



CENTRO PER LA RICERCA
SCIENTIFICA E TECNOLOGICA

38050 Povo (Trento), Italy
Tel.: +39 0461 314312
Fax: +39 0461 302040
e-mail: prdoc@itc.it – url: <http://www.itc.it>

DISTRIBUTED DESCRIPTION LOGICS: ASSIMILATING
INFORMATION FROM PEER SOURCES

Borgida A., Serafini L.

March 2003

Technical Report # 0303–12

© Istituto Trentino di Cultura, 2003

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication outside of ITC and will probably be copyrighted if accepted for publication. It has been issued as a Technical Report for early dissemination of its contents. In view of the transfert of copy right to the outside publisher, its distribution outside of ITC prior to publication should be limited to peer communications and specific requests. After outside publication, material will be available only in the form authorized by the copyright owner.

Distributed Description Logics: Assimilating Information from Peer Sources

Alex Borgida¹ and Luciano Serafini²

¹ Dept. of Computer Science
Rutgers University
New Brunswick, USA
borgida@cs.rutgers.edu

² ITC-IRST
Trento, Italy
serafinit@itc.it

Abstract. Due to the availability on the Internet of a wide variety of sources of information on closely related topics, the problem of providing seamless, integrated access to such sources has become (again) a major research challenge. Although this problem has been studied for several decades, especially in the database community, there is a need for a more refined approach in those cases where the original sources maintain their own independent view of the world. In particular, we motivate with examples the utility of directed non-injective mappings between the individuals in the domains of multiple Information Sources. Since Description Logics have already served successfully in information integration and as ontology languages, we extend this formalism with the ability to handle the above kinds of mappings. The result is called Distributed Description Logics, and we investigate examples, desirable properties, and formal definitions, providing at the end a number of theorems concerning its theoretical and computational aspects.

1 Introduction

A significant problem of modern information management is the integration of information from multiple sources. The standard version presumes a framework where users are accessing through a single interface data from several information sources (local ISs), which can include databases, web data, files, etc. The important goal here is to provide seamless access to the data, making the users unaware of the original source of the information. This is achieved by providing a single global schema, which traditionally was the result of merging the local schemata. And query processing consists of identifying the relevant ISs, translating the user's query into collections of queries over local ISs, and collating the answers, expressed in terms of the global schema. In more traditional approaches, the global schema is defined by integrating the local schemas, and afterwards defining its contents through views over the local ISs, making query processing relatively easy. More recently, Levy et al [25] have investigated an

approach where the global, conceptual schema is developed independently, and then local IS are defined in terms of it.

A somewhat different, but related, approach is one which preserves the identity of each local IS and its user interface. However, the local system wishes to import information available in other sources, which are related to it directly through some kinds of assertion, or indirectly through chains of such relations. This approach is more appropriate for loosely related information sources, such as information appearing on the web, or distributed agents, where each source of information is independent. This paper is mostly concerned with investigating this second, “peer to peer” kind of combination.

The traditional view of database integration holds that the semantic world-views of local IS may exhibit miss-matches (often called “conflicts” or “semantic heterogeneities”) which need to be resolved in order to allow information from one source to be properly visible in the other source.

Saltor and Rodriguez [31] [31] identify three high-level categories of such semantic heterogeneities, which are useful for our presentation: (i) heterogeneities between object classes, (ii) heterogeneities between class structures, and (iii) heterogeneities between object instances. The first two categories relate to the schema of the IS, and have been thoroughly studied. Some of the conflicts in the third category deal with “the facts”: e.g., two IS may record the capital of China as Beijing and Peking respectively. In other situations, however, there may be more interesting systematic relationships between individuals. For example, consider the case when one IS contains personal information (e.g., from credit card purchases), while another one contains census information, which only records information about *households*. The correspondence between households and the people in it is not the identity relation, neither is it a simple functional bijection. Yet it will be important to establish and record this relationship if the two IS are to be integrated. In Section 2, we provide further examples of complex correspondences between the domains of multiple IS in a federated system.

Description Logics are formalisms for knowledge representation and reasoning [4]. They have been used for a variety of roles in databases [7], including the description of the “data semantics” / “meta data” in the form of conceptual schemata. In turn, such DL conceptual schemas play a central role in many recent proposals for database integration, and more general information integration (e.g., [14, 20, 2, 25, 11, 28, 5]).

On the other hand, the term taxonomies used in web sites such as DMOZ, Yahoo, or e-commerce stores, can be viewed as fly-weight Description Logic ontologies. More importantly, DMAL+OIL [17] – the current leading contender for being the ontology language for the semantic web, is a clear example of a Description Logic (even if its syntax is not). Moreover, Tim Berners-Lee’s vision of the semantic web [3] explicitly abandons the notion of a universal ontology, and embraces the kind of “distributed” ontologies we have in mind.

For these reasons, we will carry out our investigations concerning complex correspondences between IS domains, in the context of DLs.

We start by providing several examples which motivate the new kinds of relationships between peer IS, and review related in the database and ontology literature. In Section 3 we then introduce Description Logics and Distributed Description Logics (DDL), showing how some examples are handled by the them. In Section 5, we review some desirable properties of distributed Information Systems, and discuss how DDL measure up to them. Formal definitions are given in Section 4, in Section 6 provide a theorem which shows that under some rather general conditions, DDL reasoning can often be translated into reasoning in a single, global but ordinary DL. Finally, we investigate the properties of DDL in the case of ontologies that are simple taxonomies of atomic terms, such as those found on dmoz, google, yahoo, etc., and provide an algorithm for reasoning with them which is no harder than in the original DLs.

2 Motivating Examples

We will be considering a variety of possible correspondences between individuals in the domains of discourse of two IS, IS_1 and IS_2 say, some of which were noted already in [21].

In the simplest case of integration, the same individual (e.g., the name "Toronto"), occurs in both IS. Somewhat more complex is the situation where one needs to identify/match different representations of the same real-world individual. In the case of scaling conflicts (e.g., use of different units of measure), this relationship is quite systematic, and can be described through mathematical equations. In other cases this needs to be achieved through the use of key attributes or heuristics (e.g., persons having the same name). Either way, the result can be thought of as a new binary relation between objects in the two IS, which is assumed to be a bijection.

Several complications may arise even in this situation. Consider the case when the unit of measure is currency: the conversion function `Euro_to_Dollars`, is not the inverse of the function `Dollars_to_Euro`, because banks add a surcharge to all transactions. Therefore a single conversion relation is not sufficient, and we must acknowledge the need for *directional mappings* between the domains, e.g., one from Dollars to Euro, and another from Euro to Dollars. A different complication can arise in the case when the mapping between the domains cannot be described extensionally. For example, suppose that one school assigns simple letter grades, e.g., A , B , C , while another school allows them to be qualified by plus and minus, e.g., $A+$. An A at the first school corresponds to one of $\{A+, A, A-\}$ at the other school, but there is no way to tell which. Note however that this partial information is still important: having a grade of "A+, A or A-" is known to be better than having a grade of B ! The following examples explore further intricacies of the relationship between domain elements in different IS.

Example 1. Suppose `BasicC`, `IntermC` and `AdvancedC` are 3 increasingly difficult courses on some topic. University $Univ_1$ offers `BasicC` and `AdvancedC`, while university $Univ_2$ offers `IntermC`. The universities are concerned about what classes a

student has completed, in order to check the pre-requisites of other courses they are enrolling in, or to meet degree requirements. In particular, both universities allow a course x to be substituted for another course y if x is harder than y , *and* covers most of the material of y (say 80%). The universities also allow credits earned at one to be *transferred* to the other. Univ₁ may decide to accept `IntermC` as a substitute for `BasicC`; on the other hand, Univ₂ may only accept `AdvancedC` as a substitute for `IntermC`. Suppose that courses are modeled as concepts whose extension is the set of students who have completed them. Then we have a situation where we want the instances of `IntermC` to be included among the instances of `BasicC` according to Univ₁, while `IntermC` should subsume the instances of `AdvancedC` according to Univ₂. Despite this, Univ₁ may not necessarily want to view `AdvancedC` as a subclass of `BasicC`, since the courses might disagree on more than 80% of the material.

Example 2. Suppose IS_1 has information about married couples, while IS_2 has information about persons. We therefore need to express correspondences between individuals in the two domains, e.g., between `couple23` in IS_1 , and each of `Gianni` and `Mary`, in IS_2 . But there are more general relationships between the information in the two IS. For example, we know that each couple involves exactly two persons.

In this case, IS_1 contains information about individuals that are in some sense *abstractions* over individuals in IS_2 . Similar examples will arise in most other situations where the so-called “materialization abstraction” [29] occurs: one individual standing for a class of individuals in some more refined view of the world.

Example 3. Consider a situation where there are two IS: $IS_{Harvard}$ and IS_{MIT} , serving the needs of college libraries in some town. The libraries have information about *copies of books*, which can be taken by borrowers or are available on the shelf. On the other hand, $IS_{Student}$ is a database accessed by students who want to know which library they should go to if they need some book. Notice that the student does not care about which copies of a book are available, so we have once again an abstraction: the student’s `Tractatus` corresponds to `TractatusCopy1,...` in $IS_{Harvard}$, as well as `TractatusCopy2` in IS_{MIT} . Moreover, the student only wants to know about some material being located at MIT if there is a copy of it currently on the shelf at the MIT library.

The above examples reinforce the need to consider in greater detail the role of the mapping between the domain of objects in the IS’s being integrated into a federation. First, one needs a pair of general relations (not just functions) to connect each pair of IS, because information flow is “directional”. Second, there are two aspects of these mappings:

- How are specific individual objects related to each other? (e.g., `couple23` in IS_1 and `Mary` in IS_2).
- What general statements can one make about the mappings of individuals? (e.g., `Couple` instances in IS_1 correspond to exactly two `Person` instances in IS_2).

Although our work deals with hitherto unexplored aspects, it is useful to see some of the considerable prior work devoted to the integration of information from multiple sources¹

2.1 Related Work on Information Integration

There has been a great deal of work on the problem of integrating database schemas and databases in general. In the beginning, this work was motivated by the process of integrating user views in order to arrive at a “corporate” schema that satisfies the needs of everyone. Later, additional motivation was provided by the advent of heterogeneous/federated databases, and especially data warehouses, which by their nature integrate possibly radically different sources.

The key questions in this research include how to match up both schema and data-level information between multiple databases, and most effort has been devoted to heuristics and tools for *finding* such relationships. (See [20] for a fairly exhaustive list of conflicts at the schema level, and [30] for a recent survey of solutions.)

More relevant to the present paper is work concerned with languages for *expressing* the relationship between elements of different IS, especially as they relate to the instance level.

Kent [21] provides an extensive list of problems that arise due to data-level mismatches between databases, recognizing the need for both domain relationships (e.g., currencies) and context-dependent use of them (e.g., salaries vs. stock prices in different currencies). He examines complex solutions that use domain mapping functions orchestrated by integrator functions, and expresses them in the Iris database programming language.

SchemaLog [24] is just one representative of the class of declarative languages for relating multi-databases that use powerful data restructuring facilities. (The paper has a fine section reviewing other similar approaches.) Its higher-order syntax allows querying the schema, as well as inter-relating schema and value identifiers. Similar comments apply to work on integrating heterogeneous semi-structured data sources (see [15] for a survey).

A different declarative approach to the specific problem of data mediation is illustrated in [32], where meta-attributes and rules are used to deal with complex value conversions. A desirable feature of the approach is that it introduces the notion of “contexts”, which allows for the *automatic invocation* of conversion functions.

Information integration has also been studied in the field of Artificial Intelligence, where *ontologies* are essential components of the knowledge-rich environment required by problems such as natural language understanding. An ontology [33] is supposed to be a collection of shared term definitions, agreed upon by some community of users. The need to integrate ontologies arose from the beginning,

¹ We leave to the end discussion of work which is particularly close to ours, having been carried out in the Description Logic framework.

as in [23], which integrates two dictionaries. Considerable work has been done recently on this problem, resulting again mostly in heuristic tools and methodologies supporting integration, although [34] is an exception: he shows how one can take local assertions of how an agent sees itself connecting to global ontologies, and derive a more global connection. Klein [22] provides a recent review of the literature, distinguishing among others between ontology *integration*, which results in a single ontology, from the weaker ontology *combination*, where the original ontologies are kept. In both cases one must perform ontology *alignment*, which involves relating some of the terms in the two ontologies. The closely related terms can be connected by generalization-specialization or disjointness assertions, which had also been used to relate database schema elements.

3 Distributed Description Logics

3.1 Description Logics

For those not familiar with DLs, we start with a review of those aspects that are relevant to this paper.

Description Logics are a family of object-centered knowledge representation formalisms which, as mentioned earlier, have proven to be useful in the design and querying of Information Systems [7], including information integration [14, 2, 25, 11, 28, 5] as well as representing ontologies.

Description logics view the world as being populated by individuals that can be grouped into classes, called *concepts*, and that can be related to each other by binary relationships, called *roles*. A specific DL provides a specific set of “constructors” for building more complex concepts and roles, much like a programming language type system provides type constructors for building complex types from simpler ones. For example, concept constructors such as conjunction (written as $A \sqcap B$) and value restriction (written as $\forall r.C$) can be used to describe object-oriented style classes with multiple superclasses and type-like constraints on members/attributes. The following examples are meant to provide intuitive sense for the above two constructors, and the possibilities in combining in a DL.

- primitive concepts PERSON, UNIVERSITY, INTEGER
- primitive roles attends, hasAge, hasLocation
- $\forall \text{hasAge.Integer}$ (* Objects whose hasAge role/attribute has only integer values *)
- $\forall \text{attends.UNIVERSITY}$ (* Objects whose attends role values are instances of UNIVERSITY *)
- $\forall \text{hasLocation.NEW_ENGLAND}$ (* Objects located in places that are considered to be New England *)
- $\forall \text{attends.}(UNIVERSITY \sqcap \forall \text{hasLocation.NEW_ENGLAND})$ (* Objects attending universities in New England *)
- $PERSON \sqcap \forall \text{attends.}(UNIVERSITY \sqcap \forall \text{hasLocation.NEW_ENGLAND})$ (* Persons attending universities in New England *)

As seen in the above examples, a description usually corresponds to a noun-phrase (a unary formula in logic). In some DLs, it is possible to construct more complex roles as well. For example, one can use attends^- to refer to the inverse of the attends role, which might otherwise be called hasEnrolled .

One can then use description terms for several different tasks. First, one can *define* new concepts starting from its members. For example, NEW-ENGLAND may itself be defined as $\forall \text{hasAddr}.\forall \text{inState}.\{\text{Maine}, \text{Vermont}, \dots\}$. Second, one can claim that one description, D , *subsumes* or *is more general than* another one, C , written as $C \sqsubseteq D$, meaning that anything that satisfies the conditions of C must, by necessity, satisfy the conditions of D . This is useful in asserting necessary conditions, such as $\text{STUDENT} \sqsubseteq \text{PERSON}$, rather than the necessary and sufficient conditions provided by definitions.

One can also use \sqsubseteq to make more general statements, sometimes called axioms, relating various terms in a complex ontology. For example, if $\exists p.C$ is a concept constructor representing objects that have at least on p role filler in the concept C , then $(\text{STUDENT} \sqcap \forall \text{attends}.\text{IVY-LEAGUE-U}) \sqsubseteq (\text{PERSON} \sqcap \exists \text{hasParents}.\text{RICH-PERSON})$ says that students attending Ivy League universities must have at least one rich parent. Collections of such assertions specify the terminology used to describe some application domain. Such a collection is called a *T-box*, and resembles the schema of a database, or an ontology. Subsumption has additional uses. According to its formal definition, it is possible to deduce new subsumptions, just like it is possible to deduce new implications from a logical theory. For example, $\forall \text{attends}.\text{UNIVERSITY}$ subsumes $\text{PERSON} \sqcap \forall \text{attends}.\text{IVY-LEAGUE-U}$, assuming that UNIVERSITY can be deduced to subsume IVY-LEAGUE-U . This provides the specification of an algorithm which computes whether some (new or old) description subsumes another one. This is useful in organizing the terminology of the domain into the familiar IS-A hierarchy, and to detect whether some description is incoherent, in the sense that it cannot hold of any individual. (Incoherence is detected by checking for subsumption by the empty concept).

Third, one can assert the membership of an *individual* in a concept, as a way of giving it a partial description. For example, $\text{STUDENT} \sqcap \neg \text{MALE}(\text{Anna})$ asserts that Anna is a student who is not male. In addition, one can assert the inter-relatedness of two individuals, e.g., $\text{attends}(\text{Anna}, \text{Harvard})$. Collections of assertions about individuals partially describe some state of world, and form an *A-box*, which resembles a database state (the collection of tuples in relations, for example). As with subsumption, not only can one assert membership in concepts, but one can deduce/compute whether some arbitrary individual is an instance of an arbitrary description, possibly given some A-box and T-box. For example, $\exists \text{attends}.\text{IVY} - \text{LEAGUE}(\text{Anna})$ can be deduced from the above two assertions, assuming that Harvard in turn is, or can be deduced to be, an instance of IVY-LEAGUE .

