

Optimizing Description Logic Reasoning with Nominals: First Results

Evren Sirin, Bernardo Cuenca Grau, Bijan Parsia
Maryland Information and Network Dynamics Lab.
8400 Baltimore Av.
College Park, MD, 20740 USA

evren@cs.umd.edu, bernardo@mindlab.umd.edu, bparsia@isr.umd.edu

Abstract

OWL-DL is a World Wide Web Consortium standard for representing ontologies on the Semantic Web. It can be seen as a syntactic variant of the Description Logic $\mathcal{SHOIN}(\mathcal{D})$, with an OWL-DL ontology corresponding to a $\mathcal{SHOIN}(\mathcal{D})$ knowledge base. The very recent accomplishment of a decision procedure for $\mathcal{SHOIN}(\mathcal{D})$ poses the challenge of turning the decision procedure into a practical implementation. In particular, we emphasize the need of new optimization techniques for *nominals*, especially in the presence of large number of individuals in the KB.

In this paper, we present new techniques for optimizing DL reasoning in the presence of nominals in the TBox and individuals in a large ABox. We have integrated our optimizations in the open-source Pellet reasoner, which is sound and complete for $\mathcal{SHOIN}(\mathcal{D})$, and found that they suffice for efficiently classifying the famous Wine Ontology. We also show that these optimization techniques produce significant performance improvements in other widely used ontologies containing nominals, such as the OWL-S and AKT ontologies.

1 Introduction and Motivation

OWL-DL became a World Wide Web Consortium standard for representing ontologies on the Semantic Web in February, 2004. As the group approached completion, there were two deep controversies with regard to the expressivity of the language: first, there was, at that point, no decision procedure for OWL-DL, a language many felt had decidability as its main justification, and, secondly, the example ontology in the OWL specifications [10], the Wine Ontology, which tried to exercise every feature of OWL-DL, was not processable by existing or anticipated reasoner. Of particular concern were the presence of a large number of *nominals*, that is, individuals appearing in concept definitions. At the time, there were few reasoners, if any, that could handle nominals at all. In this paper, we present a suite of optimizations implemented in our OWL-DL reasoner, Pellet [9], that suffice to render the Wine ontology (and most current ontologies with nominals) a solved problem.

OWL-DL can be seen as a syntactic variant of the Description Logic $\mathcal{SHOIN}(\mathcal{D})$, with an OWL-DL ontology corresponding to a $\mathcal{SHOIN}(\mathcal{D})$ knowledge base ¹. The logic $\mathcal{SHOIN}(\mathcal{D})$ is a decidable fragment of First Order Logic (FOL) and extends the Description Logic \mathcal{S} (the DL providing transitive roles, all the boolean operators on concepts as well as existential and universal restrictions) with *unqualified number restrictions* (\mathcal{N}), *nominals* (\mathcal{O}), *inverses on roles* (\mathcal{I}), *role hierarchies* (\mathcal{H}) and *datatypes* (\mathcal{D}).

Although tableau-based decision procedures for prominent fragments of $\mathcal{SHOIN}(\mathcal{D})$, such as $\mathcal{SHIN}(\mathcal{D})$ [4] and $\mathcal{SHON}(\mathcal{D})$ [2] have been known for quite a long time, the design of a decision procedure for $\mathcal{SHOIN}(\mathcal{D})$ has been accomplished only very recently [5].

Expressive description logics, in particular the ones mentioned above, are known to have very high worst-case complexity. As a consequence, there exists a significant gap between the design of a decision procedure and the achievement of a practical implementation. Naive implementations are doomed to failure. In order to achieve acceptable performance, modern DL reasoners, such as FaCT, RACER, DLP and Pellet, implement a suite of *optimization techniques* [1]. These optimizations lead to a significant improvement in the empirical performance of the reasoner and have proved effective in wide variety of realistic applications.

However, at the current stage of research and deployment, existing optimizations have been implemented and proved useful for the description logic $\mathcal{SHIN}(\mathcal{D})$. From an implementation point of view, the recent achievement

¹We refer the reader to [5] and [7] for a detailed discussion of \mathcal{SHOIN} and OWL-DL respectively. We also recommend [3] for a thorough discussion on the relationship between OWL and expressive Description Logics.

of a decision procedure for $\mathcal{SHOIN}(\mathcal{D})$ poses new challenges:

- While many optimization techniques are completely independent of the DL supported by the reasoner, others are valid for certain logics only. In particular, some major optimizations for reasoning with large ABoxes rely on the absence of nominals in the definition of concepts. Moreover, in the presence of nominals, ABox assertions can affect concept satisfiability and TBox classification. In other words, nominals break the traditional “separation” between TBox and ABox in DLs. As a consequence, ontologies with nominals in the TBox and large number of instances in the ABox are likely to compromise the performance of DL reasoners.
- Nominals are not supported by state of the art DL reasoners, with the only exception of the Pellet system. Thus, there is very little experience in developing techniques for dealing with nominals efficiently in practice. In particular, to the best of our knowledge, no optimizations specific for nominals have been designed and tested yet.

From a logical point of view, the *nominal constructor* [2] [8] transforms the object name o into the concept description $\{o\}$, which is evaluated, by every model-theoretic interpretation, to a singleton set with o as its only element. So far, nominals have been partially approximated in DL reasoners by treating them as pair-wise disjoint atomic concepts, commonly called *pseudo-nominals*. However, this technique is known to lead to incorrect inferences in some cases.

From a modeling point of view, nominals are used in a significant number of ontologies available on the Semantic Web. The OWL-DL specification [7] contains two modeling constructs specific for nominals, which illustrate their main uses in Ontology Engineering.

- The *OneOf* construct allows to define a concept by finite enumeration of its elements. For example, the atomic concept *Continent* can be defined, using nominals, as follows:

$$Continent \equiv \{europe, asia, america, antartica, africa, oceania\}$$

where the elements of the enumeration are individuals in the KB.

