Describing Knowledge Representation Schemes: a Formal Account

Giorgos Flouris, Dimitris Plexousakis and Grigoris Antoniou
{fgeo, dp, antoniou}@ics.forth.gr
Institute of Computer Science
Foundation for Research and Technology - Hellas
Science and Technology Park of Crete P.O.Box 1385
GR 711 10 Heraklion, Crete, Greece
Tel: +30 2810 391600
Fax: +30 2810 391601

TR-320, April 2003,
ICS-FORTH

Abstract. The representation and manipulation of knowledge has been drawing a great deal of attention since the early days of computer science, resulting in the introduction of numerous different Knowledge Representation schemes (KR-schemes). Given the great variety of such available schemes, it would be desirable to have a uniform way of treating them. In this report, we propose a formal, unifying framework for dealing with KR-schemes. This framework is useful for a formal definition of the reduction of one Knowledge Representation model to another. Based on the notion of reduction, a formal method of comparing KR-schemes in terms of expressive power is proposed and some of its properties are explored.

Introduction

Virtually every application in computer science uses some kind of knowledge, usually referred to as the data of the application. This fact makes the representation and manipulation of knowledge and data a very important consideration. In some cases, knowledge is represented as raw data; more often it is stored using complex structures such as various forms of logic, active rules, semantic graphs or other constructs which also reflect the relationship between the data. The interest being drawn in this problem has resulted in the introduction of several different approaches, each one driven by a different need and placed on a different context. Selecting the proper KR-scheme (or building a new one from scratch) for any given application is a difficult problem, whose solution requires a general method of comparison of KR-schemes.

The vast variety of approaches has led to the impression that there can be no uniform way of dealing with KR-schemes. For the same reason, there have not been any general comparison attempts, except in a specific context or under the light of some specific application. The absence of comparison methods is mainly due to the absence of a general model of representation; one cannot compare two schemes unless they are both expressed in a uniform way. Logic is a very general model that can be used for knowledge representation (KR), so most comparison attempts use some form of logic as the "common ground" of comparison, as pointed out in [4].

In this report we will address the problem of comparing KR-schemes in full generality, not restricting ourselves in logic and without setting any constraints on the type of the two KR-schemes under question. The comparison will be made in terms of expressive power; we will not concern ourselves in other metrics such as intuitive correctness, processing speed, storage space required etc. Our approach can be divided in two main parts. Firstly, we will show that all KR-schemes share some common properties and exploit these properties in order to define a formal, general and uniform way of describing what a KR-scheme is. Secondly, given such a model, we will introduce a formal way of expressing our intuitive notion of a KR-scheme being more expressive than the other. Using these two tools, we will prove some general results regarding KR-schemes.

Defining a KR-scheme

Informal Definitions

Before attempting to formally define a KR-scheme, we will try to informally describe it. There have only been a few such definitions in the literature. One of the most complete attempts to define KR and other relevant terms is made in [9]. In that book, KR is defined as *the field of study within AI concerned with using formal symbols to represent a collection of propositions believed by some putative agent*. In this context, the term *agent* refers to the entity that uses the system. We will usually prefer the term *user* instead.

Using this definition for KR, we could say that a KR-scheme is the set of symbols that are used to represent the collection of propositions (knowledge) of the agent (user). This is not enough though; according to [6], "one can characterize a representational language as one which has (or can be given) a semantic theory". Indeed, a set of symbols is nothing more than a set of symbols. It does not contain any type of knowledge, unless we explicitly assign each symbol to some type of knowledge. For example the symbol "6" obviously refers to number 6. The same goes for the symbols "six" or "VI" in other contexts. However, the symbol "6" means nothing by itself unless this assignment is made, for pretty much the same reasons that the symbol "six" means nothing to a person unless he speaks English (so he knows that "six" is the English word for 6). Thus, a KR-scheme (representational language) is *a formal description of the symbols used for the representation, as well as their real world semantics*.

A KB is used to store the knowledge of the agent. It can be viewed as an instance of a KR-scheme, in the sense that it uses the symbols and the semantics of the given KR-scheme. In this context we can say that a KB is to a KR-scheme what a DB is to its schema. In this report, by the term KB, we will refer to *the symbols representing the knowledge currently stored in the KR-scheme*. As will be seen later, a schema also contains updates, queries and other structures, which are not part of a KB.

The term *user* of a KB will refer generally to *the entity that uses the KB*. This implies that a user is not necessarily a human being; a robot or a sophisticated software that uses a Knowledge Base (KB) to draw information is classified as a user; a data mining software uncovering patterns from a database (DB) is also a user; a web crawler is a user of the knowledge in the Internet; and a mediator software is a user of the KBs it accesses.

Uncovering Common Properties

Representing the Knowledge Base

Despite the vast variety of KR-schemes currently available, there exists a set of common properties found in all such schemes. To formally describe a KR-scheme, we must identify and exploit such similarities. Initially, we observe that all KR-schemes contain a KB that is used to store knowledge regarding a domain of interest. This knowledge is stored in a specific and well-defined format. For example, in a propositional KB, the knowledge is represented by a propositional expression or a set of propositional expressions; in a relational DB, the knowledge is represented by a set of tables, which are instances of a certain DB schema; in a semantic network, knowledge is represented by a specially formatted graph.

This format is provided by the designer of the KR-scheme and defines a (usually infinite) set of available possibilities for the KB. Any KB that can be produced under this KR-scheme is a member of this set, which will be called the *Knowledge Base set* (or the *KB set*) and denoted by S_K. The set S_K contains all the possible variety of information that can be stored in the specific KR-scheme. Notice that S_K only contains symbols, so no semantic theory is attached to it.

A Formal Description of Queries

Another common property of KR-schemes is that they allow an interaction with the user who would like to know the current contents of the KB. During this interaction the user poses a question (usually referred to as a *query*) to the system and expects an answer. Once again, the different queries that can be posed upon the KB are specified by the designer of the KR-scheme; all KR-schemes provide a query language, with a specific syntax, to the user in order to formulate his queries.

The possible answers are likewise limited. Some systems allow only *Closed Queries*; in this case the only possible answers are usually YES or NO. In some cases, for example in many-valued logics, there can be some more possible answers, such as MAYBE, POSSIBLY etc. If the Open World Assumption (OWA) is used then the value UNKNOWN is possible as well. In other systems, where *Open Queries* are allowed, the answer can be something much more complex. One example is the relational model, where the answer can be a tuple or a set of tuples. Despite the variety of the different answers, the set of possibilities is once again well-defined.

Using the same thoughts as with the KB set, we will define two more sets, the *Query set*, S_Q, containing all the possible queries upon the KB and the *Answer set*, S_A, containing all the possible answers to a query. In the above discussion, we implied the existence of an algorithm that evaluates the user query against the KB and decides on the proper answer. This algorithm, like any algorithm, can be formally modeled with a function. This function will be called the *Query Function* and denoted by $ASK: S_K \times S_Q \rightarrow S_A$. The query answering algorithm is another structure that must be provided by the KR-scheme designer; without it the system cannot answer any queries.

It is important to note here that querying the KB is the only way a user has to identify the contents of the KB, because the user is usually neither interested nor allowed to see the whole KB. The KB is usually a set of structures not understandable by the user and, in most cases, it is too big for him to handle. Moreover, security or privacy considerations make such kind of interaction prohibitive in general. In the rare cases where the user is allowed to see the whole KB, we can model this action by

the query: "give me the contents of the KB", and the ASK function returning the contents of the KB.

The ASK function is a way to provide the symbols comprising a KB with real world semantics. As already mentioned, the symbols by themselves have no meaning. The ASK function assigns to each KB (symbol) a set of answers (one answer per possible query), which "characterizes" the KB. The Answer set (S_A) is supposed to contain symbols understandable by the user without further explanation, thus giving meaning to the original symbols in the KB (which need not be identifiable). The Query and Answer sets are supposed to be designed in such a way as to allow the user to extract the full knowledge stored in a KB. Any type of knowledge that the user cannot ask about is invisible to him; it could as well not exist at all.

A Formal Description of Updates

Querying is not the only interaction a user may have with a KB. All KR-schemes provide the user with the ability to change the contents of a KB. A KB must be able to change dynamically for several reasons. For example, mistakes may have occurred during the input; or some new information previously unavailable, unknown, or classified may now become available; or the world represented by the KB may change. In all these cases the contents of the KB must change in order to properly represent the real world. A KR-scheme must allow the user to make these changes.

To include this type of interaction in our model we will need a new set, the *Update set*, S_U, containing all the possible updates a user may make upon the KB. In order to calculate the new, updated KB, an algorithm is needed, represented with a function, as usual. This function will be called the *Update function*, and denoted by TELL:$S_K \times S_U \rightarrow S_K$.

Once again, the TELL function provides the update semantics of the KB. It actually represents a set of "what if" hypotheses: given the KB $K \in S_K$, what would happen if the new knowledge (update) $U \in S_U$ became known? The combination of ASK and TELL functions provides the KB with the semantics needed to be classified as a KR-scheme.

Constraints and Justifications

Notice that we pose no restrictions on the contents of the above sets, but we will require that they are non-empty. It is easy to see that all the sets are necessary for the KR-scheme. A KR-scheme allowing no KBs has no way to express information (store knowledge). There is no reasonable KR-scheme allowing no updates; this would make it extremely inflexible, thus unsuitable for most applications. Even if we want a KB that cannot be changed by any means, we could simply create an arbitrary set S_U and define TELL as: TELL(K,U)=K for all $K \in S_K$, $U \in S_U$, which is actually the same. The query and answer sets cannot be empty, because this would disallow queries. What is the use of a KB that provides no output to the user?

As far as the two functions are concerned, ASK and TELL must be total, in order to have an answer for any possible KB-query combination and an updated KB for any possible KB-update combination. This requirement has strong intuitive grounds; it would not make sense to have queries or updates that cannot be dealt with. This would cause errors. It can be argued that there could exist queries that are inapplicable to a certain KB. This case can be modeled by creating an artificial answer, say ERROR$\in S_A$ or INAPPLICABLE$\in S_A$ and returning this answer in the inapplicable cases. Similarly, if an update is inapplicable (for example, if it violates an integrity constraint), then we could define TELL as having no effect in this particular case (TELL(K,U)=K). Apart from the constraint of totality, the ASK and TELL

functions can be freely defined; no rationality constraints are imposed upon them in order to preserve generality.

Formally Defining a KR-scheme

The above sets and functions uniquely characterize a KR-scheme. The following definition summarizes the thoughts expressed in the preceding sections:
Definition 1 A KR-scheme is a 6-tuple (S_K, S_U, S_Q, S_A, ASK, TELL), where S_K, S_U, S_Q and S_A are non-empty sets and ASK and TELL are total functions defined as: $ASK:S_K \times S_Q \rightarrow S_A$, $TELL:S_K \times S_U \rightarrow S_K$.

One may argue that this definition, though general, does not take into account some of the most sophisticated advances in DB technology, such as triggers, user profiles, active rules or integrity constraints. In fact, it does! As will be made clear in the following examples, the concept of the set is so general that virtually any type of information can be in a set. The same argument allows us to include any type of construct imaginable, such as semantic networks, different forms of logic, relational schemas etc.

It can also be argued that the user interaction is too weak. We should possibly include operations that query (or update) structures not normally considered as part of the KB (for example user profiles). Similarly, we would like to allow sequences of updates (transactions) that are committed (or rolled back) as one operation, or parameterize the query algorithm depending on the user, as some users may have more privileges upon the KB than others. In fact the above operations can be all modeled in our framework. The ASK and TELL operations are in a sense "overloaded", including operations not usually considered as queries or updates respectively.

Indeed, all types of user interaction can be classified in two types: "read" operations and "write" operations. All operations that only read the KB without making any changes upon it are considered "read" operations and could be modeled using the ASK function. Similarly, all operations that in any way change the contents of the KB are considered "write" operations and could be modeled using the TELL function. For example the question "give me the persons that hold administrative privileges upon the DB" is not considered a query in the conventional sense; in our framework it is a query, thus a member of the S_Q set. The ASK function should be designed in such a way as to be able to process it and the Answer set, S_A, should contain all the possible answers to such a query. Similarly the action: "give administrative privileges to X" is an update in our sense and should be a member of the update set. The TELL function should respond by changing the KB in such a way as to give X administrative privileges, if X exists as a user. Transactions can be likewise modeled by considering that a sequence of updates is also an update (belongs in S_U).

As already stated, the contents of the sets S_K, S_U, S_Q of our structure contain no knowledge by themselves. They merely contain the symbols (or symbol sequences) used to describe the knowledge, thus they constitute the "symbol level" of a KR-scheme. On the other hand, the set S_A and the functions ASK and TELL give meaning to these symbols, thus constituting the "knowledge level" of a KR-scheme.

Examples of KR-schemes

Introduction

The above arguments show the generality of our model. We conjecture (even though this cannot be proved) that all KR-schemes can be modeled using this 6-tuple. A few examples will provide some insight on the reasons behind this conjecture. In the following, we will attempt to model a monotonic and a non-monotonic logic-based KR-scheme, the relational model in its simplest form, as well as in its expanded form that appears in commercial DBMSs, containing integrity constraints, triggers etc. Moreover, we will show that even DB schemes may be viewed as data to be stored in a KB, thus showing that this model can even contain meta-data information.

Propositional KBs (AGM Model)

In this example, we will try to model propositional KBs based on the AGM paradigm, as described in [1]. Assume a propositional language L and the set of all the well formed formulas (wff) in L, denoted by L^*. According to the AGM model, a KB is a set of propositions, so the KB set is $S_K=P(L^*)$, the powerset of L^*. Any proposition in $p \in L^*$ can be an update or a query, so $S_U=S_Q=L^*$. We will use the Closed World Assumption, where the possible answers to any query are either YES or NO, so $S_A=\{YES, NO\}$. In the Closed World Assumption, the Query function ASK is defined as: ASK(K,Q)=YES iff $K \models Q$; else ASK(K,Q)=NO. The AGM paradigm does not specify any particular update (belief revision) algorithm; it only gives a set of postulates that should be satisfied by one. Therefore the AGM model describes a class of different KR-schemes, depending on the belief revision algorithm (equivalently TELL function) selected.

Defeasible KBs

A defeasible KB is a KB based on defeasible logic, whose syntax and properties are described in [10], among other works. We denote by D the set of all well formed defeasible formulas. A defeasible KB is a set of such formulas, thus the KB set is the powerset of D ($S_K=P(D)$). Any rule (formula) can be an update, thus $S_U=D$. Queries can be of the form $\{+\partial L, -\partial L, +\Delta L, -\Delta L\}$, where L is a literal of the language, querying whether the given literal L can or cannot be strongly or defeasibly proved in the given KB. The answer to such a query can be either YES or NO, thus $S_A=\{YES, NO\}$. The ASK function is the function checking whether a query of the above form can be proved by the contents of the KB using any valid proof. An update in defeasible KBs is a very straightforward procedure; it simply constitutes of the annexation of the new proposition into the KB. Thus $TELL(K,U)=K \cup \{U\}$.

Relational Model (Simple)

The case of the relational model is somewhat more complex. We will initially take the simple model, where only tables are used (integrity constraints, triggers etc are not allowed). We will assume that SQL is the underlying data manipulation language. Each DB schema actually defines a KR-scheme of its own. Suppose any particular DB schema S. Then the KB set S_K contains all the possible instances of S, S_U contains all the acceptable UPDATE, INSERT and DELETE operations upon S and S_Q contains all the possible SELECT operations upon S. If we would like to make our model more exact, we should include in S_U all possible transactions, including those that perform more than one atomic operation upon the DB. These transactions can be modeled as sequences of atomic operations. The answer to a query can be any set of tuples. So, S_A is quite complex, containing all the possible tuples that can be

created, even tuples that are not instances of any table definition of the schema S. The ASK function is the algorithm that evaluates user queries and the TELL function is the algorithm that performs the updates upon the DB; both are implemented in all DBMSs.

Relational Model (Expanded)

A more complex system should also include user profiles, triggers, integrity constraints, views etc. A KB should include this kind of information; thus S_K should be expanded to include such objects. Under this expanded model a KB consists of its data (as in the simple model) along with a set of user profiles, a set of triggers, a set of integrity constraints and a set of views. Similarly, the Update set S_U should be expanded to allow changes upon such objects (for example containing UPDATE TRIGGER operations). The Query set S_Q should be expanded to allow questions upon such objects (for example: "what are the preconditions of trigger T?"). Similarly, the Answer set should be expanded to be able to answer questions regarding these objects and the ASK and TELL functions should implement all the above expansions, by actually executing the queries and updates upon these objects. The implementations of SQL that appear in commercial DBMSs include such facilities.

Schemes of a Relational Model

Proceeding one step further, we could also consider each DB schema as a KB on its own right (at a meta-level). Commercial DBMSs allow the creation and manipulation of the DB schema, by allowing the user to create tables or change the definition of existing ones and store the schema in a certain structure. This structure can be considered an instance of the KR-scheme of DB schemes and it can also be modeled using our approach. In this example, it is important to discriminate between a KR-scheme and a DB scheme: we are trying to model a KR-scheme which describes (contains) DB schemes. The KB set (S_K) of this KR-scheme should contain all the possible sets of tuple definitions (DB schemas) that can be created using elements from a given, fixed domain D. The Update set comprises all the commands that alter a schema by changing table definitions and creating or deleting existing table definitions (implementations of SQL in commercial DBMSs include this facility). Likewise, the Query set should contain all the possible questions upon the schema definition. The Answer set should contain all the possible answers to such questions, while the ASK and TELL functions represent the algorithms implementing these operations.

Comparing Expressive Power

Reduction as a Means of Comparison

With the above formalism at hand, comparing two given KR-schemes is possible, even if these two schemes seem to have nothing in common. In effect, we have created a common model which can be used in order for this comparison to take place. One can imagine several ways to formally compare expressive power. We will introduce two different ones. Deciding on the method that most properly fits our intuitive notion of a scheme being more expressive than another is a matter of current research.