A (DL) *knowledge base* \mathbf{K} will then be a pair $\langle \mathbf{T}, \mathbf{A} \rangle$, where \mathbf{T} is a terminology and \mathbf{A} is an A-box, which uses only descriptions valid according to \mathbf{T} . It is then usual to ask specific subsumption or membership questions with such a \mathbf{K} as a background theory.

Description Logics have been quite successful as Conceptual Modeling languages, capturing the semantics of the world which is being modeled by the Information Source. This can be done by direct modeling, or as a result of translating from some more traditional conceptual data models such as the Extended Entity Relationship model. For example, suppose we start from a typical ER relationship ENROLLMENT, which relates to entity STUDENT, with cardinality upper bound 5, and to entity COURSE.

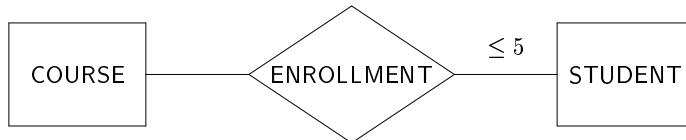


Fig. 1. ER diagram

Then, by introducing roles *who* and *what*, one can express the ER diagram by the following subsumptions [9]:

1. $\text{ENROLLMENT} \sqsubseteq (\forall \text{who}.\text{STUDENT} \sqcap (= 1 \text{ who}) \sqcap \forall \text{what}.\text{COURSE} \sqcap (= 1 \text{ what}))$
 (* *who and what are functions connecting each enrollment object to a student and a course* *)
2. $\text{STUDENT} \sqsubseteq \forall \text{who}^-. \text{ENROLLMENT} \sqcap (\leq 5 \text{ who}^-)$ (* *who only connects enrollments to students, and a student is in at most 5 such connections; expressed using who^- , the inverse of the *who* role* *)
3. $\text{COURSE} \sqsubseteq \forall \text{what}^-. \text{ENROLLMENT}$ (* *what only connects enrollments to courses* *)

We are now ready to proceed with the new notions we wish to introduce.

3.2 Distributed Description Logics

Suppose we have a collection of information sources IS_i , each using some (potentially different) description logic \mathcal{DL}_i . (The IS_i could be full DL knowledge bases \mathbf{K}_i , or just T-boxes \mathbf{T}_i .) Let us now try to express connections between them, as a way of aligning them.

The pioneering work of Catarci and Lenzerini [14] proposed the continued use of description logics. In particular, subsumption assertions could relate descriptions in different knowledge bases: $\text{GradStudent}_2 \sqsubseteq_{int} \text{Student}_1$ would indicate that every graduate student in the part of the world described by IS_2 was also a student in the overlapping part of the world described by IS_1 . However, the semantics in [14] implies that inter-schema assertions only have an effect on those *individuals that are shared* between the respective IS domains – i.e., the correspondence between the domain elements is identity. The reason for this is,

in part, that in current DL the definition of a new concept can only retrieve a subset of the existing set of individuals, rather than create new individuals.

In order to deal with our more complex examples, we turn for inspiration to the work of Ghidini and Serafini [16] on Distributed First Order Logic. The idea is to introduce, at least conceptually, some binary relations \mathbf{r}_{ij} , called *domain relation*, describing the correspondences between the pairs of domains IS_i and IS_j , and to use so-called *bridge rules* to constrain these relationships *in an implicit manner*.

In order to support directionality, bridge rules in a set \mathfrak{B}_{jk} will be viewed as describing “flow of information” from IS_j to IS_k *from the point of view of IS_k* ; i.e., IS_k is “importing” information from IS_j), and hence \mathfrak{B}_{jk} may be different from \mathfrak{B}_{kj} .

Based on studies in [16], here are some patterns of constraints on the correspondence relationships that one might like to express using bridge rules:

1. Every instance of concept A (A -instances) in IS_1 corresponds only to a G -instances in IS_2 .
2. All G -instances in IS_2 have a corresponding A -instance in IS_1 .
3. Each A -instance has at least/at most n corresponding G -instances in IS_2 .
4. The domain relation from IS_1 to IS_2 is the identity relationship.
5. The domain relations between IS_1 and IS_2 are simmetric, i.e., $\mathbf{r}_{12} = \mathbf{r}_{21}^{-1}$.

In this paper we will study the first two kinds of bridge rules.

In order to help keep us from confusing descriptions from various IS_i , we start by labeling each description E in \mathcal{DL}_i with the index i (written as $i:E$). (However, when talking about subsumption within a single IS_i , we will use the more readable $i:A \sqsubseteq B$, instead of the formally correct $i:A \sqsubseteq i:B$.) We now introduce two kinds of bridge rules:

Definition 1. *Given concepts C and E of \mathcal{DL}_i and \mathcal{DL}_j respectively, a bridge rule from i to j is an expression of one of the following two forms:*

$$i:C \xrightarrow{\sqsubseteq} j:E, \text{ called into-bridge rule}$$

$$i:C \xrightarrow{\supseteq} j:E, \text{ called onto-bridge rule}$$

An into-bridge rule specifies that C -objects in IS_i correspond only to E -objects in IS_j (according to the \mathbf{r}_{ij} relation), while an onto-bridge rule states that the every object in E has a corresponding pre-image in concept C of IS_1 . The intuition of the interpretation of the into- and onto-bridge rules is shown in Figure 2 and Figure 3

In Example 2, one would have the bridge rule $1:\text{COUPLE} \xrightarrow{\sqsubseteq} 2:\text{PERSON}$ to indicate that every couple has corresponding persons, but would not normally include the bridge rule $1:\text{COUPLE} \xrightarrow{\supseteq} 2:\text{PERSON}$, because there may be unmarried persons, who would not be partners in any couple.

Informally, a distributed T-box is then a collection of T-boxes \mathbf{T}_i (representing the local IS_i), together with into- and onto- bridge rules, grouped into sets $\mathfrak{B}_{i,j}$ indicating the direction in which they transfer information.

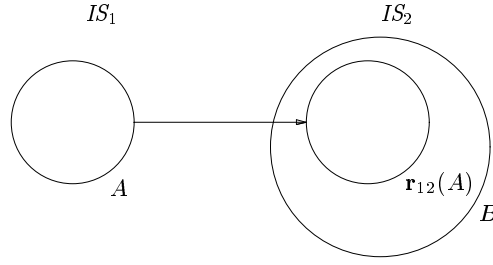


Fig. 2. Intuitive interpretation of $A \stackrel{\sqsubseteq}{\rightarrow} B$

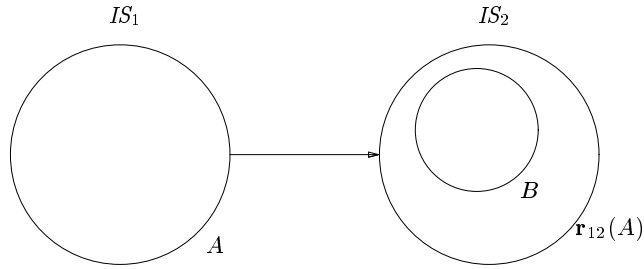


Fig. 3. Intuitive interpretation of $A \stackrel{\sqsupseteq}{\rightarrow} B$

3.3 Some examples revisited

We will now recast some of the earlier examples into the notation of distributed description logics (hereafter DDL), and briefly consider some of the subtle aspects of the representation.

To begin with, the course correspondences in Example 1 yield two bridge rules:

$$\begin{aligned} 1 : \text{AdvancedC} &\stackrel{\sqsubseteq}{\rightarrow} 2 : \text{IntermC} \\ 2 : \text{IntermC} &\stackrel{\sqsupseteq}{\rightarrow} 1 : \text{BasicC} \end{aligned}$$

These allow each IS to import appropriate information from the other, but does not entail $1 : \text{AdvancedC} \sqsubseteq 1 : \text{BasicC}$, because the two bridge rules concern mappings in opposite directions. This is a good example of the peer-to-peer approach, where each IS imports information independently.

Consider next Example 3, involving libraries. We have three T-boxes \mathbf{T}_h , \mathbf{T}_m and \mathbf{T}_s . \mathbf{T}_h and \mathbf{T}_m describe the information systems of the libraries. They both have concepts **BOOK**, corresponding to copies of books that can be loaned, and concepts **PERSON**, to model the borrowers. The role **taken_by** is meant to record who has borrowed a book, so that the concept **BOOK_ON_SHELF**, in \mathbf{T}_h , can be defined as follows

$$\text{BOOK_ON_SHELF} \equiv \text{BOOK} \sqcap \neg \exists \text{taken_by.PERSOON}$$

The T-box \mathbf{T}_s , modeling the students' information system, also includes a concept called **BOOK**, but its meaning is different, since, as we mentioned before, these are abstractions over the libraries' copies of the book. The following bridge rules are intended to capture the fact that students see books based on copies from the respective libraries

$$h:\text{BOOK} \xrightarrow{\sqsubseteq} s:\text{BOOK} \quad (1)$$

$$m:\text{BOOK} \xrightarrow{\sqsubseteq} s:\text{BOOK} \quad (2)$$

Note that this does not imply that *all* books at the Harvard library can be seen in the students database (the mapping \mathbf{r}_{hs} may be partial). Nor does it imply that the same book copy cannot be mapped to several books in $\text{IS}_{Student}$ – this would require a different kind of bridge rule, one expressing that the correspondence has a certain cardinality.

In addition, \mathbf{T}_s uses the role `located_at` to capture the name of the library where the student should go to in order to get the material in question, assuming it is available there. For convenience, \mathbf{T}_s contains a concept, **AVAILABLE_BOOK**, that lets students tell quickly if some book is available:

$$\text{AVAILABLE_BOOK} \equiv \text{BOOK} \sqcap \exists \text{located_at}$$

Since students only want to hear about material located at a library if there are some copies of it on the shelf there, we use onto bridge rules as follows²:

$$h:\text{BOOK_ON_SHELF} \xrightarrow{\sqsubseteq} s:\exists \text{located_at}.\{\text{Harvard}\} \quad (3)$$

$$m:\text{BOOK_ON_SHELF} \xrightarrow{\sqsubseteq} s:\exists \text{located_at}.\{\text{Mit}\} \quad (4)$$

As we shall see later, one of consequences of DDL reasoning with the bridge rules (1–4) is the following subsumption

$$s:\exists \text{located_at}.\{\text{Harvard}\} \sqsubseteq \text{AVAILABLE_BOOK}$$

This follows because the bridge rules allow us to infer in $\text{IS}_{Student}$ that anything that is located at the Harvard library must be a book, and hence an instance of the concept **AVAILABLE_BOOK** defined earlier.

3.4 Distributed A-boxes

In order to deal with correspondences between specific individuals, we can follow two approaches. First, if the description logic is sufficiently expressive, we can state such correspondences by using bridge rules. For example, the correspondence of `couple23` to Gianni and Mary in Example 2, can be expressed by bridge

² Technical note: although we use enumerated concepts, such as $\{\text{Harvard}\}$, which are not in SHIQ , their elements are string constants, which do not have properties of their own. It is known that such language constructs can be eliminated by using mutually exclusive primitive concepts

rules $1 : \{\text{couple23}\} \xrightarrow{\sqsubseteq} 2 : \{\text{Gianni, Mary}\}$ and $1 : \{\text{couple23}\} \xrightarrow{\sqsupseteq} 2 : \{\text{Gianni, Mary}\}$, if the description logics support concepts formed by enumeration.

Otherwise, we need to introduce the individual-level equivalent of bridge-rules.

Definition 2. *If x is an individual in \mathcal{DL}_i , while y_1, y_2, \dots are individuals of \mathcal{DL}_j , then*

- a (partial) individual correspondence is an expression $i : x \mapsto j : y$
- a complete individual correspondence is an expression $i : x \mapsto j : \{y_1, y_2, \dots\}$

The first kind of assertion indicates that the domain relation r_{ij} includes the correspondence between x and y . Note that $1 : \text{couple23} \mapsto 2 : \text{Gianni}$ and $1 : \text{couple23} \mapsto 2 : \text{Mary}$ do not capture fully the relationship between couple23 , Gianni and Mary , because additional objects may still be in correspondence with couple23 . Hence the need for complete correspondences: $1 : \text{couple23} \mapsto 2 : \{\text{Gianni, Mary}\}$

Distributed A-boxes than connect the individual A-boxes of each IS, through the same bridge rules we used for distributed T-boxes.

4 Formal Definitions

We present next the formal specification of the above notions, starting with the semantics of ordinary DLs.

4.1 Formal Syntax and Semantics of Description Logics

A typical DL, such as \mathcal{SHIQ}^3 [19], would start with atomic concepts A , as well as constants **ANYTHING** and **NOTHING**, denoting the universe and the empty set respectively, and then build more complex descriptions according to the recursive syntax in the second column of Figure 4. An additional concept constructor that we may use, but is not available in \mathcal{SHIQ} , is enumeration, which describes a concept by enumerating its instances: $\{IN_1, \dots, IN_n\}$.

We begin with the notion of interpretation/structure, which is familiar to anyone who has studied predicate calculus. Specifically, an interpretation \mathcal{I} is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, which treats concepts and roles as unary and binary predicates, assigning subsets $A^{\mathcal{I}}$ of the non-empty domain $\Delta^{\mathcal{I}}$ to atomic concepts, and subsets $R^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to atomic roles, as well as distinct values of $\Delta^{\mathcal{I}}$ to different named individuals. The interpretation then proceeds recursively, driven by the syntax of complex concept and role constructors, as shown in column three of Figure 4. (In the case of recursive concept definitions, the interpretation assigns extensions to all descriptions, and then complex concepts constrain the valid interpretations.)

We can now define formally subsumption, and then introduce convenient notation to talk about interpretations “satisfying” T-boxes.

³ This particular DL is of interest, among others, since it is the basis of the proposed DAML+OIL language for expressing web-ontologies.

Concept construct name	Syntax	Semantics
primitive concept	A	$A^{\mathcal{I}}$
top concept	ANYTHING	$\Delta^{\mathcal{I}}$
bottom concept	NOTHING	\emptyset
conjunction	$C_1 \sqcap \dots \sqcap C_n$	$C_1^{\mathcal{I}} \cap \dots \cap C_n^{\mathcal{I}}$
disjunction	$C_1 \sqcup \dots \sqcup C_n$	$C_1^{\mathcal{I}} \cup \dots \cup C_n^{\mathcal{I}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
value restriction	$\forall R.C$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$
exists restriction	$\exists R.C$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \neq \emptyset\}$
number restrictions	$\geq n R$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \geq n\}$
	$\leq n R$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \leq n\}$
qualified number restriction	$\geq n R.C$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \geq n\}$
	$\leq n R.C$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \leq n\}$
Role construct name	Role	Semantics
primitive role	P	$P^{\mathcal{I}}$
role inverse	R^-	$\{(y, x) \mid (x, y) \in R^{\mathcal{I}}\}$

Fig. 4. Syntax and semantics of the \mathcal{SHIQ} Description Logic [19]

- $\mathcal{I} \models C \sqsubseteq D$ (* read as D subsumes C in interpretation \mathcal{I} *) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- $\mathcal{I} \models \mathbf{T}$ (* The definitions and subsumptions in \mathbf{T} are satisfied in \mathcal{I} *) if $\mathcal{I} \models A \sqsubseteq B$, for all $A \sqsubseteq B$ in \mathbf{T} .
- $\mathbf{T} \models C \sqsubseteq D$ (* D subsumes C with background theory/ T -box \mathbf{T} *) if $\mathcal{I} \models C \sqsubseteq D$ for all interpretations \mathcal{I} such that $\mathcal{I} \models \mathbf{T}$.

These definitions are extended to A-boxes according to the rules

- $\mathcal{I} \models C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$.
- $\mathcal{I} \models p(a, b)$ if $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in p^{\mathcal{I}}$.
- $\mathcal{I} \models \mathbf{A}$ if $\mathcal{I} \models \pi$ for every assertion $\pi = C(a), p(a, b) \in \mathbf{A}$.
- $\mathbf{K} \models C(a)$ iff $\mathcal{I} \models C(a)$ for all interpretations \mathcal{I} such that $\mathcal{I} \models \mathbf{K}$. Similarly for $p(a, b)$.

A particular description logic has a complexity of reasoning associated with the various questions that one wants to answer, such as subsumption or concept membership. This complexity depends on the set of constructors offered by that language. Much of the effort in DL research has been in identifying different sets of such constructors which have at least decidable inferences, and characterizing their computational complexity. We remark on the variable-free nature of the DL formalisms, which limit their expressive power but at the same time contribute to their ability to reason efficiently even with incomplete information (see [7]).

