we advocate an additional level of abstraction, and an additional semantic triangle. The second semantic triangle is added because we reckon it is necessary to distinguish between the abstractions or interpretations that are evoked by object representations and the abstractions that are abstractions of those evoked abstractions. These abstractions and abstractions of abstractions are represented in Type Hierarchies that form the knowledge architecture that should allow us to represent and report faithfully and consistently. The representations and their interpretations or abstractions might be seen as the descriptive infrastructure that enables us to record real world phenomena in information systems as manageable data. All of this is represented in Figure 4.

To get a compact picture, we tilted the semantic triangles from figure 3 for figure 4. This means Secondness and Thirdness have switched places. The synthetic⁴ sign in figure 4 shows how an object has a State. A State is an interpretation that is present in the information system. States are like records in conventional databases. The difference is that not every object has a unique State. In stead, every object has a unique Instance that is a representation, while the State is an interpretation. This way we separate representation and identity from interpretation and semantics. The advantage is homonymy and synonymy become controllable variables, as for example coding errors will occur less frequent and will be retrieved more easily. Another advantage is that modeling compromises for valuable and invaluable resources are made explicit, e.g. there probably is a unique State for each Instance that represents a vintage car object. On the other hand, though each of 5000 nails may be unique, we consider them all as having the same State. Their identity or Instance might be a number that is assigned with a FIFO⁵ or LIFO⁶ method. This kind of Instance does not allow unique identification but enables the minimal functionality of (ac)counting. Hence, Every unique object has a unique Instance and every object has a State, consequently, every Instance 'evokes' a State. A conventional database record is actually a combination of an Instance and a State. The Instance contains the representamen for the identity of the recorded object. The State describes the object properties that do not determine its identity.

The analytic sign reveals how Types can be seen as collections of States, through Type Hierarchies. Type Hierarchies provide us with a representation that allows us to manage the quantity of information delivered by the descriptive system shown in the synthetic sign by reducing the number of data that is represented at once. This reduction is achieved by structuring data in a hierarchy that introduces symbols for collections of data. The difference between State and Types is that Types are nodes in Type Hierarchy trees, while States are leafs of Type Hierarchy trees. Consequently, States are a special kind of Types. States are the most detailed descriptions of objects. As a consequence, every object has exactly one State and can have multiple Types. Thanks to their reduced level of detail, Types refer to multiple States at once. Hence, Types can be considered collections of States while each State is related to one or more Instances, each standing for one object.

Continuant vs. Occurrent

The Continuant and Occurrent categories in figure 1 also originate in Sowa's ontology. Consequently, we cite Sowa's website. **Continuant** is "*An entity whose identity continues to be recognizable over some extended interval of time. Formally, Continuant is a primitive that satisfies the following axioms: (1) A continuant x has only spatial parts and no temporal parts. At any time t when x exists, all of x exists at the same time t . New parts of a continuant x may be acquired and old parts may be lost, as when a snake sheds its skin. Parts that have been lost may cease to exist, but everything that remains a part of x continues to exist at the same time as x . (2) The identity conditions for a continuant are independent of time. If c is a subtype of Continuant, then the identity predicate $Id_c(x,y)$ for identifying two instances x and y of type c does not depend on time. A physical continuant is an object, and an abstract continuant is a schema that may be used to characterize some object.*" **Occurrent** is "*An entity that does not have a stable identity during any interval of time. Formally, Occurrent is a primitive that satisfies the following axioms: (1) The temporal parts of an occurrent, which are called stages, exist at different times. (2) The spatial parts of an occurrent, which are called participants, may exist at the same time, but an occurrent may have different participants at different stages. (3) There are no identity conditions that can be used to identify two occurrents that are observed in non-overlapping space-time regions. A person's lifetime, for example, is an occurrent. Different stages of a life cannot be reliably identified unless some continuant, such as the person's fingerprints or DNA, is recognized by suitable identity conditions at each stage. Even then, the identification depends on an inference that presupposes the uniqueness of the identity conditions.*"

This definition is problematic, as it is biased by a time perspective, because the definitions evoke 'an interval of time'. This 'interval of time' is not further specified. Specifying an 'interval of time' would yield an operational definition and not an ontological definition, as different time intervals yield different categorizations. When we consider 'an interval of time' a second or a blink of an eye, most of things we observe would be Continuant, while most things would be

Occurrent if we expressed 'an interval of time' in geological time where millennia are no more than a blink of an eye. In a Cartesian⁷ time perspective, all things are created sometime and all things disappear sometime. In a relativistic time perspective, things get too complicated to be represented in information systems. This problem is illustrated by the fact that a snake is a Continuant in the example for axiom one and a human's lifetime is an Occurrent in the example for axiom three, while both are living creatures that are created and die. We do not see how a Continuant's lifetime – which seems to be an Occurrent – can be separated from itself by a definition that is determined by a time notion, as we consider the Continuant and its Occurrent lifetime concurrent.

To overcome this ambiguity, we focus on implicit inferences in Sowa's definition and suggest to replace the notions Continuant and Occurrent from Sowa's ontology by State⁸ and Transformation in the BWW ontology, because this dichotomy yields the same notions evading the time 'trap'. We cite Bunge-Wand-Weber: "**State**: *The vector of values for all property functions of a thing is the state of a thing.*" "**Transformation** is a mapping from one state to another state." (Green and Rosemann 2000; Wand and Weber 1995)

When we consider States Continuant and Transformations Occurrent, this could lead to the following – Sowa based – operational definition for Continuants and Occurrents. **Continuant** Types⁹ are Types that are related with Instances that can receive a unique identity with a sufficiently specified description of the objects location in space, regardless of time. This means for example a resource can be uniquely identified by its location in a warehouse or data can be uniquely identified through their location on storage media. We suppose a sufficiently detailed location description entails registration of transportation or substitution. In reality this means we suppose the Resource Instance we find on a certain location in a warehouse is the same we stored there earlier. If the resource is replaced by an action that was not recorded in the system, this assumption is wrong. We do not consider events that are not registered in a system part of the information systems architecture domain, but part of the operational management domain, therefore these kind of problems are not tackled here.

Occurrent Types are Types that are related with Instances that can receive a unique identity by a sufficiently specified description that has to contain both their spatial and temporal dimensions. Those spatial dimensions are determined by the Continuant Instances that are a precondition for the Occurrent Instance. In other words all Continuant Instances that are needed for an Occurrent Instance need to be at the location of the Occurrent Instance at the time the Occurrent Instance occurs. While two sequential observations – with a different location in time and the same location in space – of a Continuant object lead to the hypothesis that the same Continuant object is observed, two sequential observations – with

the same location in space but a different location in time – of an Occurrent object lead to the hypothesis that two distinct Occurrents object were observed.

Property vs. Thing

We consider the Property Thing dichotomy, such as for instance present in the Bunge-Wand-Weber ontology, as only implicitly treated in Sowa's ontology. We cite Sowa's definition for the root of his ontology, on his website. "*T: The universal type, which has no differentiae.*" In which 'differentiae' are "*The properties, features, or attributes that distinguish a type from other types that have a common super-type. The term comes from Aristotle's method of defining new types by stating the genus or super-type and stating the differentiae that distinguish the new type from its super-type.*"

Again, we turn to the Bunge-Wand-Weber ontology for an alternative definition. "*A **thing** is the elementary unit in the BWW ontology model. The real world is made up of things. Two or more thing (composite or simple) can be associated into a composite thing.*" "*Things possess **properties**. A property is modeled via a function that maps the thing into some value. For example, the attribute "weight" represents a property that all humans possess. In this regard, weight is an attribute standing for a property in general. If we focus on the weight of a specific individual, however, we would be concerned with a property in particular. A property of a composite thing that belongs to a component thing is called a hereditary property. Otherwise it is called an emergent property. Some properties are inherent properties of individual things. Such properties are called intrinsic. Other properties are properties of pairs or many things. Such properties are called mutual. Non-binding mutual properties are those properties shared by two or more things that do not "make a difference" to the things involved; for example order relations or equivalence relations. By contrast, binding mutual properties are those properties shared by two or more things that do "make a difference" to the things involved. Attributes are the names that we use to represent properties of things.*"(Green and Rosemann 2000; Wand and Weber 1995)

To identify the difference between properties and things, we focus on the phrase "Things possess properties" meaning there has to be a Secondness or relation assigning Properties to Things. This means both Properties and Things need to be items of Firstness. This sounds logical as the property 'red' for example has a meaning on its own however it may be related with the notions 'color', 'sports car' and 'tomato'. For this reason we add a Property column to Sowa's ontology.

Surprisingly, the property definition in BWW reveals some remarkable links with Sowa's ontology as presented earlier. For example the notions 'property in particular' and 'property in general' can be linked with the notions Physical and

Abstract in Sowa's Ontology, as properties in particular represent a property of an object, while properties in general can be considered interpretations of those properties in particular. For example 'red' represents a property of a certain tomato object and '83kg' represents a property of certain person object. They are physical properties or properties in particular, as they represent something. An average person would probably suppose 'red' is a 'color' and '83kg' is a 'weight'. He or she would also know there are other colors and weights in the 'weight' and 'color' domain. This makes 'weight' and 'color' Abstract Properties as they interpret a collection of Physical Properties.

Another surprising link with Sowa is the notion 'mutual property'. We consider a 'mutual property' a property that mediates between two things. As the definition describes two things that are related by a property, and both properties and things are considered Firstness, mutual properties yield the notion of Thirdness.

Context-providing vs. Context-slave

The previously discussed concepts and dichotomies are useful to identify constraints, but are yet not fine-grained enough to solve the ontological clarity problems with the REA ontology mentioned in the introduction as they do not distinguish Agents from Resources. Therefore, we introduce a new dichotomy that can be used to separate Agents from Resources. Our starting perspective is that Agents can own information systems, while Resources can't. In fact, every human possesses an informal information system in its brain. In the formal information systems context, we will focus on the ability of Agents to possess an information system or a subsystem of a larger information system. Possession of information systems could be defined using CRUD¹⁰ matrices, as only Agents and their systems can have authority and ability to perform these actions.

To extend the applicability of this 'possession', we define **Context-providing** things as things that are able to create, update or delete **Context-slave** things. In particular, an Agent can have authority and ability to create (produce), update (modify or use) or delete (consume) Resources. Of course, Agents also have ability and authority to create, read, update and delete information and information systems that can be considered Resources. In fact, Context-providers can initiate Context-slave Occurrents and own Context-slave Continuants, while Context-slaves cannot. Context-slaves and Context-providers can both be affected by Occurrents.

The Context-providing and Context-slave dichotomy provided so far cannot be literally applied to Properties. Therefore, we reveal Context-providing and Context-slave as relative and hierarchic. Resources are always controlled by an Agent. On the other hand, Agents can be controlled by other Agents. (e.g. A manager has the authority and ability to instruct a blue collar worker.) This means that Agents can act as Resources in certain conditions. Consequently every information system has exactly one Context-providing Agent, while its sub-systems might be controlled by Subordinate Agents,

which McCarthy (1982) calls Subordinate Units. This ‘possession’ is an extension of the ‘control’ in McCarthy (1982). Please notice we identify the ‘Economic Agent’ and ‘Economic Unit’ notions in McCarthy (1982).

Validating the preceding analysis we could state that Resources can be considered leafs of control trees, because Resources cannot control other Resources neither can they control Agents. Generalizing this statement, we utter Context-slave Things are leafs of control trees that are populated with Context-providing Things. Figure 5 shows how enterprises can be Resources for a conglomerate when we limit our view to level 0 and 1, as leafs of the control trees consisting of level 0 and 1 are enterprises. When we include level 2, the strategic business units (SBU) are Resources for the Context-providing Agent called ‘conglomerate’. For ‘SBU #2’ ‘wood’ and ‘metal’ are Resources, as they are leafs of this control tree. ‘SBU #2’'s control tree starts at level 2 and ends at level 3. In this level 2 and 3 view, ‘SBU #1’ and ‘SBU #2’ are ‘Outside party’ (McCarthy 1982) Agents to each other. When the control tree for ‘conglomerate’ contains level 0 to 3, ‘wood’ and ‘metal’ are also Resources for the ‘conglomerate’ Context-providing Agent. In the level 0 to 3 view, ‘SBU #1’ and ‘SBU #2’ are ‘Inside party’ (McCarthy 1982) Agents to each other.