A straightforward comparison is not generally possible, as the two KR-schemes may use totally different symbols and semantics. To overcome this problem, both comparison methods use the notion of *reduction*. They are based on the general

idea that being able to express correctly the information appearing in one scheme in terms of the other is an indication that the latter scheme is more expressive.

More specifically, consider two KR-schemes $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$, $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ and suppose that we want to prove that S_2 is more expressive than S_1. If this is true, then we should be able to "substitute" each symbol (or symbol sequence) in S_1 with a symbol (or symbol sequence) in S_2. If this can be done for every type of information expressible in the former scheme, then the latter scheme can express at least the type of information that the former can, so it is at least as expressive as the former.

If S_2 is strictly more expressive than S_1 then it may be able to express information (knowledge) not available in S_1; in this case the opposite reduction (from S_2 to S_1) is not possible. It might also be the case that S_1 and S_2 are incomparable; this could happen if each scheme contains symbols (knowledge) which cannot be mapped to the symbols of the other system properly. This is an expected property, as the ordering cannot be total.

The substitution of the symbols in S_1 with symbols in S_2 cannot be made arbitrarily; we should guarantee that there is no loss of information during the transition. Loss of information appears when the semantics of the symbols in S_1 and the semantics of their "substitutes" in S_2 are different. We must preserve the semantics of the original symbols to avoid loss of information; in other words, we may change the symbols, but not the knowledge they represent. We must formally define what properties must hold for the "transition algorithm" in order to preserve the semantics. There are at least two ways to do so, depending on our intuitive notion on what "preserving the semantics" means, resulting in two different types of reduction.

Normal Reduction

Introduction

Initially, we will consider a rather strong form of reduction, that we will call *normal reduction*. Under this notion, two KBs are considered to have the same semantics if and only if all the possible query and update results related to these two KBs are the "same". We will later show why this is a strong requirement and relax it by defining a form of reduction based on the "behavior" of the two KBs under queries.

More formally, consider again the schemes $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ and $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$. Each possible KB in S_1 (any $K_1 \in S_{K1}$) must be mapped to a KB in S_2 ($K_2 \in S_{K2}$). This mapping is a total function, which will be called *Knowledge Base Reduction function* (or *KB Reduction function*) and denoted by $f_K:S_{K1} \rightarrow S_{K2}$. This mapping by itself is not sufficient, because a KR-scheme also consists of updates, queries and answers to queries. Thus, similar assignments must be made for the Update and Query sets, using the *Update Reduction function*, $f_U:S_{U1} \rightarrow S_{U2}$, and the *Query Reduction function*, $f_Q:S_{Q1} \rightarrow S_{Q2}$, respectively, which must be total as well. For the Answer set, the opposite assignment must be made; once the substitution of S_1 with S_2 has been made, answers will be obtained by S_2 (using function ASK_2), and will belong in set S_{A2}. So we need a way to "translate" these answers to answers that S_1 understands (belonging in S_{A1}). To do so, each possible answer in the substituting system (S_2), will be mapped to one answer in the substitutable system (S_1) by the *Answer Reduction function*, $f_A:S_{A2} \rightarrow S_{A1}$. This function must be total as well.

As one can imagine, the existence of these four functions by itself should not constitute evidence that S_2 is more expressive than S_1. After all, we can define such functions for any pair of schemes. We need to impose restrictions on these functions, in order to guarantee that the semantics of the original system (S_1) are preserved during the transition to S_2. These restrictions are closely related to the symbol manipulation functions ASK and TELL and based on the intuition previously phrased, namely that two KBs are equivalent (have the same semantics) if they give the "same" results to all updates and queries.

Preserving the Semantics of Queries

The first restriction has to do with the query answering mechanism. Assume (in S_1) a KB $K \in S_{K1}$, a query $Q \in S_{Q1}$, and the answer $A=ASK_1(K,Q) \in S_{A1}$. We denote by K' the assigned KB of K in S_2 and by Q' the assigned query of Q in S_2. Formally: $K'=f_K(K) \in S_{K2}$, $Q'=f_Q(Q) \in S_{Q2}$. Finally, let A' be the answer of the query Q' under the KB K', so $A'=ASK_2(K',Q') \in S_{A2}$.

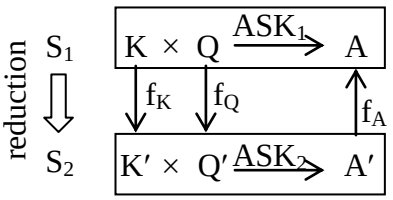

Figure 1: Query Preservation property

To preserve the semantics these two answers (A and A') must be the "same", in the sense that A must be the assigned answer of A' under the Answer Reduction function. So the restriction that must hold in order to preserve the semantics of queries during the reduction is: $A=f_A(A')$. Expanding this expression using the definitions of the previous paragraph we get (see also Figure 1):

$$ASK_1(K,Q)=f_A(ASK_2(f_K(K),f_Q(Q)))$$
$$\text{for all } K \in S_{K1},\ Q \in S_{Q1}$$

This will be called the *Query Preservation property*.

Preserving the Semantics of Updates

The second restriction is related to the update mechanism. Consider two KBs K_1, $K_2 \in S_{K1}$ and an update $U \in S_{U1}$, such that $K_2=TELL_1(K_1,U)$. As before, let K_1', K_2' be the assigned KBs of K_1, K_2 respectively and U' the assigned update of U in S_2 (formally: $K_1'=f_K(K_1) \in S_{K2}$, $K_2'=f_K(K_2) \in S_{K2}$, $U'=f_K(U) \in S_{U2}$).

Since the equation $K_2=TELL_1(K_1,U)$ holds, K_2 is the result of the update of K_1 with U in S_1. To preserve the semantics during the transition to S_2, we must ensure that K_2' is the result of the update of K_1' with U' is S_2. Thus, the following equation must hold: $K_2'=TELL_2(K_1',U')$. Expanding this expression, we get:

$$\text{For any } K_1,\ K_2 \in S_{K1},\ U \in S_{U1} \text{ such that}$$
$$K_2=TELL_1(K_1,U) \text{ it holds that: } f_K(K_2)=TELL_2(f_K(K_1),f_U(U))$$

This property may look adequate at first glance but there is an annoying fact about it. Consider the trivial case where scheme S_2 is very poor, allowing only one KB, ie $S_{K2}=\{K_0\}$ for some K_0. Then, necessarily, for all $K \in S_{K1}$ we get: $f_K(K)=K_0$. Also for all $U' \in S_{U2}$ it necessarily holds that $TELL_2(K_0,U')=K_0$.

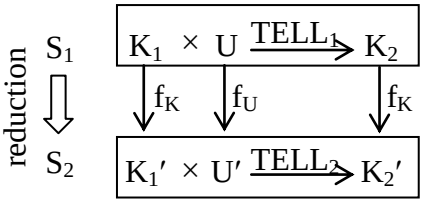

Figure 2: Update Preservation property

Combining these two properties, we can conclude that, for any S_1, the restriction $f_K(K_2)=TELL_2(f_K(K_1),f_U(U))$ holds for all K_1, $K_2 \in S_{K1}$, $U \in S_{U1}$.

This result is counter-intuitive and obviously unacceptable for our purposes, leading us to the conclusion that this property, though necessary, is not sufficient to capture the intuition of the preservation of semantics. What the above example shows is that we lack the opposite property; in effect, we need that, in scheme S_2, whenever $K_2'=TELL_2(K_1',U')$ for two KBs K_1', $K_2' \in S_{K2}$ and an update $U' \in S_{U2}$, the same must hold for their respective KBs and update in S_1. This is exactly the opposite implication of the above property. Summarizing the above, the property that guarantees the preservation of the update semantics during the reduction is the following (see also Figure 2):

For any K_1, $K_2 \in S_{K1}$, $U \in S_{U1}$ it is the case that
$K_2=TELL_1(K_1,U)$ if and only if $f_K(K_2)=TELL_2(f_K(K_1),f_U(U))$

This will be called the *Update Preservation property*.

Formal Definition

The above two properties (Query and Update Preservation properties) guarantee that the semantics of each KB is preserved for both types of user interaction. The formal definition of *normal reduction* is as follows:

Definition 2 Consider two KR-schemes $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ and $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$. If there exists a 4-tuple of total functions (f_K, f_U, f_Q, f_A), $f_K:S_{K1} \to S_{K2}$, $f_U:S_{U1} \to S_{U2}$, $f_Q:S_{Q1} \to S_{Q2}$, $f_A:S_{A2} \to S_{A1}$, such that the Query and Update Preservation properties hold, then we say that *S_1 can be normally reduced to S_2*. The 4-tuple (f_K, f_U, f_Q, f_A) will be called the *reduction algorithm*. If S_1 can be reduced to S_2 then we say that *S_2 is at least as expressive as S_1*, denoted by $S_1 \leq_r S_2$. If $S_1 \leq_r S_2$ and $S_2 \leq_r S_1$ then S_1 and S_2 will be called *equivalent*, denoted by $S_1 \cong_r S_2$. If $S_1 \leq_r S_2$ but $S_2 \not\leq_r S_1$ then we say that *S_2 is more expressive than S_1*, denoted by $S_1 <_r S_2$.

Behavioral Reduction

Introduction

As already noticed, normal reduction is a rather strong form of reduction. We can define a weaker form, based on the perception a user has about the KB. This weaker form of reduction will be based on the *behavior* of the KB, hence the name. It is based on the observation that, if the user cannot "notice the difference" between a KB K and its assigned KB K′ (in the second scheme), then the reduction can be considered successful; the semantics of the KB K are preserved, as far as the user is concerned.

More formally, let us consider the schemes $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ and $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$. Behavioral reduction uses the four functions f_K, f_U, f_Q and f_A as defined in the previous section to "translate" the information of S_1 in terms of S_2. However, the properties that these functions must have is different this time. In behavioral reduction, we consider that the semantics of a KB is determined by the queries' answers upon the KB. If two KBs give the same answers to all queries, then they carry the same information. To ensure that, the Query Preservation property is enough, as already described.

However, a KB can be updated as well. We must ensure that after any update, the answers given to queries are still the "same". If this is not true, then the user could perform an update followed by a query and get "different" results; this would allow

him to notice the transition. The Update Preservation property is more restrictive than necessary in this case, and must be dropped. All we need is to require that after an update the resulting KBs still give the same query answers. This should be true for any sequence of updates as well. Suppose any KB K in S_1 and its respective KB (say K′) in S_2. For any number of updates in S_1 (say U_1, U_2, …, U_m) and their assigned updates in S_2 (say U_1', U_2', …, U_m') the result of the sequential update of K with U_1, U_2, …, U_m and the result of the sequential update of K′ with U_1', U_2', …, U_m' should give the same answers to all queries. If this is not true for any specific sequence of updates, then the user could perform these updates followed by a carefully selected query and get "different" results; this would allow him to notice the transition which constitutes a violation of our initial objective of "behavioral preservation of semantics".

Put another way, the main difference between behavioral and normal reduction has to do with the update results. In normal reduction the update results need to be the "same", under the reduction algorithm. In behavioral reduction, the update results need only to have the same behavioral semantics, but not necessarily be the same. In behavioral reduction a kind of recursion appears: two KBs have the same behavioral semantics if and only if they give the "same" answers to queries and all the update results related to these KBs have the same behavioral semantics. This recursion does not appear in normal reduction.

Preserving the Semantics of Queries

As already stated in the previous section, in behavioral reduction we need that a KB in S_1 and its respective in S_2 give the "same" results to the "same" queries. There can be no loosening in the Query Preservation property for behavioral reduction. In order to preserve the "behavioral" semantics of queries the property used in normal reduction must hold:

$$ASK_1(K,Q)=f_A(ASK_2(f_K(K),f_Q(Q)))$$
$$\text{for all } K\in S_{K1},\ Q\in S_{Q1}$$

Preserving the Semantics of Updates

The update semantics can be preserved with a weaker constraint than the Update Preservation property. All we need in behavioral reduction is that the KB occurring after an update in S_1, is behaviorally the same as the one occurring after the "same" update in S_2. So, suppose a KB $K_1\in S_{K1}$, an update $U\in S_{U1}$ in S_1 and their respective KB $K_1'=f_K(K_1)\in S_{K2}$ and update $U'=f_U(U)\in S_{U2}$ in S_2. We need that the updated KBs $K_2=TELL_1(K_1,U)\in S_{K1}$ and $K_2'=TELL_2(K_1',U')\in S_{K2}$ are behaviorally the same.

As already mentioned, the behavior of the two KBs $K_2\in S_{K1}$, $K_2'\in S_{K2}$ is determined by the answers they give to the queries. So, suppose a query $Q\in S_{Q1}$ and its respective query $Q'=f_Q(Q)\in S_{Q2}$ in S_2. We need that the answers $A=ASK_1(K_2,Q)\in S_{A1}$ and $A'=ASK_2(K_2',Q)\in S_{A2}$ in S_1 and S_2 are the "same" in the sense that $A=f_A(A')$. If this is not true, then the user could update the original KB (K_1) with U, then ask the query Q and have a different behavior (answer) in the two systems, thus being able to notice the difference between the original KBs K_1 and K_1'. Expanding the above expression, we get:

$$ASK_1(K_2,Q)=f_A(ASK_2(K_2',Q'))$$
or equivalently:
$$ASK_1(TELL_1(K_1,U),Q)=f_A(ASK_2(TELL_2(K_1',U'),Q'))$$

or equivalently:

$$ASK_1(TELL_1(K_1,U),Q)=f_A(ASK_2(TELL_2(f_K(K_1),f_U(U)),f_Q(Q)))$$

The above expression holding for all $K_1 \in S_{K1}$, $U \in S_{U1}$, $Q \in S_{Q1}$ is a necessary condition, but it is not yet strong enough to support the notion of behavioral reduction. Even though the updated KBs $K_2 \in S_{K1}$, $K_2' \in S_{K2}$ give the "same" answers to all the possible queries, they can be themselves updated. If the resulting KBs (after the second update) do not give the same answers to all the queries, then the user can actually "notice the difference" between the two KBs K_2 and K_2' (and consequently between K_1 and K_1'). The behavioral semantics of the symbols in S_1 are not preserved during the transition to S_2 in this case.

To cover this possibility, we need to expand the above expression to deal with sequences of updates as well, for the reasons already phrased in a previous section. To make the argument more clear, suppose any finite sequence of updates (in S_1) $U_1, U_2, \ldots, U_m \in S_{K1}$ (for some $m>0$) and the respective sequence (in S_2) $U_1'=f_U(U_1) \in S_{U2}$, $U_2'=f_U(U_2) \in S_{U2}$, …, $U_m'=f_U(U_m) \in S_{U2}$. Assume any KB in S_1 $K_0 \in S_{K1}$ and its respective KB in S_2 $K_0'=f_K(K_0) \in S_{K2}$. Finally, let $K_1, K_2, \ldots, K_m \in S_{K1}$ be a sequence of KBs of S_1, such that $K_1=TELL_1(K_0,U_1)$, $K_2=TELL_1(K_1,U_2)$, …, $K_m=TELL_1(K_{m-1},U_m)$ and $K_1', K_2', \ldots, K_m' \in S_{K2}$ be a sequence of KBs of S_2, such that $K_1'=TELL_2(K_0',U_1')$, $K_2'=TELL_2(K_1',U_2')$, …, $K_m'=TELL_2(K_{m-1}',U_m')$. The intuitive notion of behavioral reduction requires that the KBs K_1 and K_1' give the "same" answers to the "same" queries, in the sense of the Query Preservation property. The same must hold for the KBs K_2 and K_2', K_3 and K_3' and so on. In other words the constraint just expressed must hold not only for all updates, but for all sequences of updates as well. If there exists a sequence of updates that does not satisfy this property, then the transition of system S_1 to S_2 could be noticed by the user if he performs the specific sequence of updates and then performs a selected query; the two KBs are no longer indistinguishable by the user.

To express the above condition more formally, we will need the iterated TELL function, which calculates the result of a sequence of updates upon a certain KB:

Definition 3 Consider an update function $TELL:S_K \times S_U \rightarrow S_K$. We define, for any $m \geq 0$, the *iterated TELL function*, $TELL^m$, recursively, as follows:

$TELL^0:S_K \rightarrow S_K$, $TELL^0(K)=K$ for all $K \in S_K$.

$TELL^1:S_K \times S_U \rightarrow S_K$, $TELL^1(K,U)=TELL(K,U)$ for all $K \in S_K$, $U \in S_U$.

$TELL^m:S_K \times S_U{}^m \rightarrow S_K$, $TELL^m(K,U_1,\ldots,U_m)=TELL(TELL^{m-1}(K,U_1,\ldots,U_{m-1}),U_m)$, for all $K \in S_K$, $U_1, \ldots, U_m \in S_U$.

Using the iterated TELL operator, we can express the above considerations more formally as follows:

$$ASK_1(TELL_1{}^m(K,U_1,\ldots,U_m),Q)=f_A(ASK_2(TELL_2{}^m(f_K(K),f_U(U_1),\ldots,f_U(U_m)),f_Q(Q)))$$

for all $m \geq 0$, $K \in S_{K1}$, $U_1,\ldots,U_m \in S_{U1}$, $Q \in S_{Q1}$

This constraint will be called the *Behavior Preservation property*.