4.2 Formal syntax and semantics of Distributed Description Logics

The following is just a precise restatement of the notion of DDL introduced in the previous section.

Definition 3. Given a set I of indexes, let $\{\mathcal{DL}_i\}_{i \in I}$ be a collection of description logics. A distributed T-box (DTB) $\mathfrak{T} = \langle \{\mathbf{T}_i\}_{i \in I}, \mathfrak{B} \rangle$ consists of a set of ordinary DL T-boxes $\{\mathbf{T}_i\}_{i \in I}$, and a set $\mathfrak{B} = \{\mathfrak{B}_{ij}\}_{i \neq j \in I}$ of bridge rules from i to j . For every $k \in I$, all descriptions in \mathbf{T}_k must be in the corresponding language \mathcal{DL}_k , and for every bridge rule $i : A \xrightarrow{\sqsubseteq} j : B$ or $i : A \xrightarrow{\sqsupseteq} j : B$ in \mathfrak{B}_{ij} , the concepts A and B must be in the languages \mathcal{DL}_i and \mathcal{DL}_j respectively.

A distributed A-box (DAB) $\mathfrak{A} = \langle \{\mathbf{A}_i\}_{i \in I}, \mathfrak{C} \rangle$ consists of a set of A-boxes $\{\mathbf{A}_i\}_{i \in I}$, and a set $\mathfrak{C} = \{\mathfrak{C}_{ij}\}_{i \neq j \in I}$ of partial and complete individual correspondences from i to j . For every $k \in I$, all descriptions in \mathbf{A}_k must be in the corresponding language \mathcal{DL}_k , and for every correspondence rule $i : x \mapsto j : y$ or $i : x \bar{\mapsto} j : \{y_1, y_2, \dots\}$ in \mathfrak{C}_{ij} , the individual name x must be in \mathcal{DL}_i , and y_1, y_2, \dots be in \mathcal{DL}_j .

A DDL knowledge base is then a pair $\langle \mathfrak{T}, \mathfrak{A} \rangle$, consisting of a distributed T-box and a distributed A-box

We provide semantics for distributed description logics by using local interpretations for the individual information systems, and connecting their domains using relations \mathbf{r}_{ij} .

Definition 4. A distributed interpretation $\mathfrak{I} = \langle \{\mathcal{I}_i\}_{i \in I}, \mathbf{r} \rangle$ of \mathfrak{T} consists of interpretations \mathcal{I}_i for \mathcal{DL}_i over domain $\Delta^{\mathcal{I}_i}$, and a function \mathbf{r} associating to each $i, j \in I$ a binary relation $\mathbf{r}_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$. We use $\mathbf{r}_{ij}(d)$ to denote $\{d' \in \Delta^{\mathcal{I}_j} \mid \langle d, d' \rangle \in \mathbf{r}_{ij}\}$, and for any $D \subseteq \Delta^{\mathcal{I}_i}$, we use $\mathbf{r}_{ij}(D)$ to denote $\bigcup_{d \in D} \mathbf{r}_{ij}(d)$.

Definition 5. A distributed interpretation \mathfrak{I} d-satisfies (written $\mathfrak{I} \models_d$) the elements of a DTB $\mathfrak{T} = \langle \{\mathbf{T}_i\}_{i \in I}, \mathfrak{B} \rangle$ according to the following clauses: For every $i, j \in I$

- $\mathfrak{I} \models_d i : A \xrightarrow{\sqsubseteq} j : G$, if $\mathbf{r}_{ij}(A^{\mathcal{I}_i}) \subseteq G^{\mathcal{I}_j}$ (* Satisfiability of into-bridge rules *)
- $\mathfrak{I} \models_d i : B \xrightarrow{\sqsupseteq} j : H$, if $\mathbf{r}_{ij}(B^{\mathcal{I}_i}) \supseteq H^{\mathcal{I}_j}$ (* Satisfiability of onto-bridge rules *)
- $\mathfrak{I} \models_d i : A \sqsubseteq B$, if $\mathcal{I}_i \models A \sqsubseteq B$ (* Satisfiability of local subsumptions *)
- $\mathfrak{I} \models_d \mathbf{T}_i$ if $\mathcal{I}_i \models \mathbf{T}_i$. (* Satisfiability of local T-boxes *)
- $\mathfrak{I} \models_d \mathfrak{T}$ if, for every $i \in I$, $\mathfrak{I} \models_d \mathbf{T}_i$, and \mathfrak{I} d-satisfies every bridge rule in \mathfrak{B} .

Finally, $\mathfrak{T} \models_d i : C \sqsubseteq D$ if, for every distributed interpretation \mathfrak{I} , $\mathfrak{I} \models_d \mathfrak{T}$ implies $\mathfrak{I} \models_d i : C \sqsubseteq D$.

Concerning individuals, we have the following

Definition 6. A distributed interpretation \mathfrak{I} d-satisfies the elements of a DAB $\mathfrak{A} = \langle \{\mathbf{A}_i\}_{i \in I}, \mathfrak{C} \rangle$ according to the following clauses: For every $i, j \in I$

- $\mathfrak{I} \models_d i : x \mapsto j : y$, if $y^{\mathcal{I}_j} \in \mathbf{r}_{ij}(x^{\mathcal{I}_i})$ (* Satisfiability of individual correspondences *)
- $\mathfrak{I} \models_d i : x \bar{\mapsto} \{y_1, y_2, \dots\}$ if $\mathbf{r}_{ij}(x^{\mathcal{I}_i}) = \{y_1^{\mathcal{I}_j}, y_2^{\mathcal{I}_j}, \dots\}$ (* Satisfiability of complete correspondences *)
- $\mathfrak{I} \models_d i : C(a)$, if $\mathcal{I}_i \models C(a)$ (* Satisfiability of local assertions *)
- $\mathfrak{I} \models_d i : p(a, b)$, if $\mathcal{I}_i \models p(a, b)$ (* Satisfiability of local assertions *)

- $\mathcal{I} \models_d \mathbf{A}_i$ iff $\mathcal{I} \models_d \pi$ for every assertion $\pi = C(a), p(a, b)$ in \mathbf{A}_i . (* Satisfiability of local \mathbf{A} -boxes *)
- $\mathcal{I} \models_d \mathfrak{A}$ if, for every $i \in I$, $\mathcal{I} \models_d \mathbf{A}_i$, and \mathcal{I} d -satisfies every individual correspondence in \mathcal{C} .

Finally, $\mathfrak{A} \models_d i : C(a)$ if, for every distributed interpretation \mathcal{I} , $\mathcal{I} \models_d \mathfrak{A}$ implies $\mathcal{I} \models_d i : C(a)$. Similarly for $p(a, b)$.

4.3 Library Example Revisited Formally

Ignoring one of the libraries, \mathbf{T}_m say, in order to simplify matters, we can define the distributed T-box $\mathfrak{T}_{lib} = \langle \mathbf{T}_h, \mathbf{T}_s, \mathfrak{B}_{hs} \rangle$, where \mathfrak{B}_{hs} contains the single bridge rules:

$$h : \text{BOOK_ON_SHELF} \xrightarrow{\exists} s : \exists \text{located_at.}\{\text{Harvard}\} \quad (5)$$

Figure 5 provides an example distributed interpretation \mathcal{I}_{lib} for \mathfrak{T}_{lib} .

$$\begin{aligned} \Delta^{\mathcal{I}_h} &= \{\text{Tractatus}(1), \text{Tractatus}(2), \text{DB_Pples}, \text{Mario}\} \\ \text{BOOK}^{\mathcal{I}_h} &= \{\text{Tractatus}(1), \text{Tractatus}(2), \text{DB_Pples}\} \\ \text{PERSON}^{\mathcal{I}_h} &= \{\text{Mario}\} \\ \text{taken_by}^{\mathcal{I}_h} &= \{\langle \text{Tractatus}(1), \text{Mario} \rangle\} \\ \Delta^{\mathcal{I}_s} &= \{\text{Tractatus}, \text{Philosophical_Investigations}, \text{Harvard}, \text{Mit}\} \\ \text{BOOK}^{\mathcal{I}_s} &= \{\text{Tractatus}, \text{Philosophical_Investigations}\} \\ \text{located_at}^{\mathcal{I}_s} &= \left\{ \begin{array}{l} \langle \text{Tractatus}, \text{Harvard} \rangle \\ \langle \text{Philosophical_Investigations}, \text{Mit} \rangle \end{array} \right\} \\ \mathbf{r}_{hs} &= \left\{ \begin{array}{l} \langle \text{Tractatus}(1), \text{Tractatus} \rangle \\ \langle \text{Tractatus}(2), \text{Tractatus} \rangle \end{array} \right\} \end{aligned}$$

Fig. 5. Example of distributed interpretation for \mathfrak{T}_{lib}

Note that bridge rule is satisfied by \mathcal{I}_{lib} even though $\text{BOOK}^{\mathcal{I}_h}$ is not contained in $\text{BOOK}^{\mathcal{I}_s}$; indeed, $\mathbf{r}_{hs}(\text{BOOK}^{\mathcal{I}_h}) = \{\text{Tractatus}\}$, is a subset of $\text{BOOK}^{\mathcal{I}_s}$, which is $\{\text{Tractatus}, \text{Philosophical_Investigations}\}$. \mathcal{I}_{lib} also satisfies bridge rule (3), since $\mathbf{r}_{hs}(\text{BOOK_ON_SHELF}^{\mathcal{I}_h}) = \mathbf{r}_{hs}(\{\langle \text{Tractatus}(2), \text{DB_Pples} \rangle\}) = \{\text{Tractatus}\}$, which is a superset of $(\exists \text{located_at.}\{\text{Harvard}\})^{\mathcal{I}_s} = \{\text{Tractatus}\}$. Recall that \mathfrak{T}_{lib} entails

$$s : \exists \text{located_at.}\{\text{Harvard}\} \sqsubseteq \text{AVAILABLE_BOOK}$$

which can be verified by considering all possible distributed interpretations.

The above example exhibits a common pattern of inference in our DDL:

starting from
 A subsumes B in IS_1
 A is mapped *into* G by a bridge rule
 B is mapped *onto* H by a bridge rule
conclude that
 G subsumes H in IS_2

The following is another example of this inference: suppose \mathbf{T}_1 contains the subsumption assertion

$$\text{Villa} \sqsubseteq \text{SecondResidence}$$

and there are bridge rules

$$1: \text{SecondResidence} \xrightarrow{\sqsubseteq} 2: \text{Dwelling}$$

$$1: \text{Villa} \xrightarrow{\supseteq} 2: \text{Cottage}$$

One can then conclude that $2: \text{Cottage} \sqsubseteq \text{Dwelling}$. This inference is illustrated by the diagram in Figure 4.3, which should also provide some of the intuition behind it. (The horizontal arrows describe the bridge rules.)

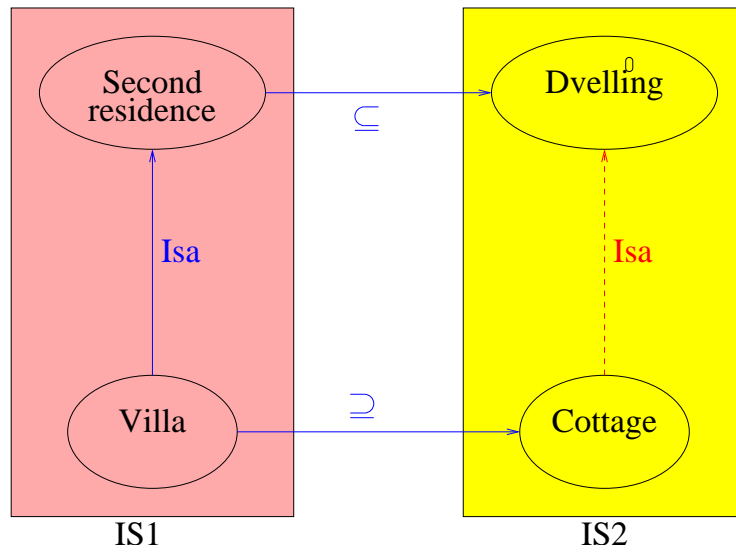


Fig. 6. Example of Inferred Subsumption Relationship (dashed line) in DDL

5 Some properties of DDL

In this section, we will restrict our attention to the simplest kinds of DDL, namely distributed T-boxes involving only two IS, and a single set of bridge rules between them: $\mathfrak{T}_{12} = \langle \mathbf{T}_1, \mathbf{T}_2, \mathfrak{B}_{12} \rangle$.

The following is a list of intuitively desirable properties for such a system, based on our motivation:

1. When deducing things at IS_i in the distributed system, all local information should be available. Formally, if $\mathbf{T}_i \models X \sqsubseteq Y$, then $\mathfrak{T}_{12} \models_d i: X \sqsubseteq Y$, for $i = 1, 2$.
2. In the absence of bridge rules, no information should pass between the component systems. Formally, if $\langle \mathbf{T}_1, \mathbf{T}_2, \emptyset \rangle \models_d i: X \sqsubseteq Y$ then $\mathbf{T}_i \models X \sqsubseteq Y$, for $i = 1, 2$.
3. A DDL should exhibit “directionality”/“no backflow”: we have said that \mathfrak{B}_{12} contains bridge rules that are setup to provide information flow from IS_1 to IS_2 . Therefore, such a set up should not affect, by itself, reasoning in IS_1 . Formally, we would like to have that if $\mathfrak{T}_{12} \models_d 1: A \sqsubseteq B$, then $\mathbf{T}_1 \models A \sqsubseteq B$. This would also allow for more effective reasoning because there would be no need for a feedback loop between new inferences in IS_2 and those in IS_1 .
4. A distributed system should not allow local inconsistencies to “pollute” the entire system, in the sense that if the information at IS_i is not satisfiable, then deductions at other sites should not be affected. Formally, this would mean that if \mathbf{T}_i is inconsistent then $\mathfrak{T}_{12} \models_d j: X \sqsubseteq Y$ iff $\mathbf{T}_j \models X \sqsubseteq Y$ for $j = 1, 2$.

It can be easily seen that our definition of DDL does have the first two properties, if the component T-boxes are consistent.

As far as “backflow” is concerned, it is unfortunately possible to use onto bridge rules to enforce certain properties of descriptions in IS_1 . For example, a rule such as $1: A \xrightarrow{\sqsubseteq} 2: \text{ANYTHING}$ requires every distributed interpretation to have the property $\mathbf{r}_{12}(A^{\mathcal{I}_1}) \neq \emptyset$, because the extension of the ANYTHING concept is never empty, and hence there must always be individuals in $A^{\mathcal{I}_1}$ that correspond to it. This kind of reasoning can sometimes be translated into new subsumptions for IS_1 , which depend on IS_2 . To show this, let us introduce a role constant, $\text{Universal}_{\text{ROLE}}$. It relates every possible pair of objects: $\text{Universal}_{\text{ROLE}}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$. Now consider the DTB \mathfrak{T} , with $\mathbf{T}_2 = \{ \text{ANYTHING} \sqsubseteq G \}$ and one bridge rule $\{ 1: A \xrightarrow{\sqsubseteq} 2: G \}$. It d-entails the formula $1: \text{ANYTHING} \sqsubseteq \exists \text{Universal}_{\text{ROLE}}. A$ because, if A 's interpretation is never empty then there is always some A -object to which every object can be related by $\text{Universal}_{\text{ROLE}}$.

Interestingly, it is possible to prove that for some DDLs no backflow will occur. For example, if the components of a DTB \mathfrak{T}_{12} only involve the \mathcal{SHIQ} description logic, (which does not support $\text{Universal}_{\text{ROLE}}$, of course), then for consistent \mathfrak{T}_{12} , $\mathfrak{T}_{12} \models_d 1: A \sqsubseteq B$ iff $\mathbf{T}_1 \models A \sqsubseteq B$. (We do not present this result formally in this paper since there is no space for its formal derivation.)

5.1 Inconsistency in DDL

As far as inconsistency propagation is concerned, unfortunately if one of the component T-boxes is unsatisfiable then there is no distributed interpretation satisfying the entire DTB, and therefore every possible assertion is automatically d-entailed by an unsatisfiable DTB. (This kind of inconsistency propagation is not peculiar to DDL – it is a feature of most logic-based formal models of IS with multiple components.)

Looking at the issue more closely in the context of DTBs. From the definition, it can be seen that a DTB \mathfrak{T} is unsatisfiable if each distributed interpretation \mathfrak{J} does not satisfy either some local T-box, some bridge rule (for instance $1:\text{NOTHING} \xrightarrow{\sqsubseteq} 2:\text{ANYTHING}$), or some combination thereof (for instance $\{A \sqsubseteq \text{NOTHING}\}, \{\text{ANYTHING} \sqsubseteq G\}, \{1:A \xrightarrow{\sqsubseteq} 2:G\}$).