Applying this ‘possession’ on Property level, Context-providing Properties are identical to all Agents in a (sub-) system, while Context-slave Properties can differ according to different Subordinate Agents (McCarthy 1982) of a Context-providing Agent. The notion Context-providing Agents combined with Context-providing Properties yields the notion of ‘independent view’ in Hruby (2006) because those properties are identical to all Agents in the Context, including the Context-providing Agent itself. We provide some explanation. The Context-providing Properties assigned to Context-slaves in a system are identically perceived by all Agents in this system. This means the Context-providing Properties’ meaning is determined by the highest Agent in the hierarchy, being the Context-providing Agent. For all Subordinate Agents and their sub-systems, the meaning of a Context-providing Property is determined outside their system and outside their authority. (e.g. A transfer of a Resource from the Context-providing Agent’s Context to another Agent is seen as a Decrement Event by all subordinate Agents’ in the Context.)

Context-slave Properties on the other hand are not determined by the highest Agent in the hierarchy. When we consider a tax administration the Context-providing Agent for an economy, a transfer of a Resource from one enterprise to another enterprise – both enterprises being Subordinate Agents of the tax administration – the transfer is neither an Increment neither a Decrement Event for the administration. For the enterprises – both owning a subsystem of the tax administrations information system – on the other hand, this transfer is either an Increment or Decrement Event. As this information is irrelevant for the higher level Agent it is hidden. This yields the notion tacit knowledge in Nonaka (Nonaka 1991; Nonaka et al. 1996). The value that is added through the enterprise’s transformation process, however, should be

explicit knowledge as it is relevant for the Context-providing Agent being the tax administration. If there is also a tacit value, hidden from the tax administration, this may be considered fraud. Notice a Property can be Context-providing or explicit for an Agent or its system, while this same Property may be Context-slave or tacit for a Context-providing Agent of the Context-providing Agent for the first Agent. The Context-providing Agent for a system is the highest Agent represented in its Agent hierarchy and determines the boundaries of his system.

Figure 6 shows how 'soft' is a Context-providing Property for all entities on levels 1 to 3, determined by 'conglomerate'. This means 'wood' is 'soft' for all Agents in the hierarchy, for which 'wood' is a Resource. 'Metal' on the other hand, is 'cheap' for 'SBU #1' and 'expensive' for 'SBU #2'. This means 'metal' will only get a label 'cheap' or 'expensive' when a super-ordinate Agent like 'conglomerate' chooses the 'SBU #1' or 'SBU #2' view. In the other case 'metal' will be neither 'cheap' nor 'expensive'. When level 3 is not considered, the Properties 'soft', 'cheap' and 'expensive' disappear, as they are not relevant because the Resource they characterize is not represented anymore.

III. Methods

In their 2002 paper, Geerts and McCarthy applied a method for checking REA's completeness by mapping REA constructs to Constructs in Sowa's ontology. We think this method is based on implicit assumptions that we will try to reveal. The table that is the result of this completeness check was already represented in figure 1. In a second sub, we will indicate this same method can be used for addressing validity issues when it is slightly adapted. We do not address reliability and consistency issues in this paper.

Completeness issues

We suppose REA aims to provide a complete language to formulate business models. We think this attempt is structured as in Figures 2 and 7. Operations are registered in operational sentences. These sentences consist of business language items that fit in a business model. The business model determines the semantics for the business language items. A business model is formulated in a language. In our case, we use the REA language. This language has a meta-model that determines its semantics. The meta-model is formulated in a meta-language. Geerts and McCarthy (2002) chose Sowa's language as a language to describe REA semantics. In figure 1 we notice Sowa's ontology lacks semantics to describe REA semantic as for example the EconomicAgent and EconomicResource REA language constructs map both to the Object Sowa language construct. In section 4 we will provide a meta-model with full semantics for REA by extending Sowa's ontology. This extension will be based on the evaluation in section 2, but first we will provide a method that will help us with this operation.

Figure 8 provides us with a grid that represents our ontological completeness evaluation method. This method is based on the type-I and type-II error framework in statistics (Myers and Melcher 1969). The elementary framework is extended with the notions of ontological deficiencies provided by Wand and Weber (2002). Figure 8 shows how an ontology that has exactly one ontological construct for every grammatical construct is an ideal ontology. We also see “**Construct Overload** exists in a modeling grammar – that we call language – if one grammatical construct represents more than one ontological construct.” Ontological constructs are multidimensional categories in the meta-model of the language. “**Construct excess** exists in a modeling grammar when a grammatical construct is present that does not map into any ontological construct.” “**Construct redundancy** exists if more than one grammatical construct represents the same ontological construct.” “**Construct deficit** exists unless there is at least one modeling grammatical construct for each ontological construct.” (Wand and Weber 2002). The column “Several ontological constructs” reveals the grammar is too coarse grained or the ontology too fine grained. The row “Several grammatical constructs” reveals the grammar is too fine grained or the ontology too coarse grained. When we see REA as our modeling grammar that is checked against Sowa’s ontology, we see the second situation occurring. Sowa’s ontology is too coarse grained to specify the REA language completely. This is not surprising as Sowa’s ontology is a top-level ontology. Consequently, the Sowa’s meta-meta-model is underspecified and needs additional constraints to provide a meta-model for REA.

Validity issues

To get accepted, every theory needs validation. While acceptance is a social process, it is supported by methodological processes that provide validation. In this sub, we raise some questions and try to structure them. The terms introduced here come from Zeller and Carmines (Carmines and Zeller 1979) and were adapted to an information systems context.

Internal validity is the criterion that judges how well an ontology represents the domain it is supposed to represent.

External validity is the criterion that determines whether an ontology is capable of representing other domains than the domain it is supposed to represent. Internal validity can be split into several sub-criteria. The first one is face validity.

Face validity describes to what extent non-expert users think the ontology is capable of representing the domain it should represent. Face validity could be seen as an ontology’s main selling argument. **Content validity** determines whether experts think the ontology is capable of representing the domain it should represent. Non-expert users and experts base their judgment on their body of knowledge. We expect experts to found their judgment on a large body of relevant knowledge and an elaborate study. We do not expect this for non-expert users.

Criterion validity describes how well an ontology provides guidance for the classification of objects. This has to do with reliability as objects that were classified in the wrong category, give rise to inconsistencies in the (operational) model.

This **reliability** has to do with the extent to which classification of objects into categories is free of modelers bias (Jaquet

2006) and chance. If criterion validity is high, classification is well-defined and obvious, hence classification reliability is high. Criterion validity can be split into concurrent and predictive validity. **Concurrent validity** tells us how well objects can be mapped to their category thanks to the provided classification criteria. (e.g. the classification criteria for Sowa's ontology are the sets {Firstness, Secondness, Thirdness}, {Physical, Abstract} and {Continuant, Occurrent} while our classification criteria are discussed in section 2). **Predictive validity** describes how well the set of criteria – called constraints here – is able to discover unidentified ontological or grammatical constructs. (e.g. Mendeleev's table had a very high concurrent validity as it was able to classify all known elements and it also had a very high predictive validity as it predicted which other elements would be discovered in the future.)

A fifth internal validity sub-criterion is construct validity. **Construct validity** determines to what extent the constructs of an ontology map to constructs in another ontology. For instance, Geerts and McCarthy (2002) measured the construct validity of REA constructs by mapping them to constructs in Sowa's ontology. Construct validity however is more than just mapping constructs in one ontology to constructs in another ontology as the mapping needs to be preceded by an expectation that is based on prior research. For instance, Geerts and McCarthy expected that the REA business domain ontology would map to Sowa's top-level ontology, as the business domain might be expected to be part of the top-level domain.

The sixth and seventh internal validity sub-criterion are convergent validity and discriminator validity. **Convergent validity** tells us that two ontologies that model the same domain are expected to map easily. For instance, as both REA and E3 (Gordijn 2002) are said to be business ontologies, we expect them to map easily. In the worst case we expect them to be complementary, but this would mean none of them satisfies the criterion of maximal ontological completeness (MOC) (Green and Rosemann 2000). **Discriminator validity** determines that two ontologies that model a different, non-overlapping domain map poorly. For instance, a business ontology and an ontology for medical applications can be expected to map poorly. This situation may be expected to map with the construct minimal ontological overlap (MOO) (Green and Rosemann 2000) that prevents us from creating fat ontologies and stimulates us to create succinct domain ontologies that can be used as complementary modeling grammars if necessary.

The concurrent validity notion presented earlier can easily be checked with the framework introduced in figure 9. Figure 9 tells us that a valid ontology should be able to provide each single object in the real world domain with a symbol in the operational system. For our interpretation of REA this criterion is met because every Instance stands for one single object. Since each Instance needs to evoke a State, this framework is fit for empirical ontology testing.

IV. Applications

Refined REA ontology (Firstness)

Figure 10 represents our refined interpretation of the first row in figure 1. The newly introduced notions are written in italic. The first thing we notice is that the ‘Commitment’ construct has been removed from the ontology. This is because we see a Commitment as a Resource, hereby acknowledging Lampe’s critique (Lampe 2002). The reclassification of Commitment originates in the fact that we see a Commitment as a Continuant and not as an Occurrent, as Commitments can be uniquely identified over an interval of time. This means the value of a Commitment is not limited to the moment it is created. In fact a Commitment bears value between the moment it is created (an Increment Event for the beneficiary party and a Decrement Event for the debtor) and the moment it is consumed, which is the execution or fulfillment. From a legal point of view it is not relevant whether the Commitment is materialized or not, as oral agreements can have legal consequences in some business practices. In addition an indication of time is not needed for a unique identification of a Commitment.

A second thing we notice is the introduction of a Context-providing Occurrent Thing called Phenomenon. Phenomena are Occurrents that are not initiated by the Context-providing Agent in the model, nor by one of its subordinate Agents. In fact this ‘Phenomenon’ might be initiated by a Context-providing Agent of the Context-providing Agent for the system under examination. Since this Superior Context-providing Agent is outside the scope of the system under examination, the Events the Superior Context-providing Agent introduces are beyond the authority and control of the Agents in the system. Hence, we consider it better to distinguish these Occurrents from Event that are under the control and authority of at least one Agent in the system. These Phenomena can also be used for weather-accounting in agriculture or global warming accounting as these phenomena are indirectly controlled (or at least influenced) by the world population, which can be modeled as the most Superior Agent.

This Agent hierarchy approach yields a particular definition for a universe, as a universe is not a monolithic ‘out there’ world, but a collection of Agents. This universe is represented in figure 11. This means a systems universe is a collection of Agents, more precisely all known Agents except the Context-providing Agent for the system and the Agents that are ‘Inside party’ Agents for the system. In figure 11 the environment of the red ‘A’ Agent’s system is the collection of all black ‘A’s. Due to the hierarchic nature of this universe, red ‘A’ Agent in figure 11 can be decomposed into a collection of ‘Inside party’ Subordinate red ‘A’ Agents in figure 12, each of these red ‘A’s is a Context-providing Agent for its own system that is a sub-system of the system that is possessed by the red ‘A’ in figure 11.

Thanks to the introduction of Phenomena not all levels of detail in the description of a universe are exclusively populated with Agents. This situation is represented in figure 13. The Phenomena ‘P’ in this universe are Events that are under the control and authority of Agents that are Superior to the Context-providing Agents represented in this scheme. Hence, none of the Context-providing Agents represented can influence these Phenomena, and these Phenomena should be treated as equals of these Agents.

In the Property column, we introduced a distinction between Features and Attributes. Features are identical to all Agents in a systems Agent hierarchy, including the Context-providing Agent. Attributes can differ according to Agents involved. When the Agents involved are part of one Agent hierarchy, their knowledge is tacit for their Context-providing Agent. In this context, tacit means the Context-providing Agent can only access this information when he has authority to limit its view to the view of one of its Subordinate Agents. In this case the view of the Subordinate Agent and the limited view of the Superior Agent are identical. The difference is the Superior Agent has the ability to switch views. In reality this means the actions an employee takes as ‘organ’ of an enterprise are seen as actions of an enterprise, since the enterprise is the Context-providing Agent. The actions undertaken by an employee as independent person are tacit for the enterprise, since the Context-providing Agent is the employee.