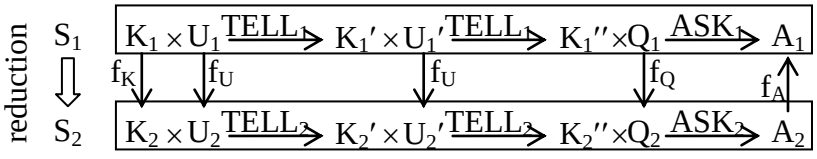

Figure 3: Behavior Preservation property for m=2

Formal Definition

The Behavior Preservation property does not ensure that the transition keeps the knowledge of the original system intact, but it does ensure that the behavior of the two systems will be identical and indistinguishable by the user, thus it preserves the behavioral semantics of a KB. This is done by ensuring that, for any KB and any finite sequence of updates (regardless of its length), the two systems S_1 and S_2 will give the "same" answers to any query. This guarantees that the user can never notice the transition of S_1 to S_2 and formally expresses the intuition behind the behavioral reduction. Finally, one can notice that the Query Preservation property is a special case of the Behavior Preservation property (for m=0). The formal definition of *behavioral reduction* is as follows:

Definition 4 Consider two KR-schemes $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ and $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$. If there exists a 4-tuple of total functions (f_K, f_U, f_Q, f_A), $f_K:S_{K1}\rightarrow S_{K2}$, $f_U:S_{U1}\rightarrow S_{U2}$, $f_Q:S_{Q1}\rightarrow S_{Q2}$, $f_A:S_{A2}\rightarrow S_{A1}$, such that the Behavior Preservation property holds, then we say that *S_1 can be behaviorally reduced to S_2*. The 4-tuple (f_K, f_U, f_Q, f_A) will be called the *reduction algorithm*. If S_1 can be reduced to S_2 then we say that *S_2 is behaviorally at least as expressive as S_1*, denoted by $S_1\leq_b S_2$. If $S_1\leq_b S_2$ and $S_2\leq_b S_1$ then S_1 and S_2 will be called *behaviorally equivalent*, denoted by $S_1\cong_b S_2$. If $S_1\leq_b S_2$ but $S_2\not\leq_b S_1$ then we say that *S_2 is behaviorally more expressive than S_1*, denoted by $S_1<_b S_2$.

Discussion

Properties of the Relations

We can immediately deduce some properties of the above expressiveness relations. Initially, we will prove some lemmas that will be helpful in the following discussion. In all of the following propositions, we will implicitly assume that two schemes $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$, $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ are given, as well as a 4-tuple of total functions (f_K, f_U, f_Q, f_A) as in Definition 2.

Lemma 1 Suppose two KR-schemes S_1, S_2, such that the Update Preservation property holds. Then for all $K\in S_{K1}$, $U\in S_{U1}$ it holds that:
$f_K(TELL_1(K,U))=TELL_2(f_K(K),f_U(U))$.

Proof
Suppose that $K'=TELL_1(K,U)$. By the Update Preservation property, we get that:
$f_K(K')=TELL_2(f_K(K),f_U(U))$, thus:
$f_K(TELL_1(K,U))=TELL_2(f_K(K),f_U(U))$.

Lemma 1 shows that when the Update Preservation property holds, the update of a KB K with an update U in S_1, followed by the "translation" of the result in terms of S_2 gives the same result as the "translation" of K and U in terms of S_2 followed by the update of $f_K(K)$ with $f_U(U)$ (in S_2). Moreover, the following generalization holds:

Lemma 2 Suppose two KR-schemes S_1, S_2, such that the Update Preservation property holds. Then for all $m\geq 0$, $K\in S_{K1}$, $U_1,\ldots,U_m\in S_{U1}$ it holds that:
$f_K(TELL_1^m(K,U_1,\ldots,U_m))=TELL_2^m(f_K(K),f_U(U_1),\ldots,f_U(U_m))$.

Proof
For m=0:
$f_K(TELL_1^0(K))=f_K(K)$ and
$TELL_2^0(f_K(K))=f_K(K)$, so the lemma holds for m=0.
Suppose that it holds for m=0,1,…,n–1. We will prove that it holds for m=n.

Indeed:
$TELL_2^n(f_K(K),f_U(U_1),\ldots,f_U(U_n))=$
$=TELL_2(TELL_2^{n-1}(f_K(K),f_U(U_1),\ldots,f_U(U_{n-1})),f_U(U_n))=$ (by definition)
$=TELL_2(f_K(TELL_1^{n-1}(K,U_1,\ldots,U_{n-1})),f_U(U_n))=$ (by induction, for m=n−1)
$=f_K(TELL_1(TELL_1^{n-1}(K,U_1,\ldots,U_{n-1})),U_n)=$ (by Lemma 1)
$=f_K(TELL_1^n(K,U_1,\ldots,U_n))$ by definition, and the proof is complete.

Using this lemma, we can prove the intuitively expected property that behavioral reduction is easier to achieve than normal reduction:

Proposition 1 Suppose two KR-schemes S_1, S_2. If $S_1 \leq_r S_2$, then $S_1 \leq_b S_2$. Similarly, if $S_1 \cong_r S_2$ then $S_1 \cong_b S_2$.

Proof

Since $S_1 \leq_r S_2$, the Update Preservation property holds, so by Lemma 2, we have that for all $m \geq 0$, $K \in S_{K1}$, $U_1,\ldots,U_m \in S_{U1}$:
$f_K(TELL_1^m(K,U_1,\ldots,U_m))=TELL_2^m(f_K(K),f_U(U_1),\ldots,f_U(U_m))$.
So, assume any $m \geq 0$, $U_1,\ldots,U_m \in S_{U1}$, $K \in S_{K1}$, $Q \in S_{Q1}$.
Suppose that: $K'=TELL_1^m(K,U_1,\ldots,U_m)$. Then:
$f_A(ASK_2(TELL_2(f_K(K),f_U(U_1),\ldots,f_U(U_m)),f_Q(Q)))=$
$=f_A(ASK_2(f_K(TELL_1^m(K,U_1,\ldots,U_m)),f_Q(Q)))=$ (by Lemma 2)
$=f_A(ASK_2(f_K(K'),f_Q(Q)))=$ (by definition of K′)
$=ASK_1(K',Q)=$ (by the Query Preservation property)
$=ASK_1(TELL_1^m(K,U_1,\ldots,U_m),Q)$ (by definition of K′)
So the Behavior Preservation property holds, thus $S_1 \leq_b S_2$.
For the equivalence, the result is obvious using the definition and the property just proved.

In the following two lemmas, we will prove that the expressiveness relations are transitive. Using this result, we can break a complex reduction into simpler ones.

Lemma 3 The relation $\leq_r$ is transitive.

Proof

Suppose the KR-schemes $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$, $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$, $S_3=(S_{K3}, S_{U3}, S_{Q3}, S_{A3}, ASK_3, TELL_3)$, such that $S_1 \leq_r S_2$ and $S_2 \leq_r S_3$. We will prove that $S_1 \leq_r S_3$.
Since $S_1 \leq_r S_2$, there exists a reduction algorithm $(f_{K12}, f_{U12}, f_{Q12}, f_{A12})$.
Similarly, since $S_2 \leq_r S_3$, there exists a reduction algorithm $(f_{K23}, f_{U23}, f_{Q23}, f_{A23})$.
We define the functions:
$f_{K13}:S_{K1} \to S_{K3}$, such that $f_{K13}(K)=f_{K23}(f_{K12}(K))$ for all $K \in S_{K1}$,
$f_{U13}:S_{U1} \to S_{U3}$, such that $f_{U13}(U)=f_{U23}(f_{U12}(U))$ for all $U \in S_{U1}$,
$f_{Q13}:S_{Q1} \to S_{Q3}$, such that $f_{Q13}(Q)=f_{Q23}(f_{Q12}(Q))$ for all $Q \in S_{Q1}$,
$f_{A13}:S_{A3} \to S_{A1}$, such that $f_{A13}(A)=f_{A12}(f_{A23}(A))$ for all $A \in S_{A3}$.
We will prove that $(f_{K13}, f_{U13}, f_{Q13}, f_{A13})$ is a reduction algorithm.
For any $K_1 \in S_{K1}$, $Q_1 \in S_{Q1}$, we denote by $K_2=f_{K12}(K_1)$, $Q_2=f_{Q12}(Q_1)$. Then:
$f_{A13}(ASK_3(f_{K13}(K_1),f_{Q13}(Q_1)))=$
$=f_{A12}(f_{A23}(ASK_3(f_{K12}(f_{K23}(K_1)),f_{Q12}(f_{Q23}(Q_1)))))=$ (by definitions)
$=f_{A12}(f_{A23}(ASK_3(f_{K12}(K_2),f_{Q12}(Q_2))))=$ (by the notation above)
$=f_{A12}(ASK_2(K_2,Q_2))=$ (since $S_2 \leq_r S_3$)
$=f_{A12}(ASK_2(f_{K12}(K_1),f_{Q12}(Q_1)))=$ (by the notation above)
$=ASK_1(K_1,Q_1)$ since $S_1 \leq_r S_2$, so the Query Preservation property holds.

Now assume any $K_1 \in S_{K1}$, $U_1 \in S_{U1}$ and $K_1' = TELL_1(K_1,U_1) \in S_{K1}$.
Since $S_1 \leq_r S_2$ this expression is equivalent to: $f_{K12}(K_1') = TELL_2(f_{K12}(K_1), f_{U12}(U_1))$.
Denote by $K_2' = f_{K12}(K_1') \in S_{K2}$, $K_2 = f_{K12}(K_1) \in S_{K2}$, $U_2 = f_{U12}(U_1) \in S_{U2}$.
Then we equivalently get that: $K_2' = TELL_2(K_2,U_2)$.
By the fact that $S_2 \leq_r S_3$ this is equivalent to: $f_{K23}(K_2') = TELL_3(f_{K23}(K_2), f_{U23}(U_2))$.
By substitution we equivalently get:
$f_{K23}(f_{K12}(K_1')) = TELL_3(f_{K23}(f_{K12}(K_1)), f_{U23}(f_{U12}(U_1)))$
or equivalently:
$f_{K13}(K_1') = TELL_3(f_{K13}(K_1), f_{U13}(U_1))$.
Summarizing, we have that:
$K_1' = TELL_1(K_1,U_1)$ if and only if $f_{K13}(K_1') = TELL_3(f_{K13}(K_1), f_{U13}(U_1))$, which is the Update Preservation property.
Thus, the tuple (f_{K13}, f_{U13}, f_{Q13}, f_{A13}) is indeed a reduction algorithm, so we get that: $S_1 \leq_r S_3$.

Lemma 4 The relation $\leq_b$ is transitive.
Proof
Suppose the KR-schemes $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$, $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$, $S_3=(S_{K3}, S_{U3}, S_{Q3}, S_{A3}, ASK_3, TELL_3)$, such that $S_1 \leq_b S_2$ and $S_2 \leq_b S_3$. We will prove that $S_1 \leq_b S_3$.
Since $S_1 \leq_b S_2$, there exists a reduction algorithm (f_{K12}, f_{U12}, f_{Q12}, f_{A12}).
Similarly, since $S_2 \leq_b S_3$, there exists a reduction algorithm (f_{K23}, f_{U23}, f_{Q23}, f_{A23}).
We define the functions:
$f_{K13}: S_{K1} \rightarrow S_{K3}$, such that $f_{K13}(K) = f_{K23}(f_{K12}(K))$ for all $K \in S_{K1}$,
$f_{U13}: S_{U1} \rightarrow S_{U3}$, such that $f_{U13}(U) = f_{U23}(f_{U12}(U))$ for all $U \in S_{U1}$,
$f_{Q13}: S_{Q1} \rightarrow S_{Q3}$, such that $f_{Q13}(Q) = f_{Q23}(f_{Q12}(Q))$ for all $Q \in S_{Q1}$,
$f_{A13}: S_{A3} \rightarrow S_{A1}$, such that $f_{A13}(A) = f_{A12}(f_{A23}(A))$ for all $A \in S_{A3}$.
We will prove that (f_{K13}, f_{U13}, f_{Q13}, f_{A13}) is a reduction algorithm.
Assume any $m \geq 0$, $U_{11}, U_{12}, \ldots, U_{1m} \in S_{U1}$, $K_1 \in S_{K1}$, $Q_1 \in S_{Q1}$.
We denote by $U_{21} = f_{U12}(U_{11})$, $U_{22} = f_{U12}(U_{12})$, …, $U_{2m} = f_{U12}(U_{1m})$, $K_2 = f_{K12}(K_1)$, $Q_2 = f_{Q12}(Q_1)$. Then:
$f_{A13}(ASK_3(TELL_3^m(f_{K13}(K_1), f_{U13}(U_{11}), \ldots, f_{U13}(U_{1m})), f_{Q13}(Q_1))) =$
$= f_{A12}(f_{A23}(ASK_3(TELL_3^m(f_{K23}(f_{K12}(K_1)), f_{U23}(f_{U12}(U_{11})), \ldots, f_{U23}(f_{U12}(U_{1m}))), f_{Q23}(f_{Q12}(Q_1)))))$ = (by definition)
$= f_{A12}(f_{A23}(ASK_3(TELL_3^m(f_{K23}(K_2), f_{U23}(U_{21}), \ldots, f_{U23}(U_{2m})), f_{Q23}(Q_2))))$ = (by notation)
$= f_{A12}(ASK_2(TELL_2^m(K_2, U_{21}, \ldots, U_{2m}), Q_2))$ = (since $S_2 \leq_b S_3$)
$= f_{A12}(ASK_2(TELL_2^m(f_{K12}(K_1), f_{U12}(U_{11}), \ldots, f_{U12}(U_{1m})), f_{Q12}(Q_1)))$ = (by notation)
$= ASK_1(TELL_1^m(K_1, U_{11}, \ldots, U_{1m}), Q_1)$ since $S_1 \leq_b S_2$, so the Behavior Preservation Property holds.
Thus we have that $S_1 \leq_b S_3$.

The following proposition summarizes the most important properties of the expressiveness relations:
Proposition 2 The relations $\leq_r$ and $\leq_b$ are reflexive and transitive. The relations $\cong_r$ and $\cong_b$ are symmetric, reflexive and transitive.
Proof
Assume a KR-scheme $S=(S_K, S_U, S_Q, S_A, ASK, TELL)$ and the identity functions:
$f_K: S_K \rightarrow S_K$, $f_K(K) = K$, for all $K \in S_K$,

$f_U:S_U \rightarrow S_U$, $f_U(U)=U$, for all $U \in S_U$,
$f_Q:S_Q \rightarrow S_Q$, $f_Q(Q)=Q$, for all $Q \in S_Q$,
$f_A:S_A \rightarrow S_A$, $f_A(A)=A$, for all $A \in S_A$.
It is trivial to see that the Query, Update and Behavior Preservation properties hold for the KR-schemes $S_1=S$ and $S_2=S$ and the reduction algorithm (f_K, f_U, f_Q, f_A). Thus:
$S \leq_r S$, $S \leq_b S$, $S \cong_r S$, $S \cong_b S$, so the relations $\leq_r$, $\leq_b$, $\cong_r$, $\cong_b$ are reflexive.
The relation $\leq_r$ is transitive as proven in Lemma 3. An immediate corollary of this fact is that the relation $\cong_r$ is transitive as well.
The relation $\leq_b$ is transitive as proven in Lemma 4. An immediate corollary of this fact is that the relation $\cong_b$ is transitive as well.
Now, suppose that $S_1 \cong_r S_2$. Then $S_1 \leq_r S_2$ and $S_2 \leq_r S_1$, so $S_2 \cong_r S_1$, thus $\cong_r$ is symmetric.
Using similar arguments, if $S_1 \cong_b S_2$, then $S_2 \cong_b S_1$, thus $\cong_b$ is symmetric as well.
The proof is complete.

Open World Assumption and Closed World Assumption

Some examples may help uncover some more properties of these relations. At first, assume any KR-scheme based on some form of logic (propositional, first-order etc). We will denote by L^* the set of all well formed formulas of the selected logic and S_{OWA}, S_{CWA} the KR-schemes that occur by using L^* and the Open World Assumption (OWA) and Closed World Assumption (CWA) respectively. We can prove that $S_{CWA} <_r S_{OWA}$.

At first, we need to define the two schemes. Initially, we notice that the KB, Update and Query sets are the same for both schemes, as well as the TELL function. Differences exist in the Answer set and the Query function (ASK). We define $S_{OWA}=(S_K, S_U, S_Q, S_{AOWA}, ASK_{OWA}, TELL)$ and $S_{CWA}=(S_K, S_U, S_Q, S_{ACWA}, ASK_{CWA}, TELL)$, where $S_K=S_U=S_Q=L^*$, and TELL is some function that performs updates (any function would do). For the KR-scheme based on the OWA, the possible answers are: $S_{AOWA}=\{YES, NO, UNKNOWN\}$. For a given KB $K \in S_K$ and a query $Q \in S_Q$ the Query function is defined as follows:

$ASK_{OWA}(K,Q)=YES$, iff $K \models Q$,

$ASK_{OWA}(K,Q)=NO$, iff $K \models \neg Q$,
$ASK_{OWA}(K,Q)=UNKNOWN$, otherwise.
Regarding the scheme based on the CWA, the possible answers are only YES or NO, so $S_{ACWA}=\{YES, NO\}$. For a given KB $K \in S_K$ and a query $Q \in S_Q$ the Query function under the CWA is:

$ASK_{OWA}(K,Q)=YES$, iff $K \models Q$,
$ASK_{OWA}(K,Q)=UNKNOWN$, otherwise.

In order to make the desired reduction, we define the following functions:
$f_K:S_K \rightarrow S_K$, $f_K(K)=K$ for all $K \in S_K$,
$f_U:S_U \rightarrow S_U$, $f_U(U)=U$ for all $U \in S_U$,
$f_Q:S_Q \rightarrow S_Q$, $f_Q(Q)=Q$ for all $Q \in S_Q$,
$f_A:S_{AOWA} \rightarrow S_{ACWA}$, $f_A(YES)=YES$, $f_A(NO)=f_A(UNKNOWN)=NO$.
We can prove that the Query and Update Preservation properties hold for the reduction algorithm (f_K, f_U, f_Q, f_A) and the schemes S_{CWA}, S_{OWA}. Indeed for all $K \in S_K$, $Q \in S_Q$, $f_K(K)=K$ and $f_Q(Q)=Q$, so:

$ASK_{CWA}(K,Q)=YES$ if and only if $K \models Q$, which is equivalent to $f_K(K) \models f_Q(Q)$. The latter expression is true if and only if (by definition) $ASK_{OWA}(f_K(K),f_Q(Q))=YES$, which (by definition of f_A) is equivalent to $f_A(ASK_{OWA}(f_K(K),f_Q(Q)))=YES$.