Conversely, if some \mathbf{T}_i of \mathfrak{T} is not satisfiable, then the entire DTB is unsatisfiable, because there is no distributed interpretation for it, and hence everything can be deduced from \mathfrak{T} , because d-entailment quantifies over the set of distributed interpretations, which in this case is *empty*. This is an unsatisfactory state of affairs for distributed IS. Let us consider some possibilities in the case of \mathfrak{T}_{12} .

- If just \mathbf{T}_2 is unsatisfiable, we do not want this to affect reasoning in IS_1 , especially because of the “no back-flow” stance. So reasoning in \mathfrak{T}_{12} should reduce to reasoning in \mathbf{T}_1 .
- If just \mathbf{T}_1 is unsatisfiable, there is still some desire for this inconsistency not to “infect” the reasoning of IS_2 , at least not to the point that *all* conclusions of the form $2:X \sqsubseteq Y$ are entailed by \mathfrak{T}_{12} .
- To help the integrator, it would be desirable to be able to distinguish the above cases from the one where the inconsistency is due the effect of bridge rules.

Although a simple syntactic solution is available – just check each local T-box for consistency, and define d-interpretations to exclude such inconsistent component IS. We would prefer to find a semantic solution having these properties in order to be sure of its coherence. So let us introduce a new, special DL interpretation, $\mathcal{I}^\delta = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}^\delta} \rangle$, where $\Delta^{\mathcal{I}}$ is the original non empty domain, and $\cdot^{\mathcal{I}^\delta}$ makes the denotation of every description, even **NOTHING**, be the whole domain $\Delta^{\mathcal{I}}$. Intuitively \mathcal{I}^δ provides an interpretation even to a locally inconsistent T-box; indeed, in \mathcal{I}^δ , every subsumption $A \sqsubseteq B$ is satisfied, including **ANYTHING** \sqsubseteq **NOTHING**. This means that even if a DTB has an inconsistent T-box \mathbf{T}_k , it will have *some* distributed interpretations $\{\mathfrak{J}_i\}_{i \in I}$ – ones where $\mathfrak{J}_k = \mathcal{I}^\delta$; this means that in the definition of d-entailment we will not be quantifying over the empty set of distributed interpretations. Moreover, the addition of this special interpretation does not change the set of theorems and the set of logical consequences of any component DL IS, which therefore maintains all its formal and computational properties.

Suppose we repeat all previous definitions, but using this more encompassing notion of satisfaction, to obtain δ -satisfies and δ -entails \models_δ .

The following proposition shows that when \mathbf{T}_2 is inconsistent, we have the desired effect, and more generally, we always get “no backflow”.

Proposition 1. *If $\mathfrak{T}_{12} \models_\delta 1:A \sqsubseteq B$ then $\mathbf{T}_1 \models A \sqsubseteq B$.*

Proof. Recall that the general definition of $\mathbf{T} \models M \sqsubseteq N$ is that for every interpretation \mathcal{I} , if $\mathcal{I} \models \mathbf{T}$, then $\mathcal{I} \models M \sqsubseteq N$. So let \mathcal{I}_1 be an arbitrary model of \mathbf{T}_1 . We must show that $\mathcal{I}_1 \models 1:A \sqsubseteq B$. Consider the distributed model $\{\{\mathcal{I}_1, \mathcal{I}^\delta\}, \mathbf{r}_{12} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}^\delta}\}$: it satisfies \mathbf{T}_1 by assumption, and it satisfies \mathbf{T}_2 by our prior comment concerning \mathcal{I}^δ . Moreover, the right hand side of all bridge rules corresponds to the set $\Delta^{\mathcal{I}^\delta}$, so that they are satisfied too. Therefore, since $\mathfrak{T}_{12} \models_\delta 1:A \sqsubseteq B$, we must also have $\{\mathcal{I}_1, \mathcal{I}^\delta\} \models_\delta 1:A \sqsubseteq B$, which includes $\mathcal{I}_1 \models A \sqsubseteq B$.

In the case when \mathbf{T}_1 is inconsistent we have:

Proposition 2. *If \mathbf{T}_1 is an inconsistent T-box, then if $\mathfrak{T}_{12} = \langle \mathbf{T}_1, \mathbf{T}_2, \mathfrak{B}_{12} \rangle \models_\delta 2:E \sqsubseteq F$ then $\mathbf{T}'_2 \models E \sqsubseteq F$, where \mathbf{T}'_2 is obtained by extending \mathbf{T}_2 with $\{G \sqsubseteq H \mid \text{if } 1:A \xrightarrow{\sqsubseteq} 2:H \text{ and } 1:B \xrightarrow{\sqsupseteq} 2:G \text{ in } \mathfrak{B}_{12} \text{ for some } A \text{ and } B\}$.*

This shows that when \mathbf{T}_1 is inconsistent, results in \mathbf{T}_2 are only affected by the bridge rules.

Proof. Let \mathcal{I}_2 be a model of \mathbf{T}'_2 and let $\mathfrak{J} = (\{\mathcal{I}^\delta, \mathcal{I}_2\}, \Delta^{\mathcal{I}_1} \times (\cap H_k^{\mathcal{I}_2}))$ be the distributed interpretation where $\{1:A_k \xrightarrow{\sqsubseteq} 2:H_k\}$ is the set of all-into bridge rules in \mathfrak{B}_{12} . Clearly $\mathfrak{J} \models_\delta \mathbf{T}_i$. As far as bridge rules, note that by construction, $\mathbf{r}_{12}(\Delta^{\mathcal{I}_1}) = \cap H_k$, so all into-rules are automatically satisfied. On the other hand, since every concept $2:G$ appearing on the right side of an onto rule is required to have the property $G \sqsubseteq H_k$ for all k , then $\mathbf{r}_{12}(G) \subseteq \cap H_k$, so the mapping satisfies all onto rules, except possibly the ones where $1:B$ denotes the empty set. However, in \mathcal{I}^δ , nothing denotes the empty set, not even NOTHING, so this case cannot arise. Therefore, since $\mathfrak{J} \models_\delta \mathfrak{T}_{12}$, we get by hypothesis that $\mathcal{I}_2 \models E \sqsubseteq F$.

6 Relating DDL and ordinary DL

We are interested in finding a connection between DLs and DDLs, which would allow us to transfer theoretical results (such as complexity analysis) and reasoning techniques from the extensive current DL literature.

We will do so by building a “global” DL, which encodes the information available in the local T-boxes and the bridge rules of the DDL. To do so, we start by creating a language, \mathcal{GDL} , for descriptions in this global DL. Suppose one is given a family of description logics $\{\mathcal{DL}_i\}_{i \in I}$. For any primitive concept A (resp. role R) of \mathcal{DL}_i , let $i:A$ (resp. $i:R$) be a primitive concept (resp. role) of \mathcal{GDL} . Moreover, \mathcal{GDL} will use all the concept and role constructors appearing

in any \mathcal{DL}_i . Therefore \mathcal{GDL} permits the equivalents of at least all composite descriptions of \mathcal{DL}_i . \mathcal{GDL} has the usual top and bottom concepts, ANYTHING and NOTHING, which in this case are distinguished from the tops and bottoms of the hierarchies in \mathcal{DL}_i , which are now ordinary concepts, with names $i:\text{ANYTHING}$ and $i:\text{NOTHING}$. (To emphasize this, we will use instead the symbols Top_i and Bot_i .) \mathcal{GDL} has a special set of role symbols R_{ij} , which will be used to simulate the domain relations, as well as an additional role symbol \hat{P} , used strictly for technical reasons.

We start by providing a translation $\#()$ from \mathcal{DL}_i concepts/roles to concepts/roles in the global DL, and then extending it to map entire DDL T-boxes. Not unexpectedly, such a translation will be based on the recursive structure of concepts, which is based on the use of DL *operators*. To emphasize this, we will view concept and role constructors as functors, ρ , that take as arguments simpler descriptions. For example, $\forall p.D$ can be viewed as **all**(*<role expression>*, *<concept expression>*), with **all** being the constructor. The mapping $\#$ is defined recursively as follows:

1. $\#(i, M) = i:M$ for atomic concepts and roles M , as well as $M = \text{ANYTHING}$ and $M = \text{NOTHING}$. (* *Atomic base cases.* *)
2. if ρ is a concept constructor taking k arguments, then $\#(i, \rho(M_1, \dots, M_k)) = \text{Top}_i \sqcap \rho(\#(i, M_1), \dots, \#(i, M_k))$. (* *Complex concepts are translated by first translating the constructor's arguments, and then intersecting with the domain Top_i .* *)
3. if ρ is a role constructor taking k arguments, then $\#(i, \rho(M_1, \dots, M_k)) = \rho(\#(i, M_1), \dots, \#(i, M_k))$. (* *Complex roles are translated by first translating the arguments.*⁴ *)

For example, $\#(i, \text{PERSON} \sqcap \forall \text{likes}^- . \text{TEACHER})$ produces

$$\text{Top}_i \sqcap i: \text{PERSON} \sqcap \forall (i: \text{likes})^- . (\text{Top}_i \sqcap i: \text{TEACHER})$$

We are now ready to produce the global DL T-box:

Definition 7. *Applying $\#$ to a DTB $\mathfrak{T} = \{\{\mathbf{T}_i\}_{i \in I}, \mathfrak{B}\}$, yields a T-box $\#(\mathfrak{T})$ in the language \mathcal{GDL} , consisting of the following axioms:*

1. (* *Copies of axioms from local T-boxes.* *)
 $\#(i, A) \sqsubseteq \#(i, B)$ for all $i: A \sqsubseteq B \in \mathbf{T}_i$;
2. (* *Translations of into bridge rules as value restrictions on R_{ij} .* *)
 $\#(i, A) \sqsubseteq \forall R_{ij} . \#(j, G)$ for every into bridge rule $i: A \xrightarrow{\sqsubseteq} j: G \in \mathfrak{B}$;
3. (* *Translations of onto bridge rules as existential restrictions on the inverse of R_{ij} .* *)
 $\#(j, H) \sqsubseteq \exists R_{ij}^- . \#(i, A)$ for every onto bridge rule $i: A \xrightarrow{\sqsupseteq} j: G \in \mathfrak{B}$;

⁴ We do not use the equivalent of top role and role conjunction instead of ANYTHING and \sqcap in the last definition, because these constructors are rarely present in DLs, unlike ANYTHING and \sqcap , which are ubiquitous.

4. (* Restrictions on role R_{ij} so it connects only objects in \mathbf{T}_i and \mathbf{T}_j . *)
 $\text{ANYTHING} \sqsubseteq \forall R_{ij}. \text{Top}_j$ (* the range of R_{ij} is $\Delta^{\mathcal{I}_j}$ *)
 $\neg(\text{Top}_i) \sqsubseteq \forall R_{ij}. \text{NOTHING}$ (* R_{ij} is undefined outside $\Delta^{\mathcal{I}_i}$ *)
5. (* Restricting Bot_i to always be the incoherent concept. *)
 $\text{Bot}_i \sqsubseteq \text{NOTHING}$.
6. (* Ensuring that Top_i is the proper local top of its IS-A hierarchy *)
 $i:A \sqsubseteq \text{Top}_i$, for every atomic concept A of \mathcal{DL}_i
7. (* Ensuring that Top_i is not empty *)
 $\text{ANYTHING} \sqsubseteq \exists \hat{P}. \text{Top}_i$
8. (* Ensuring that every i-role p has as domain and range Top_i *)
 $\text{Top}_i \sqsubseteq \forall(i:p). (\text{Top}_i)$ for every role p of \mathcal{DL}_i (* the range of $i:p$ is in $\Delta^{\mathcal{I}_i}$ *);
 $\neg(\text{Top}_i) \sqsubseteq \forall(i:p). \text{NOTHING}$ (* $i:p$ is undefined outside $\Delta^{\mathcal{I}_i}$ *)

We propose next that under some circumstances d-entailment can be reduced to ordinary DL-reasoning through the use of the above translation; namely that $\#(\mathfrak{I}) \models \#(i, X) \sqsubseteq \#(i, Y)$ if and only if $\mathfrak{I} \models_d i: X \sqsubseteq Y$.

Its proof would be based, as usual, on constructing an appropriate interpretation for $\#(\mathfrak{I})$, given one for \mathfrak{I} , and conversely. In turn, the proof of correctness of the constructions relies on induction on the structure of concepts. So the proof depends on the concept and role constructors of the particular \mathcal{DL} 's involved. In fact, the proof is quite similar for a great variety of constructors, ρ , so we will try to state it in more general terms in order to abstract out this common part of the proof.

We start by noting that the semantics of most descriptions is compositional, so that the denotation of $\rho(\text{arg}_1, \dots, \text{arg}_n)$ is a function, f_ρ , of the denotation of its arguments. For example, $\mathbf{all}(\mathbf{R}, \mathbf{C})^{\mathcal{I}}$ is equal to $f_{\text{all}}(R^{\mathcal{I}}, C^{\mathcal{I}})$ for an associated “semantic function” $f_{\text{all}}(XR, XC) = \{d \in \Delta^{\mathcal{I}} \mid XR(d) \subseteq XC\}$. However, sometimes such a function depends on more than the interpretation of its arguments — in this case the set $\Delta^{\mathcal{I}}$. Although we can imagine more general scenarios, we will consider here only constructors that depend in some explicit way on $\Delta^{\mathcal{I}}$. We will require that the function f_ρ make this dependence explicit by taking an additional argument, DY , which is to be instantiated by $\Delta^{\mathcal{I}}$ when getting the denotation of a particular concept constructed with ρ . For example, f_{all} should take three arguments, XR , XC and DY , with $f_{\text{all}}(XR, XC, DY)$ defined by $\{d \in DY \mid XR(d) \subseteq XC\}$; and $(\forall p.C)^{\mathcal{I}}$ would be expressed as $f_{\text{all}}(p^{\mathcal{I}}, C^{\mathcal{I}}, \Delta^{\mathcal{I}})$.

Semantic functions with the following property will be of interest to us:

Definition 8. Let ρ be a concept constructor whose semantics can be expressed as $\rho(\text{arg}_1, \dots, \text{arg}_n)^{\mathcal{I}} = f_\rho(\text{arg}_1^{\mathcal{I}}, \dots, \text{arg}_n^{\mathcal{I}}, \Delta^{\mathcal{I}})$, for a function $f_\rho(X_1, \dots, X_n, DY)$ whose definition contains no references to \mathcal{I} .⁵ Let $B_1, \dots, B_n, W, \Delta$ be sets such that $W \subseteq \Delta$, and each B_j is either a subset of W or of $W \times W$, $1 \leq j \leq n$. Then ρ is called a local constructor if f_ρ satisfies the condition

$$W \cap f_\rho(B_1, \dots, B_n, \Delta) = f_\rho(B_1, \dots, B_n, W)$$

⁵ In the case when one of the arguments arg_k of the constructor ρ is not a concept or a role description — e.g., for number restrictions $\leq np$, one argument is an integer — $\text{arg}_k^{\mathcal{I}}$ is extended to evaluate to arg_k .

when it is a concept constructors, or

$$f_\rho(B_1, \dots, B_n, \Delta) = f_\rho(B_1, \dots, B_n, W)$$

when it is a role constructor.

The main result is then

Theorem 1. *Suppose \mathfrak{T} is a DTB in $\{\mathcal{DL}_i\}$, where every concept and role constructor is local. Then $\#(\mathfrak{T}) \models \#(i, X) \sqsubseteq \#(i, Y)$ if and only if $\mathfrak{T} \models_d i: X \sqsubseteq Y$.*

Proof. “If”. Let $\mathcal{I} = \langle \{\mathcal{I}^i\}, \{\mathbf{r}_{ij}\} \rangle$ be a d-interpretation. Define $\mathcal{I}^\#$ to be the interpretation with domain $\Delta^{\mathcal{I}^\#} = \sqcup_i \Delta^{\mathcal{I}^i}$ (the disjoint union of the domains of \mathcal{DL}_i), which interprets $i: A$ in \mathcal{GDL} according to the rule $\#(i, A)^{\mathcal{I}^\#} = A^{\mathcal{I}^i}$, whenever A is an atomic concept, atomic role, ANYTHING and NOTHING. Moreover, let $R_{ij}^{\mathcal{I}^\#}$ equal \mathbf{r}_{ij} , and $\hat{P}^{\mathcal{I}^\#} = \Delta^{\mathcal{I}^\#} \times \Delta^{\mathcal{I}^\#}$.