We also introduce a distinction between Persistent and Temporal Properties. This distinction depends on the level of detail that is chosen. Persistent Properties are Properties that describe the State of an object; hence they are the properties of a Continuant. Temporal Properties are related with Transformation or Occurrents. In fact Occurrents know both Persistent and Temporal Properties, while Continuants only know Persistent Properties. We explain. The Properties that are not changed by the Transformation inherent at an Occurrent are Persistent for the Occurrent. This means Persistent Properties of an Occurrent do not differ between the Occurrent’s precondition and its post-condition. The set of Persistent Properties of an Occurrent is the set of Properties that is shared between the State preceding and the State following the Transformation inherent in the Occurrent. The Temporal Properties of an Occurrent vanish or occur during the Transformation inherent in the Occurrent. It is possible to have Temporal Properties that occur and vanish during a Transformation. This means they do not occur on the precondition State, neither do they occur on the post-condition State, but they occur on an intermediate State in the Process, in which a Process is a sequence of Occurrents. These Occurrents are called stages by Sowa (2007).

Type Hierarchies (Thirdness)

Type Hierarchies represent States, Types and their hierarchic relations. Type Hierarchies are constructed with two operations. We call them Aggregation and Abstraction. **Aggregation** is the operation that assigns multiple Physical Properties to one Abstract Property. Aggregation extends the domain of an Abstract Property with additional Physical

Properties, increasing choice between Physical Properties for one domain. This operation is represented in figure 14.

Abstraction is the operation that removes Abstract Properties from a sub-type to obtain a super-type. The collection of Abstract Properties owned by a super-type that is an Abstraction of its subtypes is the intersection of the collections of Abstract Properties owned by its subtypes. The collection of Physical Properties an Abstract Property has – called domain – is not affected by Abstraction. Abstraction is shown in figure 15.

One can model with States and Types. The difference is modeling with States is modeling as detailed as possible and modeling with Types means modeling on a higher level of abstraction. We will show the consequences of collapsing States into Types and hence sub-Types into super-Types. In figure 16 we see how Event Types indicated with E[0] are collapsed into one Event Type indicated with E[1] and how Resource Types indicated with R[0] are collapsed into Resource Types indicated with R[1]. The Types labeled with [1] are super-types for the Types labeled with [0]; hence the Types labeled with [0] are subtypes of the Types labeled with [1]. This abstraction yield figure 17, where R[1] represents 3 States or Types in figure 16. The Events that indicate the Transformations between those States or Types are collapsed into one unary relation, called E[1], on R[1]. This means the Events in E[1] do not change the classification of a Resource in R[1] if they are performed on this Resource in R[1] at this level of abstraction. Events that are not unary relations on a Resource change the Resource's classification at that given level of abstraction.

Instances vs. States

To describe an operational system, one needs more than the abstract framework presented earlier. In fact, we need to be able to identify objects uniquely and represent them in our system. This identification and representation is performed with Instances. We match Instances with their States as States are leafs of Type Hierarchies; therefore they are the most detailed kind of Type and provide a partition of all States in the system, for all objects in the domain and their Instances. Figure 18 shows how this operation is performed. It shows how Instance 'Box of nails # 345' is transformed into a collection of nails by the Event Instance 'Open box # 345'. The abstract operation is represented at Knowledge Level.

Property vs. Thing

Introducing Properties as Independent Items enables us to model Property functions explicitly, as seen in figure 19. Figure 20 represents a Property function for a Type. In figure 21 Property functions are modeled with one single 'P', while Things are modeled with one single 'T'. Figure 21 shows us the difference in the semantics of a conventional database on the left, and a database with REA semantics on the right. The difference is the notion 'querying' is replaced by 'associating'. Associating is an extension of the notion 'coupling' in the BWW ontology. We cite BWW. *"Two things are said to be coupled (or interact) if one thing acts on the other. Furthermore, those two things are said to share a*

binding mutual property (or relation); that is, they participate in a relation that ‘makes a difference’ to the thing.”

(Green and Rosemann 2000; Wand and Weber 1995) This notion is represented in the lower right corner of figure 21.

Grouping – called coupling in BWB – occurs when two or more Things (Instances, States or Types) share the same Property function. Not all of these grouped Things need to share the same Type. (e.g. A sports car and a tomato can be grouped as they share the Property function ‘color: red’.)

Typification occurs when two or more Things (Instances, States or Types) share the same set of Property functions. These Property functions can be modeled like in figure 19 or 20, depending on the chosen level of detail. (e.g. nail Instance #6454 and nail Instance #5672 share the same Type when the intersection of their States and the nail Type counts all Abstract Properties of the nail Type and at least one Physical Property from the nail Type property domain for each Abstract Property in the intersection.)

The difference between these semantics and the semantics of a conventional database containing records is that these Instances, States and Types act as if they were all represented in one big table, or as if tables were dynamic entities that could be split and merged without limitations. In conventional databases each Instance has a record with a collection of unique property functions. This means the same object property red can be represented by the property functions ‘color: red’, ‘Color: red’, ‘color: Red’, ‘Color: Red’ ... without the system ever noticing the synonymy. This convention makes homonymy and synonymy hard to control in a conventional database setting. In the REA setting, Property functions are unique in the system, just like Things. This makes homonymy and synonymy easily controllable.

Figure 21 also represents how querying is actually Typification and Grouping at the same time. When we ‘Select * from T’, we choose a table and perform Typification by choosing table T. Table T is represented by a red rectangle. When we perform ‘where P is P’ we select a Property function, and represent all T from table T that possess this Property function. That there are other records in other tables containing the same Property function cannot be discovered by a conventional querying method as the table has to be chosen before the Property function is chosen. With sheer Grouping discovering Instances that map with the same Property function without sharing the same Type can be performed.

V. Conclusion and Future Work.

In this paper, we refined Geerts’ and McCarthy’s (2002) ontological evaluation that maps REA with Sowa’s ontology. We have also established that REA maps at least partially with the Bung-Wand-Weber ontology. This supports the hypothesis that REA is a valid business ontology. We also provided arguments for representing Properties as Independent

items of Firstness. Furthermore we proposed a method for evaluating an ontology's granularity. In addition, we made the difference between Instances, States and Types clear. Instances bear an object's identity. Types describe the objects that can be represented by Instances, and States are a special kind of Types in a way that every Instance has exactly one State, as States are leafs of Type Hierarchies. Additionally, we established that Type Hierarchies can be constructed with Abstraction and Aggregation. Finally, we revealed how data can be extracted from a REA database by Typification and Grouping.

In the future we will provide an algebra for Secondness that will enable us to represent items of Secondness in an elegant way. This algebra should be able to represent all items of Secondness like Duality, Participation, StockFlow, Custody, Linkage, Association ... and semantics like hierarchy, sequence and concurrency. This algebra should also enable us to collapse fine grained process representations like Petri-nets into coarse grained value perspectives like E3-value models. Eventually, we will elaborate the methodological issues introduced here in brief and eliminate the construct overload indicated by the green ovals in figure 1.

¹ The category of all independent items.(Sowa 2000)

² www.jfsowa.com (Sowa 2007)

³ If one can say x has y, neither x nor y is independent.

⁴ Synthetic and analytic are words that were denoted by the 'Wiener Kreis'. Analytic predicates are non-trivial predicates that cannot be verified nor falsified, while Synthetic predicates can.

⁵ First In First Out.

⁶ Last In First Out.

⁷ A non-relativistic approach where time is constant. This is our traditional notion of time, as we consider the duration of a second determined.

⁸ In this case, State is seen as a snapshot of an object during its lifetime. The snapshot is a description of all the objects Properties. This yields the same 'State' notion as the one introduced earlier.

⁹ States are a special kind of Types.

¹⁰ Create, Read, Update, and Delete. (Martin 1989)

- Carmines, E. G., and R. A. Zeller. 1979. *Reliability and validity assessment*. Beverly Hills ; London: Sage Publications.
- Dunn, C. L., J. O. Cherrington, and A. S. Hollander. 2005. *Enterprise information systems : a pattern-based approach*. 3rd ed. Boston: McGraw-Hill/Irwin.
- Geerts, G. L., and W. E. McCarthy. 2002. An ontological analysis of the economic primitives of the extended-REA enterprise information architecture. *International Journal of Accounting Information Systems* 3 (1):1-16.
- Gordijn, J. 2002. Value-based requirements Engineering: Exploring innovative e-commerce ideas, Exact Sciences, Free University of Amsterdam, Amsterdam.
- Green, P., and M. Rosemann. 2000. Integrated process modeling: An ontological evaluation. *Information Systems* 25 (2):73-87.
- Hruby, P. 2006. *Model-driven design using business patterns*. Berlin: Springer.
- ISO/IEC. 2006. ISO/IEC 15944-4: 2006 Information technology - Business agreement semantic descriptive techniques -- Part 4: Open-ed business transaction ontology
- Jaquet, M. 2006. A Property Driven Approach towards Describing Semantics of REA Entities. *Proceedings of the 2nd International REA Technology Workshop*.
- Lampe, J. C. 2002. Discussion of an ontological analysis of the economic primitives of the extended-REA enterprise information architecture. *International Journal of Accounting Information Systems* 3 (1):17-34.
- Martin, J. 1989. *Information engineering*. 3 vols. Englewood Cliffs, N.J.: Prentice Hall.
- McCarthy, W. E. 1982. The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. *Accounting Review* 57 (3):554.
- Myers, B. L., and A. J. Melcher. 1969. ON THE CHOICE OF RISK LEVELS IN MANAGERIAL DECISION-MAKING. *Management Science* 16 (2):B-31-B-39.
- Nickles, T. 1980a. *Scientific discovery, case studies*. Dordrecht, Holland ; Boston Hingham, MA: D. Reidel Pub. Co. ; sold and distributed in the U.S.A. and Canada by Kluwer Boston.
- . 1980b. *Scientific discovery, logic, and rationality*. Dordrecht, Holland ; Boston Hingham, MA: D. Reidel Pub. Co. ; sold and distributed in the U.S.A. and Canada by Kluwer Boston.
- Nonaka, I. 1991. The Knowledge-Creating Company. *Harvard Business Review* 69 (6):96-104.
- Nonaka, I., K. Umemoto, and D. Senoo. 1996. From information processing to knowledge creation: A Paradigm shift in business management. *Technology in Society* 18 (2):203-218.
- Poels, G., A. Maes, F. Gailly, and R. Paemeleire. 2007. The pragmatic quality of Resources-Events-Agents diagrams: an experimental evaluation. In *Accepted for Information Systems Journal*.
- Sowa, J. F. 2000. *Knowledge representation : logical, philosophical, and computational foundations*. Pacific Grove: Brooks/Cole.
- . 2007. *Top-Level Categories*. <http://www.jfsowa.com/ontology/toplevel.htm>, 08/28/2001 18:36:40 2007 [cited 1-25 2007]. Available from <http://www.jfsowa.com/ontology/toplevel.htm>.
- Wand, Y., and R. Weber. 1995. On the deep structure of information systems. *Information Systems Journal* (5):203-223.
- . 2002. Research Commentary: Information Systems and Conceptual Modeling--A Research Agenda. *Information Systems Research* 13 (4):363-376.

	Physical Categories		Abstract Categories	
	<i>Continuant</i>	<i>Occurrent</i>	<i>Continuant</i>	<i>Occurrent</i>
	<u>Object</u>	<u>Process</u>	<u>Schema</u>	<u>Script</u>
<i>1st</i> <i>(Independent)</i>	<i>EconomicAgent (A)</i> <i>EconomicResource (R)</i>	<i>EconomicEvent (E)</i> <i>Commitment (C)</i>	<i>AgentType (AT)</i> <i>ResourceType (RT)</i>	<i>EventType (ET)</i> <i>CommitmentType (CT)</i>
	<u>Juncture</u>	<u>Participation</u>	<u>Description</u>	<u>History</u>
<i>2nd</i> <i>(Relative)</i>	<i>Association (A-A)</i> <i>Custody (A-R)</i> <i>Linkage (R-R)</i>	<i>StockFlow (E-R)</i> <i>Duality (E-E)</i> <i>Accountability (E-A)</i> <i>Executes (C-E)</i> <i>Involvement (C-A)</i> <i>Reserved (C-R)</i> <i>Reciprocal (C-C)</i>	<i>Typification (A-AT)</i> <i>(R-RT)</i> <i>Characterization (AT-AT)</i> <i>(AT-RT)</i> <i>(RT-RT)</i>	<i>Typification (E-ET)</i> <i>(C-CT)</i> <i>Scenario (ET-RT)</i> <i>(ET-ET)</i> <i>(ET-AT)</i> <i>(CT-ET)</i> <i>(CT-AT)</i> <i>(CT-RT)</i> <i>(CT-CT)</i>
	<u>Structure</u>	<u>Situation</u>	<u>Reason</u>	<u>Purpose</u>
<i>3rd</i> <i>(Mediating)</i>	<i>Responsibility</i> <i>Partnering</i> <i>Configuration</i>	<i>Exchange</i> <i>Conversion</i> <i>Contracting</i> <i>Scheduling</i>	<i>Segmentation</i> <i>Policy</i> <i>Substitutability</i> <i>Complementarity</i> <i>Configuration</i>	<i>Standardization</i> <i>Policy</i> <i>Strategy</i>

Legend:

Red ovals: Construct overload for 'Object', 'Process', 'Schema' and 'Script' constructs in Sowa's ontology for REA independent constructs.