- The *hasValue* construct is used as a shorthand for an existential restriction on a nominal concept. This construct can be used to describe catholics as persons who follow the Pope, or Rock’n’Roll fans as the persons who venerate Elvis:

$$Catholic \sqsubseteq Person \sqcap \exists follows.\{pope\}$$

$$RockFan \sqsubseteq Person \sqcap \exists hasIdol.\{elvis\}$$

One prominent example of the use of nominals for modeling is the ontology used in the OWL documentation: the Wine Ontology [10].

This ontology extensively relies on the *OneOf* and *hasValue* constructs for describing different kinds of wines according to various criteria, like the area they are produced in, the kinds of grapes they contain, their flavor and color, etc. For example, a “Cabernet Franc Wine” is defined to be a dry, red wine, with moderate flavor and medium body and which is made with Cabernet Franc grapes

$$\begin{aligned} CabernetFranc &\equiv Wine \sqcap \exists madeFrom.\{cabFrancGrape\} \sqcap \leq 1 madeFrom \\ CabernetFranc &\sqsubseteq \exists hasColor.\{red\} \sqcap \exists hasFlavor.\{moderate\} \sqcap \exists hasBody.\{medium\} \end{aligned}$$

Potential wine flavors, colors, etc are defined using an enumeration. For example:

$$WineFlavor \equiv \{delicate, moderate, strong\}$$

The Wine ontology contains only 138 concepts and 206 individuals and hence it is a relatively small knowledge base. However, its classification has remained, so far, an open problem for DL reasoners.

What makes the Wine ontology hard for automated reasoning? First, nominals break the traditional TBox-ABox separation. As a consequence, the computational cost of every new individual in the ontology is very high: a relatively small number of individuals (a couple of hundreds) affects reasoning performance dramatically; second, the ontology contains a significant number of General Concept Inclusion Axioms (GCIs) associated to nominals that cannot be handled by current absorption techniques. As a result, tableau expansions become very expensive computationally and hence every additional satisfiability test performed during classification is likely to be very expensive.

In this paper, we present new techniques for optimizing DL reasoning. These techniques aim at alleviating the impact of the sources of complexity mentioned above. We have integrated our optimizations in the open-source Pellet reasoner, which implements the *SHOIN(D)* decision procedure presented in [5] and found that they suffice for efficiently classifying the Wine Ontology. In this paper, we also show that these optimization techniques produce significant performance improvements in other widely used ontologies containing nominals, such as the OWL-S and AKT ontologies.

2 Novel Optimizations

In this section, we present a novel suite of optimization techniques:

- *Nominal Absorption* aims at localizing non-determinism in the KB caused by General Concept Inclusion Axioms involving nominals.

- *Partial Backjumping* allows to prune the search graph by discarding “redundant” non-deterministic choices.
- *Learning-based Disjunct Selection* is a heuristic to guide the search based on a simple learning algorithm.
- *Nominal-based Pseudo-model Merging* allows to reduce the number of satisfiability tests performed during classification by taking advantage of the semantics of *hasValue* restrictions.
- *Lazy Forest Generation* aims at sparing the application of some expansion rules during the satisfiability checking for an atomic concept by creating nominal nodes *only* when needed in the tableaux expansion
- *Forest Caching* stores the saturated tableaux expansion constructed during the initial KB consistency check and reuses it for subsequent concept satisfiability and subsumption tests

Partial Backjumping and Learning-based Disjunct Selection are completely independent of the DL under consideration; the other techniques are only effective in the presence of nominals in the KB. In what follows, we describe these techniques in detail.

2.1 Nominal Absorption

General Concept Inclusion Axioms (GCIs) are hard to reason with, given the the high degree of non-determinism they introduce. For each GCI, one disjunction is added to the label of *each* node in a tableaux expansion, which causes an exponential blow-up in the search space. As a consequence, even a reduced number of GCIs can degrade the performance of a DL reasoner significantly.

Absorption is an optimization technique that tries to eliminate GCIs as possible from a KB by replacing them with primitive definitions. Absorption has revealed a key technique in the past for processing DL ontologies, such as the GALEN medical ontology.

As stated before, the two main uses of nominals for modeling are the definition of concepts by finite enumeration of its elements (the OWL *OneOf* construct) and the definition of concepts in terms of existential restrictions on a nominal (the OWL *hasValue* construct). For both cases, we provide an extension of existing absorption techniques.

Let us start with enumerations. Consider the concept *WineColor* in the Wine Ontology, defined as follows:

$$\begin{aligned} \textit{WineColor} &\equiv \{red, rose, white\} \\ \textit{WineColor} &\sqsubseteq \textit{WineDescriptor} \end{aligned}$$

Both axioms involve a GCI that is not captured by currently available absorption techniques and hence, the disjunction:

$$\neg WineColor \sqcup \{red, rose, white\}$$

would be added to every node in the tableau expansion. On the other hand, an enumeration is equivalent to the disjunction of its elements, i.e.:

$$\{rose, red, white\} \equiv \{rose\} \sqcup \{red\} \sqcup \{white\}$$

This leads to an additional difficulty: enumerations are likely to introduce a significant number of backtracking points. These disjunctions, when added to every node of the tableau expansion, cause the search space to grow exponentially with the number of elements in the enumeration. Thus, the presence of these non-absorbable GCIs is doomed to significantly affect reasoning performance.

Nominal absorption is a novel optimization technique that transforms these definitions into a primitive definition and a set of ABox assertions. The technique relies on the following equivalence:

$$C = \{a_1, \dots, a_n\} \Leftrightarrow C \sqsubseteq \{a_1, \dots, a_n\}, \text{ and } C(a_1), \text{ and } \dots, \text{ and } C(a_n)$$

Where $C(a_1), \dots, C(a_n)$ are ABox assertions and $C \sqsubseteq \{a_1, \dots, a_n\}$ is a primitive definition for C . Note that the set $C(a_1), \dots, C(a_n)$ of ABox assertions is equivalent to the GCI $\{a_1, \dots, a_n\} \sqsubseteq C$. In our example, the assertions:

$$WineColor(red); WineColor(rose); WineColor(white)$$

state that *red*, *rose* and *white* are instances of the concept *WineColor* (not necessarily the only ones). Thus, the enumeration $\{red, rose, white\}$ is subsumed by *WineColor*.