We then start by verifying that

$$\#(i, M)^{\mathcal{I}^\#} = M^{\mathcal{I}^i}$$

for arbitrary concepts M in \mathcal{DL}_i . The proof is by structural induction on M . The base cases for atomic concepts and roles hold by construction. So consider some composite concept description $\rho(arg_1, \dots, arg_n)$, where the arguments are descriptions or objects with invariant value w.r.t. interpretation (e.g., numbers). Then

$$\begin{aligned} \#(i, \rho(arg_1, \dots, arg_n))^{\mathcal{I}^\#} &= (Top_i \sqcap \rho(\#(i, arg_1), \dots, \#(i, arg_n)))^{\mathcal{I}^\#} \\ &= \Delta^{\mathcal{I}^i} \cap f_\rho(\#(i, arg_1)^{\mathcal{I}^\#}, \dots, \#(i, arg_n)^{\mathcal{I}^\#}, \Delta^{\mathcal{I}^\#}) \\ &= \Delta^{\mathcal{I}^i} \cap f_\rho(arg_1^{\mathcal{I}^i}, \dots, arg_n^{\mathcal{I}^i}, \Delta^{\mathcal{I}^\#}) \end{aligned}$$

On the other hand, $\rho(arg_1, \dots, arg_n)^{\mathcal{I}^i} = f_\rho(arg_1^{\mathcal{I}^i}, \dots, arg_n^{\mathcal{I}^i}, \Delta^{\mathcal{I}^i})$. By letting W be $\Delta^{\mathcal{I}^i}$, B_i be $arg_i^{\mathcal{I}^i}$, and Δ be $\Delta^{\mathcal{I}^\#}$ in the definition of local constructor, we can have that $\Delta^{\mathcal{I}^i} \cap f_\rho(arg_1^{\mathcal{I}^i}, \dots, arg_n^{\mathcal{I}^i}, \Delta^{\mathcal{I}^\#}) = f_\rho(arg_1^{\mathcal{I}^i}, \dots, arg_n^{\mathcal{I}^i}, \Delta^{\mathcal{I}^i})$, which is equal to $\rho(arg_1, \dots, arg_n)^{\mathcal{I}^i}$. An almost identical argument holds for role constructors ρ .

Now let $\mathcal{I} = \langle \{\mathcal{I}^i\}, \{\mathbf{r}_{ij}\} \rangle$ be a d-interpretation that satisfies \mathfrak{T} . We first need to show that $\mathcal{I}^\# \models \#(\mathfrak{T})$. So consider the various items in the definition of $\#(\mathfrak{T})$. Since $\#(i, M)^{\mathcal{I}^\#} = M^{\mathcal{I}^i}$, then $\#(i, V)^{\mathcal{I}^\#} \subseteq \#(i, W)^{\mathcal{I}^\#}$ if and only if $V^{\mathcal{I}^i} \subseteq W^{\mathcal{I}^i}$, and hence $\mathcal{I}^\# \models \#(i, V) \sqsubseteq \#(i, W)$ iff $\mathcal{I}^i \models V \sqsubseteq W$. As a result, $\mathcal{I}^\#$ satisfies the subsumptions in item 1. $\mathcal{I}^\#$ satisfies items 2 and 3 because \mathcal{I} satisfies the bridge rules of \mathfrak{T} , and item 4 by the definition of $\mathcal{I}^\#$. Axioms in item 7 are satisfied because the \mathcal{I}^i are interpretations, and therefore their domains $\Delta^{\mathcal{I}^i}$ are nonempty, and in turn these are the interpretations of Top_i . The remaining items follow from the fact that each \mathcal{I}^i is a model of theory \mathbf{T}_i , with domain $\Delta^{\mathcal{I}^i}$. Therefore $\mathcal{I}^\#$ satisfies $\#(\mathfrak{T})$, and hence $\mathcal{I}^\# \models \#(i, X) \sqsubseteq \#(i, Y)$. By using

again the fact that $V^{\mathcal{I}^i} = \#(i, V)^{\mathcal{I}^\#}$, we therefore get $\mathcal{I}^i \models X \sqsubseteq Y$, and hence that $\mathcal{J} \models_d i: X \sqsubseteq i: Y$.

The “only if” part of the proof, starts from an arbitrary interpretation $\mathcal{I}^\#$ of $\#(\mathfrak{T})$. Let $\Delta^{\mathcal{I}^i}$ be $\#(i, \text{ANYTHING})^{\mathcal{I}^\#}$, and define the mappings $\cdot^{\mathcal{I}^i} : \mathcal{DL}_i \rightarrow \Delta^{\mathcal{I}^i}$ to be $M^{\mathcal{I}^i} = \#(i, M)^{\mathcal{I}^\#}$ for atomic concepts and roles M , as well as ANYTHING.

First, we need to show that $(\Delta_i, \cdot^{\mathcal{I}^i})$ is indeed an interpretation of \mathcal{DL}_i . This requires verifying that

- $\Delta^{\mathcal{I}^i}$ is a non-empty set; (* true, because otherwise axiom 7 of $\#(\mathfrak{T})$ cannot be satisfied by $\mathcal{I}^\#$ *)
- $\text{ANYTHING}^{\mathcal{I}^i} = \Delta^{\mathcal{I}^i}$ (* By the definition of $\Delta^{\mathcal{I}^i}$ *)
- $\text{NOTHING}^{\mathcal{I}^i} = \emptyset$ (* By axiom 5 in the definition of $\#(\mathfrak{T})$ *)
- $A^{\mathcal{I}^i} \subseteq \Delta^{\mathcal{I}^i}$ for atomic concepts A (* By axioms 6 *)
- $p^{\mathcal{I}^i} \subseteq \Delta^{\mathcal{I}^i} \times \Delta^{\mathcal{I}^i}$ for atomic roles p . (* By axioms 8 *)

Then one proves that $\#(i, X)^{\mathcal{I}^\#} = X^{\mathcal{I}^i}$ using the same argument as in the “if” case. And the rest of the proof follows the same way as in the previous part.

The following corollary uses the properties of the abstract constructor ρ to describe a collection of familiar concept and role constructors which allow the above translation of DDL reasoning to DL reasoning to go through.

Corollary 1. *The equivalence $\#(\mathfrak{T}) \models \#(i, X) \sqsubseteq \#(i, Y)$ if and only if $\mathfrak{T} \models_d i: X \sqsubseteq Y$ holds when \mathcal{DL}_i offers a subset of the following concept constructors: $\{C \sqcap D, C \sqcup D, \neg C, \forall p.C, \geq n p D, \leq n p D, \text{role1} = \text{role2}\}$ and role constructors $\{\text{role}^-, \sqcap, \sqcup, \circ\}$*

Proof. First, we provide for each constructor the definition of the corresponding semantic function.

Concept constructor	semantics function
$C_1 \sqcap C_2$	$f_{and}(XC1, XC2, DY) = X1 \cap X2$
$C_1 \sqcup C_2$	$f_{or}(XC1, XC2, DY) = X1 \cup X2$
$\neg C$	$f_{not}(XC, DY) = \{d \in DY \mid d \notin XC\}$
$\forall R.C$	$f_{forall}(XR, XC, DY) = \{d \in DY \mid XR(d) \subseteq XC\}$
$\exists R.C$	$f_{exists}(XR, XC, DY) = \{d \in DY \mid XR(d) \cap XC \neq \emptyset\}$
$\geq n R.C$	$f_{qatleast}(n, XR, XC, DY) = \{d \in DY \mid XR(d) \cap C \geq n\}$
$\leq n R.C$	$f_{qatmost}(n, XR, XC, DY) = \{d \in DY \mid XR(d) \cap C \leq n\}$
$R_1 = R_2$	$f_{sameas}(XR_1, XR_2, DY) = \{d \in DY \mid XR_1(d) = XR_2(d)\}$
role constructor	semantics function
R^-	$f_{inverse}(XR, DY) = \{(y, x) \mid (x, y) \in XR\}$
$R_1 \circ R_2$	$f_{compose}(XR_1, XR_2, DY) = \{(x, z) \mid y \in XR_1(XR_2(x))\}$
$R_1 \sqcap R_2$	$f_{role_and}(XR_1, XR_2, DY) = XR_1 \cap XR_2$
$R_1 \sqcup R_2$	$f_{role_or}(XR_1, XR_2, DY) = XR_1 \cup XR_2$

It is straightforward to demonstrate that each constructor is local, once one notes that DY is not used in the semantics of role constructors, and in the case of concept constructors, it appears only in the form $\{d \in DY \mid \dots\}$, which has the property that $W \cap \{d \in \delta \mid \dots\} = \{d \in W \mid \dots\}$.

We are now ready to use Theorem 1, to obtain some significant rewards:

Proposition 3. *A DDL such that all \mathcal{DL}_i are contained in some decidable description logic \mathcal{DL}_0 , with only primitive roles, which supports (i) qualified existential restriction, and (ii) arbitrary subsumption assertions in T -boxes, can use the decision procedure of \mathcal{DL}_0 to decide unsatisfiability and d -entailment.*

Proof. We know that reasoning in \mathfrak{T}_{12} is equivalent to ordinary DL reasoning in $\#(\mathfrak{T}_{12})$. The proof relies on two observations: (a) (by design) every axiom in $\#(\mathfrak{T}_{12})$ involving R_{ij} is either of form $\alpha \sqsubseteq \forall R_{ij}.\beta$ or $\exists R_{ij}.\delta \sqsubseteq \gamma$; (b) an axiom of the form $\alpha \sqsubseteq \forall p.\beta$ is equivalent to $\exists p^-. \alpha \sqsubseteq \beta$. This allows all axioms involving R_{ij} to be rewritten to involve only qualified existentials over R_{ij}^- , at which point we can replace R_{ij}^- by some new role S_{ij} . This removes the need for inverse roles and universal restrictions.

We get as corollaries that DDLs with \mathcal{DL}_i that are in \mathcal{ALCN} [8], \mathcal{DLR} [10], or \mathcal{SHIQ} [19] can use their reasoners for determining \models_d .

The following are some examples of concept and role constructors that do not satisfy the conditions required in Theorem 1

- Role complement, the identity role, or any role constructor whose semantic function cannot be written without using DY .
- A hypothetical concept constructor $\mathbf{each}(R)$, which denotes objects that are related by role R to every concept in the domain of interpretation. Such a constructor would have $f_{\mathbf{each}}(XR, DY) = \{d \in DY \mid XR(d) = DY\}$, and $W \cap f_{\mathbf{each}}(V, \Delta) \neq f_{\mathbf{each}}(V, W)$ because of the second DY in the body.

The above construction and theorem can be extended to deal with individuals (and hence A -boxes) by (a) adding to the language $\mathcal{GD}\mathcal{L}$ individual names $i:ind$, corresponding to individuals appearing in \mathcal{DL}_i ; (b) extending $\#$ so that $\#(i, ind) = i:ind$; and (c) adding the axioms $Top_i(i:ind)$, indicating that i -individuals are restricted to be interpreted in the sub-domain engendered by the representation of each local top concept. Moreover, the corollary can be shown to apply to constructors involving individuals such as \mathbf{fills} and $\mathbf{one_of}$:

Syntax	Semantic function
$R(ind_1, \dots, ind_n)$	$f_{fills}(XR, \{v_1, \dots, v_n\}, DY) = \{d \in DY \mid \{v_1, \dots, v_n\} \subseteq XR(d)\}$
$\{ind_1, \dots, ind_n\}$	$f_{oneof}(v_1, \dots, v_n) = \{v_1, \dots, v_n\}$

7 Atomic DDL

In many practical applications on the web, an ontology is simply a list of words organized in a generalization hierarchy. Such a scheme is visible in the topic areas of search sites such `dmoz`, `google` and `yahoo!`. Taxonomies of terms are also widely used in e-commerce sites, e.g. Eclass `www.eclass-standard.net/` and UNSPSC `www.unspsc.com`. Such “lightweight” ontologies correspond to DLs where there are no concept constructors, only atomic names. Hence there is no opportunity to define complex descriptions, and the T-box describes essentially a hierarchy of identifiers.

Although the computation of subsumption reduces to computing transitive closure for such T-boxes, there are several additional services provided by actual systems:

- They present visually a reduced partial order, where there are no redundant links: If $a \rightarrow b$ and $b \rightarrow c$ then there is no link $a \rightarrow c$.
- The answer to subsumption questions $X \sqsubseteq Y$ may take constant time for tree hierarchies, and potentially only a little more for those that are “almost” like trees, through preprocessing of the final hierarchy [1, 6].

There is an obvious need to integrate even such simple ontologies, if one wants to provide access to a variety of related resources. For example, [27] show how the integration of the ontologies of two software repositories, TuCows and Downloads.com, can provide seamless retrieval of software from multiple sources. However, such ontologies, especially in e-commerce, would be expected to change daily, and would definitely not want to be tied together in some “distributed” system. Therefore our approach of peer-to-peer coupling of IS seems reasonable in this case.

As in other work, for example [20], the process starts by expressing statements that link terms in the two ontologies, usually using additional subsumption relationships holding between terms in the two ontologies. One can then deduce the relationships of two terms by reasoning in the T-box which is the union of the two original ontologies, and these “cross-relating” subsumptions. The framework of DDLs provides a more refined tool, through the use of into- and onto-bridge rules. The question in this case is how one can reason effectively with the resulting DDL. The natural course is to use Theorem 1 to transform the problem into ordinary DL reasoning in the global DL. The most unsettling aspect of this approach is that even if the original \mathcal{DL}_i only had atomic concepts, the $\#(\mathfrak{T})$ theory uses qualified existential quantifiers (\exists), for which it is known that subsumption is computationally difficult in general. This is in contrast with the situation for atomic hierarchies, mentioned above.

There is however some hope for our reasoning in $\#(\mathfrak{T})$: if one looks carefully, the translation does not contain any *nested* existential restrictions. Exploiting this hole, we can then prove the following interesting result:

Theorem 2. *Given a DTB $\mathfrak{T}_{12} = \langle \mathbf{T}_1, \mathbf{T}_2, \mathfrak{B}_{12} \rangle$, where \mathbf{T}_1 , \mathbf{T}_2 , and \mathfrak{B}_{12} involve only atomic concepts. Then $\mathfrak{T} \models_d 2:G \sqsubseteq H$ if and only if*

$$\#(\mathfrak{T}) \vdash 2:G \sqsubseteq 2:H$$

where the inference rules are

$$\begin{array}{l}
\text{NOTHING} \sqsubseteq X \quad (\text{bottom}) \qquad X \sqsubseteq \text{ANYTHING} \quad (\text{top}) \\
\exists p. \text{NOTHING} \sqsubseteq \text{NOTHING} \quad (\text{some-bottom}) \qquad X \sqsubseteq X \quad (\text{reflex}) \\
\frac{X \sqsubseteq Y}{\exists p. X \sqsubseteq \exists p. Y} \quad (\text{some-isa}) \qquad \frac{X \sqsubseteq Y \quad Y \sqsubseteq Z}{X \sqsubseteq Z} \quad (\text{trans})
\end{array}$$

In general we continue to be interested in the conclusions $\mathfrak{T} \models 2 : E \sqsubseteq F$ derivable when IS_2 integrates into its terminology/ontology that of IS_1 , by the addition of into- and onto-bridge rules of the form $1 : A \xrightarrow{\sqsubseteq} 2 : H$ and $1 : B \xrightarrow{\sqsupseteq} 2 : G$ respectively.⁶ According to the above Theorem 2, proofs of $g(\mathfrak{T}) \vdash 2 : E \sqsubseteq 2 : F$ are essentially chains of subsumptions in \mathbf{T}_2 , interspersed with “switches up” to IS_1 via onto-rules, and then switches back down via into-rules as shown in Figure 7 (This seems obvious from the beginning, but it is surprisingly hard to

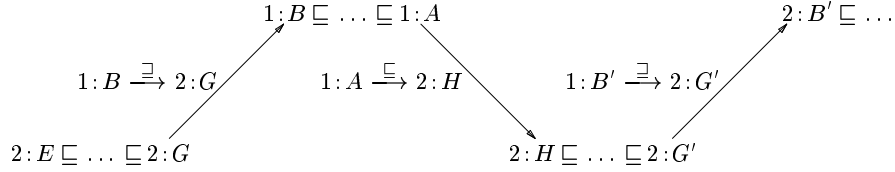


Fig. 7. Intuitive shape of subsumption proofs in DDL

prove formally that there are no other possible inferences – hence the importance of the Theorem.)

Recall that in assimilating information from other terminologies, we wanted to obtain reduced hierarchies, and preserve fast access by preprocessing of the final hierarchy. We could proceed by a single wholesale action, according to the previous theorem, or we could add subsumption and bridge statements incrementally. In either case, we seem to waste the work already done for processing IS_1 , which could be very large. The following shows how one can make a much smaller number of additions – bounded by the square of only the number of bridge rules.

Returning to our observation above about the form of a $\#(\mathfrak{T})$ proof, it seems like we can precompute the cases when “switches up/down are possible”. In particular, collect all onto bridge rules $B_i \xrightarrow{\sqsupseteq} G_i$ and into bridge rules $A_j \xrightarrow{\sqsubseteq} H_j$, and whenever $\mathbf{T}_1 \vdash B_i \sqsubseteq A_j$, incorporate the corresponding subsumption $G_i \sqsubseteq H_j$ in \mathbf{T}_2 . Assuming that \sqsubseteq computation is fast in \mathbf{T}_1 , finding such additions is a quick process, and as desired, it results in at most k^2 additional links where k is the number of bridge rules.