Thus, we conclude that:
$ASK_{CWA}(K,Q)$=YES if and only if $f_A(ASK_{OWA}(f_K(K),f_Q(Q)))$=YES.
Using this equivalence and the fact that S_{ACWA}={YES, NO}, we can trivially get that:
$f_A(ASK_{OWA}(f_K(K),f_Q(Q)))=ASK_{CWA}(K,Q)$, so the Query Preservation property holds.
Since the TELL functions of the two schemes are the same and the KB and update reduction functions (f_K, f_U) are the identity functions in their respective domains, we can trivially conclude that the Update Preservation property holds. So: $S_{CWA} \leq_r S_{OWA}$.

The opposite reduction is not possible. To see that, notice that S_{AOWA} is finite and contains more elements than S_{ACWA}. This means that for any function $f_A:S_{ACWA} \rightarrow S_{AOWA}$, there will exist an element $A_0 \in S_{AOWA}$ for which there will exist no element $A_0' \in S_{ACWA}$ such that $f_A(A_0')=A_0$. However, there exist at least one pair $K_0 \in S_K$, $Q_0 \in S_Q$ such that $ASK_{OWA}(K_0,Q_0)=A_0$. Now suppose that the Query Preservation property holds. Then:
$A_0=ASK_{OWA}(K_0,Q_0)=f_A(ASK_{CWA}(f_K(K_0),f_Q(Q_0)))$.
However, there is no element $A_0' \in S_{ACWA}$ such that $f_A(A_0')=A_0$, so the above relation cannot be true. Using this result, we conclude that $S_{CWA} <_r S_{OWA}$.

The argument used to rule out the possibility that $S_{OWA} \leq_r S_{CWA}$ is a special case of a more complex condition having to do with the size of the four sets of a KR-scheme. The stronger scheme is required to use "larger" sets than the weaker one, or else the reduction is not possible. A more formal and exact presentation of this fact appears in Corollary 1.

Redundancy

Uncovering Redundancy

Introduction

In many KR-schemes, the user is able to express the same information (KB, update, query) with several different ways. From a practical point of view, being able to express the same information in different ways is usually desirable, because it facilitates the user and provides the system with a friendlier interface. From the theoretical viewpoint though, such "duplicate" elements are useless and could be removed from the KR-scheme without loss of expressive power.

Our model allows us to find such "redundant" elements in a scheme and eliminate them. This elimination should cause no loss of expressive power, so the original scheme and the one with the removed elements should be equivalent. In other words, for any scheme containing redundancy, we should be able to create an equivalent one that does not contain redundancy. For the Answer set, redundancy appears when the set contains answers that the system never actually gives. Such answers are useless and could be dropped, without loss of expressive power.

As in the case with the reduction, we can define two types of redundancy (normal and behavioral), depending on what we mean by the term "same" knowledge. Redundancy may appear in any of the four sets comprising a KR-scheme. In the following sections, we will define all eight types of redundancy and prove that redundant elements add no expressive power to a KR-scheme.

Normal Redundancy

Initially, we will consider the normal case. In this case, one KB $K_1 \in S_K$ contains information that is redundant for S_K if there exists another KB, say $K_2 \in S_K$,

which gives the same answers to all queries and provides the same results to all updates. Formally, this requirement can be expressed as follows:

For some $K_1, K_2 \in S_K$, $K_1 \neq K_2$ it holds that:
$ASK(K_1,Q)=ASK(K_2,Q)$ for all $Q \in S_Q$ and
$TELL(K_1,U)=TELL(K_2,U)$ for all $U \in S_U$

Unfortunately, this definition does not work perfectly well. This happens because the KBs K_1 and K_2 may themselves be the results of another update. Suppose that there exists a KB $K_0 \in S_K$ and an update $U_0 \in S_U$ such that $TELL(K_0,U_0)=K_1$. Since it cannot possibly be the case that $TELL(K_0,U_0)=K_1$ and $TELL(K_0,U_0)=K_2$ at the same time, the two KBs K_1 and K_2 have slightly different semantics in the case of the normal reduction. An intuitively correct definition should overrule this case, by forcing that there exist no $K_0 \in S_K$, $U_0 \in S_U$ such that $TELL(K_0,U_0)=K_1$ or $TELL(K_0,U_0)=K_2$.

The notion of redundancy is simpler to express when it comes to updates. Suppose two updates $U_1, U_2 \in S_U$. If the update of any KB $K \in S_K$ with U_1 gives the same result as the update of K with U_2, then the two updates U_1, U_2 contain the same information. This requirement can be formally expressed as follows:

For some $U_1, U_2 \in S_U$, $U_1 \neq U_2$, it holds that:
$TELL(K,U_1)=TELL(K,U_2)$ for all $K \in S_K$

For queries, the same argumentation may be used. Two queries contain the same information if they result in the same answer, regardless of the current KB. Formally, this can be expressed as follows:

For some $Q_1, Q_2 \in S_Q$, $Q_1 \neq Q_2$, it holds that:
$ASK(K,Q_1)=ASK(K,Q_2)$ for all $K \in S_K$

The final form of normal redundancy refers to the Answer set. The notion here is slightly different, as each answer has a value of its own. This means that one cannot discard any answer as "useless", because it provides the user with some information regarding the KB. However, the Answer set may be unnecessarily large, if it contains answers that the system will not give under any circumstances. So, for an Answer set to contain redundancy, the following must be true:

For some $A \in S_A$ and for all $K \in S_K$, $Q \in S_Q$:
$ASK(K,Q) \neq A$

Summarizing the above notions, we have the following definition of normal redundancy:

Definition 5 Let $S=(S_K, S_U, S_Q, S_A, ASK, TELL)$ be a KR-scheme. We say that S:

- Contains *Normal KB redundancy* if there exist $K_1, K_2 \in S_K$, $K_1 \neq K_2$ such that:
  - For all $Q \in S_Q$, $ASK(K_1,Q)=ASK(K_2,Q)$
  - For all $U \in S_U$, $TELL(K_1,U)=TELL(K_2,U)$
  - For all $K_0 \in S_K$, $U_0 \in S_U$, $TELL(K_0,U_0) \neq K_1$
  - For all $K_0 \in S_K$, $U_0 \in S_U$, $TELL(K_0,U_0) \neq K_2$
- Contains *Normal Update redundancy* if there exist $U_1, U_2 \in S_U$, $U_1 \neq U_2$ such that for all $K \in S_K$, $TELL(K,U_1)=TELL(K,U_2)$.
- Contains *Normal Query redundancy* if there exist $Q_1, Q_2 \in S_Q$, $Q_1 \neq Q_2$ such that for all $K \in S_K$, $ASK(K,Q_1)=ASK(K,Q_2)$.

- Contains *Normal Answer redundancy* if there exist $A \in S_A$ such that for all $K \in S_K$, $Q \in S_Q$, $ASK(K,Q) \neq A$.
- Contains *Normal redundancy* if it contains any type of normal redundancy (KB, Update, Query or Answer).

Behavioral Redundancy

In the behavioral case, the semantics need not be preserved in the sense of normal case; we only need the behavior of two KBs to be the same to be classified as redundant. So, two KBs carry the same information if they give the same answers to all queries and, when updated with any update, or any sequence of updates, the results give the same answers to all queries as well. The constraint for the behavioral redundancy is based on the Behavior Preservation property and can be expressed as follows:

$$\text{For some } K_1, K_2 \in S_K,\ K_1 \neq K_2 \text{ it holds that:}$$
$$ASK(TELL^m(K_1,U_1,\ldots,U_m),Q) = ASK(TELL^m(K_2,U_1,\ldots,U_m),Q)$$
$$\text{for all } m \geq 0,\ U_1,\ldots,U_m \in S_U,\ Q \in S_Q$$

The case of the updates is similar. Two updates U_1, $U_2 \in S_U$ carry the same information (behaviorally) if, for any KB $K \in S_K$, the update results $TELL(K,U_1)$ and $TELL(K,U_2)$ carry the same (behavioral) information. This is true if they satisfy the property just expressed about behaviorally redundant KBs. So, formally, we can express behavioral update redundancy as follows:

$$\text{For some } U_1, U_2 \in S_U,\ U_1 \neq U_2 \text{ it holds that:}$$
$$ASK(TELL^{m+1}(K,U_1,U_1',\ldots,U_m'),Q) = ASK(TELL^{m+1}(K,U_2,U_1',\ldots,U_m'),Q)$$
$$\text{for all } m \geq 0,\ K \in S_K,\ U_1',\ldots,U_m' \in S_U,\ Q \in S_Q$$

As far as the query evaluation mechanism is concerned, the behavioral and the normal case are no different. The method used to preserve the semantics of queries is the same in both cases, so one would expect that for structures having to do solely with the query mechanism, the definition of normal and behavioral redundancy should coincide. Indeed, this is true; the definitions of the behavioral query and answer redundancy are the same as the respective definitions of the normal redundancy.

Summarizing the above notions, we have the following definition of behavioral redundancy:

Definition 6 Let $S=(S_K, S_U, S_Q, S_A, ASK, TELL)$ be a KR-scheme. We say that S:

- Contains *Behavioral KB redundancy* if there exist K_1, $K_2 \in S_K$, $K_1 \neq K_2$ such that $ASK(TELL^m(K_1,U_1,\ldots,U_m),Q) = ASK(TELL^m(K_2,U_1,\ldots,U_m),Q)$ for all $m \geq 0$, $U_1,\ldots,U_m \in S_U$, $Q \in S_Q$.
- Contains *Behavioral Update redundancy* if there exist U_1, $U_2 \in S_U$, $U_1 \neq U_2$ such that $ASK(TELL^{m+1}(K,U_1,U_1',\ldots,U_m'),Q) = ASK(TELL^{m+1}(K,U_2,U_1',\ldots,U_m'),Q)$ for all $m \geq 0$, $K \in S_K$, $U_1',\ldots,U_m' \in S_U$, $Q \in S_Q$.
- Contains *Behavioral Query redundancy* if it contains Normal Query redundancy.
- Contains *Behavioral Answer redundancy* if it contains Normal Answer redundancy.
- Contains *Behavioral redundancy* if it contains any type of behavioral redundancy (KB, Update, Query or Answer).

Results on Redundancy

Redundancy Can Be Eliminated

Now that we have a method to detect redundant elements, we would expect that for all schemes we could eliminate redundancy, resulting in simpler and equivalent schemes. We would also expect that no further elimination of elements should be possible. If more elements could be reduced without loss of expressive power, those elements should be redundant in the first place. Such considerations will be more formally expressed in Proposition 4 and Corollary 1. Before formulating and proving these propositions, some lemmas are required, mostly being special cases of these propositions.

Lemma 5 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme. Then, there exists a scheme $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ such that $S_{K1}\supseteq S_{K2}$, $S_{U1}=S_{U2}$, $S_{Q1}=S_{Q2}$, $S_{A1}=S_{A2}$, S_2 does not contain normal KB redundancy and $S_1\cong_r S_2$.

Proof

We define the set: $D=\{K\in S_{K1} \mid$ there exist no $K_0\in S_{K1}$, $U_0\in S_{U1}$: $TELL(K_0, U_0)=K\}$, ie the set of KBs that are not the results of any update in S_{K1}.

If $D=\varnothing$, then S_1 obviously contains no KB redundancy, so the schema $S_2=S_1$ satisfies the lemma.

If $D\neq\varnothing$, then we define (for the needs of this proof) an equivalence relation $\cong$ in D as follows:

For $K_1, K_2\in D$, $K_1\cong K_2$ if and only if

- For all $Q\in S_{Q1}$ $ASK_1(K_1,Q)=ASK_1(K_2,Q)$ and
- For all $U\in S_{U1}$ $TELL_1(K_1,U)=TELL_1(K_2,U)$

It can be trivially proven that:

- For all $K\in D$, $K\cong K$
- For any $K_1, K_2\in D$, if $K_1\cong K_2$ then $K_2\cong K_1$
- For any $K_1, K_2, K_3\in D$, if $K_1\cong K_2$ and $K_2\cong K_3$ then $K_1\cong K_3$

Thus $\cong$ is indeed an equivalence relation upon D, so D can be broken down into equivalence classes. For each class, we select one "representative" (arbitrarily) and define a function $g_0:D\rightarrow D$ such that for any $K\in D$ $g_0(K)=K'$, where $K'\cong K$ and K' is the representative of the equivalence class that K belongs to.

Moreover, we define the function $g:S_{K1}\rightarrow S_{K1}$, such that:

$g(K)=g_0(K)$ if $K\in D$

$g(K)=K$ if $K\notin D$

Let S_{K2} be the set: $S_{K2}=g(S_{K1})$, ie the range of g. Obviously $S_{K2}\subseteq S_{K1}$.

We define the KR-scheme: $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$, where:

- $S_{K2}=g(S_{K1})$
- $S_{U2}=S_{U1}$
- $S_{Q2}=S_{Q1}$
- $S_{A2}=S_{A1}$
- $ASK_2(K,Q)=ASK_1(K,Q)$ for all $K\in S_{K2}$, $Q\in S_{Q2}$
- $TELL_2(K,U)=TELL_1(K,U)$ for all $K\in S_{K2}$, $U\in S_{U2}$

Initially, we must prove that the functions ASK_2 and $TELL_2$ are properly defined, ie they give results in the sets S_{A2}, S_{K2} respectively. For ASK_2 this is obvious, as $S_{A2}=S_{A1}$. For $TELL_2$, take any $K\in S_{K2}$, $U\in S_{U2}$ and $K'=TELL_1(K,U)\in S_{K1}$. Since there exist $K\in S_{K2}\subseteq S_{K1}$, $U\in S_{U2}=S_{U1}$ such that $K'=TELL_1(K,U)$, we conclude that $K'\notin D$.

However, $K' \in S_{K1}$, thus $g(K')=K'$, so $K' \in g(S_{K1})=S_{K2}$. Thus $TELL_2(K,U) \in S_{K2}$ for all $K \in S_{K2}$, $U \in S_{U2}$.
Therefore, S_2 is properly defined. We will prove that it satisfies the lemma.
Firstly, suppose that it contains normal KB redundancy. So there exist K_1, $K_2 \in S_{K2}$, $K_1 \neq K_2$, such that:

1. For all $Q \in S_{Q2}$, $ASK_2(K_1,Q)=ASK_2(K_2,Q)$
2. For all $U \in S_{U2}$, $TELL_2(K_1,U)=TELL_2(K_2,U)$
3. For all $K_0 \in S_{K2}$, $U_0 \in S_{U2}$, $TELL_2(K_0,U_0) \neq K_1$
4. For all $K_0 \in S_{K2}$, $U_0 \in S_{U2}$, $TELL_2(K_0,U_0) \neq K_2$

Since K_1, $K_2 \in S_{K2}$, $S_{K2} \subseteq S_{K1}$, $S_{U1}=S_{U2}$ and by the third and fourth properties above we conclude that K_1, $K_2 \in D$.
Combining this fact with the fact that $S_{Q1}=S_{Q2}$, $S_{U1}=S_{U2}$ and the first and second properties above we conclude that $K_1 \cong K_2$.
Since K_1, $K_2 \in S_{K2} \cap D$ both K_1 and K_2 are representatives of an equivalence class in D; however $K_1 \cong K_2$, and this can only hold if $K_1=K_2$, which is overruled by the hypothesis. We have reached a contradiction, thus S_2 does not contain normal KB redundancy.
Secondly, we must prove that $S_1 \leq_r S_2$. We define the functions:

- $f_K: S_{K1} \rightarrow S_{K2}$, $f_K(K)=g(K)$ for all $K \in S_{K1}$
- $f_U: S_{U1} \rightarrow S_{U2}$, $f_U(U)=U$ for all $U \in S_{U1}$
- $f_Q: S_{Q1} \rightarrow S_{Q2}$, $f_Q(Q)=Q$ for all $Q \in S_{Q1}$
- $f_A: S_{A2} \rightarrow S_{A1}$, $f_A(A)=A$ for all $A \in S_{A2}$

Then for any $K \in S_{K1}$, $Q \in S_{Q1}$, we get by definition that:
$f_A(ASK_2(f_K(K),f_Q(Q)))=ASK_1(g(K),Q)$.
If $K \notin D$ then $g(K)=K$, while if $K \in D$ then $g(K)=g_0(K)$ which is the representative of K in its equivalence class. In either case: $ASK_1(K,Q)=ASK_1(g(K),Q)$ for all $Q \in S_{Q1}$. Thus:
$f_A(ASK_2(f_K(K),f_Q(Q)))=ASK_1(K,Q)$, so the Query Preservation property holds.
Moreover, for any K_1, $K_2 \in S_{K1}$, $U \in S_{U1}$, suppose that:
$K_2=TELL_1(K_1,U)$
Obviously $f_U(U)=U$ and $K_2 \notin D$, so $g(K_2)=K_2$.
Moreover, if $K_1 \notin D$ then $g(K_1)=K_1$, while if $K_1 \in D$ then $g(K)=g_0(K)$ which is the representative of K_1 in its equivalence class. In either case: $TELL_1(K_1,U)=TELL_1(g(K_1),U)$ for all $U \in S_{U1}$.
Combining the above:
$f_K(K_2)=g(K_2)=K_2=TELL_1(K_1,U)=TELL_1(g(K_1),U)=TELL_2(f_K(K),f_U(U))$, thus:
if $K_2=TELL_1(K_1,U)$ then $f_K(K_2)=TELL_2(f_K(K),f_U(U))$.
Suppose now that:
$f_K(K_2)=TELL_2(f_K(K_1),f_U(U))$.
Initially, we observe that:
$TELL_2(f_K(K_1),f_U(U))=TELL_1(g(K_1),U)$ by the definitions.
As before, we can prove that $TELL_1(K_1,U)=TELL_1(g(K_1),U)$ for all $U \in S_{U1}$, so:
$TELL_2(f_K(K_1),f_U(U))=TELL_1(K_1,U)$.
Applying this relation we get:
$g(K_2)=f_K(K_2)=TELL_2(f_K(K_1),f_U(U))=TELL_1(K_1,U)$, thus $g(K_2) \notin D$.
Now, if $K_2 \in D$ then $g(K_2)=g_0(K_2) \in D$, which is a contradiction, so $K_2 \notin D$, thus: $f_K(K_2)=g(K_2)=K_2$, thus:

if $f_K(K_2)=TELL_2(f_K(K_1),f_U(U))$ then $K_2=TELL_1(K_1,U)$, so the Update Preservation Property holds.
Thus, we conclude that $S_1 \leq_r S_2$.
Finally, we need to prove that $S_2 \leq_r S_1$. We define the functions:

- $f_K:S_{K2} \rightarrow S_{K1}$, $f_K(K)=K$ for all $K \in S_{K2}$
- $f_U:S_{U2} \rightarrow S_{U1}$, $f_U(U)=U$ for all $U \in S_{U2}$
- $f_Q:S_{Q2} \rightarrow S_{Q1}$, $f_Q(Q)=Q$ for all $Q \in S_{Q2}$
- $f_A:S_{A1} \rightarrow S_{A2}$, $f_A(A)=A$ for all $A \in S_{A1}$

Then:
$f_A(ASK_1(f_K(K),f_Q(Q)))=ASK_2(K,Q)$ by the definitions, so the Query Preservation property holds.
Moreover:
$K_2=f_K(K_2)$ for all $K_2 \in S_{K2}$ and
$TELL_1(f_K(K_1),f_U(U))=TELL_2(K_1,U)$ for all $K_1 \in S_{K2}$, $U \in S_{U2}$ by the definitions.
Using the two equalities above it is obvious that for all K_1, $K_2 \in S_{K2}$, $U \in S_{U2}$:
$K_2=TELL_2(K_1,U)$ if and only if $f_K(K_2)=TELL_1(f_K(K_1),f_U(U))$, so the Update Preservation property holds, thus: $S_2 \leq_r S_1$.
We conclude that $S_1 \cong_r S_2$ and S_2 has the properties that the lemma requires, so the proof is complete.

The above lemma shows that when a KR-scheme contains normal KB redundancy, we could eliminate some elements from the KB set and still get an equivalent KR-scheme. We could keep on removing elements until the KR-scheme no longer contains KB redundancy. In Corollary 1, we will prove that this is the simplest equivalent scheme we could possibly create; any further elimination of elements will result in less expressive KR-schemes, thus resulting to loss of expressive power. In the following three lemmas we will prove that the same property holds for the other types of redundancy as well:
Lemma 6 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme. Then, there exists a scheme $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ such that $S_{K1}=S_{K2}$, $S_{U1} \supseteq S_{U2}$, $S_{Q1}=S_{Q2}$, $S_{A1}=S_{A2}$, S_2 does not contain normal update redundancy and $S_1 \cong_r S_2$.
Proof
If S_1 does not contain normal update redundancy, then the KR-scheme $S_2=S_1$, obviously satisfies the lemma.
Suppose that S_1 contains normal update redundancy.
We define (for the needs of this proof) an equivalence relation $\cong$ in S_{U1} as follows:
for U_1, $U_2 \in S_{U1}$, $U_1 \cong U_2$ if and only if for all $K \in S_{K1}$ $TELL_1(K,U_1)=TELL_1(K,U_2)$.
It can be trivially proven that:

- For all $U \in S_{U1}$, $U \cong U$
- For any U_1, $U_2 \in S_{U1}$, if $U_1 \cong U_2$ then $U_2 \cong U_1$
- For any U_1, U_2, $U_3 \in S_{U1}$, if $U_1 \cong U_2$ and $U_2 \cong U_3$ then $U_1 \cong U_3$

Thus $\cong$ is indeed an equivalence relation upon S_{U1}, so S_{U1} can be broken down into equivalence classes. For each class, we select one "representative" (arbitrarily) and define a function $g:S_{U1} \rightarrow S_{U1}$ such that for any $U \in S_{U1}$ $g(U)=U'$, where $U' \cong U$ and U' is the representative of the equivalence class that U belongs to.
Let S_{U2} be the set: $S_{U2}=g(S_{U1})$, ie the range of g. Obviously $S_{U2} \subseteq S_{U1}$.
We define the KR-scheme: $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$, where:

- $S_{K2}=S_{K1}$
- $S_{U2}=g(S_{U1})$

- $S_{Q2}=S_{Q1}$
- $S_{A2}=S_{A1}$
- $ASK_2(K,Q)=ASK_1(K,Q)$ for all $K\in S_{K2}$, $Q\in S_{Q2}$
- $TELL_2(K,U)=TELL_1(K,U)$ for all $K\in S_{K2}$, $U\in S_{U2}$

The functions ASK_2 and $TELL_2$ are properly defined, ie they give results in the sets S_{A2}, S_{K2} respectively, because $S_{K2}=S_{K1}$ and $S_{A2}=S_{A1}$, so S_2 is properly defined. We will prove that it satisfies the lemma.

Firstly, suppose that it contains normal update redundancy. So there exist U_1, $U_2\in S_{U2}$, $U_1\neq U_2$, such that for all $K\in S_{K2}$, $TELL_2(K,U_1)=TELL_2(K,U_2)$ or, equivalently, $TELL_1(K,U_1)=TELL_1(K,U_2)$. In this case, $U_1\cong U_2$. Moreover U_1, $U_2\in S_{U2}$, so U_1 and U_2 are both representatives of the same equivalence class (since $U_1\cong U_2$) of S_{U1}, which is a contradiction, thus S_2 does not contain normal update redundancy.

Secondly, we must prove that $S_1\leq_r S_2$. We define the functions:

- $f_K:S_{K1}\rightarrow S_{K2}$, $f_K(K)=K$ for all $K\in S_{K1}$
- $f_U:S_{U1}\rightarrow S_{U2}$, $f_U(U)=g(U)$ for all $U\in S_{U1}$
- $f_Q:S_{Q1}\rightarrow S_{Q2}$, $f_Q(Q)=Q$ for all $Q\in S_{Q1}$
- $f_A:S_{A2}\rightarrow S_{A1}$, $f_A(A)=A$ for all $A\in S_{A2}$

Then for any $K\in S_{K1}$, $Q\in S_{Q1}$, we get by definition that:

$f_A(ASK_2(f_K(K),f_Q(Q)))=ASK_1(K,Q)$, so the Query Preservation property holds.

Moreover, for any K_1, $K_2\in S_{K1}$, $U\in S_{U1}$, it holds that:

$K_2=f_K(K_2)$ and

$TELL_2(f_K(K_1),f_U(U))=TELL_1(K_1,g(U))=TELL_1(K_1,U)$ because $U\cong g(U)$ by definition.

Using the two equalities above, we conclude that $K_2=TELL_1(K_1,U)$ if and only if $f_K(K_2)=TELL_2(f_K(K_1),f_U(U))$, so the Update Preservation Property holds.

Thus, we conclude that $S_1\leq_r S_2$.

Finally, we need to prove that $S_2\leq_r S_1$. We define the functions:

- $f_K:S_{K2}\rightarrow S_{K1}$, $f_K(K)=K$ for all $K\in S_{K2}$
- $f_U:S_{U2}\rightarrow S_{U1}$, $f_U(U)=U$ for all $U\in S_{U2}$
- $f_Q:S_{Q2}\rightarrow S_{Q1}$, $f_Q(Q)=Q$ for all $Q\in S_{Q2}$
- $f_A:S_{A1}\rightarrow S_{A2}$, $f_A(A)=A$ for all $A\in S_{A1}$

Then:

$f_A(ASK_1(f_K(K),f_Q(Q)))=ASK_2(K,Q)$ by the definitions, so the Query Preservation property holds.

Moreover:

$K_2=f_K(K_2)$ for all $K_2\in S_{K2}$ and

$TELL_1(f_K(K_1),f_U(U))=TELL_2(K_1,U)$ for all $K_1\in S_{K2}$, $U\in S_{U2}$ by the definitions.

Using the two equalities above it is obvious that for all K_1, $K_2\in S_{K2}$, $U\in S_{U2}$:

$K_2=TELL_2(K_1,U)$ if and only if $f_K(K_2)=TELL_1(f_K(K_1),f_U(U))$, so the Update Preservation property holds, thus: $S_2\leq_r S_1$.

We conclude that $S_1\cong_r S_2$ and S_2 has the properties that the lemma requires, so the proof is complete.

Lemma 7 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme. Then, there exists a scheme $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ such that $S_{K1}=S_{K2}$, $S_{U1}=S_{U2}$, $S_{Q1}\supseteq S_{Q2}$, $S_{A1}=S_{A2}$, S_2 does not contain normal query redundancy and $S_1\cong_r S_2$.

Proof

If S_1 does not contain normal query redundancy, then the KR-scheme $S_2=S_1$, obviously satisfies the lemma.

Suppose that S_1 contains normal query redundancy.
We define (for the needs of this proof) an equivalence relation $\cong$ in S_{Q1} as follows:
for $Q_1, Q_2 \in S_{Q1}$, $Q_1 \cong Q_2$ if and only if for all $K \in S_{K1}$ $ASK_1(K,Q_1)=ASK_1(K,Q_2)$.
It can be trivially proven that:

- For all $Q \in S_{Q1}$, $Q \cong Q$
- For any $Q_1, Q_2 \in S_{Q1}$, if $Q_1 \cong Q_2$ then $Q_2 \cong Q_1$
- For any $Q_1, Q_2, Q_3 \in S_{Q1}$, if $Q_1 \cong Q_2$ and $Q_2 \cong Q_3$ then $Q_1 \cong Q_3$

Thus $\cong$ is indeed an equivalence relation upon S_{Q1}, so S_{Q1} can be broken down into equivalence classes. For each class, we select one "representative" (arbitrarily) and define a function $g:S_{Q1} \rightarrow S_{Q1}$ such that for any $Q \in S_{Q1}$ $g(Q)=Q'$, where $Q' \cong Q$ and Q' is the representative of the equivalence class that Q belongs to.
Let S_{Q2} be the set: $S_{Q2}=g(S_{Q1})$, ie the range of g. Obviously $S_{Q2} \subseteq S_{Q1}$.
We define the KR-scheme: $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$, where:

- $S_{K2}=S_{K1}$
- $S_{U2}=S_{U1}$
- $S_{Q2}=g(S_{Q1})$
- $S_{A2}=S_{A1}$
- $ASK_2(K,Q)=ASK_1(K,Q)$ for all $K \in S_{K2}$, $Q \in S_{Q2}$
- $TELL_2(K,U)=TELL_1(K,U)$ for all $K \in S_{K2}$, $U \in S_{U2}$

The functions ASK_2 and $TELL_2$ are properly defined, ie they give results in the sets S_{A2}, S_{K2} respectively, because $S_{K2}=S_{K1}$ and $S_{A2}=S_{A1}$, so S_2 is properly defined. We will prove that it satisfies the lemma.
Firstly, suppose that it contains normal query redundancy. So there exist $Q_1, Q_2 \in S_{Q2}$, $Q_1 \neq Q_2$, such that for all $K \in S_{K2}$, $ASK_1(K,Q_1)=ASK_1(K,Q_2)$. In this case, $Q_1 \cong Q_2$. Moreover $Q_1, Q_2 \in S_{Q2}$, so Q_1 and Q_2 are both representatives of the same equivalence class (since $Q_1 \cong Q_2$) of S_{Q1}, which is a contradiction, thus S_2 does not contain normal query redundancy.
Secondly, we must prove that $S_1 \leq_r S_2$. We define the functions:

- $f_K:S_{K1} \rightarrow S_{K2}$, $f_K(K)=K$ for all $K \in S_{K1}$
- $f_U:S_{U1} \rightarrow S_{U2}$, $f_U(U)=U$ for all $U \in S_{U1}$
- $f_Q:S_{Q1} \rightarrow S_{Q2}$, $f_Q(Q)=g(Q)$ for all $Q \in S_{Q1}$
- $f_A:S_{A2} \rightarrow S_{A1}$, $f_A(A)=A$ for all $A \in S_{A2}$

Then for any $K \in S_{K1}$, $Q \in S_{Q1}$:
$f_A(ASK_2(f_K(K),f_Q(Q)))=ASK_1(K,g(Q))=ASK_1(K,Q)$ (by the definitions and the fact that $Q \cong g(Q)$ by definition), so the Query Preservation property holds.
Moreover, for any $K_1, K_2 \in S_{K1}$, $U \in S_{U1}$, it holds that:
$K_2=f_K(K_2)$ and
$TELL_2(f_K(K_1),f_U(U))=TELL_1(K_1,U)$ by the definitions.
Using the two equalities above, we conclude that $K_2=TELL_1(K_1,U)$ if and only if $f_K(K_2)=TELL_2(f_K(K_1),f_U(U))$, so the Update Preservation Property holds.
Thus, we conclude that $S_1 \leq_r S_2$.
Finally, we need to prove that $S_2 \leq_r S_1$. We define the functions:

- $f_K:S_{K2} \rightarrow S_{K1}$, $f_K(K)=K$ for all $K \in S_{K2}$
- $f_U:S_{U2} \rightarrow S_{U1}$, $f_U(U)=U$ for all $U \in S_{U2}$
- $f_Q:S_{Q2} \rightarrow S_{Q1}$, $f_Q(Q)=Q$ for all $Q \in S_{Q2}$
- $f_A:S_{A1} \rightarrow S_{A2}$, $f_A(A)=A$ for all $A \in S_{A1}$

Then:

$f_A(ASK_1(f_K(K),f_Q(Q)))=ASK_2(K,Q)$ by the definitions, so the Query Preservation property holds.
Moreover:
$K_2=f_K(K_2)$ for all $K_2 \in S_{K2}$ and
$TELL_1(f_K(K_1),f_U(U))=TELL_2(K_1,U)$ for all $K_1 \in S_{K2}$, $U \in S_{U2}$ by the definitions.
Using the two equalities above it is obvious that for all K_1, $K_2 \in S_{K2}$, $U \in S_{U2}$:
$K_2=TELL_2(K_1,U)$ if and only if $f_K(K_2)=TELL_1(f_K(K_1),f_U(U))$, so the Update Preservation property holds, thus: $S_2 \leq_r S_1$.
We conclude that $S_1 \cong_r S_2$ and S_2 has the properties that the lemma requires, so the proof is complete.

Lemma 8 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme. Then, there exists a scheme $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ such that $S_{K1}=S_{K2}$, $S_{U1}=S_{U2}$, $S_{Q1}=S_{Q2}$, $S_{A1} \supseteq S_{A2}$, S_2 does not contain normal answer redundancy and $S_1 \cong_r S_2$.
Proof
We define the set $S_0=\{A \in S_{A1} \mid$ there exist $K \in S_{K1}$, $Q \in S_{Q1}$ such that $ASK_1(K,Q)=A\}$, ie the range of ASK_1 and the KR-scheme: $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$, where:

- $S_{K2}=S_{K1}$
- $S_{U2}=S_{U1}$
- $S_{Q2}=S_{Q1}$
- $S_{A2}=S_0$
- $ASK_2(K,Q)=ASK_1(K,Q)$ for all $K \in S_{K2}$, $Q \in S_{Q2}$
- $TELL_2(K,U)=TELL_1(K,U)$ for all $K \in S_{K2}$, $U \in S_{U2}$

The functions ASK_2 and $TELL_2$ are properly defined, ie they give results in the sets S_{A2}, S_{K2} respectively; the definition of S_{A2}, ASK_2 and $TELL_2$ along with the fact that $S_{K2}=S_{K1}$ guarantees that. So S_2 is properly defined. We will prove that it satisfies the lemma.
Firstly, it does not contain normal answer redundancy, as is obvious by the definition of S_{A2} and ASK_2.
Secondly, we must prove that $S_1 \leq_r S_2$. We define the functions:

- $f_K: S_{K1} \rightarrow S_{K2}$, $f_K(K)=K$ for all $K \in S_{K1}$
- $f_U: S_{U1} \rightarrow S_{U2}$, $f_U(U)=U$ for all $U \in S_{U1}$
- $f_Q: S_{Q1} \rightarrow S_{Q2}$, $f_Q(Q)=Q$ for all $Q \in S_{Q1}$
- $f_A: S_{A2} \rightarrow S_{A1}$, $f_A(A)=A$ for all $A \in S_{A2}$

Then for any $K \in S_{K1}$, $Q \in S_{Q1}$:
$f_A(ASK_2(f_K(K),f_Q(Q)))=ASK_1(K,Q)$ (by the definitions), so the Query Preservation property holds.
Moreover, for any K_1, $K_2 \in S_{K1}$, $U \in S_{U1}$, it holds that:
$K_2=f_K(K_2)$ and
$TELL_2(f_K(K_1),f_U(U))=TELL_1(K_1,U)$ by the definitions.
Using the two equalities above, we conclude that $K_2=TELL_1(K_1,U)$ if and only if $f_K(K_2)=TELL_2(f_K(K_1),f_U(U))$, so the Update Preservation Property holds.
Thus, we conclude that $S_1 \leq_r S_2$.
Finally, we need to prove that $S_2 \leq_r S_1$. We arbitrarily select any $A_0 \in S_{A2}$. We define the functions:

- $f_K: S_{K2} \rightarrow S_{K1}$, $f_K(K)=K$ for all $K \in S_{K2}$
- $f_U: S_{U2} \rightarrow S_{U1}$, $f_U(U)=U$ for all $U \in S_{U2}$

- $f_Q:S_{Q2}\rightarrow S_{Q1}$, $f_Q(Q)=Q$ for all $Q\in S_{Q2}$
- $f_A:S_{A1}\rightarrow S_{A2}$, $f_A(A)=A$ for all $A\in S_{A2}$ and $f_A(A)=A_0$ for all $A\in S_{A1}\setminus S_{A2}$

Then, since $ASK_1(f_K(K),f_Q(Q))\in S_0=S_{A2}$ by definition, we get that:
$f_A(ASK_1(f_K(K),f_Q(Q)))=ASK_2(K,Q)$ by the definitions, so the Query Preservation property holds.
Moreover:
$K_2=f_K(K_2)$ for all $K_2\in S_{K2}$ and
$TELL_1(f_K(K_1),f_U(U))=TELL_2(K_1,U)$ for all $K_1\in S_{K2}$, $U\in S_{U2}$ by the definitions.
Using the two equalities above it is obvious that for all K_1, $K_2\in S_{K2}$, $U\in S_{U2}$:
$K_2=TELL_2(K_1,U)$ if and only if $f_K(K_2)=TELL_1(f_K(K_1),f_U(U))$, so the Update Preservation property holds, thus: $S_2\leq_r S_1$.
We conclude that $S_1\cong_r S_2$ and S_2 has the properties that the lemma requires, so the proof is complete.