Let us consider now the case of *hasValue* restrictions. Axioms in the following form are commonly found in the Wine ontology:

$$Riesling \equiv Wine \sqcap \exists madeFrom. \{RieslingGrape\} \sqcap \leq 1 madeFrom$$

Considering that there are other subclass axioms in the ontology with the concept *Riesling* in its left hand side, we are again left with GCI's. Standard absorption techniques can take care of such cases by absorbing the axiom into the definition of *Wine* concept, i.e. the concept $(Riesling \sqcup \forall madeFrom. \neg \{RieslingGrape\} \sqcup \geq 2 madeFrom)$ is added to the definition of *Wine*. However, this disjunctive definition to the *Wine* concept introduces a backtracking point in the tableau expansion for every node containing *Wine* in its label. Absorption introduces around 30 of such disjunctions relative to the *Wine* concept, which significantly increases the search space.

However, the semantics of nominals allows a more effective absorption of the above axiom by taking profit of the following equivalence²:

²See the appendix for a proof

- | |
|--|
| <ul style="list-style-type: none"> (i) <i>Initialize</i> Create a set $G = \{C, \neg D\}$ for the axiom $C \sqsubseteq D$. (ii) <i>Concept absorption</i> If $A \in G$ where A is atomic, replace $(A \sqsubseteq C) \in T_u$ with $A \sqsubseteq \sqcap\{C, \neg(\sqcap(G \setminus \{A\}))\}$ to T_u and exit. (iii) <i>OneOf absorption</i> If $C \in G$ where C is an enumeration $\{o_1, \dots, o_n\}$, add $\neg(\sqcap G)$ to each individual o_i and exit. (iv) <i>HasValue absorption</i> If $C \in G$ where C is in the form $\exists p.\{o_1, \dots, o_n\}$ then add $\forall p^-. \neg(\sqcap G)$ to each individual o_i and exit. (v) <i>Simplification</i> If $A \in G$ (resp. $\neg A \in G$) where $(A \equiv D) \in T_u$, then substitute A (resp. $\neg A$) with D (resp. $\neg D$) and go to (ii). (vi) <i>Conjunction simplification</i> If $C \in G$ where C is in the form $(C_1 \sqcap \dots \sqcap C_n)$, then set G to $G \sqcup \{C_1, \dots, C_n\}$ and go to (ii). (vii) <i>Recursion</i> If $C \in G$ where C is in the form $(C_1 \sqcup \dots \sqcup C_n)$, then for every C_i try recursively absorbing $\neg C_i \sqcup G \setminus \{C\}$, else fail. |
|--|

Figure 1: Standard absorption algorithm extended with nominal absorption

$$\exists p.\{o\} \sqsubseteq C \Leftrightarrow \{o\} \sqsubseteq \forall p^-.C \Leftrightarrow (\forall p^-.C)(o)$$

which would yield the following ABox assertion for this example:

$$(\forall madeFrom^-. (Riesling \sqcup \neg Wine \sqcup \geq 2 madeFrom))(RieslingGrape)$$

The resulting axiom still contains the same number of disjuncts, but this time the effect is localized to the individuals related to *RieslingGrape* which are significantly less than the *Wine* instances.

Figure 1 describes the standard absorption algorithm extended with nominal absorption.

2.2 Learning-based Disjunct Selection

When a disjunction in the label of the node is being expanded, the order in which disjuncts are selected can make a drastic change in the performance of the tableau reasoner. Many different heuristics have been developed for DPLL SAT algorithms to minimize the size of the search tree. However, it has been shown in the DL literature that such heuristics generally counter-interact with other optimizations like dependency-directed backjumping [1].

An investigation of real world ontologies reveals that, in many cases, there are some disjunctions that inherently have one possible expansion. However, this is detected by the reasoner only after numerous tableaux rule applications. Moreover, this expensive cycle is typically repeated for individuals with similar characteristics. Let us illustrate this case with an example from OWL-S ontologies:

$$\begin{aligned} Process &= AtomicProcess \sqcup CompositeProcess \sqcup SimpleProcess \\ AtomicProcess &\sqsubseteq \neg CompositeProcess \end{aligned}$$

$$\begin{aligned} & \text{CompositeProcess} \equiv \leq 1.\text{composedOf} \sqcap \geq 1.\text{composedOf} \\ \top \sqsubseteq \forall \text{composedOf}.\text{ControlConstruct} \sqcap \forall \text{composedOf}^{\neg}.\text{CompositeProcess} \end{aligned}$$

These axioms go through standard preprocessing steps, e.g. normalization and absorption, and finally we get the following axiom³:

$$\text{Process} \sqsubseteq \geq 2.\text{composedOf} \sqcup \text{CompositeProcess} \sqcup \leq 0.\text{composedOf}$$

During tableaux expansion, for any *AtomicProcess* instance we will face to expand this disjunction. Obviously, there is only one right selection here ($\leq 0.\text{composedOf}$) since the first disjunct is unsatisfiable by definition and the second disjunct causes a clash, when combined with *AtomicProcess*. However, a DL reasoner will observe this fact only after applying several other rules, in this case the \geq -rule and *unfolding-rule*. When these rule applications are interleaved with other rule applications, several other disjunctions might have been expanded for a different number of individuals, which causes a significant amount of wasted computation. Moreover, OWL-S knowledge bases would typically have lots of *AtomicProcess* instances and, consequently, these steps would be repeated for each of such instances, which degrades performance significantly.

The learning-based disjunct selection technique aims to minimize the wasted computation by avoiding inherently clash-generating expansions. The idea is to reuse the clash-free expansions for instances with similar characteristics. The heuristic is to sort the disjuncts based on how many clashes they caused during rule applications. Note that when the dependency sets for concepts are being maintained it is quite easy to detect if a certain disjunction expansion caused the clash or not.