⁶ For convenience, we will drop the prefixes $1 :$ from A and B , as well as prefixes $2 :$ from H and G .

Note that if in one of the above cases, $\vdash H'_j \sqsubseteq G'_i$ also holds, then indeed H'_j and G'_i are equivalent as far as IS_2 sees it in the DDL; but, this “bow-tie”-like situation [33] does **not** imply in DDL that the corresponding A'_j and B'_i must also be identical, or that there is an inconsistency. This is the advantage of allowing for the non-identity directional mapping connecting the domains of the two ISs.

Note also that in general one can pre-process bridge rules to eliminate redundancies by observing that if $A \xrightarrow{\sqsubseteq} H_1$, and $A \xrightarrow{\sqsubseteq} H_2$, where $H_1 \sqsubseteq H_2$, then the later rule is redundant, and can be removed from the DDL. More generally, if $A_1 \xrightarrow{\sqsubseteq} H_1$ and $A_2 \xrightarrow{\sqsubseteq} H_2$, then we can remove the $A_2 \xrightarrow{\sqsubseteq} H_2$ rule if $A_2 \sqsubseteq A_1$ and $H_1 \sqsubseteq H_2$. Similarly, if one has onto-rules $B_1 \xrightarrow{\sqsupseteq} G_1$ and $B_2 \xrightarrow{\sqsupseteq} G_2$, then $B_1 \sqsubseteq B_2$ and $G_2 \sqsubseteq G_1$ implies that the second onto-rule can be removed.

8 Summary and Related Work

The seminal work of Catarci and Lenzerini [14] on integrating ISs described by DLs, made an implicit assumption that the local ISs have the same notion of what individual objects are, and that there was only one set of (subsumption) assertions relating IS_1 and IS_2 . We have argued that in cases such as loosely allied/peer-to-peer IS, when there is no single global view, these conditions need to be relaxed, by allowing general correspondence relationships between objects in the local domains, and by having “directed” import assertions.

Even more closely related to this paper, Calvanese et al [12] present an interesting and detailed mechanism for specifying data integration in the context of data warehouses. Although the conceptual data model is specified in a DL, which in some sense provides global terminology, the local ISs have relation-like schemata, which are connected via Datalog-like rules augmented by important information concerning keys and domains. The aspect that distinguishes their work, and corresponds more closely to ours, is the specification of “*reconciliation correspondences*”, including ones for conversion, matching and merging of specific data. These can be thought of as describing different aspects of instance-level connections, similar to our domain relationships. Their work is characterized by the use of rules, with a declarative part and a procedural part, performing operations such as conversions. While this is very useful for practical applications (e.g., converting currencies), it is not tied back to the conceptual schema, and how it might implicitly affect relationships there.

In contrast to the above, and approaches reviewed earlier, our work on DDL extends the *reasoning* available on ordinary schemas to the case of multiple schemas aligned by arbitrary binary correspondences between individuals. Such reasoning can be used for a wide variety of tasks, including query (re)formulation, and dealing with partially-specified individuals, as surveyed in [7]. Moreover, as we emphasized in the beginning, our approach lends itself to a situation where the multiple ontologies/IS are not being merged/integrate, but rather information from one source is being assimilated by another. Stuckenschmidt [34] provides

an additional nice motivating example of this, in the use of local ontologies for agents, and at the same time provides some testimony to the potential appeal of our approach by actually using Distributed Description Logics, as these were presented in an earlier workshop.

This paper has also identified other desirable properties of aligned DL, such as “no backflow”, and localizing the effect of inconsistencies so that one IS does not “infect” the reasoning of the entire system.

Among the interesting results obtained are a translation of DDL reasoning to DL reasoning. The representation of the target DL knowledge base requires only the presence of the qualified existential restriction DL-constructor $\exists p.C$ and the ability to reason with (acyclic) background axioms, over and above the needs of the local IS. This allows both complexity results and reasoning algorithms to be transferred for DDL whose local theories come from decidable logics such as *SHIQ* and *DLR*.

Finally, we have taken a closer look at the case of DDL without concept constructors, i.e., ones in which there are only atomic concepts organized in a subsumption hierarchy. These are of interest since they correspond to the majority of the ontologies currently used, especially on the web. In this case, we showed that there is indeed no “backflow”, and the reasoning needed to compute d-entailment is like that needed to compute regular entailment, with the addition of a relatively small number of pre-computed links to the hierarchy of the ontology that “imports” information.

Among the many open questions are ones concerning the behaviour of DDL with more than two local T-boxes, both in the presence and absence of cycles in the domain relations. And the investigation of a greater variety of bridge rules, especially ones that are of both practical use and can preserve the very useful DDL-to-DL translation theorem.

References

1. Rakesh Agrawal, Alexander Borgida, H. V. Jagadish. Efficient Management of Transitive Relationships in Large Data and Knowledge Bases. *SIGMOD Conference 1989*: 253-262
2. Y. Arens, C.A. Knoblock, W.M. Shen. Query Reformulation for Dynamic Information Integration. *J. Intelligent Information Systems 6(2/3)*: 99-130 (1996)
3. Tim Berners-Lee. What the Semantic Web can represent. (An parenthetical discussion to the Web Architecture at 50,000 feet. and the Semantic Web roadmap.) September 1998. <http://www.w3.org/DesignIssues/RDFnot.html>
4. *The Description Logic Handbook* (Eds. F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider), Cambridge University Press, January 2003
5. S. Bergamaschi, S. Castano, M. Vincini, D. Beneventano. Semantic integration of heterogeneous information sources. *Data and Knowledge Engineering 36(3)*: 215-249 (2001).
6. Martin F. van Bommel, T. J. Beck: Incremental Encoding of Multiple Inheritance Hierarchies. *CIKM 1999*: 507-513
7. A. Borgida. Description logics in data management. *IEEE Trans. on Knowledge and Data Engineering*, 7(5):671-682, 1995.

8. Buchheit, M., Donini, F.M. and Schaerf, A. Decidable Reasoning in Terminological Knowledge Representation Systems. *J. Artificial Intelligence Research*, Volume 1, pages 109-138.
9. D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.
10. Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2-13, 1998.
11. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Information integration: Conceptual modeling and reasoning support. *Proc. CoopIS'98*, pp.280–291, 1998.
12. Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Data integration in data warehousing. *Int. J. of Cooperative Information Systems*, 10(3):237-271, 2001
13. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. A principled approach to data integration and reconciliation in data warehousing. In *Proc. Intl. Workshop on Design and Management of Data Warehouses (DMDW'99)*, Heidelberg, Germany, 1999.
14. T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *International Journal of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.
15. D. Florescu, A. Levy, and A. Mendelzon. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Record*, 27(3):59-74, Sept. 1998.
16. C. Ghidini and L. Serafini. Distributed First Order Logics. In D. Gabbay and M. de Rijke, editors, *Frontiers Of Combining Systems 2*, Studies in Logic and Computation, pages 121–140. Research Studies Press, 1998.
17. I. Horrocks, P. Patel Schneider, and F. van Harmelen. Reviewing the design of DAML+OIL: language for the semantic web. In *Proc. AAAI'02*, 2002.
18. Deborah L. McGuinness, Richard Fikes, James Hendler and Lynn Andrea Stein. DAML+OIL: An Ontology Language for the Semantic Web. *IEEE Intelligent Systems*, Vol. 17, No. 5, pages 72-80, September/October 2002.
19. I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. Logic and Computation*, 9(3):385–410, 1999.
20. V. Kashyap and A. Sheth. Semantic and Schematic Similarities Between Database Objects: A Context-Based Approach. *VLDB Journal* 5(4): 276-304, 1996.
21. W. Kent. Solving Domain Mismatch and Schema Mismatch Problems with an Object-Oriented Database Programming Language. *Proc. VLDB'91*, pp. 147-160, 1991.
22. M. Klein. Combining and relating ontologies: an analysis of problems and solutions. *Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing* Seattle, USA, August 4-5, 2001.
23. K. Knight and S.K. Luck. Building a Large-scale Knowledge Base for Machine Translation. *Proc. AAAI'94*
24. L. Lakshmanan, F. Sadri, and I.N. Subramanian. Logic and Algebraic Languages for Interoperability in Multi-database Systems. *Journal of Logic Programming* 33(2):101–149, Nov. 1997.
25. A.Y. Levy, A. Rajaraman , J.J. Ordille. Query answering algorithms for information agents *Proc. AAAI'96*

26. Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder. "An Environment for Merging and Testing Large Ontologies" *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, Breckenridge, Colorado, USA 12-15 April 2000
27. Eduardo Mena, Arantza Illarramendi, Alfredo Goi, "Automatic Ontology Construction for a Multiagent-Based Software Gathering Service", Proc. Workshop on Cooperative Information Agents IV, Springer LNCS 1860, pp. 232-243
28. E. Mena, V. Kashyap, A. Illarramendi and A. Sheth. Imprecise Answers on Highly Open and Distributed Environments: An Approach based on Information Loss for Multi-Ontology Based Query Processing. *Int. Journal of Cooperative Information Systems* 9(4):403-425, December 2000.
29. A. Pirotte, E. Zimanyi, D. Massart, and T. Yakusheva. Materialization: A Powerful and Ubiquitous Abstraction Pattern. In *Proc. VLDB'94*, pp. 630-641, 1994.
30. E. Rahm, and P.A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal* 10(4):334-350. Dec. 2001.
31. F. Salto and E. Rodríguez. On Intelligent Access to Heterogeneous Information. In F. Baader, M.A. Jeusfeld & W. Nutt (eds), *Proc. KRDB'97*, Athens, Greece, August 30, 1997, pp. 15:1-15:7
32. E. Sciore, M. Siegel, and A. Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM TODS* 19(2):254-290, June 1994.
33. J. Sowa. Building, Sharing, and Merging Ontologies. <http://www.jfsowa.com/ontology/ontoshar.htm>
34. H. Stuckenschmidt. Exploiting Partially Shared Ontologies for Multi Agent Communication. *Cooperative Information Agents VI, (CIA 2002)*, Springer LNCS 2446.

A Proof of theorems

A.1 Reasoning in Description Logics

Standard reasoning questions involving descriptions include checking subsumption and satisfiability. Most reasoning algorithms for DLs fall into one of two categories:

1. Normalize-and-compare: concepts are put into some normal form which reduces redundancy and makes explicit information that is implicit; subsumption is then checked by comparing normalized concepts. The later tends to be an efficient operation, so such implementations work for languages whose expressive power is limited (and hence computational complexity is low), or in cases when the reasoner is incomplete. In this case unsatisfiability of a concept is detected when it is normalized to the empty/inconsistent concept.
2. Tableaux: the satisfiability of a concept is tested by trying to construct a model for it—an interpretation in which the denotation of the concept is not empty. This approach has been applied to more expressive DLs, and one of its chief advantages is that the reasoner can be proven to be *complete*. In tableaux provers, determining whether a subsumption $C \sqsubseteq D$ holds, is reduced to testing the concept $C \sqcap \neg D$ for *unsatisfiability*.

In this paper we will be interested in tableau reasoning, and particularly the details for \mathcal{ALC} , because it supports complete reasoning with theories [8]. This is useful for demonstrating indirectly certain properties of more restricted DLs, for which standard completeness proof techniques would not apply because, for example, they lack negation. A typical DL tableau algorithm starts by taking the description M to be tested for satisfiability, and starts a partial model by asserting M of some individual named x_0 (written $M[x_0]$ here). The algorithm then repeatedly takes some candidate partial model (a.k.a. "constraint system"), and tries to grow it into a more complete one by repeatedly using *expansion rules*⁷. An example of an expansion rule, is adding $A[x]$ and $B[x]$ to the partial model/set containing the assertion $A \sqcap B[x]$. If *all* attempts to construct a model lead to a contradiction (e.g., the co-occurrence of inconsistent descriptions, $B[x]$ and $\neg B[x]$), then the original concept is unsatisfiable.

Some of the expansion rules are "nondeterministic", in the sense that they suggest alternative successor partial models. For example, the expansion rule for concept disjunction adds *either* $A[x]$ or $B[x]$, when $A \sqcup B[x]$ is present. Since, as we said, we want to examine all attempts to find models, we will keep track of all the expansion rule applications in a *proof tree*, whose nodes n , are associated with partial models $\mathcal{P}(n)$, and whose edges are implicitly labeled by the expansion rule application producing it.

Each partial model will be a collection of assertions of the form $\alpha[x]$ or $p(w, z)$, where x, w, z are called individuals, and α is an arbitrary description, while p is a role identifier. The application of an expansion rule will introduce one or two nodes n , whose $\mathcal{P}(n)$ will contain a new individual and/or an assertion about it.

Notation: In our diagrams, we will sometime write beside node n part of the $\mathcal{P}(n)$ set (e.g., $\{A \sqcap B[x], D[y], \dots\}$); and if n_2 is a child node of n_1 , then we will only indicate with $+\alpha$, that $\mathcal{P}(n_2) = \mathcal{P}(n_1) \cup \{\alpha\}$. The expansion rules for the \mathcal{ALC} language are given in Figure 8, where we will leave unexplained technical terms concerning "blocking" since they will not play a role in our analyses. In the rules we suppose n_1 to be some leaf node in a proof tree, and w some individual in its partial model $\mathcal{P}(n_1)$. The node n_1 will be said to have a *clash*, if for some individual w either $\perp[w] \in \mathcal{P}(n_1)$ or $\{A[w], \neg A[w]\} \subseteq \mathcal{P}(n_1)$ for some atomic A .

A node will be called *closed* if it has a *clash* on it, and a proof tree will be called closed if all its leaf nodes have clashes. Such a proof tree represents a refutation, and it indicates that the formulas at its root form an unsatisfiable set. Since subsumption is reduced to testing for unsatisfiability, we will be interested in closed proof trees. Figures 9 shows some examples of proof trees.

We will be using the above expansion rules to check the well-formedness of a proof tree. Since the secondary conditions (in parentheses) for applying each of the above rules are present only to make the algorithm terminate, it is therefore acceptable to have a proof tree that has some superfluous branches resulting from applying these rules with only the first condition satisfied.

⁷ Our terminology and ' notation will be somewhat non-standard, in order to facilitate presentation of proofs and to reduce confusion with other notation in this paper. For example, $M[x]$ is normally written $x:M$, but we are already using $:$ to add labels.

and-rule	<p>if $M \sqcap N[w] \in \mathcal{P}(n_1)$ (and w is not blocked, and $\{M[w], N[w]\} \not\subseteq \mathcal{P}(n_1)$) then add to the proof tree a unique child node n_2 for n_1, such that $\mathcal{P}(n_2) = \mathcal{P}(n_1) \cup \{M[w], N[w]\}$</p>
or-rule	<p>if $M \sqcup N[w] \in \mathcal{P}(n_1)$ and (and w is not blocked, and $\{M[w], N[w]\} \cap \mathcal{P}(n_1) = \emptyset$) then add to the proof tree two child nodes n_2 and n_3 for n_1 such that $\mathcal{P}(n_2) = \mathcal{P}(n_1) \cup \{M[w]\}$ and $\mathcal{P}(n_3) = \mathcal{P}(n_1) \cup \{N[w]\}$</p>
some-rule	<p>if $\exists p.C[w] \in \mathcal{P}(n_1)$, (and w is not blocked, and there is no z such that $p(w, z) \in \mathcal{P}(n_1)$ and $C[z] \in \mathcal{P}(n_1)$) then add to the proof tree a unique child node n_2 for n_1, such that $\mathcal{P}(n_2) = \mathcal{P}(n_1) \cup \{p(w, y), C[y]\}$, where y is a new individual, not appearing anywhere else in the proof tree. Such a node n_2 will be called a <i>y-introduction</i></p>
all-rule	<p>if $\forall p.C[w] \in \mathcal{P}(n_1)$ (and w is not blocked, and there is a y such that $p(y, w) \in \mathcal{P}(n_1)$ but $C[y] \notin \mathcal{P}(n_1)$) then add to the proof tree a unique child node n_2 for n_1, such that $\mathcal{P}(n_2) = \mathcal{P}(n_1) \cup \{C[y]\}$</p>
theory	<p>if there is a set T of descriptions that apply to any individual (because they encode a T-box theory using $\neg M \sqsubseteq N$ for $M \sqsubseteq N$) then every time a new individual x is added, add all assertions $\neg M[x] \sqcup N[x]$ if $M \sqsubseteq N \in T$, as well as $N[x]$ for any $N \in T$, whose presence is due to axioms of the form $\top \sqsubseteq N$. (This will be denoted $T[[x]]$, and we will usually leave it implicit.)</p>

Fig. 8. Tableaux rules for \mathcal{ALC}

The goal of this section is to prove the following theorem

Theorem 3. *Given a DTB $\mathfrak{T}_{12} = \langle \mathbf{T}_1, \mathbf{T}_2, \mathfrak{B}_{12} \rangle$, where \mathbf{T}_1 , \mathbf{T}_2 , and \mathfrak{B}_{12} involve only atomic concepts. Then $\mathfrak{T} \models_d 2 : G \sqsubseteq H$ if and only if*

$$\#(\mathfrak{T}) \vdash 2 : G \sqsubseteq 2 : H$$

where the inference rules are

$$\perp \sqsubseteq X \quad (\text{bottom}) \qquad X \sqsubseteq \top \quad (\text{top})$$

$$\exists p.\perp \sqsubseteq \perp \quad (\text{some-bottom}) \qquad X \sqsubseteq X \quad (\text{reflex})$$

$$\frac{X \sqsubseteq Y}{\exists p.X \sqsubseteq \exists p.Y} \quad (\text{some-isa}) \qquad \frac{X \sqsubseteq Y \quad Y \sqsubseteq Z}{X \sqsubseteq Z} \quad (\text{trans})$$

Our approach will be the following: Given a DDL \mathfrak{T} , we know by Theorem 1 that $\mathfrak{T} \models_d i : M \sqsubseteq N$ if and only if $\#(\mathfrak{T}) \models \#(i, M) \sqsubseteq \#(i, N)$. So we will be working with the DL theory $\#(\mathfrak{T})$.