Proposition 3 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme. Then, there exists a scheme $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ such that $S_{K1}\supseteq S_{K2}$, $S_{U1}\supseteq S_{U2}$, $S_{Q1}\supseteq S_{Q2}$, $S_{A1}\supseteq S_{A2}$, S_2 does not contain normal redundancy and $S_1\cong_r S_2$.
Proof
The proof is a simple application of the last four lemmas and the transitivity property of $\cong_r$ (Proposition 2).

Proposition 3 actually says that we lose nothing by restricting our attention to non-redundant schemes only. When a KR-scheme does not contain normal redundancy, then it satisfies this proposition itself. However, if it does contain some type of normal redundancy, then we can get rid of the redundant elements without loss of expressive power. Regarding the answer set, useless answers can be removed at no cost.

The same property can be proved for the behavioral case. If a KR-scheme contains behavioral redundancy, then we can create a behaviorally equivalent KR-scheme that does not. The proof of this proposition will be done in a similar way, breaking it into four lemmas as before:
Lemma 9 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme. Then, there exists a scheme $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ such that $S_{K1}\supseteq S_{K2}$, $S_{U1}=S_{U2}$, $S_{Q1}=S_{Q2}$, $S_{A1}=S_{A2}$, S_2 does not contain behavioral KB redundancy and $S_1\cong_b S_2$.
Proof
Initially, we define (for the needs of this proof) an equivalence relation $\cong$ in S_{K1} as follows:
For K_1, $K_2\in S_{K1}$, $K_1\cong K_2$ if and only if for all $m\geq 0$, $U_1,\ldots,U_m\in S_{U1}$, $Q\in S_{Q1}$ it holds that:
$ASK_1(TELL_1^m(K_1,U_1,\ldots,U_m),Q)=ASK_1(TELL_1^m(K_2,U_1,\ldots,U_m),Q)$.
It can be trivially proven that:
- For all $K\in D$, $K\cong K$
- For any K_1, $K_2\in D$, if $K_1\cong K_2$ then $K_2\cong K_1$
- For any K_1, K_2, $K_3\in D$, if $K_1\cong K_2$ and $K_2\cong K_3$ then $K_1\cong K_3$

Thus $\cong$ is indeed an equivalence relation upon S_{K1}, so S_{K1} can be broken down into equivalence classes. For each class, we select one "representative" (arbitrarily) and define a function $g:S_{K1}\rightarrow S_{K1}$ such that for any $K\in S_{K1}$ $g(K)=K'$, where $K'\cong K$ and K' is the representative of the equivalence class that K belongs to.
Let S_{K2} be the set: $S_{K2}=g(S_{K1})$, ie the range of g. Obviously $S_{K2}\subseteq S_{K1}$.
We define the KR-scheme: $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$, where:

- $S_{K2}=g(S_{K1})$
- $S_{U2}=S_{U1}$
- $S_{Q2}=S_{Q1}$
- $S_{A2}=S_{A1}$
- $ASK_2(K,Q)=ASK_1(K,Q)$ for all $K\in S_{K2}$, $Q\in S_{Q2}$
- $TELL_2(K,U)=g(TELL_1(K,U))$ for all $K\in S_{K2}$, $U\in S_{U2}$

Initially, we must prove that the functions ASK_2 and $TELL_2$ are properly defined, ie they give results in the sets S_{A2}, S_{K2} respectively. For ASK_2 this is obvious, as $S_{A2}=S_{A1}$. For $TELL_2$, the definition of the g function guarantees that.
Therefore, S_2 is properly defined. We will prove that it satisfies the lemma.
Before that, we can prove that for all $m\geq 0$, $K\in S_{K2}$, $U_1,\ldots,U_m\in S_{U2}$, it holds that:
$TELL_2^m(K,U_1,\ldots,U_m)=g(TELL_1^m(K,U_1,\ldots,U_m))$.
To prove this relation, we will use induction on m.
For m=0 we have:
$TELL_2^0(K)=$
$=K=$ (by definition)
$=g(K)=$ ($K\in S_{K2}=g(S_{K1})$ so K is the representative of its equivalence class, so $g(K)=K$)
$=g(TELL_1^0(K))$ (by definition)
Suppose that the result holds for m=0,1,…,n. We will prove that it holds for m=n+1.
Indeed:
$TELL_2^{n+1}(K,U_1,\ldots,U_{n+1})=$
$TELL_2(TELL_2^n(K,U_1,\ldots,U_n),U_{n+1})=$ (by definition)
$=TELL_2(g(TELL_1^n(K,U_1,\ldots,U_n)),U_{n+1})=$ (by induction, for m=n)
$=g(TELL_1(g(TELL_1^n(K,U_1,\ldots,U_n)),U_{n+1}))=$ (by induction, for m=1)
$=g(TELL_1(TELL_1^n(K,U_1,\ldots,U_n),U_{n+1}))=$
(by the fact that: $g(TELL_1^n(K,U_1,\ldots,U_n))\cong TELL_1^n(K,U_1,\ldots,U_n)$)
$=g(TELL_1^{n+1}(K,U_1,\ldots,U_{n+1}))$ by definition, so the relation holds.
Now, suppose that S_2 contains behavioral KB redundancy. So there exist K_1, $K_2\in S_{K2}$, $K_1\neq K_2$, such that for all $m\geq 0$, $U_1,\ldots,U_m\in S_{U2}$, $Q\in S_{Q2}$ it holds that:
$ASK_2(TELL_2^m(K_1,U_1,\ldots,U_m),Q)=ASK_2(TELL_2^m(K_2,U_1,\ldots,U_m),Q)$.
By the definitions and the above result on iterated updates, this is equivalent to:
$ASK_1(g(TELL_1^m(K_1,U_1,\ldots,U_m)),Q)=ASK_1(g(TELL_1^m(K_2,U_1,\ldots,U_m)),Q)$.
However, by the definition of g:
$g(TELL_1^m(K_1,U_1,\ldots,U_m))\cong TELL_1^m(K_1,U_1,\ldots,U_m)$ and
$g(TELL_1^m(K_2,U_1,\ldots,U_m))\cong TELL_1^m(K_2,U_1,\ldots,U_m)$ so the last equality is equivalent to:
$ASK_1(TELL_1^m(K_1,U_1,\ldots,U_m),Q)=ASK_1(TELL_1^m(K_2,U_1,\ldots,U_m),Q)$ which means that $K_1\cong K_2$ in S_{K1}. But, K_1, $K_2\in S_{K2}$ so both K_1 and K_2 are representatives of the same equivalence class in S_{K1}, which is a contradiction by definition. Thus S_2 does not contain behavioral KB redundancy.
Secondly, we must prove that $S_1\leq_b S_2$. We define the functions:

- $f_K:S_{K1}\rightarrow S_{K2}$, $f_K(K)=g(K)$ for all $K\in S_{K1}$
- $f_U:S_{U1}\rightarrow S_{U2}$, $f_U(U)=U$ for all $U\in S_{U1}$
- $f_Q:S_{Q1}\rightarrow S_{Q2}$, $f_Q(Q)=Q$ for all $Q\in S_{Q1}$
- $f_A:S_{A2}\rightarrow S_{A1}$, $f_A(A)=A$ for all $A\in S_{A2}$

We must prove that the Behavior Preservation property holds.
Before that, we can prove that for all $m\geq 0$, $K\in S_{K1}$, $U_1,\ldots,U_m\in S_{U1}$, $Q\in S_{Q1}$:
$TELL_2^m(f_K(K),f_U(U_1),\ldots,f_U(U_m))=f_K(TELL_1^m(K,U_1,\ldots,U_m))$.
To prove this relation, we will use induction on m.
For m=0 using the definitions we have:

$TELL_2^0(f_K(K))=f_K(K)=f_K(TELL_1^0(K))$, so the result holds for m=0.
Suppose that the result holds for m=0,1,…,n. We will prove that it holds for m=n+1.
Indeed:
$TELL_2^{n+1}(f_K(K),f_U(U_1),\ldots,f_U(U_{n+1}))=$
$=TELL_2(TELL_2^n(f_K(K),f_U(U_1),\ldots,f_U(U_n)),f_U(U_{n+1}))=$ (by definition)
$=TELL_2(f_K(TELL_1^n(K,U_1,\ldots,U_n)),f_U(U_{n+1}))=$ (by induction, for m=n)
$=g(TELL_1(f_K(TELL_1^n(K,U_1,\ldots,U_n)),U_{n+1}))=$ (by definitions)
$=f_K(TELL_1(g(TELL_1^n(K,U_1,\ldots,U_n)),U_{n+1}))=$ (by definitions)
$=f_K(TELL_1(TELL_1^n(K,U_1,\ldots,U_n),U_{n+1}))=$
(by the fact that: $g(TELL_1^n(K,U_1,\ldots,U_n))\cong TELL_1^n(K,U_1,\ldots,U_n)$)
$=f_K(TELL_1^{n+1}(K,U_1,\ldots,U_{n+1}))$ (by definitions). Thus the above result holds.
To prove that the Behavior Preservation property holds, we will use induction on m.
For m=0, for any $K\in S_{K1}$, $Q\in S_{Q1}$:
$f_A(ASK_2(f_K(K),f_Q(Q)))=$
$=ASK_1(g(K),Q)=$ (by the definitions)
$=ASK_1(K,Q)$ (by the fact that $K\cong g(K)$)
Suppose that it holds for m=0,1,…,n. We will prove that it holds for m=n+1.
$f_A(ASK_2(TELL_2^{n+1}(f_K(K),f_U(U_1),\ldots,f_U(U_{n+1})),f_Q(Q)))=$
$=ASK_1(TELL_2^{n+1}(f_K(K),f_U(U_1),\ldots,f_U(U_{n+1})),Q)=$ (by the definitions)
$=ASK_1(f_K(TELL_1^{n+1}(K,U_1,\ldots,U_{n+1})),Q)=$ (by the above result)
$=ASK_1(g(TELL_1^{n+1}(K,U_1,\ldots,U_{n+1})),Q)=$ (by definition)
$=ASK_1(TELL_1^{n+1}(K,U_1,\ldots,U_{n+1}),Q)$,
because $g(TELL_1^{n+1}(K,U_1,\ldots,U_{n+1}))\cong TELL_1^{n+1}(K,U_1,\ldots,U_{n+1})$, so the induction is complete and the Behavior Preservation property holds.
We conclude that: $S_1\leq_b S_2$.
Finally, we need to prove that $S_2\leq_b S_1$. We define the functions:

- $f_K:S_{K2}\rightarrow S_{K1}$, $f_K(K)=K$ for all $K\in S_{K2}$
- $f_U:S_{U2}\rightarrow S_{U1}$, $f_U(U)=U$ for all $U\in S_{U2}$
- $f_Q:S_{Q2}\rightarrow S_{Q1}$, $f_Q(Q)=Q$ for all $Q\in S_{Q2}$
- $f_A:S_{A1}\rightarrow S_{A2}$, $f_A(A)=A$ for all $A\in S_{A1}$

We must prove that the Behavior Preservation property holds. Indeed:
$ASK_2(TELL_2^m(K,U_1,\ldots,U_m),Q)=$
$ASK_1(g(TELL_1^m(K,U_1,\ldots,U_m)),Q))=$ (by the result on iterated updates and definitions)
$=ASK_1(TELL_1^m(K,U_1,\ldots,U_m),Q)=$
(by the fact that $g(TELL_1^m(K,U_1,\ldots,U_m))\cong TELL_1^m(K,U_1,\ldots,U_m)$)
$=f_A(ASK_1(TELL_1^m(f_K(K),f_U(U_1),\ldots,f_U(U_m)),f_Q(Q)))$ by definition, so the Behavior Preservation property holds, thus: $S_2\leq_b S_1$.
We conclude that $S_1\cong_b S_2$ and S_2 has the properties that the lemma requires, so the proof is complete.

Lemma 10 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme. Then, there exists a scheme $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ such that $S_{K1}=S_{K2}$, $S_{U1}\supseteq S_{U2}$, $S_{Q1}=S_{Q2}$, $S_{A1}=S_{A2}$, S_2 does not contain behavioral update redundancy and $S_1\cong_b S_2$.
Proof
Initially, we define (for the needs of this proof) an equivalence relation $\cong$ in S_{U1} as follows:
For $U_1, U_2\in S_{U1}$, $U_1\cong U_2$ if and only if for all $m\geq 0$, $K\in S_{K1}$, $U_1',\ldots,U_m'\in S_{U1}$, $Q\in S_{Q1}$ it holds that:

$ASK_1(TELL_1^{m+1}(K,U_1,U_1',\ldots,U_m'),Q)=ASK_1(TELL_1^{m+1}(K,U_2,U_1',\ldots,U_m'),Q)$.
It can be trivially proven that:

- For all $U\in S_{U1}$, $U\cong U$
- For any U_1, $U_2\in S_{U1}$, if $U_1\cong U_2$ then $U_2\cong U_1$
- For any U_1, U_2, $U_3\in S_{U1}$, if $U_1\cong U_2$ and $U_2\cong U_3$ then $U_1\cong U_3$

Thus $\cong$ is indeed an equivalence relation upon S_{U1}, so S_{U1} can be broken down into equivalence classes. For each class, we select one "representative" (arbitrarily) and define a function $g:S_{U1}\rightarrow S_{U1}$ such that for any $U\in S_{U1}$ $g(U)=U'$, where $U'\cong U$ and U' is the representative of the equivalence class that U belongs to.
Let S_{U2} be the set: $S_{U2}=g(S_{U1})$, ie the range of g. Obviously $S_{U2}\subseteq S_{U1}$.
We define the KR-scheme: $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$, where:

- $S_{K2}=S_{K1}$
- $S_{U2}=g(S_{U1})$
- $S_{Q2}=S_{Q1}$
- $S_{A2}=S_{A1}$
- $ASK_2(K,Q)=ASK_1(K,Q)$ for all $K\in S_{K2}$, $Q\in S_{Q2}$
- $TELL_2(K,U)=TELL_1(K,U)$ for all $K\in S_{K2}$, $U\in S_{U2}$

Initially, it is obvious that the functions ASK_2 and $TELL_2$ are properly defined, ie they give results in the sets S_{A2}, S_{K2} respectively, since $S_{A2}=S_{A1}$ and $S_{K2}=S_{K1}$.
Therefore, S_2 is properly defined. We will prove that it satisfies the lemma.
Firstly, we must prove that S_2 does not contain behavioral update redundancy. We will need the following result: for all $m\geq 0$, $K\in S_{K1}$, $U_1,\ldots,U_m\in S_{U1}$, $Q\in S_{Q1}$, it holds that: $ASK_1(TELL_1^m(K,U_1,\ldots,U_m),Q)=ASK_1(TELL_1^m(K,g(U_1),\ldots,g(U_m)),Q)$. We will prove this fact by induction on m.
For m=0, the proof is obvious.
Suppose that the result holds for m=0,1,…,n. We will prove it for m=n+1. Indeed:
$ASK_1(TELL_1^{n+1}(K,g(U_1),\ldots,g(U_{n+1})),Q)=$
$=ASK_1(TELL_1(TELL_1^n(K,g(U_1),\ldots,g(U_n)),g(U_{n+1})),Q)=$ (by definition)
$=ASK_1(TELL_1(TELL_1^n(K,U_1,\ldots,U_n),g(U_{n+1})),Q)=$ (by induction, for m=n)
$=ASK_1(TELL_1(TELL_1^n(K,U_1,\ldots,U_n),U_{n+1}),Q)=$ (by induction, for m=1)
$=ASK_1(TELL_1^{n+1}(K,U_1,\ldots,U_{n+1}),Q)$ by definition, so the result holds.
Using this result, we can prove that S_2 does not contain behavioral update redundancy. Suppose any two updates U_1, $U_2\in S_{U1}$. By the result above we conclude that, for any $m\geq 0$, $K\in S_{K1}$, $U_1',\ldots,U_m'\in S_{U1}$, $Q\in S_{Q1}$ the relation:
$ASK_1(TELL_1^{m+1}(K,U_1,U_1',\ldots,U_m'),Q)=ASK_1(TELL_1^{m+1}(K,U_2,U_1',\ldots,U_m'),Q)$
holds if and only if:
$ASK_1(TELL_1^{m+1}(K,U_1,g(U_1'),\ldots,g(U_m')),Q)=ASK_1(TELL_1^{m+1}(K,U_2,g(U_1'),\ldots,g(U_m')),Q)$
This relation, along with the definition of the sets S_{K2}, S_{U2}, S_{Q2}, implies that for any U_1, $U_2\in S_{U1}$, $U_1\cong U_2$ if and only if for all $m\geq 0$, $K\in S_{K2}$, $U_1',\ldots,U_m'\in S_{U2}$, $Q\in S_{Q2}$:
$ASK_1(TELL_1^{m+1}(K,U_1,U_1',\ldots,U_m'),Q)=ASK_1(TELL_1^{m+1}(K,U_2,U_1',\ldots,U_m'),Q)$
Suppose now that S_2 contains behavioral update redundancy. So there exist U_1, $U_2\in S_{U2}$, $U_1\neq U_2$, such that for all $m\geq 0$, $K\in S_{K2}$, $U_1',\ldots,U_m'\in S_{U2}$, $Q\in S_{Q2}$ it holds that:
$ASK_2(TELL_2^{m+1}(K,U_1,U_1',\ldots,U_m'),Q)=ASK_2(TELL_2^{m+1}(K,U_2,U_1',\ldots,U_m'),Q)$.
By the definition of ASK_2, $TELL_2$, we get:
$ASK_1(TELL_1^{m+1}(K,U_1,U_1',\ldots,U_m'),Q)=ASK_1(TELL_1^{m+1}(K,U_2,U_1',\ldots,U_m'),Q)$
Which means that for the updates U_1, $U_2\in S_{U2}\subseteq S_{U1}$ it holds that $U_1\cong U_2$. So, there exist two updates U_1, $U_2\in S_{U2}$ which are the representatives of the same equivalence class

of S_{U1}, which is a contradiction. Thus S_2 does not contain behavioral update redundancy.
Secondly, we must prove that $S_1 \leq_b S_2$. We define the functions:

- $f_K: S_{K1} \rightarrow S_{K2}$, $f_K(K)=K$ for all $K \in S_{K1}$
- $f_U: S_{U1} \rightarrow S_{U2}$, $f_U(U)=g(U)$ for all $U \in S_{U1}$
- $f_Q: S_{Q1} \rightarrow S_{Q2}$, $f_Q(Q)=Q$ for all $Q \in S_{Q1}$
- $f_A: S_{A2} \rightarrow S_{A1}$, $f_A(A)=A$ for all $A \in S_{A2}$

We must prove that the Behavior Preservation property holds. Indeed:
$f_A(ASK_2(TELL_2^m(f_K(K),f_U(U_1),\ldots,f_U(U_m)),f_Q(Q)))=$
$=ASK_1(TELL_1^m(K,g(U_1),\ldots,g(U_m)),Q)=$ (by the definitions)
$=ASK_1(TELL_1^m(K,U_1,\ldots,U_m),Q)$ by the result above, so the Behavior Preservation property holds.
We conclude that: $S_1 \leq_b S_2$.
Finally, we need to prove that $S_2 \leq_b S_1$. We define the functions:

- $f_K: S_{K2} \rightarrow S_{K1}$, $f_K(K)=K$ for all $K \in S_{K2}$
- $f_U: S_{U2} \rightarrow S_{U1}$, $f_U(U)=U$ for all $U \in S_{U2}$
- $f_Q: S_{Q2} \rightarrow S_{Q1}$, $f_Q(Q)=Q$ for all $Q \in S_{Q2}$
- $f_A: S_{A1} \rightarrow S_{A2}$, $f_A(A)=A$ for all $A \in S_{A1}$

We must prove that the Behavior Preservation property holds. Indeed:
$ASK_2(TELL_2^m(K,U_1,\ldots,U_m),Q)=$
$=ASK_1(TELL_1^m(f_K(K),g(U_1),\ldots,g(U_m)),f_Q(Q))=$ (by the definitions and the above result)
$=f_A(ASK_1(TELL_1^m(f_K(K),f_U(U_1),\ldots,f_U(U_m)),f_Q(Q)))$ by definition, so the Behavior Preservation property holds, thus: $S_2 \leq_b S_1$.
We conclude that $S_1 \cong_b S_2$ and S_2 has the properties that the lemma requires, so the proof is complete.

Lemma 11 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme. Then, there exists a scheme $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ such that $S_{K1}=S_{K2}$, $S_{U1}=S_{U2}$, $S_{Q1} \supseteq S_{Q2}$, $S_{A1}=S_{A2}$, S_2 does not contain behavioral query redundancy and $S_1 \cong_b S_2$.
Proof
The proof occurs trivially by combining the results of Lemma 7, Proposition 1 and the fact that a KR-scheme contains behavioral query redundancy if and only if it contains normal query redundancy.

Lemma 12 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme. Then, there exists a scheme $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ such that $S_{K1}=S_{K2}$, $S_{U1}=S_{U2}$, $S_{Q1}=S_{Q2}$, $S_{A1} \supseteq S_{A2}$, S_2 does not contain behavioral answer redundancy and $S_1 \cong_b S_2$.
Proof
The proof occurs trivially by combining the results of Lemma 8, Proposition 1 and the fact that a KR-scheme contains behavioral answer redundancy if and only if it contains normal answer redundancy.

Proposition 4 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme. Then, there exists a scheme $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ such that $S_{K1} \supseteq S_{K2}$, $S_{U1} \supseteq S_{U2}$, $S_{Q1} \supseteq S_{Q2}$, $S_{A1} \supseteq S_{A2}$, S_2 does not contain behavioral redundancy and $S_1 \cong_b S_2$.
Proof
The proof is a simple application of the last four lemmas and the transitivity property of $\cong_b$ (Proposition 2).

This proposition is the respective result of Proposition 3 for the behavioral case. The two propositions, put together, show that redundant elements can be safely eliminated, without loss of expressive power.

One example of such a possibility is Dalal's work ([2], [3]), an algorithm for revising propositional KBs. Dalal's papers described the update mechanism, but the underlying KR-scheme was implicitly described. Propositional KBs were used, but Dalal assumed that a KB can be described by one logical proposition, unlike the AGM model where sets of formulas are used. Thus, for Dalal's scheme, $S_K=S_U=S_Q=L^*$, where L^* is the set of all well formed formulas of the underlying propositional language L. We assume Closed World Assumption, so $S_A=\{YES, NO\}$. The ASK function is the CWA query answering function, as described in the example regarding the Closed World Assumption model. The TELL function is the belief revision algorithm described in Dalal's papers.

As proved in [8], Dalal's algorithm satisfies the AGM postulates [1], so it satisfies the AGM postulate of preservation. This principle states that two equivalent KBs give equivalent results to equivalent updates. By the definition of the ASK function, two equivalent KBs give the same results to equivalent queries. Combining these two properties, we may reasonably assume that Dalal's model does contain behavioral redundancy; we could get rid of the equivalent formulas (keeping one per class) and lose nothing as far as the model's expressive power is concerned (behaviorally).

Indeed, for any two KBs K_1, $K_2 \in S_K$, with $K_1 \equiv K_2$ (where the symbol "≡" stands for logical equivalence) and for any Q_1, $Q_2 \in S_Q$ with $Q_1 \equiv Q_2$ it holds that $ASK(K_1,Q_1)=ASK(K_2,Q_2)$. Moreover, since $K_1 \equiv K_2$, for any U_1, $U_1' \in S_U$ with $U_1 \equiv U_1'$ it holds that $TELL(K_1,U_1) \equiv TELL(K_2,U_1')$. Using this equivalence, it is trivial to show using induction that for any $m \geq 0$ and any sequence of updates $U_1,\ldots,U_m,U_1',\ldots,U_m' \in S_U$, such that $U_1 \equiv U_1',\ldots,U_m \equiv U_m'$, it holds that $TELL^m(K_1,U_1,\ldots,U_m) \equiv TELL^m(K_2,U_1',\ldots,U_m')$. Combing these properties we can trivially conclude that Dalal's scheme contains KB, update and query redundancy.

Using Proposition 4, we can eliminate the redundant elements from Dalal's scheme, keeping only one formula per equivalence class, in effect reduce S_K, S_U and S_Q to $L^*/\equiv$, and get an equivalent KR-scheme. The same property can be applied in all belief revision algorithms that satisfy the AGM postulate of preservation (#5), or equivalently Dalal's Principle of Irrelevance of Syntax. By Proposition 4, all these algorithms have simpler behavioral equivalents.

Non-Redundant Elements Cannot Be Eliminated

In this subsection we will show that only redundant elements can be removed from a scheme. The removal of non-redundant elements will cause loss of expressive power. This statement is true as far as finite sets are concerned; for infinite sets, we can remove some elements, so long as the cardinality of the original set is not altered. In the following lemmas, we will formally describe the above property for both the normal and the behavioral case. Following the lemmas, we will set out some interesting corollaries and explain their intuitive meaning.

Lemma 13 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme that does not contain normal KB redundancy and $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ be a KR-scheme such that $S_1 \leq_r S_2$. Then for any reduction algorithm from S_1 to S_2: (f_K, f_U, f_Q, f_A), it holds that f_K is a 1-1 function.

Proof

Suppose that f_K is not 1-1. Then, there exist K_1, $K_2 \in S_{K1}$, such that $K_1 \neq K_2$ and $f_K(K_1)=f_K(K_2)$. Since there exists no normal KB redundancy in S_1, we conclude that at least one of the following facts holds:

1. There exists a $Q \in S_{Q1}$ such that $ASK_1(K_1,Q) \neq ASK_1(K_2,Q)$. But:
   $ASK_1(K_1,Q)=$
   $=f_A(ASK_2(f_K(K_1),f_Q(Q))=$ (by the fact that $S_1 \leq_r S_2$, Query Preservation property)
   $=f_A(ASK_2(f_K(K_2),f_Q(Q))=$ (since $f_K(K_1)=f_K(K_2)$)
   $=ASK_1(K_2,Q)$ by the fact that $S_1 \leq_r S_2$ and the Query Preservation property, which is a contradiction.
2. There exists a $U \in S_{U1}$ such that $TELL_1(K_1,U) \neq TELL_1(K_2,U)$. Denote by:
   $K_1' = TELL(K_1,U)$, $K_2' = TELL_1(K_2,U)$. Then:
   $TELL_2(f_K(K_1),f_U(U))=$
   $=TELL_2(f_K(K_2),f_U(U))=$ (since $f_K(K_1)=f_K(K_2)$)
   $=f_K(K_2')$ by the fact that $S_1 \leq_r S_2$ and the Update Preservation property.
   So: $f_K(K_2')=TELL_2(f_K(K_1),f_U(U))$, which, using the Update Preservation property again, gives that: $K_2'=TELL_1(K_1,U)$. Substituting K_2' with its equivalent we get: $TELL_1(K_1,U)=TELL_1(K_2,U)$, which is a contradiction.
3. There exists a $K_0 \in S_{K1}$, $U_0 \in S_{U1}$ such that $TELL_1(K_0,U_0)=K_1$. By the Update Preservation property, the above relation gives:
   $TELL_2(f_K(K_0),f_U(U_0))=f_K(K_1)=f_K(K_2)$. Applying the Update Preservation property again we get: $TELL_1(K_0,U_0)=K_2$ or, equivalently $K_1=K_2$, which is a contradiction.
4. There exists a $K_0 \in S_{K1}$, $U_0 \in S_{U1}$ such that $TELL_1(K_0,U_0)=K_2$. Using the same arguments as in the above case (#3) we reach a contradiction.

We reached a contradiction in all the possible cases, so f_K is 1-1.

Lemma 14 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme that does not contain normal update redundancy and $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ be a KR-scheme such that $S_1 \leq_r S_2$. Then for any reduction algorithm from S_1 to S_2: (f_K, f_U, f_Q, f_A), it holds that f_U is a 1-1 function.

Proof

Suppose that f_U is not 1-1. Then, there exist U_1, $U_2 \in S_{U1}$, such that $U_1 \neq U_2$ and $f_U(U_1)=f_U(U_2)$. Since there exists no normal update redundancy in S_1, we conclude that there exists a $K \in S_{K1}$ such that $TELL_1(K,U_1) \neq TELL_1(K,U_2)$. Denote by:
$K_1=TELL_1(K,U_1)$, $K_2=TELL_1(K,U_2)$. Then:
$TELL_2(f_K(K),f_U(U_1))=$
$=TELL_2(f_K(K),f_U(U_2))=$ (since $f_U(U_1)=f_U(U_2)$)
$=f_K(K_2)$ by the fact that $S_1 \leq_r S_2$ and the Update Preservation property.
The above relation denotes that: $f_K(K_2)=TELL_2(f_K(K),f_U(U_1))$, which, using the Update Preservation property again, gives that: $K_2=TELL_1(K,U_1)$. Substituting K_2 with its equivalent we get: $TELL_1(K,U_1)=TELL_1(K,U_2)$, which is a contradiction.

Lemma 15 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme that does not contain normal query redundancy and $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ be a KR-scheme such that $S_1 \leq_r S_2$. Then for any reduction algorithm from S_1 to S_2: (f_K, f_U, f_Q, f_A), it holds that f_Q is a 1-1 function.

Proof

Suppose that f_Q is not 1-1. Then, there exist Q_1, $Q_2 \in S_{Q1}$, such that $Q_1 \neq Q_2$ and $f_Q(Q_1)=f_Q(Q_2)$. Since there exists no normal query redundancy in S_1, we conclude that there exists a $K \in S_{K1}$ such that $ASK_1(K,Q_1) \neq ASK_1(K,Q_2)$. But:
$ASK_1(K,Q_1)=$
$=f_A(ASK_2(f_K(K),f_Q(Q_1))=$ (by the fact that $S_1 \leq_r S_2$, Query Preservation property)
$=f_A(ASK_2(f_K(K),f_Q(Q_2))=$ (since $f_Q(Q_1)=f_Q(Q_2)$)
$=ASK_1(K,Q_2)$ by the fact that $S_1 \leq_r S_2$ and the Query Preservation property, which is a contradiction.

Lemma 16 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme that does not contain normal answer redundancy and $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ be a KR-scheme such that $S_1 \leq_r S_2$. Then for any reduction algorithm from S_1 to S_2: (f_K, f_U, f_Q, f_A), it holds that f_A is an onto function.
Proof
Since there exists no normal answer redundancy in S_1, we conclude that for any $A_1 \in S_{A1}$ there exists a $K \in S_{K1}$, $Q \in S_{Q1}$ such that $A_1=ASK_1(K,Q)$. Suppose that $A_2=ASK_2(f_K(K),f_Q(Q)) \in S_{A2}$.
Then:
$A_1=ASK_1(K,Q)=$
$=f_A(ASK_2(f_K(K),f_Q(Q)))=$ (by the fact that $S_1 \leq_r S_2$, Query Preservation property)
$=f_A(A_2)$ by definition, thus $A_1=f_A(A_2)$.
Thus, for all $A_1 \in S_{A1}$, there exists a $A_2 \in S_{A2}$ such that $f_A(A_2)=A_1$, so f_A is onto.

Lemma 17 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme that does not contain behavioral KB redundancy and $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ be a KR-scheme such that $S_1 \leq_b S_2$. Then for any reduction algorithm from S_1 to S_2: (f_K, f_U, f_Q, f_A), it holds that f_K is a 1-1 function.
Proof
Suppose that f_K is not 1-1. Then, there exist K_1, $K_2 \in S_{K1}$, such that $K_1 \neq K_2$ and $f_K(K_1)=f_K(K_2)$. Since there exists no behavioral KB redundancy in S_1, there exist $m \geq 0$, $U_1,\ldots,U_m \in S_{U1}$, $Q \in S_{Q1}$ such that:
$ASK_1(TELL_1^m(K_1,U_1,\ldots,U_m),Q) \neq ASK_1(TELL_1^m(K_2,U_1,\ldots,U_m),Q)$. But:
$ASK_1(TELL_1^m(K_1,U_1,\ldots,U_m),Q)=$
$=f_A(ASK_2(TELL_2^m(f_K(K_1),f_U(U_1),\ldots,f_U(U_m)),f_Q(Q)))=$ (since $S_1 \leq_b S_2$)
$=f_A(ASK_2(TELL_2^m(f_K(K_2),f_U(U_1),\ldots,f_U(U_m)),f_Q(Q)))=$ (since $f_K(K_1)=f_K(K_2)$)
$=ASK_1(TELL_1^m(K_2,U_1,\ldots,U_m),Q)$ since $S_1 \leq_b S_2$, which is a contradiction, so f_K is 1-1.

Lemma 18 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme that does not contain behavioral update redundancy and $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ be a KR-scheme such that $S_1 \leq_b S_2$. Then for any reduction algorithm from S_1 to S_2: (f_K, f_U, f_Q, f_A), it holds that f_U is a 1-1 function.
Proof
Suppose that f_U is not 1-1. Then, there exist U_1, $U_2 \in S_{U1}$, such that $U_1 \neq U_2$ and $f_U(U_1)=f_U(U_2)$. Since there exists no behavioral update redundancy in S_1, we conclude that there exist $m \geq 0$, $K \in S_{K1}$, $U_1',\ldots U_m' \in S_{U1}$, $Q \in S_{Q1}$, such that:
$ASK_1(TELL_1^{m+1}(K,U_1,U_1',\ldots,U_m'),Q) \neq ASK_1(TELL_1^{m+1}(K,U_2,U_1',\ldots,U_m'),Q)$. But:
$ASK_1(TELL_1^{m+1}(K,U_1,U_1',\ldots,U_m'),Q)=$
$=f_A(ASK_2(TELL_2^{m+1}(f_K(K),f_U(U_1),f_U(U_1'),\ldots,f_U(U_m')),f_Q(Q)))=$ (since $S_1 \leq_b S_2$)
$=f_A(ASK_2(TELL_2^{m+1}(f_K(K),f_U(U_2),f_U(U_1'),\ldots,f_U(U_m)),f_Q(Q)))=$ (since $f_U(U_1)=f_U(U_2)$)

$=ASK_1(TELL_1^{m+1}(K,U_2,U_1',\ldots,U_m'),Q)$ since $S_1 \leq_b S_2$, a contradiction, so f_U is 1-1.