Red dashed ovals: Construct overload for 'Juncture', 'Participation', 'Description' and 'History' constructs in Sowa's ontology for REA relative constructs.

Green dashed ovals: construct overload for REA constructs 'Typification', 'Characterization' and 'Scenario'.

Blue oval: Construct overload for 'Structure', 'Situation', 'Reason' and 'Purpose' constructs in Sowa's ontology for REA mediating constructs.

Figure 1. Ontological clarity issues: Construct overload (for Sowa's ontology constructs) in REA (Source: Geerts and McCarthy 2002)

Hierarchic layer	Choice
Meta-meta-model	Sowa's ontology
Meta-language	Sowa's 'language'
Meta-model	REA ontology
Language	REA language
Business model	REA enterprise model
Business language	Enterprise specific language
Operational sentence	Information system entries

Figure 2. An ontology hierarchy.

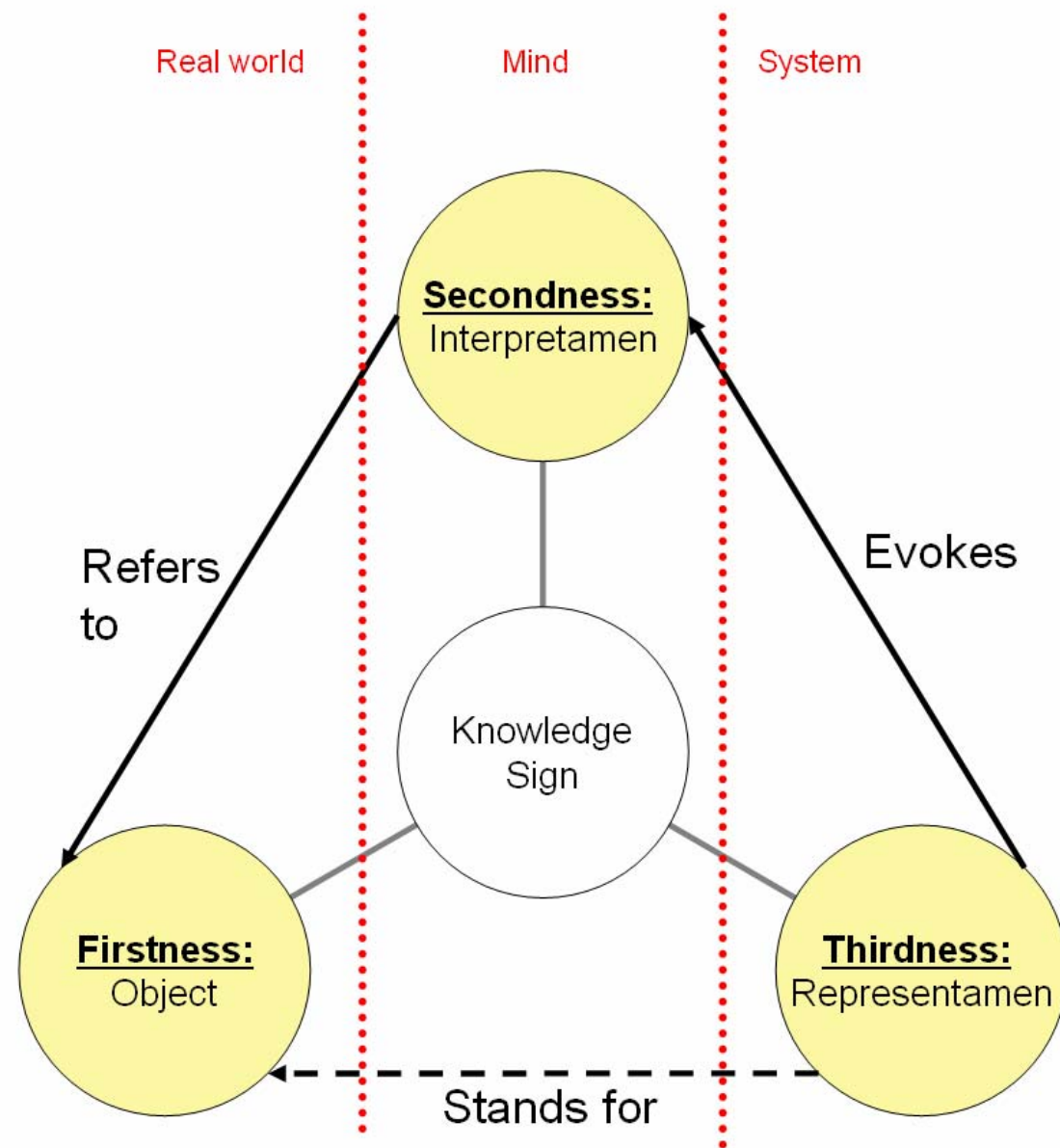


Figure 3. Peirce's semantic triangle.

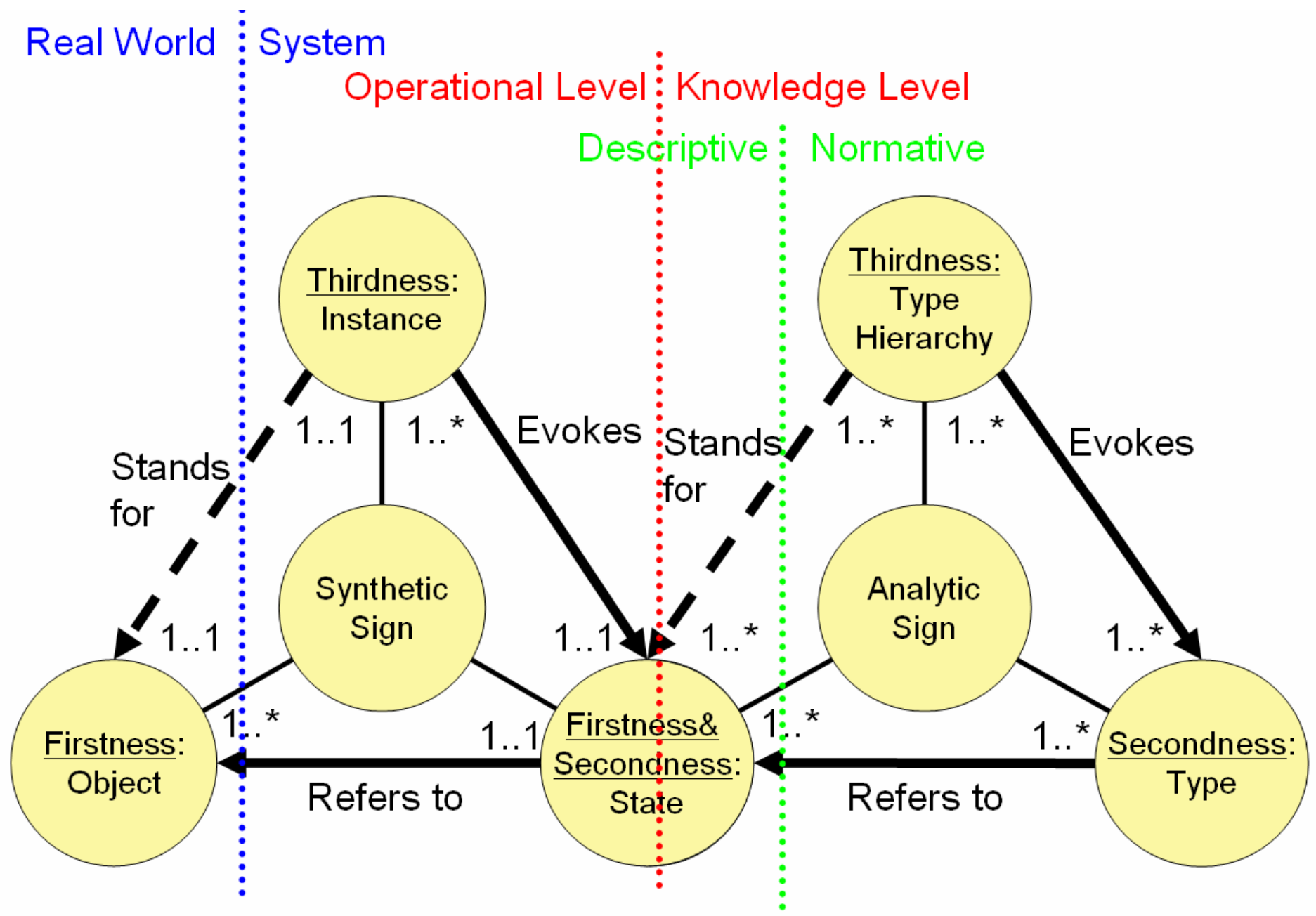


Figure 4. Double semantic triangle.

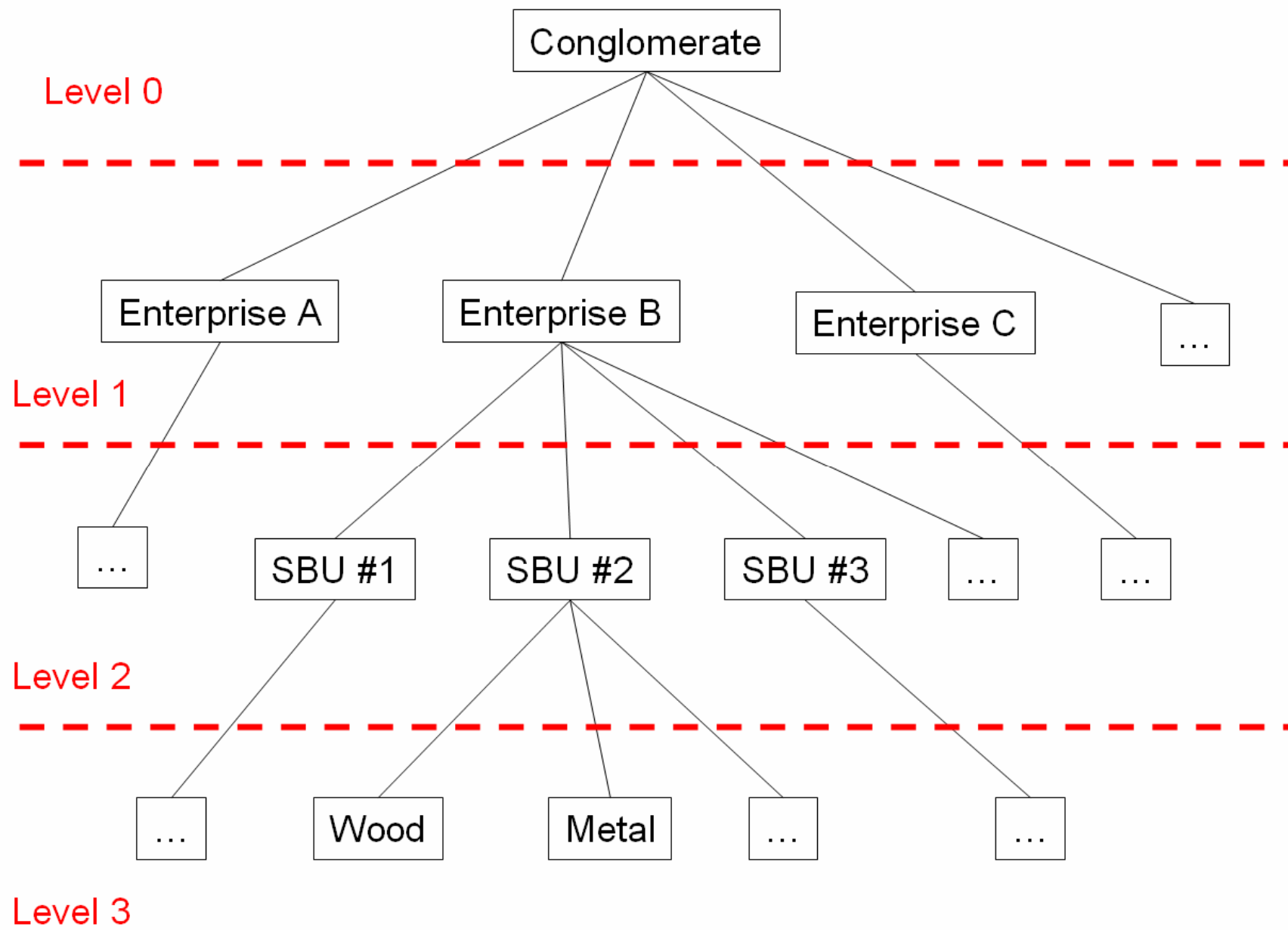


Figure 5. A control hierarchy.