<pre> function expand-disjunction(<i>x</i>, <i>D</i>) int[] stats = get-statistics(<i>D</i>) if stats not found then stats = new int[<i>n</i>] $\forall i$ stats[<i>i</i>] = 0 save-statistics(<i>D</i>, stats) Pick the next untried disjunct D_k such that stats[<i>k</i>] is minimum Add D_k to $\mathcal{L}(x)$ and continue tableau expansion if there is a clash then increment stats[<i>k</i>] </pre>
--

Figure 2: Pseudo-code of learning-based disjunct selection

Note that our technique only learns only from clashes, i.e. unsuccessful selections, but do not keep track of successful expansions. It would be nearly

³There is a possibility that absorption algorithm yields different results depending on the order axioms are processed, but this was not the case for this example.

impossible to keep track of successful expansions during completion since it is not clear when and how we can conclude a disjunction expansion was successful. On the other hand, it is possible to do a post-processing step after a clash-free completion where we iterate through the nodes in the completion graph and update the disjunction statistics for future use.

2.3 Partial Backjumping

Backjumping is a very powerful optimization technique where the branching points responsible for a clash are detected and backtracking jumps over the intervening branching points without exploring any alternatives. Obviously such alternatives will not eliminate the cause of the clash and exploring those possibilities would be unproductive. However, this does not mean that the work done in that intervening branches were not useful. It might very well be the case that we will repeat some of the steps that were done in that intervening steps. To illustrate this point, consider the following definition for concept X :

$$\begin{aligned} X \sqsubseteq (C_1 \sqcup D_1), \dots, (C_k \sqcup D_k), (\exists p.A \sqcup B), \dots, (C_n \sqcup D_n), \forall q.F \\ B \sqsubseteq \neg F, q \sqsubseteq_{\mathcal{R}} p \end{aligned}$$

During the satisfiability check of concept X , it is clear that expanding the disjunct $\exists p.A$ will eventually cause a clash due to the contradictory value restriction. However, most typically, the tableaux reasoner would first expand all the disjunctions and then apply the \exists -rule to create the successor node where the clash will occur. This means that all the other disjunctions would be expanded causing C_1, C_2, \dots, C_n to be added to the label of the node. Backjumping to the root of the clash

2.4 Nominal-based Pseudo-model Merging

Classification of named concepts in a KB is one of the most important applications of DL reasoners. Optimization techniques for classification aim at reducing as much as possible the number of subsumption tests to be performed.

Nominal-based pseudo-model merging is a novel optimization technique for classification that exploits the semantics of nominals for discovering “obvious” non-subsumptions between concepts in the KB.

In particular, this technique is especially effective if there are many concepts in the KB defined in terms of existential restrictions on nominals (or *hasValue* restrictions in OWL jargon). For example, the concept:

$$RedWine \sqsubseteq Wine \sqcap \exists hasColor.\{red\}$$

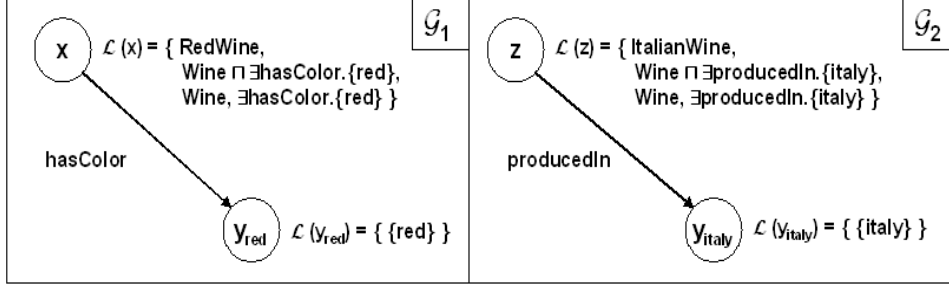


Figure 3: **Completion Graphs**

is defined in terms of the nominal concept $\{red\}$.

The nominal-based pseudo-model merging technique uses cached information relative to nominals from previous satisfiability tests to prove non-subsumption without performing a new satisfiability test.

The basic idea is to examine the edges from the blockable root node to nominal nodes in the tableau expansion and use that information for discovering non-subsumptions in the KB. For example, suppose we want to check whether Italian wines, defined as follows:

$$ItalianWine \sqsubseteq Wine \sqcap \exists producedIn.\{italy\}$$

are subsumed by red wines. The completion graphs for *ItalianWine* and *RedWine* are schematically shown in Figure 3. The concept *RedWine* is in the label of the root node x in \mathcal{G}_1 and x is connected to the nominal node $y_{\{red\}}$ through an edge that has the role *hasColor* in its label. On the other hand, in \mathcal{G}_2 , the nominal node $y_{\{red\}}$ is not neighbor of the root node z . From this information, it is possible to infer that $\mathcal{O} \not\models ItalianWine \sqsubseteq RedWine$. Note that, in case *producedIn* had been a transitive role, instead of testing for node neighborhood, we would have considered paths connecting the root node and the nominal node.

However, there is still one more important consideration to make. Let us consider the following axioms:

$$\begin{aligned} C &\equiv Wine \sqcap \exists hasSugar.\{dry, offdry\} \\ D &\equiv Wine \sqcap \exists hasSugar.\{dry\} \end{aligned}$$

We want to test whether D is subsumed by C . The graphs \mathcal{G}_1 and \mathcal{G}_2 in Figure 4 are valid completion graphs for D and C respectively. Again, the concept D is in the label of the root node x in \mathcal{G}_1 and x is connected to the nominal node $y_{\{dry\}}$ by a *hasSugar*-edge. On the other hand, in \mathcal{G}_2 , the nominal node $y_{\{dry\}}$ is not neighbor of the root node z , which has the concept C in its label. However, $\mathcal{O} \models D \sqsubseteq C$.