Lemma 19 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme that does not contain behavioral query redundancy and $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ be a KR-scheme such that $S_1 \leq_b S_2$. Then for any reduction algorithm from S_1 to S_2: (f_K, f_U, f_Q, f_A), it holds that f_Q is a 1-1 function.

Proof

Suppose that f_Q is not 1-1. Then, there exist Q_1, $Q_2 \in S_{Q1}$, such that $Q_1 \neq Q_2$ and $f_Q(Q_1)=f_Q(Q_2)$. Since there exists no behavioral query redundancy in S_1, we conclude that there exists a $K \in S_{K1}$ such that $ASK_1(K,Q_1) \neq ASK_1(K,Q_2)$. But:

$ASK_1(K,Q_1)=$

$=f_A(ASK_2(f_K(K),f_Q(Q_1))=$ (by $S_1 \leq_b S_2$, Behavior Preservation property for m=0)

$=f_A(ASK_2(f_K(K),f_Q(Q_2))=$ (since $f_Q(Q_1)=f_Q(Q_2)$)

$=ASK_1(K,Q_2)$ by the fact that $S_1 \leq_r S_2$ and the Behavior Preservation property for m=0, which is a contradiction.

Lemma 20 Let $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ be a KR-scheme that does not contain behavioral answer redundancy and $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ be a KR-scheme such that $S_1 \leq_b S_2$. Then for any reduction algorithm from S_1 to S_2: (f_K, f_U, f_Q, f_A), it holds that f_A is an onto function.

Proof

Since there exists no behavioral answer redundancy in S_1, we conclude that for any $A_1 \in S_{A1}$, there exists a $K \in S_{K1}$, $Q \in S_{Q1}$ such that $A_1=ASK_1(K,Q)$. Suppose that:

$A_2=ASK_2(f_K(K),f_Q(Q))$.

Then:

$A_1=$

$=ASK_1(K,Q)=$ (by definition)

$=f_A(ASK_2(f_K(K),f_Q(Q)))=$ (by $S_1 \leq_b S_2$, Behavior Preservation property for m=0)

$=f_A(A_2)$ by definition, thus $A_1=f_A(A_2)$.

Thus, for all $A_1 \in S_{A1}$ there exists a $A_2 \in S_{A2}$ such that $f_A(A_2)=A_1$, so f_A is onto.

Suppose any KR-scheme, say S_1, and a more expressive one, say S_2. The 8 lemmas just proved imply that for each element (symbol) in each non-redundant set of S_1 there must exist at least one distinct element (symbol) in the respective set of S_2 with the same semantics. In other words, only redundant elements can be "merged" with others during the reduction, so only redundant sets can be reduced in size without loss of expressive power. This fact is true for each of the four sets and for each of the two reduction types, giving a total of 8 lemmas. Some corollaries of these lemmas can be trivially proved:

Corollary 1 Assume two KR-schemes S_1, S_2. Then:

- If $S_1 \leq_r S_2$ and S_1 does not contain normal redundancy, then for any reduction algorithm from S_1 to S_2: (f_K, f_U, f_Q, f_A), f_K, f_U and f_Q are 1-1 and f_A is onto.
- If $S_1 \leq_b S_2$ and S_1 does not contain behavioral redundancy, then for any reduction algorithm from S_1 to S_2: (f_K, f_U, f_Q, f_A), f_K, f_U and f_Q are 1-1 and f_A is onto.

In the following, we will use the symbol $A \leq_c B$ to denote that the set B has cardinality at least equal to A and $A =_c B$ to denote that the sets A and B have the same cardinality.

Corollary 2 Assume two KR-schemes $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$, $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$. Then:

1. If $S_1 \cong_r S_2$ and S_1, S_2 do not contain normal KB redundancy, then $S_{K1}=_c S_{K2}$.
2. If $S_1 \cong_r S_2$ and S_1, S_2 do not contain normal update redundancy, then $S_{U1}=_c S_{U2}$.
3. If $S_1 \cong_r S_2$ and S_1, S_2 do not contain normal query redundancy, then $S_{Q1}=_c S_{Q2}$.
4. If $S_1 \cong_r S_2$ and S_1, S_2 do not contain normal answer redundancy, then $S_{A1}=_c S_{A2}$.
5. If $S_1 \cong_b S_2$ and S_1, S_2 do not contain behavioral KB redundancy, then $S_{K1}=_c S_{K2}$.
6. If $S_1 \cong_b S_2$ and S_1, S_2 do not contain behavioral update redundancy, then $S_{U1}=_c S_{U2}$.
7. If $S_1 \cong_b S_2$ and S_1, S_2 do not contain behavioral query redundancy, then $S_{Q1}=_c S_{Q2}$.
8. If $S_1 \cong_b S_2$ and S_1, S_2 do not contain behavioral answer redundancy, then $S_{A1}=_c S_{A2}$.

Proof

For #1, we have that $S_1 \leq_r S_2$. For any reduction algorithm (f_{K1}, f_{U1}, f_{Q1}, f_{A1}), f_{K1} is 1-1.
Similarly, $S_2 \leq_r S_1$, so for any reduction algorithm (f_{K2}, f_{U2}, f_{Q2}, f_{A2}), f_{K2} is 1-1.
Using the Schroder-Bernstein theorem and the existence of these two functions, we conclude that $S_{K1}=_c S_{K2}$.
Using the same arguments we can prove cases #2, #3, #5, #6, #7.
For #4, since $S_1 \leq_r S_2$, for any reduction algorithm (f_{K1}, f_{U1}, f_{Q1}, f_{A1}), f_{A1} is onto. For each $A_1 \in S_{A1}$, there will exist at least one $A_2 \in S_{A2}$ such that $A_1=f_{A1}(A_2)$. We define the function $f_{A1}{}': S_{A1} \rightarrow S_{A2}$, with the property that $f_{A1}{}'(f_{A1}(A))=A$ for all $A \in S_{A2}$. In effect, for each $A' \in S_{A1}$ we (arbitrarily) select one $A \in S_{A2}$ (there will exist such an $A \in S_{A2}$) such that $f_{A1}(A)=A'$ and set $f_{A1}{}'(A')=A$.
Since f_{A1} is onto, $f_{A1}{}'$ is total and 1-1.
Similarly, $S_2 \leq_r S_1$, so for any reduction algorithm (f_{K2}, f_{U2}, f_{Q2}, f_{A2}), f_{A2} is onto. Using the same arguments we create one $f_{A2}{}': S_{A2} \rightarrow S_{A1}$, which is total and 1-1.
Given these two functions ($f_{A1}{}'$, $f_{A2}{}'$) we can again apply the Schroder-Bernstein theorem and prove that $S_{A1}=_c S_{A2}$.
Similar arguments will give us #8.

This corollary implies that the richness of semantics in non-redundant KR-schemes can be determined by the richness of their respective sets. This does not necessarily mean that KR-schemes with richer sets are more expressive; in fact, one can create two (redundant or non-redundant) KR-schemes S_1, S_2 which have the same symbols (equal KB, Update, Query and Answer sets) yet be incomparable (neither $S_1 \leq_b S_2$ nor $S_2 \leq_b S_1$ holds). This case can occur if the ASK and TELL functions of the two schemes give totally different semantics in the symbols involved.

However, this set of lemmas provides us with a set of impossibility results: if the schemes involved do not contain redundancy and a 1-1 (or onto, depending on the case) function cannot be established between two of the respective sets of the two schemas, then a reduction is impossible. In the case where such functions do exist, these are the only ones that qualify, so we can immediately overrule some possible reduction algorithms in our search for one that performs the reduction properly. The existence of such functions can be easily determined using the numerous results provided to us by set theory. Similarly, if the respective sets are not isomorphic (a rather strong requirement), we cannot prove the two KR-schemes equivalent.

There have been several attempts in the literature to reduce specific KR-schemes to others. Examples can be found in [5], [7], [11], among other works. When trying such reductions it would be desirable to know in advance whether this reduction can actually occur; this would save us from the futile search of a reduction algorithm. The impossibility result above is a first step towards this direction. If the

reduction is not possible, the above results may prove useful in identifying, using formal methods instead of simple intuition, what the second KR-scheme lacks in order to be more expressive than the first (for example enriching the query language could do, while enriching the KB set could prove useless). This property is especially useful when dealing with finite sets, where the famous "Pigeonhole Principle" applies. On the other hand, the existence of such functions does not guarantee that the reduction is possible; it is merely an indication that it is.

Symbols' Effect on Expressive Power

As already stated, the four sets comprising a KR-schema represent the symbol level of a schema. One would expect that the richness of symbols (ie the cardinality of the symbol sets) comprising a scheme determines the expressive power of a scheme. We have already uncovered one major exception to this rule: redundant elements increase the size of a scheme's sets, yet add nothing to its expressive power. It may also be the case that two KR-schemes are incomparable, regardless of the cardinality of the symbol sets they use. Therefore, we cannot determine the relative expressive power of a pair of KR-schemes by simply comparing the cardinality of their symbol sets; the ASK and TELL functions play a major role in this decision.

On the other hand, a KR-scheme with poor symbol sets cannot be very expressive. We could imagine the size of the symbol sets as a kind of "upper bound" on the expressive power of the KR-scheme; whether the richness of the symbol sets will be exploited to produce a powerful enough KR-scheme depends on the definition of the ASK and TELL functions. This fact is implied by Corollary 1 and the lemmas preceding it. Moreover, we can show that when the ASK and TELL functions can be freely defined, then the expressive power of a KR-scheme can be determined by the "size" of the symbol sets.

Proposition 5 Suppose any KR-scheme $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ and four sets S_{K2}, S_{U2}, S_{Q2}, S_{A2}, such that $S_{K1}\leq_c S_{K2}$, $S_{U1}\leq_c S_{U2}$, $S_{Q1}\leq_c S_{Q2}$ and $S_{A1}\leq_c S_{A2}$. Then there exist functions $ASK_2:S_{K2}\times S_{Q2}\rightarrow S_{A2}$, $TELL_2:S_{K2}\times S_{U2}\rightarrow S_{K2}$ such that the KR-scheme $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ is at least as expressive as S_1.

Proof

Since $S_{K1}\leq_c S_{K2}$, there exists a 1-1 function $f_K:S_{K1}\rightarrow S_{K2}$. We define the function $f_K':S_{K2}\rightarrow S_{K1}$, such that $f_K'(f_K(K))=K$ for all $K\in f_K(S_{K1})$. For any $K\in S_{K2}\backslash f_K(S_{K1})$, we define $f_K'(K)$ arbitrarily.

Similarly, there exist 1-1 functions $f_U:S_{U1}\rightarrow S_{U2}$, $f_Q:S_{Q1}\rightarrow S_{Q2}$. We define analogously the functions $f_U':S_{U2}\rightarrow S_{U1}$, $f_Q':S_{Q2}\rightarrow S_{Q1}$, such that $f_U'(f_U(U))=U$ for all $U\in f_U(S_{U1})$ and $f_Q'(f_Q(Q))=Q$ for all $Q\in f_Q(S_{Q1})$ and an arbitrary value for all $U\in S_{U2}\backslash f_U(S_{U1})$, $Q\in S_{Q2}\backslash f_Q(S_{Q1})$.

Moreover, there exists a 1-1 function $f_A':S_{A1}\rightarrow S_{A2}$. We define $f_A:S_{A2}\rightarrow S_{A1}$ such that $f_A(f_A'(A))=A$ for all $A\in f_A'(S_{A1})$. We give an arbitrary value to $f_A(A)$ for any $A\in S_{A2}\backslash f_A'(S_{A1})$.

Since f_K, f_U, f_Q and f_A' are 1-1, the functions f_K', f_U', f_Q' and f_A are well defined.

Finally, for all $K\in S_{K2}$, $Q\in S_{Q2}$ we define:

$ASK_2:S_{K2}\times S_{Q2}\rightarrow S_{A2}$, such that: $ASK_2(K,Q)=f_A'(ASK_1(f_K'(K),f_Q'(Q)))$

$TELL_2:S_{K2}\times S_{U2}\rightarrow S_{K2}$, such that: $TELL_2(K,U)=f_K(TELL_1(f_K'(K),f_U'(U)))$.

We can prove that the 4-tuple (f_K, f_U, f_Q, f_A) is a reduction algorithm.

Indeed for any $K\in S_{K1}$, $Q\in S_{Q1}$:

$f_A(ASK_2(f_K(K),f_Q(Q)))=$

$=f_A(f_A'(ASK_1(f_K'(f_K(K)),f_Q'(f_Q(Q)))))=$ (by definition)

$=ASK_1(K,Q)$ by the relations above, so the Query Preservation property holds.
Moreover for any K_1, $K_2 \in S_{K1}$, $U \in S_{U1}$:
$f_K(K_2)=TELL_2(f_K(K_1),f_U(U))$ holds (by definition) if and only if:
$f_K(K_2)=f_K(TELL_1(f_K'(f_K(K_1)),f_U'(f_U(U))))$ which (by the relations above) holds if and only if:
$f_K(K_2)=f_K(TELL_1(K_1,U))$ which (by the fact that f_K is 1-1) holds if and only if:
$K_2=TELL_1(K_1,U)$.
Thus the Update and Query Preservation properties hold, so $S_1 \leq_r S_2$.

Proposition 5 implies that for any given KR-scheme, we can always create a more expressive one, given rich enough symbol sets to store the extra information. The same property holds for behavioral reduction, as expected:
Corollary 3 Suppose any KR-scheme $S_1=(S_{K1}, S_{U1}, S_{Q1}, S_{A1}, ASK_1, TELL_1)$ and four sets S_{K2}, S_{U2}, S_{Q2}, S_{A2}, such that $S_{K1} \leq_c S_{K2}$, $S_{U1} \leq_c S_{U2}$, $S_{Q1} \leq_c S_{Q2}$ and $S_{A1} \leq_c S_{A2}$. Then there exist functions $ASK_2: S_{K2} \times S_{Q2} \rightarrow S_{A2}$, $TELL_2: S_{K2} \times S_{U2} \rightarrow S_{K2}$ such that the KR-scheme $S_2=(S_{K2}, S_{U2}, S_{Q2}, S_{A2}, ASK_2, TELL_2)$ is behaviorally at least as expressive as S_1.

Conclusion and Future Work

This work is an attempt to formalize the field of Knowledge Representation by introducing a simple, but very general, model for the description of KR-schemes. This model can be used to compare KR-schemes, no matter how different, in terms of expressive power. The task of comparing two schemes is reduced to the task of finding a way to "translate" the information expressed under one scheme in terms of the other, without loss of information.

Such types of translations have already been attempted in the past. This work is intended as an aid to such attempts. Ideally, we would like to establish a result allowing us to know in advance whether a certain reduction is or is not possible, without the need of finding a reduction algorithm. This is a matter of current research.

Another interesting area of future research may be the study of different types of reductions. The reduction method, as currently defined, maps each query result of the weaker system to one query result of the stronger system. We could loosen this constraint by allowing each query result of the weaker system to be mapped to a sequence of query results in the stronger one. This sequence should uniquely determine the original result (in the weaker system) and do so correctly in all possible queries. The same loosening could be applied to the updates: each update result in the weaker system should be uniquely determined by a sequence of updates in the stronger system, instead of a single one, as is currently the case. This loosening could be applied to both the normal and the behavioral case, giving a total of four possible reduction methods (and their respective comparison relations). The consequences of the application of each of the proposed comparison methods is a matter of current research.

References

[1] Carlos Alchourron, Peter Gardenfors, David Makinson, "On the Logic of Theory Change: Partial Meet Contraction and Revision Functions", The Journal of Symbolic Logic, 50: 510-530, 1985.

[2] Mukesh Dalal, “Investigations Into a Theory of Knowledge Base Revision: Preliminary Report”, In Proceedings of the Seventh National Conference on Artificial Intelligence, 475-479, 1988.
[3] Mukesh Dalal, “Updates in Propositional Databases”, Technical Report, DCS-TR-222, Dept. of Computer Science, Rutgers University, 1988.
[4] Randall Davis, Howard Shrobe, Peter Szolovitz, “What is a Knowledge Representation?”, AI Magazine, 14(1): 17-33, 1993.
[5] Giorgos Flouris, Dimitris Plexousakis, “Belief Revision Using Table Transformation”, Technical Report ICS-FORTH, TR-290, 2001. Web: http://www.ics.forth.gr/isl/publications/paperlink/flourisTR-290.pdf.
[6] Patrick Hayes, “The logic of frames”, in Metzing, editor, Frame Conceptions and Text Understanding, de Gruyter, Berlin, 1979.
[7] Greg Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis, Michel Scholl, “RQL: A Declarative Query Language for RDF”, In Proceedings of the 11th International Conference on the WWW, Hawaii, 592-603, 2002.
[8] Hirofumi Katsuno, Alberto Mendelzon, “Propositional Knowledge Base Revision and Minimal Change”, Technical Report KRR-TR-90-3, Technical Reports on Knowledge Representation and Reasoning, University of Toronto, 1990.
[9] Hector Levesque, Gerhard Lakemeyer, “The Logic of Knowledge Bases”, MIT Press, Cambridge, Massachusetts, 2000.
[10] Michael J. Maher, Grigoris Antoniou, David Billington, “A Study of Provability in Defeasible Logic”, Australian Joint Conference on Artificial Intelligence, p. 215-226, 1998.
[11] Dimitris Plexousakis, “Semantical and Ontological Considerations in Telos: a Language for Knowledge Representation”, Computational Intelligence, 9(1): 41-72, 1993.