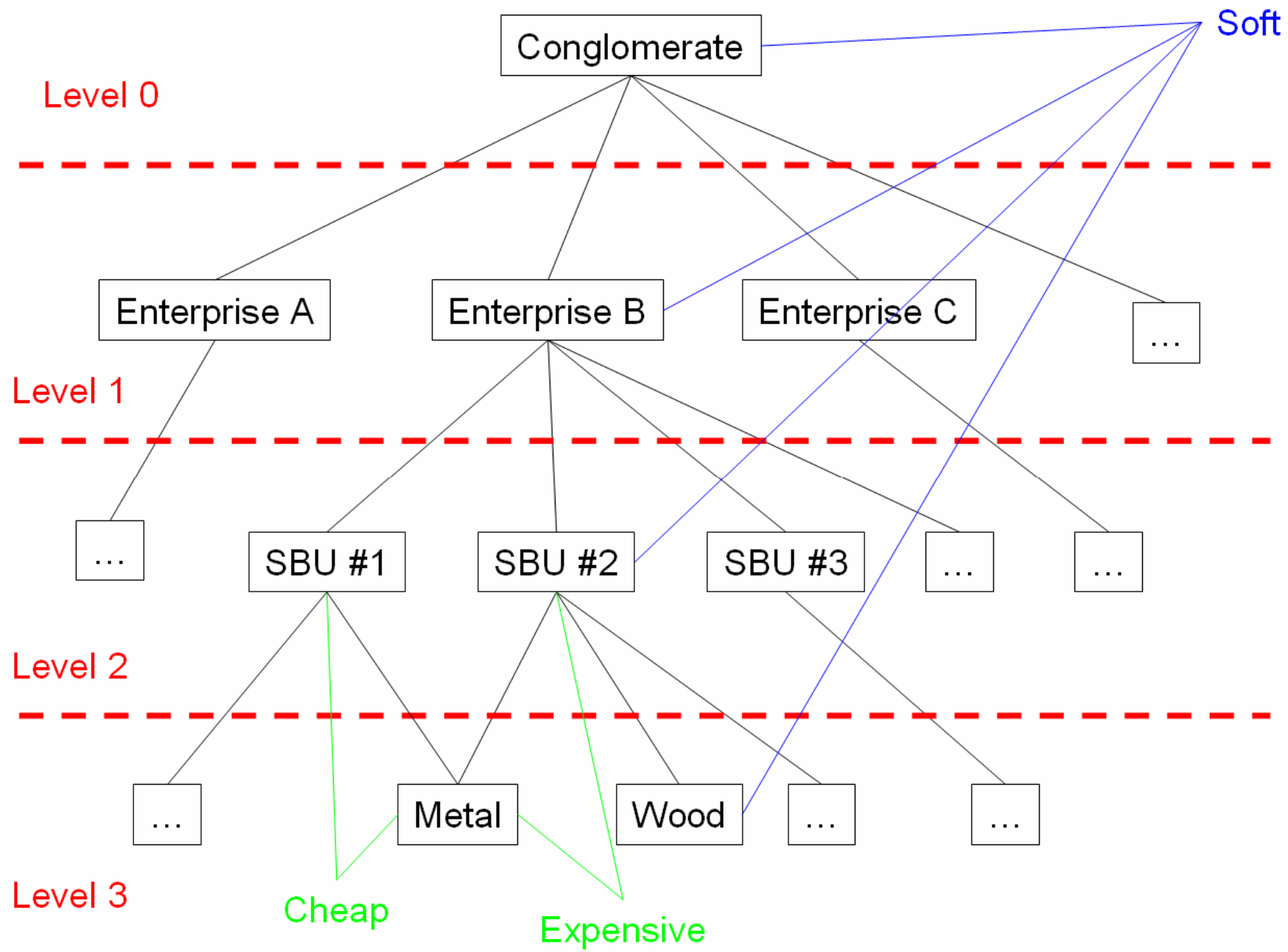


Figure 6. Context-providing and Context-slave Properties.

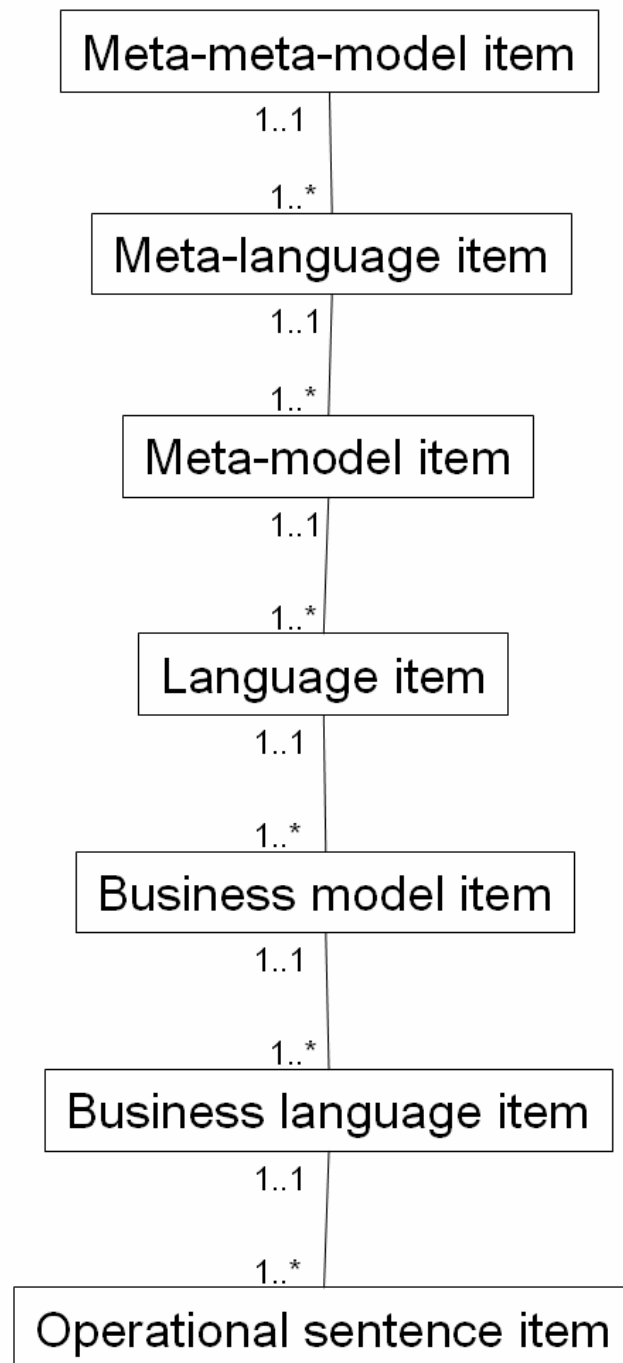


Figure 7. A model-language hierarchy.

Ontology Evaluation	Single ontological construct	Several ontological constructs	Not an ontological construct
Single grammatical construct	Ok	<u>Lack of ontological clarity:</u> Construct overload	<u>Lack of ontological clarity:</u> Construct excess
Several grammatical constructs	<u>Lack of ontological clarity:</u> Construct redundancy	<i>Decompose and check single grammatical and ontological constructs</i>	<i>Decompose and check single grammatical constructs</i>
Not a grammatical construct	<u>Ontological Incompleteness:</u> Construct deficit	<i>Decompose and check single ontological constructs</i>	Ok

Figure 8. Construct validity check.

Empiric Ontology Evaluation	Single Object in the real world domain	Several Objects in the real world domain	Not a Object in the real world domain
Single Symbol in the business language	Ok	<u>Lack of ontological clarity:</u> Construct overload	<u>Lack of ontological clarity:</u> Construct excess
Several Symbols in the business language	<u>Lack of ontological clarity:</u> Construct redundancy	<i>Decompose and check single grammatical constructs and real world objects</i>	<i>Decompose and check single grammatical constructs</i>
Not an Symbol in the business language	<u>Ontological Incompleteness:</u> Construct deficit	<i>Decompose and check single real world objects</i>	Ok

Figure 9. Concurrent validity check.

REA Firstness constraints			Thing	Property
Physical	Continuant	Context-providing	Agent Instance	<i>Persistent Feature</i>
		Context-slave	Resource Instance	<i>Persistent Attribute</i>
	Occurrent	Context-providing	<i>Phenomenon Instance</i>	<i>Temporal Feature</i>
		Context-slave	Event Instance	<i>Temporal Attribute</i>
Abstract	Continuant	Context-providing	Agent Type	<i>Persistent Feature Type</i>
		Context-slave	Resource Type	<i>Persistent Attribute Type</i>
	Occurrent	Context-providing	<i>Phenomenon Type</i>	<i>Temporal Feature Type</i>
		Context-slave	Event Type	<i>Temporal Attribute Type</i>

Figure 10. REA Firstness Table.

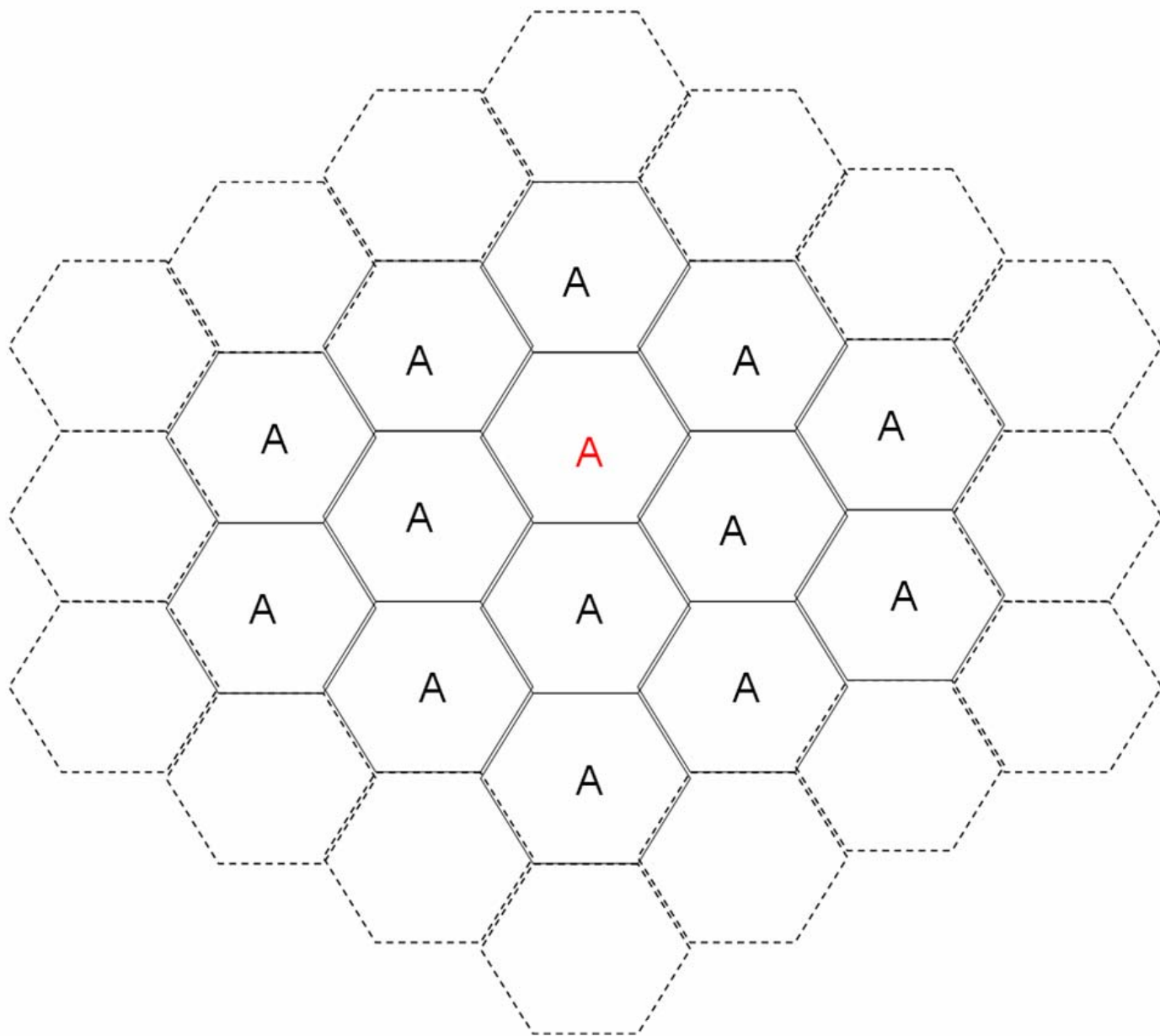


Figure 11. The REA universe.