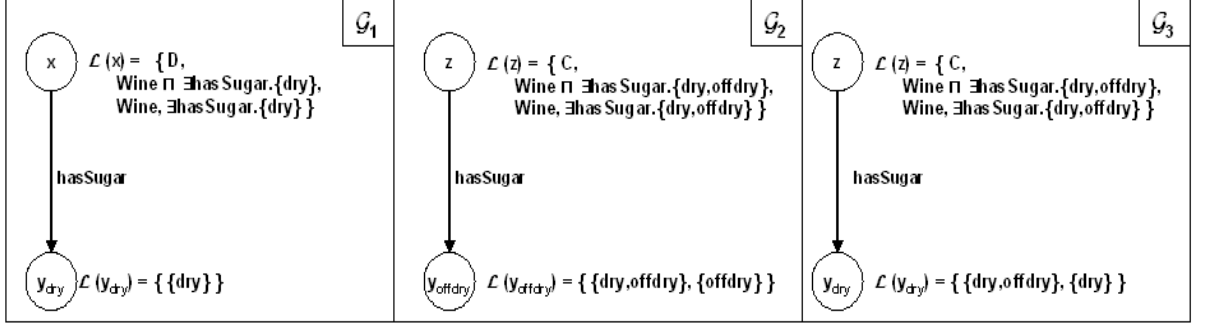


Figure 4: **Completion Graphs**

The reason why the entailment $\mathcal{O} \not\models D \sqsubseteq C$ is wrong in this case is that there is a completion graph (\mathcal{G}_3 in Figure 4) for C in which the node x , with C in its label, *does* have a *hasSugar*-edge leading to the nominal node $y_{\{dry\}}$. Therefore, in order to infer the non-subsumption, the edge to the nominal node should be present in *every* possible completion graph for C or, in other words, the presence of the edge should not depend on a non-deterministic choice in the execution of the tableau algorithm.

More formally, the technique works as follows; suppose that we want to test whether an ontology \mathcal{O} entails the subsumption relation $D \sqsubseteq C$. Suppose that \mathcal{G}_C (respectively \mathcal{G}_D) is a fully expanded and clash-free tableaux expansion representing a common model of C and \mathcal{O} (respectively a common model of D and \mathcal{O}) and that the following conditions hold:

1. C (respectively D) belongs to the label of a blockable root node x_C in \mathcal{G}_C (respectively a blockable root node x_D in \mathcal{G}_D).
2. If the role p is *simple*:
 - (a) The nominal node o is a p -neighbor of x_C in \mathcal{G}_C and the presence of such an edge does not depend on a non-deterministic choice.
 - (b) The nominal node o is not a p -neighbor of x_D in \mathcal{G}_D .
3. If the role p is not simple:
 - (a) There is a path z_0, \dots, z_k in \mathcal{G}_C with $k > 1, x = z_0, o = z_k$ and z_i a q -neighbor of z_{i-1} for $0 \leq i < k$ and some q a sub-role of p . Moreover, the presence of such a path does not depend on a non-deterministic choice.
 - (b) There is no such path from x_D to the nominal node o in \mathcal{G}_D .

Then, $\mathcal{O} \not\models D \sqsubseteq C$. The soundness of the technique is shown in the appendix.

2.5 Lazy Forest Generation

In the presence of nominals in the TBox, ABox assertions can affect concept satisfiability and classification. Thus, when checking the satisfiability of an atomic concept A after the initial KB consistency check, we need, in principle, to include in the initial completion graph for A a root nominal node x_a for each individual a in the ABox. The presence of these nodes in the initial configuration of the graph is likely to cause a large number of expansion rules to be triggered and hence may involve a significant computational overhead.

However, even in the presence of nominals in the TBox, there are typically many atomic concepts whose corresponding satisfiability check *does not* involve the application of the nominal rule and, therefore, the content of the ABox does not influence their satisfiability. For these concepts, generating the nominal nodes corresponding to the ABox individuals results in an unnecessary overhead. Since the KB is consistent, the rules triggered by the presence of the initial forest of nominal nodes will never yield to a clash in the tableau expansion for A .

Lazy forest generation avoids such a computational burden by including the forest of nominal nodes corresponding to the ABox *only* if the nominal rule is applied during a concept satisfiability check. This simple technique may yield a dramatic performance improvement, as discussed later on in Section 3.

It is important to realize that lazy forest generation may interact with *dependency-directed backjumping* and, in order to ensure the correctness of the technique, we generate the initial nominal forest everytime backjumping is applied, even if the nominal rule has not been triggered.

The reader may have noted that lazy forest generation is very conservative in two different ways: first, even if a merge is forced by the application of the nominal rule, there are cases in which it suffices to generate only a *part* of the initial nominal forest; second, the generation of the forest may not always be required after backjumping. This provides room for further improvements in the near future.

2.6 Forest Caching

The main idea underlying the forest caching technique is to store the state of the completion graph *after* the initial KB consistency check and reuse it for subsequent concept satisfiability and subsumption. Even if lazy forest generation is used, we may have to expand in many cases the nominal forest from scratch. Building the nominal forest from its initialization state may involve the application of a large number of expansion rules; repeating the process

for different concept satisfiability tests causes a significant computational overhead.

The nominal forest is first expanded during the initial KB consistency test. Caching the status of the tableaux expansion after consistency can significantly speed-up the subsequent concept satisfiability and subsumption, as will be discussed in Section 3. It is important to note that, in order to ensure correctness, we not only need to store the expanded forest, but the whole *status* of the expansion after KB consistency, including non-deterministic choices that remain to be explored. Although caching this information affects memory consumption, the overhead is not critical and pays off in terms of speed-up.

3 Empirical Results

We have integrated the optimization techniques presented in this paper into the *SHOIN*(\mathcal{D}) reasoner Pellet. In this section, we evaluate the performance of the reasoner for the tasks of consistency checking, classification and realization. A time limit of 300 seconds were set for each task. All the experiments have been performed on a Pentium Centrino 1.6GHz computer with 1.5GB memory. The maximum memory amount allowed to Java was set to 256MB for each experiment.