We will then prove that the inference rules above are sufficient to explain all refutations produced by the expansion rules of \mathcal{ALC} , which are known to be complete.

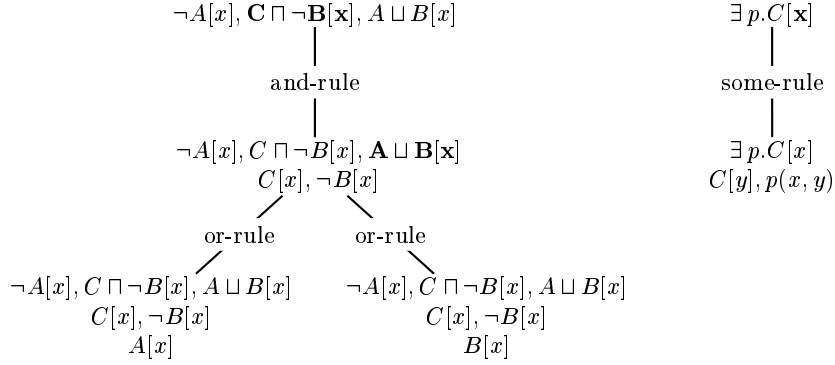


Fig. 9. Examples of proof trees in DL

Notational abbreviations and conventions:

- capital letters A, B, C, D (possibly decorated) will be used for 1-concepts $1:A, 1:B, 1:C, B, 1:D$ and will stand for either atomic identifiers or \top , or \perp , though we will write $1:\top$ as Top1 to emphasize that it is just an ordinary concept name in the proof, not \top .
- similarly, capital letters E, F, G, H will be used to stand for $2:E, 2:F, \dots$. Similarly, $1:\top$ will be written as Top2 .
- letters L, M, N, P will be used to stand for any concept of the form $Y, \neg Y, \exists p.Y$, or $\forall p.\neg Y$, where Y is an atom, Top1 , Top2 , \top or \perp .
- α will be used for a general description.
- $\exists A$ will be an abbreviation for $\exists S.A$, where, recall, $S = \text{r}_{\perp 2}$, because S is the main role symbol in the theory $\#(\mathfrak{T})$, and when there are no role symbols in \mathfrak{T} itself. (We will continue to write $\exists \hat{P}.\text{Top1}$);
- similarly, $\forall \neg G$ will be an abbreviation for $\forall S.\neg B$.

In the case when \mathcal{DL}_1 and \mathcal{DL}_2 have no concept constructors—only atomic identifiers or \perp or \top , by the definition of $\#()$, the theory $T^* = \#(\mathfrak{T})$ contains the following kinds of axioms (alongside their form as descriptions that apply to

all individuals, and their abbreviated form⁸):

Subsumption axiom	Term axiom	Term abbreviation
$1 : A \sqsubseteq 1 : B$	$\neg 1 : A \sqcup 1 : B$	$\neg A \sqcup B$
$2 : G \sqsubseteq 2 : H$	$\neg 1 : G \sqcup 1 : H$	$\neg G \sqcup H$
$1 : A \sqsubseteq \exists S.2 : H$	$\neg 1 : A \sqcup \exists S.2 : H$	$\neg A \sqcup \exists H$
$\exists S.2 : G \sqsubseteq 1 : B$	$\forall S.\neg 2 : G \sqcup 1 : B$	$B \sqcup \forall \neg G$
$\exists S.\top \sqsubseteq \text{Top}2$	$1 : \forall S \perp \sqcup \text{Top}2$	$\forall \perp \sqcup \text{Top}2$
$\exists S.\neg \text{Top}1 \sqsubseteq \perp$	$\forall S.\neg \neg \text{Top}1 \sqcup \perp$	$\forall.\text{Top}1$
$1 : C \sqsubseteq \text{Top}1$	$\neg 1 : C \sqcup \text{Top}1$	$\neg C \sqcup \text{Top}1$
$2 : E \sqsubseteq \text{Top}2$	$\neg 1 : E \sqcup \text{Top}2$	$\neg E \sqcup \text{Top}2$
$\top \sqsubseteq \exists \hat{P}.\text{Top}1$	$\neg \top \sqcup \exists \hat{P}.\text{Top}1$	$\exists \hat{P}.\text{Top}1$
$\top \sqsubseteq \exists \hat{P}.\text{Top}2$	$\neg \top \sqcup \exists \hat{P}.\text{Top}2$	$\exists \hat{P}.\text{Top}2$

The proof will require some terminology and several definitions.

Definition 9. An individual z_k is in the family of another individual z_0 if there is a chain of individuals z_i and roles p_i , $0 \leq i \leq k$, such that $p_i(z_{i-1}, z_i) \in \mathcal{P}(z_k)$.

The “family” of individual w , $\text{family}(w)$, is the set of all individuals who are in the family of w , together with w .

Basically, the family of individual w is the set of individuals whose introduction in the proof tree (via some-rules) depends on the existence of w . This definition is only for stating things more succinctly. The following are crucial properties of proof trees for our proofs:

Definition 10. Let n_1 be the root node of a proof tree tr_1 . An element $M[w]$ of $\mathcal{P}(n_1)$ will be said to be essential if tr_1 is closed and one of the following conditions holds:

- $M = \perp$
- $M = C, E, \neg D$, or $\neg F$ and M participates in a clash with its complement in tr_1 (the complement of X is $\neg X$, and of $\neg Y$ is Y).
- $M = \exists p.N$ for some role p , and a some-rule on M is used in tr_1 to add a new node n_2 under n_1 , where there is a new individual y such that $p(w, y), N[y], \exists p.N[w]$ are in $\mathcal{P}(n_2)$, and the tree rooted at n_2 is closed. (N itself might not participate in a clash, but the presence of y may permit a clash to manifest itself.)
- $M = \forall N$, and there is a node n_2 such that $\{S(w, y), \forall M\} \subset \mathcal{P}(n_2)$, and an all-rule was used to add essential N at n_2 from M .

Note that this definition is not circular since the atomic cases are not recursive, and the last two cases either reduce to the first, or require checking something else (closedness).

⁸ These have also been “normalized” by replacing $\neg \top$ with \perp , $\neg \neg M$ with M , and $\perp \sqcup M$ with M

Observe: If tr is a closed proof tree rooted at n , and $\alpha \in \mathcal{P}(n)$ is not essential, then the proof tree obtained by removing α from $\mathcal{P}(n_1)$ for all nodes n_1 in tree tr is also a valid proof tree, which is also closed.

We will need to deal with quasi-minimal proof trees at times, and for this purpose we introduce the notion of “pruning” which cuts out obviously unnecessary parts of the proof tree either at the leaf or by merging 2 nodes, and hence eliminating at least the path between them:

Definition 11. Let n_1 be a node in a proof tree.

1. If n_1 is closed, then the application of any expansion rule to it is pruned by eliminating the subtree below that node.
2. Suppose there is a node n_1 with child n_2 , where a non-essential N is being added to n_2 . Then rule application is omitted and n_1 is merged with n_2 .
3. Suppose there is a node n_1 with child n_2 , where a redundant N is being added to $\mathcal{P}(n_2)$. Then rule application is omitted, and n_1 is merged with n_2 .
4. Suppose n_1 has descendant n_3 with individuals $x \in \mathcal{P}(n_1)$ and $y \in \mathcal{P}(n_3)$, such that $x \neq y$, $y \in \text{family}(x)$ and substituting x for y in the closed proof tree tr_3 at n_3 yields another closed tree tr'_3 . Then merge n_1 with n_3 , using tr'_3 instead of tr_3 .

Observe: The pruning of a closed proof tree results in a closed proof tree.

At a high level, the demonstration of the theorem consists of taking the closed proof tree for the original subsumption $E \sqsubseteq F$ rooted at n_0 , “manipulating” into a semi-normal form, and then repeatedly reorganizing it to uncover portions of it that can be translated to derivations in the proposed proof theory. In this demonstration, we will use a series of Lemmas and Propositions, whose statement and proof appear later.

By inductively applying Lemma 2, a closed proof tree can be reorganized into a collection of small trees hanging from each other, where each little tree tr_j , $j > 0$, starts with a z_j -introduction, and the rest of the tr_j involves only expansions adding assertions of the form $Q[z_j]$.

Furthermore, by Lemma 1 in any subtree tr_j , the rules adding terms of the form $\forall \neg B[z_j]$ can be postponed till just before the introduction of the z_k in the next lower subtree tr_k .

Such closed and pruned proof trees will be said to be in *semi-normal* form.

Propositions 7,8,9 show that in the original proof tree, starting with $\{E[x_0], \neg F[x_0]\}$, either

1. E is essential; in which case by Proposition 7, either $\vdash E \sqsubseteq F$; or $\vdash E \sqsubseteq \perp$, and hence from inference rule (bottom) ($\vdash \perp \sqsubseteq F$) we get again $\vdash E \sqsubseteq F$.
2. Or E is not essential, in which case by proposition 8, $\neg F$ is also not essential. Hence by Proposition 9, $\vdash \top \sqsubseteq \perp$, which together with inference rules (top) ($\vdash E \sqsubseteq \top$), and (bottom) yield $\vdash E \sqsubseteq F$ again.

This proves the original theorem.

Lemmas, Propositions, and their Proofs. In several of the results below we will talk about “reorganizing” a closed proof-tree by moving around certain subtrees. The key to the permissibility of such reorganization is that for every subtree tr_j , $\mathcal{P}(n_j)$ is contained in $\mathcal{P}(n'_j)$, where n_j is the old root node, and n'_j is the new root node.

Lemma 1. *In a closed proof tree, it is possible to exchange the application of rule R1 creating the child node n_2 of n_1 by adding $M[w]$, with the rule R2 creating the child node n_3 of n_2 by adding $Q[z]$ (see Figures 10–12), resulting in a closed proof tree, unless one of the following conditions holds*

- (i) $Q[z]$ is the result of expanding $M[z]$, which was added by rule R1;
- (ii) $Q[z]$ is the result of expanding $\exists p.Q[x]$, which was added by rule R1.
- (iii) $Q[z]$ is the result of expanding $\forall p.Q[x]$, where z was introduced by rule R1.

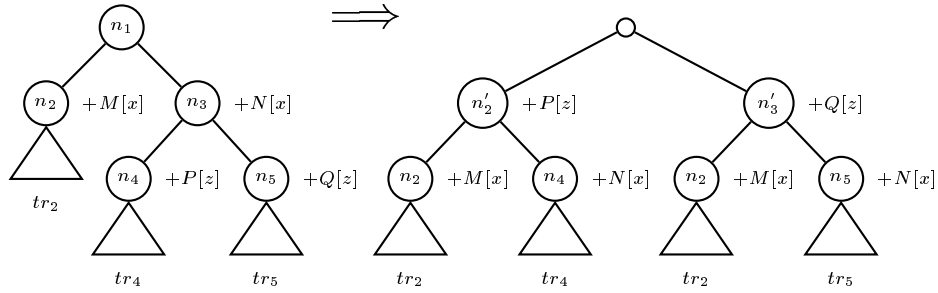


Fig. 10. Swapping expansion rules

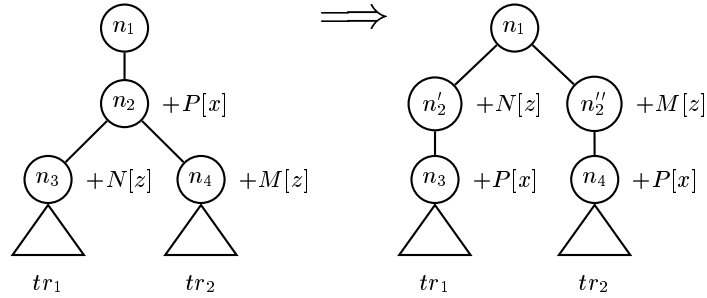


Fig. 11. Swapping expansion rules

The proof consists of checking in each case that the reorganization shown in the diagram is permissible.

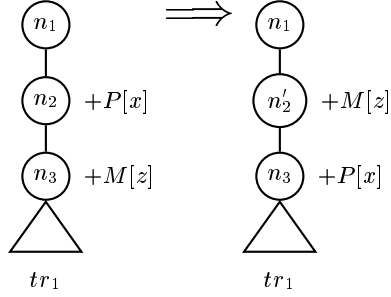


Fig. 12. Swapping expansion rules

Lemma 2. *A proof tree tr_1 with a y -introduction node inside it, can be reorganized so that the subtree tr_2 hanging from the y -introduction contains only expansions adding (expressions of the form) $Q[y']$, where $z \in \text{family}(y)$.*

Proof. First, if an $M \sqcup N[z]$ was expanded at n , then its movement up one node is blocked only by the possible introduction of z . So, if $z \notin \text{family}(y)$, the expansion can be repeatedly moved up above the introduction of y .

Then, if an $\exists M[z]$ was expanded at n to add $M[w]$, the upward movement of the expansion would only be blocked by the introduction of $\exists M[z]$. But such things are either in the original theory (e.g., $\exists \hat{P}\text{Top1}$) or the result of expanding $N \sqcup \exists M[z]$, which had all been moved up above the y -introduction already.

Finally, the movement up of an expansion of $\forall \neg B[z]$ into $\neg B[w]$ is only blocked by the addition of $\forall \neg B[z]$ or the addition of w . Since $w \notin \text{family}(y)$, only the first obstacle remains, and all the requisite introductions appear above the y -introduction already.

Lemma 3. *A seminormal pruned proof tree of size greater than 1, with an essential atom $C[x]$ at the root, n_1 , can be rewritten to have C encounter its clashing $\neg C$ immediately underneath, in the expansion of either $\neg C \sqcup M[x]$, or of $\forall p.\neg C[w]$, for some individual w such that $p(w, x) \in \mathcal{P}(n_1)$. And this can be done without increasing the size of the proof tree, nor affecting its closed, seminormal nature. Moreover M will be essential in the resulting tree, even after pruning.*

Similarly, for $\neg C$, E , and $\neg E$ replacing A above.

Proof. Since the pruned tree has size greater than 1, C cannot be \perp , nor can $\mathcal{P}(n_1)$ already contain $\neg C$.

Therefore there must be some node n_2 in the tree rooted at n_1 at which the clashing $\neg C$ is introduced, and it can only come from an assertion of the form $\neg C \sqcup M$ or $\forall \neg C$. (There are no (sub)terms of the form $\forall \hat{P}.X$ in T^* , as \hat{P} can be ignored.)

In the first case, the proof tree can be reorganized so that the clash occurs immediately below on some child, without increasing the number of nodes. Moreover, because the proof tree is seminormal, M must be essential at n_2 . M is still

essential, even if we prune the tree again, because the only way the original clash could disappear is if the path it was on was pruned. But the only difference in tr_2 is that every node now has M on it, so the only reason the path could be disturbed is because there is $\neg M$ or redundant essential M on the path to tree tr_4 . Either way, a clashing $\neg M$ is left in the resulting pruned tree.

In the second case, there must have been an essential $\forall\neg C[w]$ for some individual $w \notin \text{family}(x)$; but by semi-normal form, all introductions of these appear before the introduction of x , and hence $\forall\neg C[w] \in \mathcal{P}(n_1)$, allowing the proof tree to be begin with the expansion of $\forall\neg C[x]$, after which the subtree is already closed.

The following 3 propositions establish the form of proof subtrees resulting from y -introductions from $\exists S.N$.