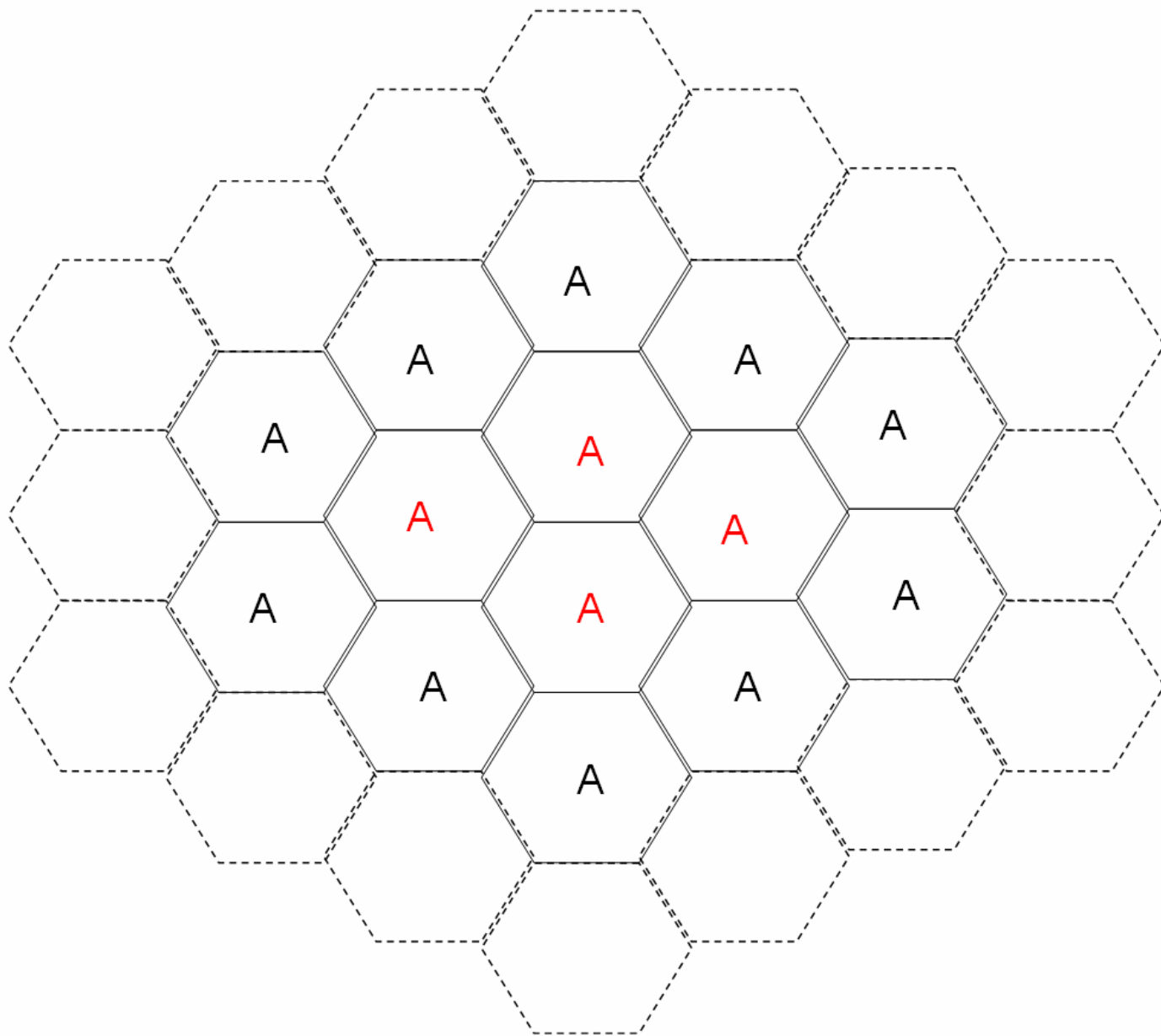


Figure 12. A decomposed A and its environment.

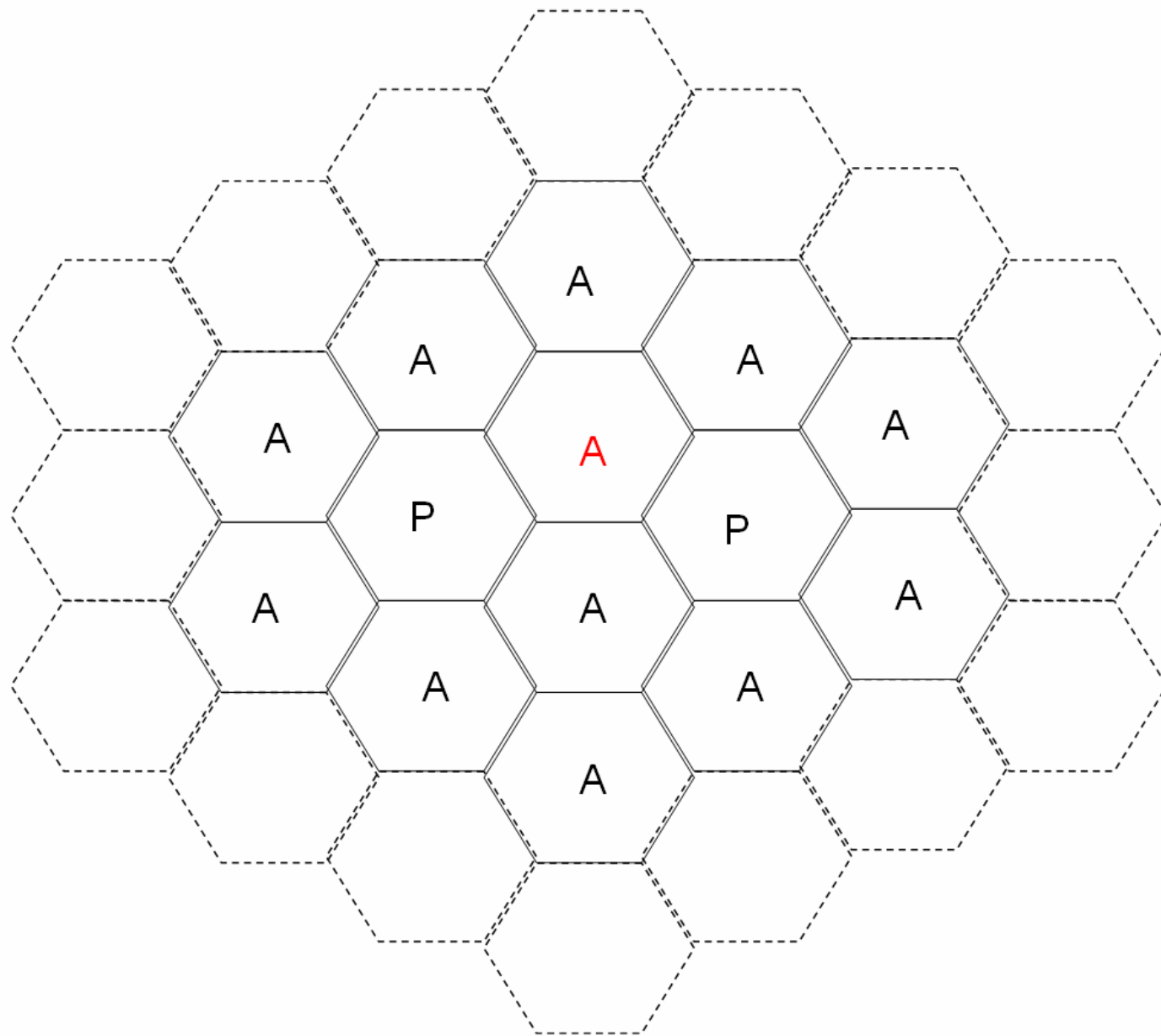


Figure 13. The REAP universe.

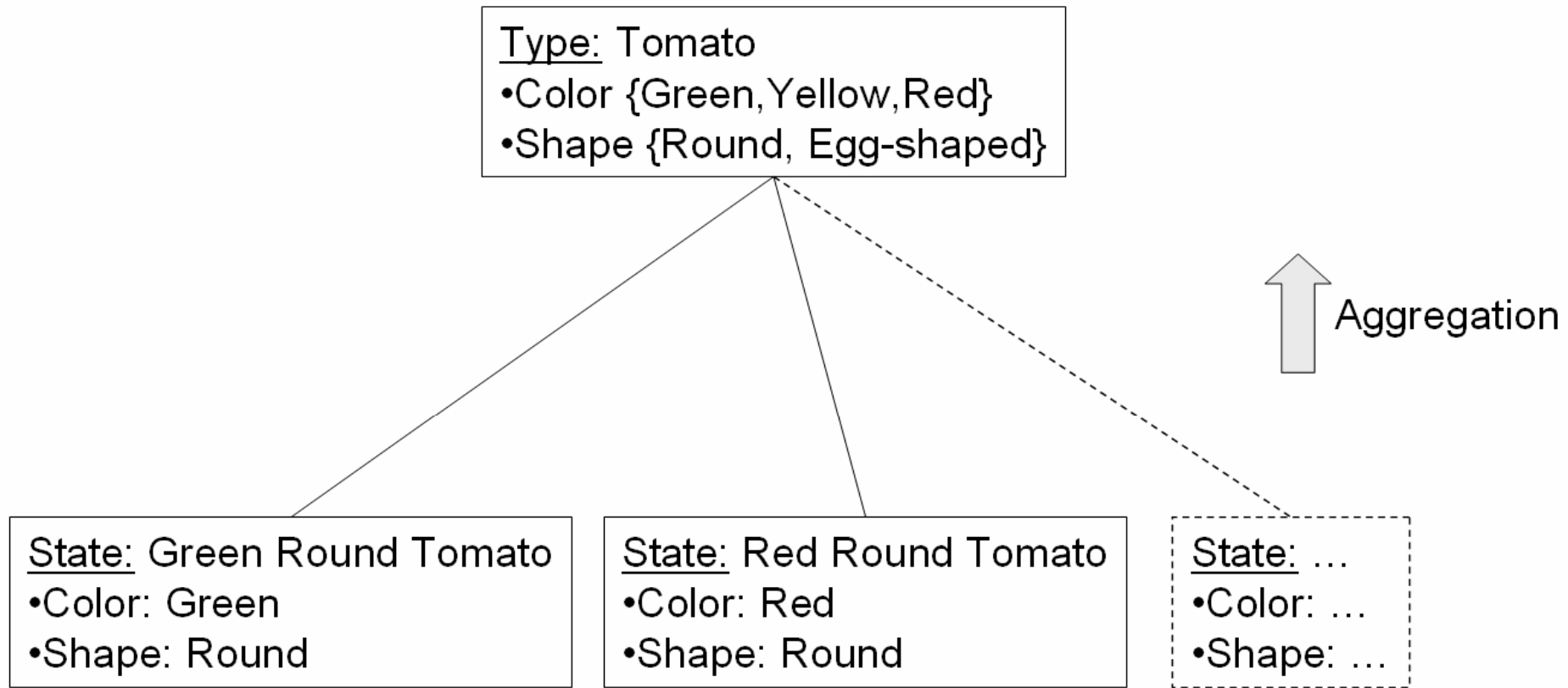


Figure 14. Aggregation.