We have run the experiments on four ontologies : the Wine Ontology, presented in the OWL documentation [10], the AKT Portal Ontology, used in the AKT project for integrating information across universities, the OWL-S ontologies, for describing Web Services, and the 3SAT ontology, included in the OWL test suite, which is an encoding of the classical 3SAT problem into OWL-DL.

In order to evaluate the impact of each optimization, we have disabled the optimizations one by one when processing each ontology. The results are shown in Figure 5. The first column indicates the enabled optimizations; the remaining columns show the times for the initial ontology consistency check, classification (including satisfiability of atomic concepts) and realization of individuals respectively.

The Wine Ontology is a medium-size ontology and it uses all of the constructs provided in OWL-DL. It contains 137 atomic concepts, 17 roles and 206 individuals. The concepts defined in the ontology are fairly complex and nominals are used profusely. With all the optimizations enabled, consistency checking takes less than a second, whereas the total processing time, including classification and realization takes approximately 20 seconds. Nominal absorption has the highest impact on performance: without any kind of

Options	Wine			OWL-S		
	Consist.	Classif.	Real.	Consist.	Classif.	Real.
OHDMB	772.0	16911.4	2154.3	377.6	2422.5	1021.5
_HDMB	16608.9	NaN	NaN	407.6	2634.7	1141.8
O_DMB	21748.2	64463.7	61412.4	387.4	2500.7	1062.5
_DMB	230463.5	NaN	NaN	388.7	2488.4	1083.6
OH_MB	3184.3	27182.1	35246.7	18006.8	2052.0	1059.5
OHD_B	766.0	32294.3	9852.3	391.4	2461.7	1089.3
OHD_M	1587.1	16533.8	5975.6	402.6	2184.0	1442.2
OHDMB_C	779.1	20973.1	2155.4	387.3	45669.9	1113.4
OHDMB_	793.2	NaN	NaN	389.4	72805.7	1116.6

Options	AKT Portal			SAT3		
	Consist.	Classif.	Real.	Consist.	Classif.	Real.
OHDMBLC	6.0	399.6	47.0	1651.5	3.0	1.0
_HDMBLC	7.0	2647.0	785.1	11478.5	3.0	6498.3
O_DMBLC	2.0	374.6	41.1	1542.1	2.0	2.1
_DMBLC	6.0	2606.7	786.3	8072.5	1.0	18493.7
OH_MBLC	1.0	1607.2	49.1	1471.1	3.0	1.0
OHD_BLC	3.0	382.6	43.2	920.5	1.0	0.0
OHD_MLC	2.0	386.4	42.0	2844.0	2.0	2.0
OHDMB_C	4.1	1030.4	44.1	1388.9	5.0	1.0
OHDMB_	0.0	1503.3	42.0	1050.4	1362.0	0.0

Figure 5: Experimental Results. All times are in milliseconds. The shorthands for the options are as follows: Nominal absorption on OneOf (O) and hasValue (H), Learning-based Disjunct Selection (D), Nominal-based Pseudo-Model Merging (M), Partial Backjumping (B), Lazy Forest Generation (L), Forest Caching (C). A dash indicates that the optimization has been disabled. All times have been computed as an average of 10 independent runs. Classification times include concept satisfiability and subsumption tests.

nominal absorption Pellet cannot classify the ontology in the specified time limit and consistency time increases by three orders of magnitude. If, in addition, partial backjumping is disabled, consistency checking is not even accomplished. Learning-based disjunct selection is especially effective for realization tests and nominal-based pseudo-model merging heavily influences classification, since it avoids a large number of subsumption tests. Lazy forest generation and forest caching have a dramatic impact on concept satisfiability and subsumption: if both optimizations are disabled, Pellet times out after the initial KB consistency test.

The OWL-S ontology is a medium-sized KB developed by the OWL-S coalition and widely used by the Semantic Web Services community. It contains 97 concepts, 191 roles and 2320 individuals, with 5 nominals. The individuals for our experiments represent Web services and have been generated in a realistic Task Computing environment [6] developed at Fujitsu Labs of America. OWL-S does use nominals, but marginally. The optimization with the most impact is disjunct selection, which makes it possible to identify similarity patterns between individuals and use them for making the right non-deterministic choices during the tableaux expansion.

The AKT portal ontology is also medium-sized. It contains 173 atomic

concepts, 142 roles and 75 individuals, with 15 nominals (all in enumerations). The descriptions are not as complex as those in Wine and nominals are used, though not heavily. Due to the presence of enumerations, nominal absorption reduces classification time. Lazy forest generation, forest caching and learning-based disjunct selection also have an influence in the results.

The 3SAT ontology uses nominals for encoding the 3SAT problem in OWL-DL. Due to the way the problem has been encoded, the ontology contains just 1 atomic concept, no roles and 20 nominals. For this case, nominal absorption and forest caching are especially effective. Both techniques speed up consistency checking time in three orders of magnitude.

Finally, we have run an experiment with a modified version of the Wine Ontology, containing *pseudo-nominals*. We have obtained the following results, after 10 independent runs and with all the optimizations enabled: 541ms for consistency, 2423ms for classification and 158648ms for realization. Note that, since the ABox does not influence reasoning in the TBox, due to the absence of nominals, consistency and classification times are faster; however, a high computational price is paid in realization since nominal-based model merging cannot be used any more. Overall, the total processing time is 1 order of magnitude *slower* with pseudo-nominals. This result indicates that *faking* nominals is more costly, especially when nominals are used heavily in the ontology.

We can summarize our results as follows: **1)** nominal absorption has proven the most useful technique and has a significant impact, even in presence of a marginal number of nominals in the ontology; **2)** Learning-based disjunct selection is particularly effective in the presence of individuals with similar characteristics, as shown in the OWL-S case; **3)** Nominal-based pseudo-model merging is only useful on ontologies with *hasValue* restrictions⁴ and affects primarily classification and realization times; **4)** Lazy forest generation and forest caching can have a dramatic influence on concept satisfiability and subsumption tests **5)** The pseudo-nominal approximation is not only unsound, but also may actually degrade the reasoner’s performance.