Proposition 4. *Let tr_1 be a seminormal closed tree rooted at n_1 , possibly having $S(x, y) \in \mathcal{P}(n_1)$. Then all essential $C[y]$ at n_1 have the property that either*

- $\vdash C \sqsubseteq \perp$, and there are no essential $\neg D[y]$ or $\forall\neg D[x]$ at n_1
- or, there is an essential $\neg C^*[y]$ or $\forall\neg C^*[x]$ in $\mathcal{P}(n_1)$ such that $\vdash C \sqsubseteq C^*$, and there are no other essential $\neg C'[x]$ or $\forall\neg C'[x]$ in $\mathcal{P}(n_1)$.

Proof. The proof is by induction on the number of nodes in the pruned proof tree. (For convenience, we use α instead of $\alpha[y]$ when this does not lead to confusion.)

Let C be essential on n_1 . If the tree size is 1, then there must be a clash on n_1 , and C must participate in it, since C is essential. Hence C is either \perp , or $\neg C$ must be in $\mathcal{P}(n_1)$, and essential. In the first case, $C = \perp$, and $\vdash \perp \sqsubseteq \perp$ by inference rule (reflex). In the second case, let C^* be C ; then we have $\vdash C \sqsubseteq C^*$ by rule (reflex) again. In either case, since the tree was pruned, there is no further expansion at a node once the first clash occurs. Hence nothing else than C , and possibly $\neg C$, that can be essential at n_1 .

Suppose now the tree size is greater than 1. Since C is essential, it must clash with $\neg C$ somewhere in the tree. Since the tree is pruned and not of size one, by Lemma 3, we can reorganize the tree to introduce $\neg C$ right under n_1 , without increasing its size. The resulting tree under n_1 can have one of two forms.

1. $\{C, \forall\neg C\} \subset \mathcal{P}(n_1)$ and n_2 is n_1 's only child, with $\neg C$ added, due to the expansion of the \forall . In this case we have $\vdash C \sqsubseteq C$ by reflexivity; plus $\forall\neg C[x]$ is the only other essential description in $\mathcal{P}(n_1)$, because nothing else is needed to close the child node.
2. Alternatively, C clashes with $\neg C$ from $(\neg C \sqcup M)[y]$ so that n_1 has two children, one with $\neg C$ on it, which is therefore closed, and another one, n_3 , with M on it. And this is due to the axiom $C \sqsubseteq M$.

The only axioms of the form $\neg C \sqcup M$ have $M = D$ (recall that letters A through D stand for concepts of the form $1:\beta$). Since the tree is pruned, D must itself be essential. Therefore, by induction at n_3 , we have two cases.

One is that $\vdash D \sqsubseteq \perp$, which together with the $C \sqsubseteq D$ used in creating the branch under n_1 , leads to $\vdash C \sqsubseteq \perp$ by transitivity. Moreover, by induction, there are no essential $\neg D[y]$ nor essential $\forall \neg D[x]$ at n_3 . Hence there are no such things at n_1 , because they could only cause clashes below n_3 .

Alternatively, there is $\neg D''$ such that $\neg D'' \in \mathcal{P}(n_3)$ or $\forall \neg D''[x] \in \mathcal{P}(n_3)$ such that $\vdash D \sqsubseteq D''$, and hence $\vdash C \sqsubseteq D''$ by transitivity. Moreover, by induction D'' is the only thing in $\mathcal{P}(n_3)$ with the property that $\neg D''$ or $\forall \neg D''[x]$ are essential. Since $\mathcal{P}(n_3) = \mathcal{P}(n_1) \cup \{D[y]\}$, so that no assertions of the form $\neg B[y]$ or $\forall \neg B[x]$ are added, then the above statements about D'' hold for n_3 replaced by n_1 , concluding the proof.

Proposition 5. *Let n_1 be a node at the root of a seminormal tree tr_1 such that there are no essential (positive atoms) $C[y] \in \mathcal{P}(n_1)$, but there may be $S(x, y) \in \mathcal{P}(nd1)$. Then there can be no essential $\neg D[y]$ on any node in tr_1 , unless $\forall \text{Top1}[x]$ is expanded to add $\text{Top1}[y]$ somewhere.*

Proof. By induction of the size of the proof tree.

Suppose there is a pruned closed tree with a single node, rooted at n_1 , such that $\neg D[y]$ is essential on it. Then $\neg D$ must clash with something positive on n_1 , since $\neg D$ is not \perp (recall $\neg \top$ were normalized to \perp , and $\perp \sqcup M$ to M , in constructing the theory T^*). Since there are no essential positive atoms of the form D on n_1 , this is not possible, contradicting the assumption that there is a closed tree of size 1.

So assume that there are no seminormal trees of size k or smaller of the kind described, but suppose there is a pruned tree of size $k+1$, where $\neg D$ is essential. Therefore $\neg D$ must clash with D somewhere below it. This D could only have come from the expansion of two kinds of sources:

- a $\forall D[x]$; but there is only one such axiom: $\forall \text{Top1}[x]$, which is claimed not have been applied;
- a $\neg C \sqcup D$, where both $\neg C$ and D are essential, since the proof tree is pruned. As in the proof of Proposition 1.1, by Lemma 3, we can reorganize the proof to yield a tree with two children at the top: one with D added, which is immediately closed, and the other, n_3 , with $\neg C$ added, but where there is a closed subtree at n_3 of size less than $k+1$. But this means that at n_3 we have a closed tree with an essential $\neg C$, and no essential positive atom (there were none at n_1 , and no positive atoms were added in the step to n_3). This contradicts the inductive assumption, so there is no such tree of size $k+1$ at n_1 .

Finally, we deal with the case when there are no essential positive or negative atoms at n_1 .

Proposition 6. *There can be no seminormal proof tree of the form Figures 13, where $A[y]$ is not essential, and $\text{Top1}[y]$ is not essential at n_3 .*

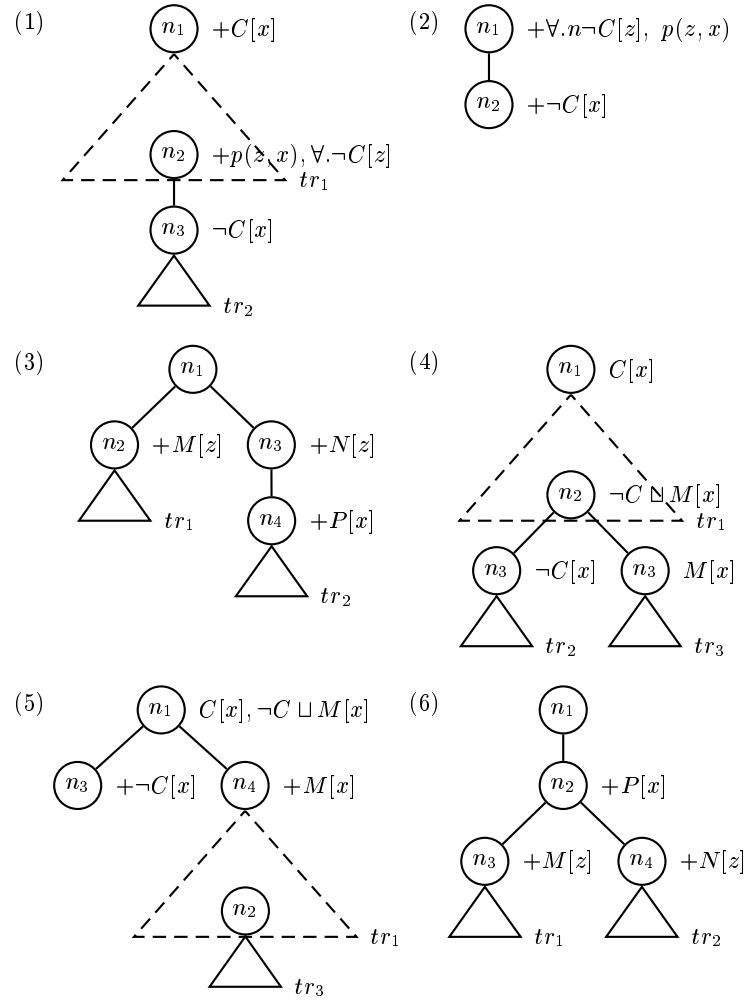


Fig. 13. Impossible proof tree, for Proposition 6

Proof. Because the proof tree is in seminormal form, the tree tr_3 can only be affected by $A[y]$, $\text{Top1}[y]$, $\forall\neg B[x]$ and axioms in $T^*[[y']]$ for $y' \in \text{family}(y)$. (This is because, by the seminormal structure of the proof tree, any other assertions $\alpha[x]$ cannot be expanded by rules in tr_3 to create new nodes under n_2 .) Since neither A nor Top1 are essential, then by Proposition 5 there can be no essential $\neg B[y]$ in the subtree tr_3 . Hence there are no essential $\forall\neg B[x]$ at n_3 , nor at n_1 . Therefore the only thing the proof tree tr_3 depends on is $T^*[[y']]$ for $y' \in \text{family}(y)$, and so there is a closed version of tr_3 where y is replaced by x . But such a situation cannot occur in a seminormal proof tree according to clause 3 of the definition of pruning.

We now repeat variants of the propositions above for nodes with essential concepts of the form $2:E$. The proofs will often be extensions of the ones above, because there are additional axioms, not just those of the form $\neg E \sqcup F$.

Proposition 7. *Let tr_1 be a seminormal proof tree rooted at n_1 . Then all essential $E[x] \in \mathcal{P}(n_1)$ have the property that either*

$$\vdash E \sqsubseteq \perp \tag{6}$$

or

$$\text{there is essential } \neg E''[x] \in \mathcal{P}(n_1) \text{ such that } \vdash E \sqsubseteq E'' \tag{7}$$

(Note that x need not be x_0).

Proof. By induction on the number of nodes in the proof tree. (As before, we drop $[x]$ for convenience.) Let E be essential on n_1 .

If the tree size is 1, the same argument applies as in Proposition 4

Suppose the tree size is $k + 1$. Therefore there is no clash yet at the root, so that there must be $\neg E$ introduced below n_1 . Since there are no terms of the form $\forall p.2:E$ in the Corollary). theory T^* , the only way such a $\neg E$ could have been introduced is via an expansion of a $\neg E \sqcup M$, due to a subsumption axiom $E \sqsubseteq M$.

Case 1: $M = F$; then the same proof applies as in proposition 4, and proof is completed.

Case 2: $M = \exists A$, with essential M . This means that there is a y -introduction at some node n_4 , with a closed subtree on it, and A or Top1 essential at n_4 or just below it (by Proposition 6).

By Propositions 4, there is at most one $\forall\neg B[x]$ that introduces $\neg B$ onto y in a pruned proof. Consider then two cases.

1. No essential $\neg B[y]$ was introduced in the proof under n_4 , which by Proposition 4 implies that $\vdash A \sqsubseteq \perp$, either because A is essential, or because an essential Top1 was introduced, leading, by Proposition 4, to $\vdash \text{Top1} \sqsubseteq \perp$; this, together with $\vdash A \sqsubseteq \text{Top1}$ (ins rule Top) also gives $\vdash A \sqsubseteq \perp$ by transitivity. Therefore, in this case we have $\vdash A \sqsubseteq \perp$, from which we get $\vdash \exists A. \sqsubseteq \exists. \perp$ by the some-isa inference rule, and in turn $\vdash \exists \perp. \sqsubseteq \perp$, from the some-empty inference rule, so that by transitivity twice we get $\vdash E \sqsubseteq \perp$, as desired.

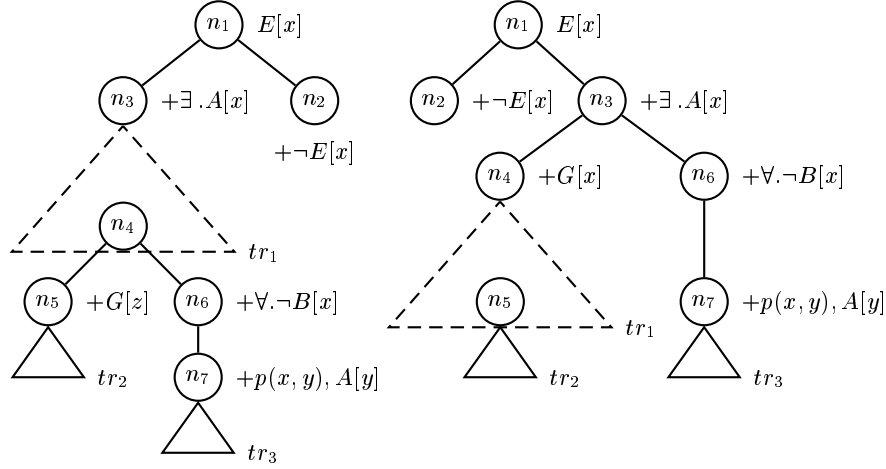


Fig. 14. Seminormal proof tree after movement up of $\neg E$ and then after making onto and into rules adjacent conflicting with E .

2. $\neg B[y]$ is introduced by the expansion of $\forall \neg B[x]$. Since the proof tree is in seminormal form, recall it has the form in Figure 14, and we have by Proposition 4 that $\vdash A \sqsubseteq B$, which leads to $\vdash \exists A \sqsubseteq \exists B$, by inference rule (Some-isa). This, together with transitivity on the two rules building the tree, lead to $\vdash E \sqsubseteq G$. The problem now is to prove that $\vdash G \sqsubseteq G''$ for some $G'' \in \mathcal{P}(n_1)$.

For this purpose, re-organize, without increasing the number of nodes in it, the proof tree to the second part of Figure 14. (There are two variants depending on the number of children.)

Since G is essential at n_4 , by induction either $\vdash G \sqsubseteq \perp$, or $\vdash G \sqsubseteq G''$ for some G'' such that $\neg G'' \in \mathcal{P}(n_4) = \mathcal{P}(n_1) \cup \{\exists A, G\}$. So by transitivity $\vdash E \sqsubseteq G''$ for $\neg G'' \in \mathcal{P}(n_1)$, since n_4 has only two, non-negated atoms added to it.

Proposition 8. *In a semi-normal proof tree there can be no subtree tr_1 rooted at node n_1 such that $\mathcal{P}(n_1)$ contains an essential $\neg E[x]$, but no essential $G[x]$*

Proof. Same proof as for Proposition 5, except in the inductive case, not only do we have to consider the case when $\neg E$ clashes with E from $E \sqcup \neg F$, but also the case when it clashes with E from $E \sqcup \forall \neg B$. Since tr_1 is closed and $\forall \neg B$ was essential, then there is some y introduced by $\exists A$, such that $\neg B[y]$ is asserted of it. Moreover the tree rooted at the introduction of y must be closed. But such a $\exists A$ can only come from $\neg H \sqcup \exists A$. So the proof tree looks like the one on the left side of Figure 14, except that the rules for $\forall \neg B$ and $\exists A$ are swapped. As before we re-balance the tree so that the onto and into rules are successors (see second tree in Figure 14). But now $\neg H$ is again essential at its node without any essential unnegated atoms in $\mathcal{P}(n_3)$, so by induction such the tree at n_3 cannot exist, since $size(tr_1) + size(tr_2)$ is less than the size of the original tree at n_1 .

Proposition 9. *If n_0 is the root of the original seminormal proof tree, with $\mathcal{P}(E[x_0], \neg F[x_0])$, and E is not essential then $\vdash \top \sqsubseteq \perp$.*

Proof. By Proposition 8, $\neg F$ cannot be essential either, so the only relevant part of $\mathcal{P}(n_0)$ is $T^*[[x_0]]$.

Consider the first branch at the root of the proof tree:

If it is caused by the expansion of $\neg E \sqcup M$ or $\neg C \sqcup M$, then the left subtree has only an essential negated atom, contradicting Propositions 5, 8.

The proof might also begin with the expansion of $G \sqcup \forall \neg B$. Since $\forall \neg B$ must be essential, then the same analysis as at the end of Proposition 8 shows that there is a subtree with only a single negated essential atom on it, and no positive ones, which contradicts Proposition 8.

This leaves the case of the first step expanding a formula of the form $\exists \hat{P}.Top_i[x_0]$. If Top_1 is essential, then by proposition 4 $\vdash Top_1 \sqsubseteq \perp$ is the only possibility. If Top_2 is essential, then by proposition 7 $\vdash Top_2 \sqsubseteq \perp$.

Either way, we have from this that $\vdash \exists \hat{P}.Top_i \sqsubseteq \exists \hat{P}.\perp$. Since we also have a rule of inference (some-isa) providing $\vdash \exists \hat{P}.\perp \sqsubseteq \perp$, and axioms in T^* of the form $\top \sqsubseteq \exists \hat{P}.Top_i$, the result holds.

This leaves the case when in a seminormal proof tree, the only place where conflicts can occur are on the node n_1 caused by the introduction of y by $\exists \hat{P}.Top_i$, but Top_i is not essential. This means that $\mathcal{P}(n_1) = T^*[[x_0]] \cup T^*[[y]]$. But then x_0 and y are indistinguishable, and hence the original proof tree could not have been pruned (x_0 could have been substituted for y), contradicting its seminormal nature.