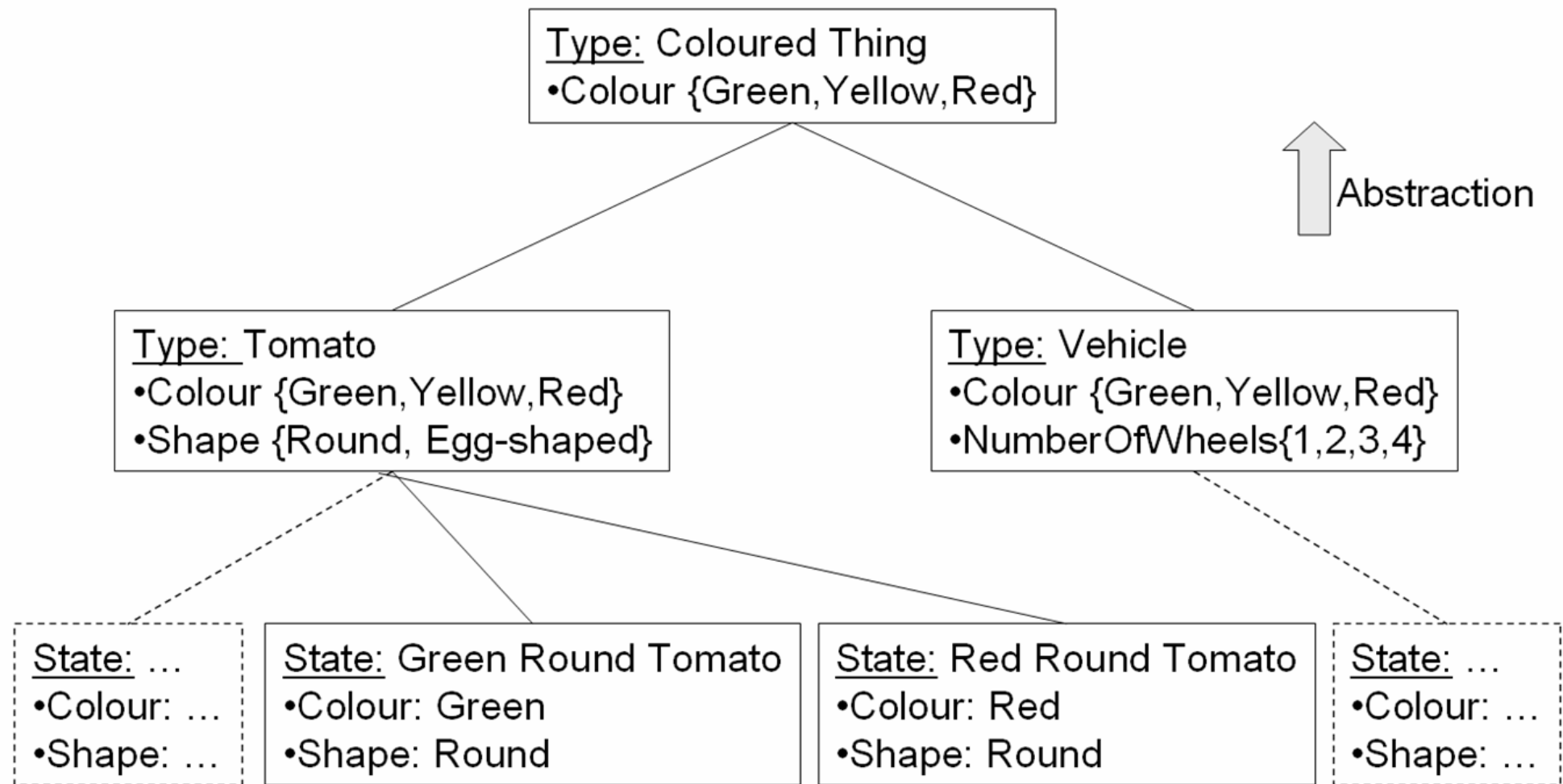


Figure 15. Abstraction.

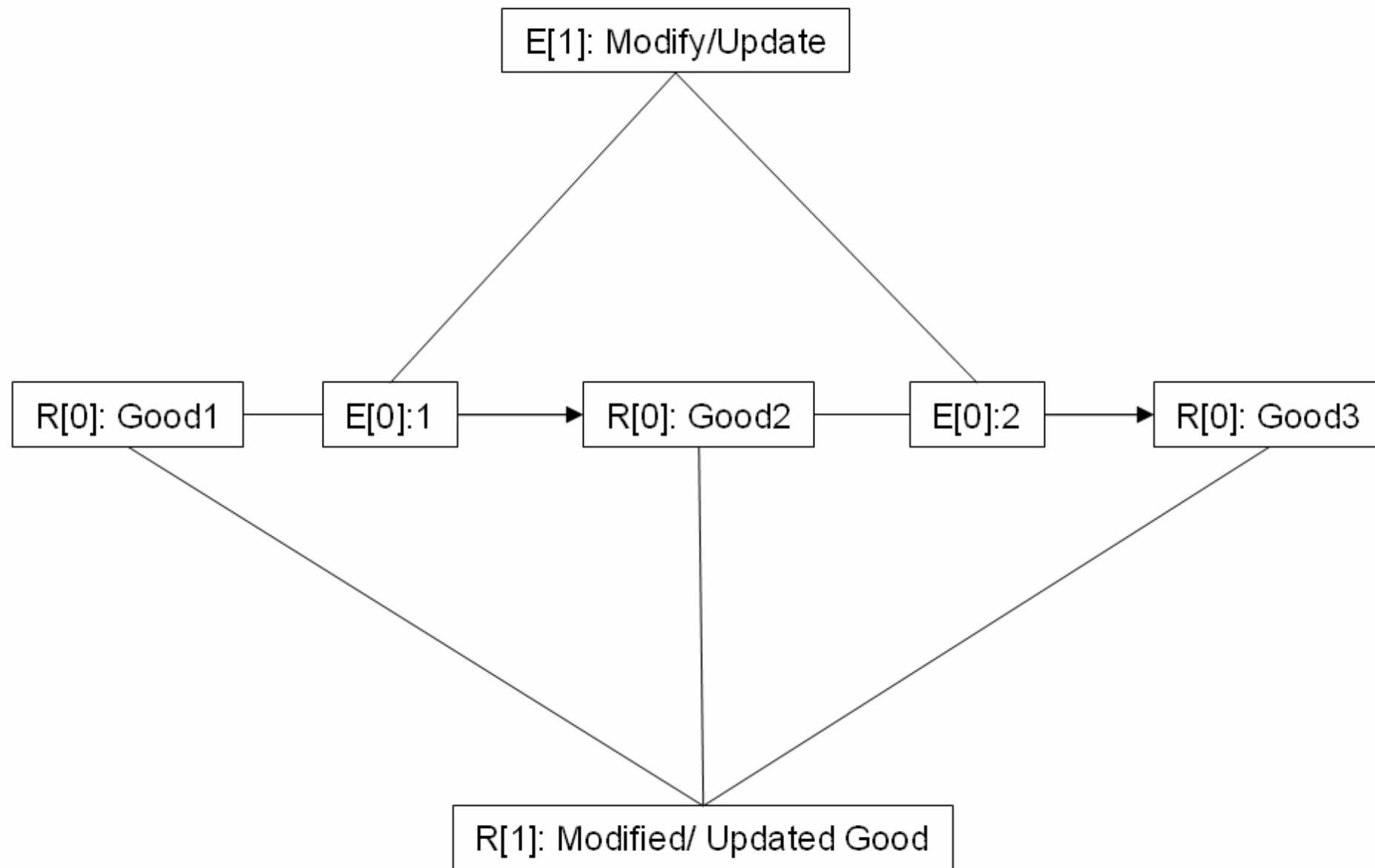


Figure 16. Collapsing a Process.

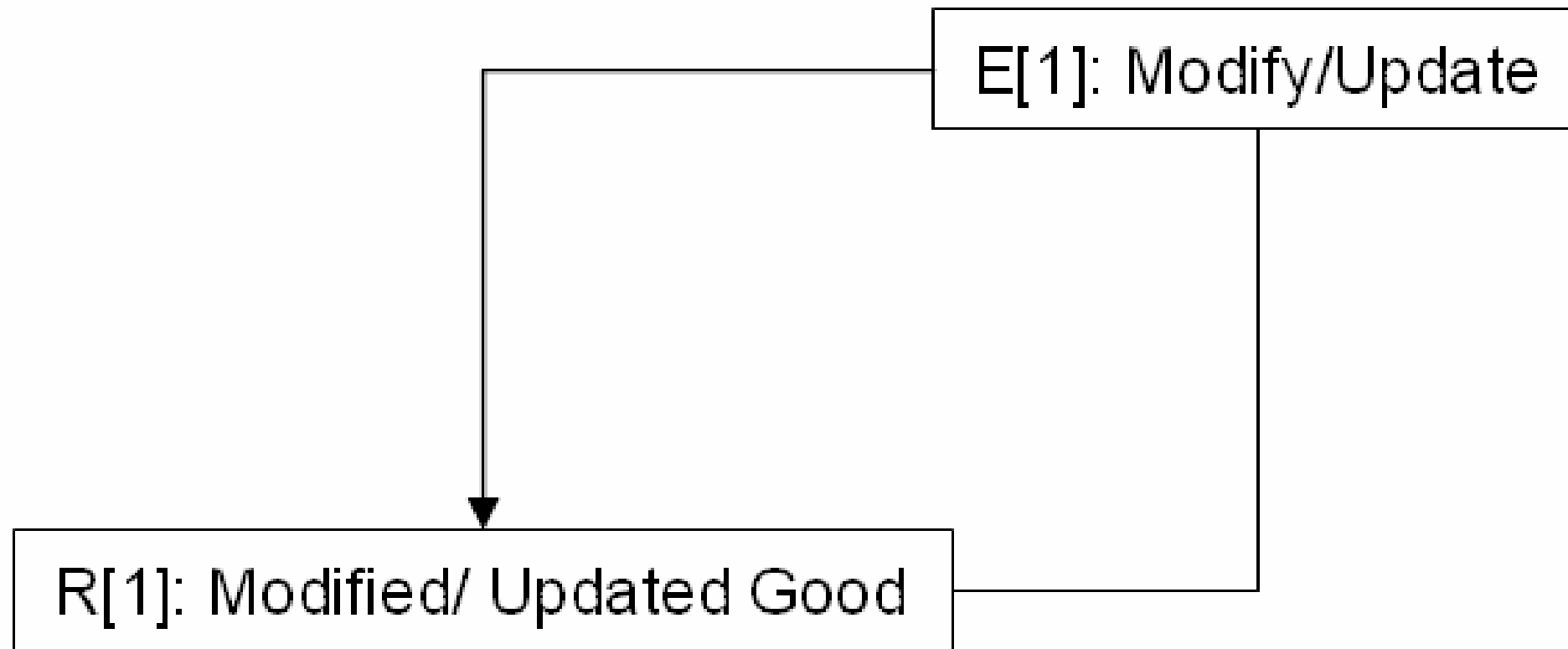


Figure 17. A collapsed Process.