4 Conclusion

In this paper, we have presented a new suite of techniques for optimizing DL reasoning in the presence of nominals in the TBox and individuals in the ABox. We have shown that these techniques dramatically improve consistency checking, classification and realization times in real-world ontologies, including the famous Wine Ontology. Contrary to the common belief of the

⁴Wine is the only ontology in our experiments that contains *hasValue* restrictions

DL community, we have proved that reasoning with “real” nominals is typically more efficient than using the pseudo-nominal approximation. Although nominals introduce non-local effects in tableaux expansions, their special semantics can be successfully exploited for optimizations.

References

- [1] I. Horrocks. Implementation and optimisation techniques. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 306–346. Cambridge University Press, 2003.
- [2] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}(\mathcal{D})$ description logic. In B. Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204. Morgan Kaufmann, 2001.
- [3] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From \mathcal{SHIQ} and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.
- [4] Ian Horrocks and Ulrike Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.
- [5] Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for SHOIQ. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*. Morgan Kaufman, 2005.
- [6] Ryusuke Masuoka, Bijan Parsia, and Yannis Labrou. Task computing - the semantic web meets pervasive computing -. In *Proceedings of 2nd International Semantic Web Conference (ISWC2003)*, October 2003.
- [7] P.F. Patel-Schneider, P. Hayes, and I.Horrocks. Web ontology language OWL Abstract Syntax and Semantics. *W3C Recommendation*, 2004.
- [8] A. Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.
- [9] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. Technical report, University of Maryland Institute for Advanced Computer Studies (UMIACS), 2005-68, 2005. Available online at <http://www.mindswap.org/papers/PelletDemo.pdf>.
- [10] M.K. Smith, C. Welty, and D.L. McGuinness. OWL Web Ontology Language Guide. *W3C Recommendation*, 2004.

A Nominal Absorption

Proposition 1 *The following GCIs are logically equivalent:*

1. $\exists p.\{o\} \sqsubseteq C$
2. $\{o\} \sqsubseteq \forall p^-.C$

Proof

Let $\mathcal{I} = (\mathcal{W}_I, \cdot^{\mathcal{I}})$ a model of 1) s.t. it does not satisfy 2). Since \mathcal{I} does not satisfy 2), then $o \notin (\forall p^-.C)^{\mathcal{I}}$ which implies that $o \in (\exists p^-. \neg C)^{\mathcal{I}}$. Thus, there exists an object $x \in \mathcal{W}_I$ s.t. $(x, o) \in p^{\mathcal{I}}$ and $x \in (\neg C)^{\mathcal{I}}$. On the other hand, since \mathcal{I} satisfies 1) and $x \in (\exists p.\{o\})^{\mathcal{I}}$, then $x \in C^{\mathcal{I}}$, which yields a contradiction.

Let $\mathcal{J} = (\mathcal{W}_J, \cdot^{\mathcal{J}})$ a model of 2) s.t. it does not satisfy 1). Since \mathcal{J} does not satisfy 1), there exists an $x \in \mathcal{W}_J$ s.t. $(x, o) \in p^{\mathcal{J}}$ and $x \notin C^{\mathcal{J}}$. On the other hand, since \mathcal{J} satisfies 2), $o \in (\forall p^-.C)^{\mathcal{J}}$ and, since $(o, x) \in (p^-)^{\mathcal{J}}$, then $x \in C^{\mathcal{J}}$, which again yields a contradiction.

Q.E.D

B Nominal-Based Pseudo-Model Merging

Lemma 1 *Let $\mathcal{G}' = (\mathbf{V}', \mathbf{E}', \mathbf{L}', \neq)$ the initial completion graph generated for the input C, \mathcal{O} before the application of any rule. Let $x \in \mathbf{V}'$ be the initial blockable root node in \mathcal{G}' with $C \in \mathcal{L}(x)$ and $o \in \mathbf{V}'$ the nominal node representing the individual o .*

Suppose that in every complete and clash free graph $\mathcal{G} = (\mathbf{V}, \mathbf{E}, \mathbf{L}, \neq)$ for C w.r.t. \mathcal{O} that can be obtained from \mathcal{G}' through the application of the expansion rules the following holds:

- $C \in \mathcal{L}(x)$
- *There exists an edge $\langle x, o \rangle \in \mathbf{E}$ with $p \in \mathbf{L}(\langle x, o \rangle)$.*

Then, if $\mathbf{T} = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ is a tableau for C w.r.t. \mathcal{O} the following holds:

$$\text{If } C \in \mathcal{L}(s) \text{ with } s \in \mathbf{S} \Rightarrow (s, o) \in \mathcal{E}(p)$$

Proof

Let $\mathbf{T} = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ be a tableau for C w.r.t. \mathcal{O} such that there exists an $s \in \mathbf{S}$ with $C \in \mathcal{L}(s)$ and $(s, o) \notin \mathcal{E}(p)$. We can establish a partial mapping:

$\pi : \text{nodes } \mathcal{G}' \Rightarrow \text{elements of } \mathbf{S}$

defined as follows:

$$\pi(x) = s$$

Such a mapping verifies the following properties (denoted by \star):

- $\mathbf{L}(x) \subseteq \mathcal{L}(\pi(x))$
- If y is an R-neighbor of x , then $(\pi(x), \pi(y)) \in \mathcal{E}(R)$
- $x \neq y$ implies $\pi(x) \neq \pi(y)$

It is easy to see that \star holds, since \mathcal{G}' has a single node x , with no edges, no inequalities with other nodes and $\mathbf{L}(x) = \{C\}$ which is necessarily a subset of $\mathcal{L}(s)$, by our assumptions.

By completeness of the *SHOIN* tableau algorithm (see [5]), whenever a rule is applicable to \mathcal{G}' it can be applied in a way that we can define an extension of π that verifies \star . Since the algorithm terminates, any sequence of rule application terminates. When no more rules are applicable, the resulting completion graph \mathcal{G} must contain an edge $\langle x, o \rangle$ with $p \in \mathcal{L}(\langle x, o \rangle)$, by our assumptions. By completeness we have a mapping π' :

$\pi' : \text{nodes } \mathcal{G} \Rightarrow \text{elements of } \mathbf{S}$

that verifies \star . Since o is a p-neighbor of x in \mathcal{G} , $\pi'(x) = s$ and $\pi'(o) = o$, we have, by \star , $(s, o) \in \mathcal{E}(p)$, which is a contradiction.