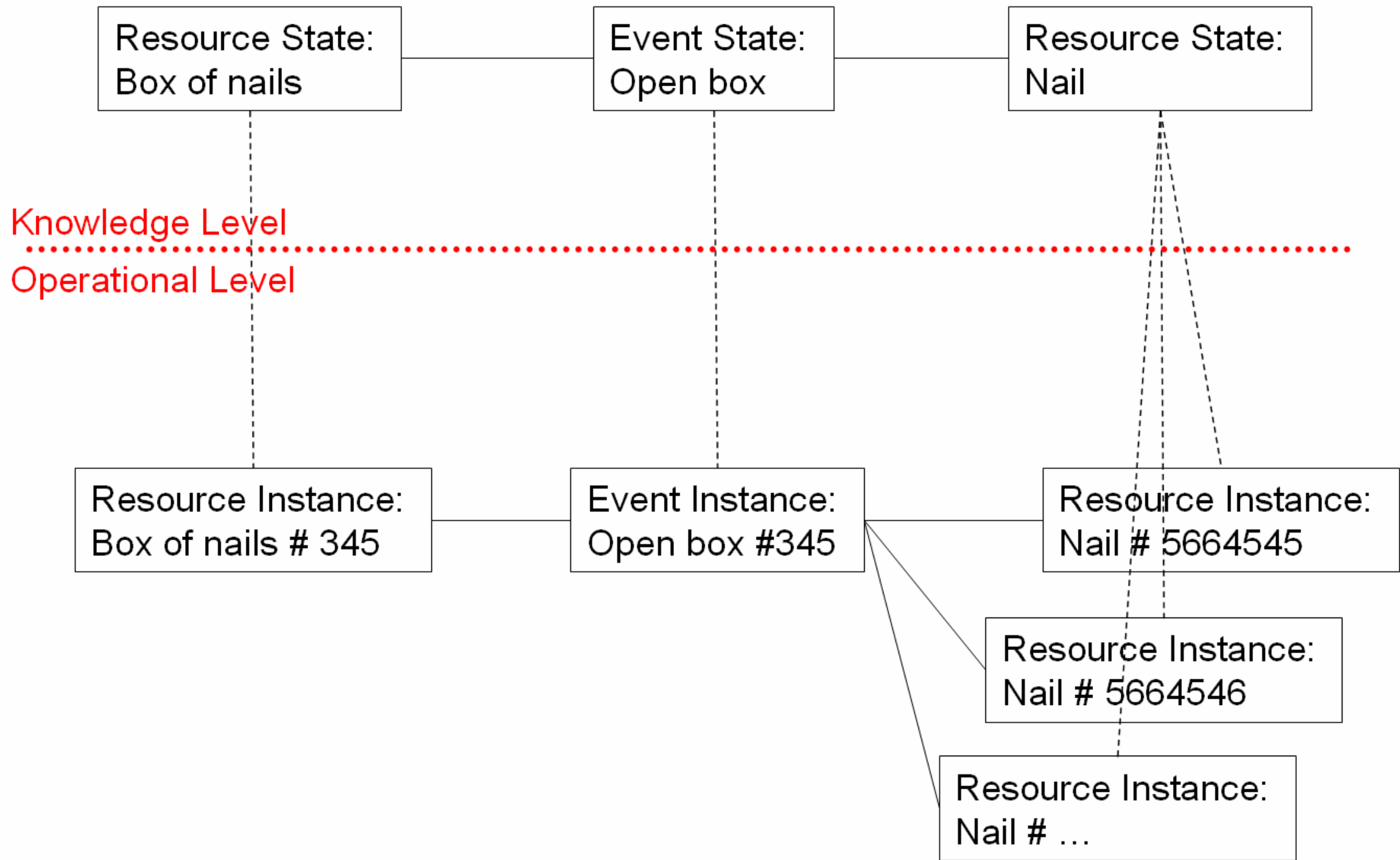


Figure 18. Mapping Instances with their States.

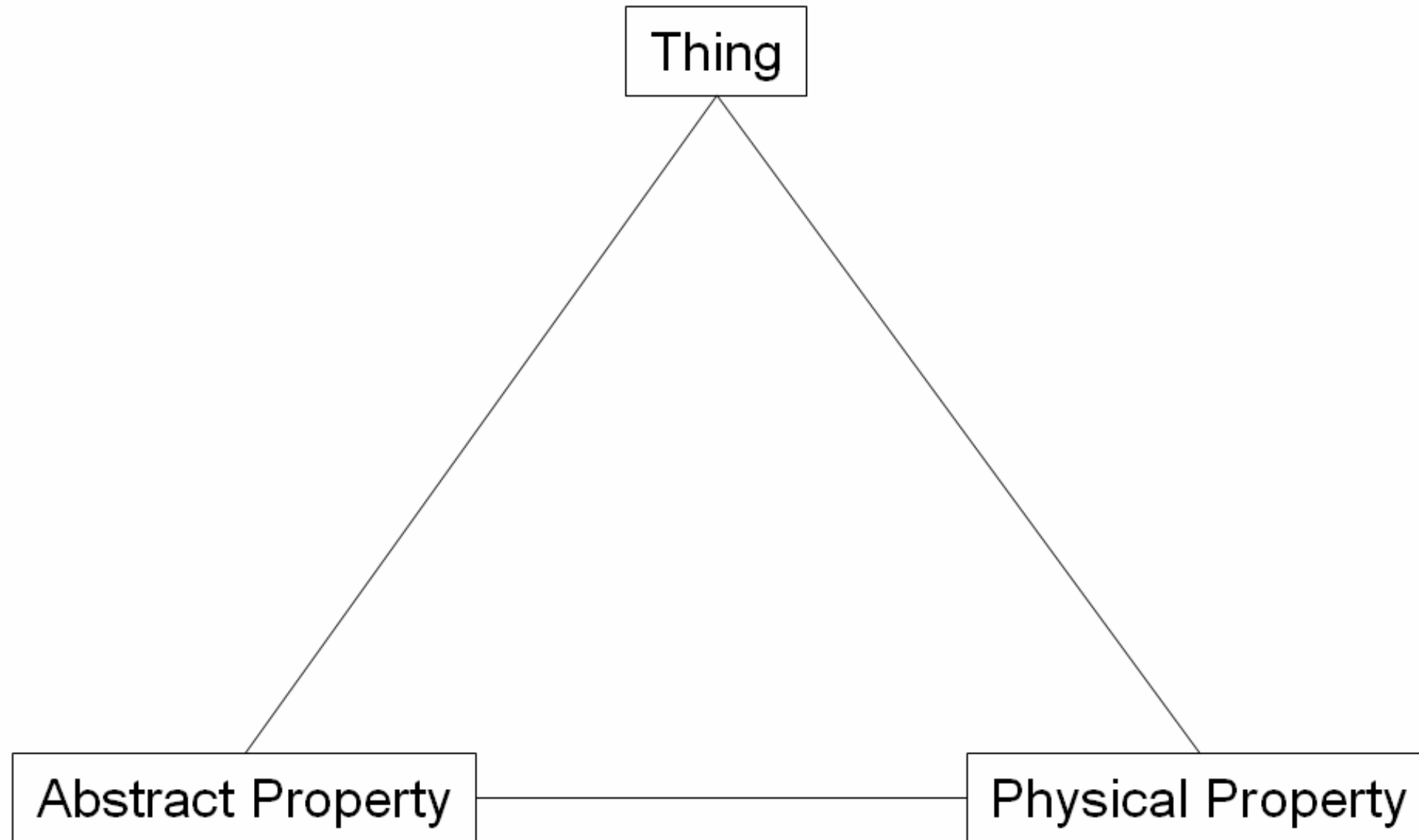


Figure 19. A Property function.

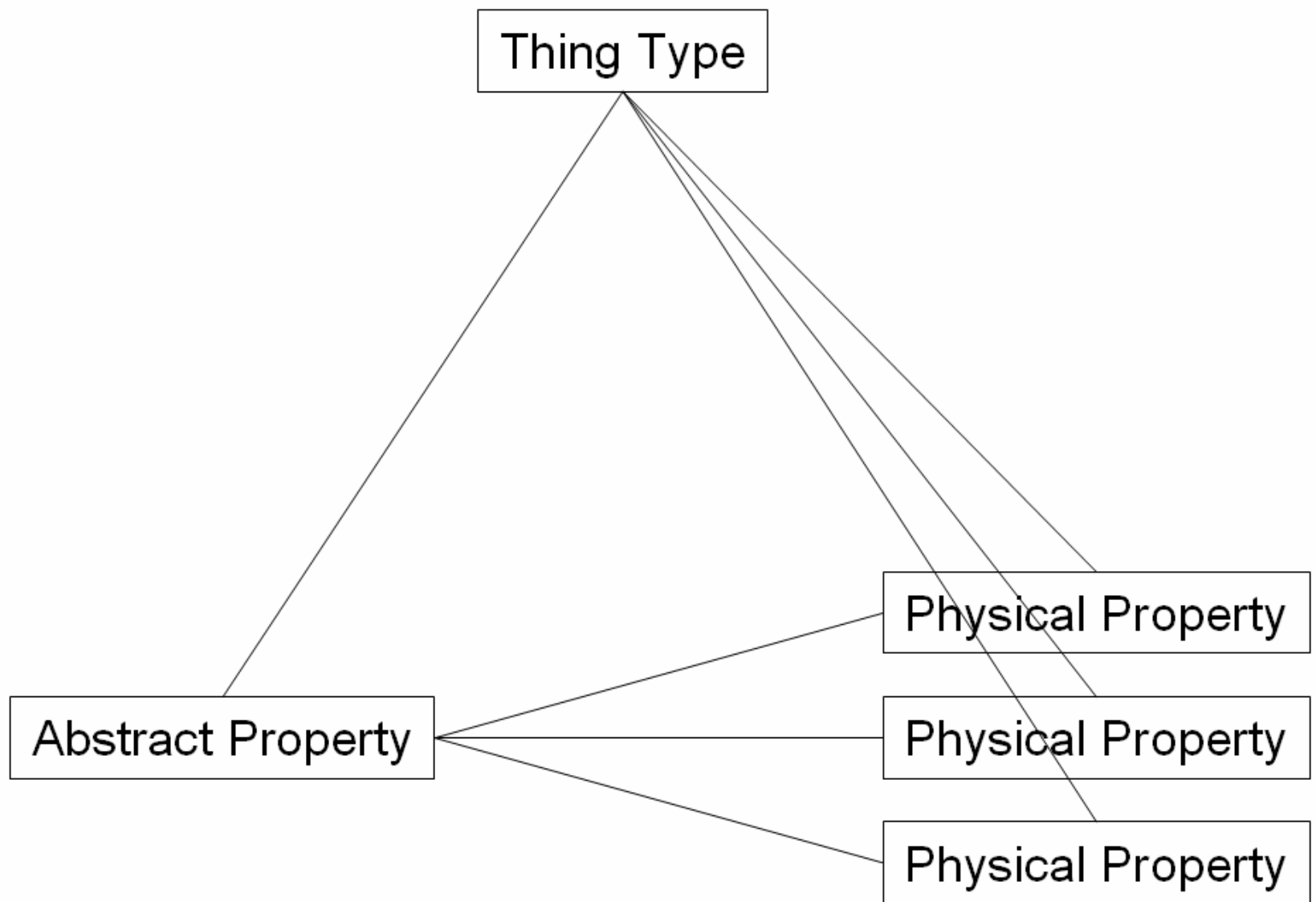
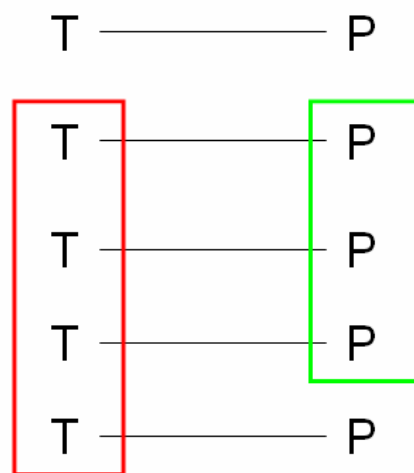


Figure 20. A Property function for a Type, generated by generalization.

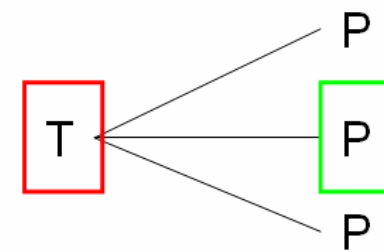
Querying



Select T from T where P is P

Associating

Typification



Grouping

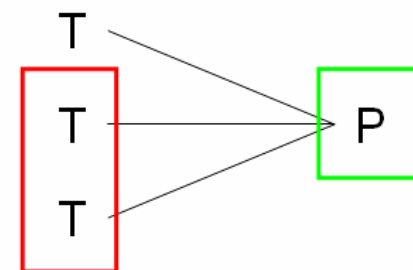


Figure 21. Database record vs. Instance and State.