Q.E.D

Lemma 2 *Suppose that every tableau $\mathbf{T} = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for C w.r.t. \mathcal{O} verifies the following:*

$$\text{If } C \in \mathcal{L}(s) \text{ with } s \in \mathbf{S} \Rightarrow (s, o) \in \mathcal{E}(p)$$

where $o \in \mathbf{S}$ represents a nominal.

Then every model \mathcal{I} of \mathcal{O} satisfies the axiom $C \sqsubseteq \exists p. \{o\}$

Proof Suppose that in every tableau $\mathbf{T} = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for C w.r.t. \mathcal{O} the

following holds:

$$\text{If } C \in \mathcal{L}(s) \text{ with } s \in \mathbf{S} \Rightarrow (s, o) \in \mathcal{E}(p)$$

Suppose there is a model $\mathcal{I} = (\mathcal{W}, \cdot^{\mathcal{I}})$ of \mathcal{O} s.t. \mathcal{I} does not satisfy the axiom $C \sqsubseteq \exists p.\{o\}$.

Since \mathcal{I} is a model of \mathcal{O} , we can build a tableau $\mathbf{T} = (\mathbf{S}, \mathcal{E}, \mathcal{L})$ as follows:

- $\mathbf{S} = \mathcal{W}$
- $\mathcal{E}(R) = R^{\mathcal{I}}$
- $\mathcal{L}(s) = \{D \in \text{clos}(C, \mathcal{O}) \mid s \in D^{\mathcal{I}}\}$

We call such a definition \star .

In [5] it is shown that a tableau defined in such a way is actually a tableau for C w.r.t. \mathcal{O} .

Since \mathcal{I} does not satisfy the axiom $C \sqsubseteq \exists p.\{o\}$, then there is an element $s \in C^{\mathcal{I}}$ s.t. $s \notin (\exists p.\{o\})^{\mathcal{I}}$, i.e. $(s, o) \notin p^{\mathcal{I}}$. But, if $s \in C^{\mathcal{I}}$, then, by \star , $C \in \mathcal{L}(s)$ and by our first assumption, $(s, o) \in \mathcal{E}(p)$. Using \star again, we have that $(s, o) \in p^{\mathcal{I}}$ and hence a contradiction.

Q.E.D

Theorem 1 *Let $\mathcal{G}' = (\mathbf{V}', \mathbf{E}', \mathbf{L}', \neq)$ the initial completion graph generated for the input C, \mathcal{O} before the application of any rule. Let $x \in \mathbf{V}'$ be the initial blockable root node in \mathcal{G}' with $C \in \mathcal{L}(x)$ and $o \in \mathbf{V}'$ the nominal node representing the individual o .*

Suppose that in every complete and clash free graph $\mathcal{G} = (\mathbf{V}, \mathbf{E}, \mathbf{L}, \neq)$ for C w.r.t. \mathcal{O} that can be obtained from \mathcal{G}' through the application of the expansion rules the following holds:

- $C \in \mathcal{L}(x)$
- *There exists an edge $\langle x, o \rangle \in \mathbf{E}$ with $p \in \mathbf{L}(\langle x, o \rangle)$.*

Then $\mathcal{O} \models C \sqsubseteq \exists p.\{o\}$

Proof Immediate by Lemmas 1 and 2. **Q.E.D**

Lemma 3 *Let $\mathcal{O} \models C \sqsubseteq \exists p.\{o\}$. Let $\mathbf{T} = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ be a tableau for C w.r.t. \mathcal{O} . Then:*

1. *If p is a simple role, then the following holds:*

$$\text{if } s \in \mathbf{S} \text{ with } C \in \mathcal{L}(s) \Rightarrow, (s, o) \in \mathcal{E}(p)$$

2. If p is not simple, there exists a role $q \sqsubseteq_{\mathcal{R}}^* p$ with $\text{Trans}(q, \mathcal{R}) = \text{true}$ and a path s_0, \dots, s_k s.t. $k > 1$, $s = s_0$, $t = s_k$ and $(s_i, s_{i+1}) \in \mathcal{E}(q)$ for $0 \leq i < k$

Proof

Suppose that $\mathbf{T} = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ is a tableau for C w.r.t. \mathcal{O} . Then, in [5] it is shown that the interpretation $\mathcal{I} = (\mathcal{W}, \cdot^{\mathcal{I}})$ defined from \mathbf{T} as follows:

- $\mathcal{W} = \mathbf{S}$
- $A^{\mathcal{I}} = \{s \mid A \in \mathcal{L}(s) \text{ for all atomic name } A \text{ occurring in } C \text{ or } \mathcal{O}\}$
- $p^{\mathcal{I}} = \{\mathcal{E}(p)^+ \text{ if } \text{Trans}(p, \mathcal{R}) = \text{true}, \text{ or } \mathcal{E}(p) \cup_{q \sqsubseteq_{\mathcal{R}}^* p} q^{\mathcal{I}} \text{ otherwise}\}$

is a model of \mathcal{O} . moreover, it is shown that:

1. If $D \in \mathcal{L}(s) \Rightarrow s \in D^{\mathcal{I}}$
2. $(s, t) \in p^{\mathcal{I}} \Leftrightarrow (s, t) \in \mathcal{E}(p)$ or there exists a role $q \sqsubseteq_{\mathcal{R}}^* p$ with $\text{Trans}(q, \mathcal{R}) = \text{true}$ and a path s_0, \dots, s_k with $k > 1$, $s = s_0$, $t = s_k$ and $(s_i, s_{i+1}) \in \mathcal{E}(q)$ for $0 \leq i < k$. Moreover, if p is simple, $p^{\mathcal{I}} = \mathcal{E}(p)$

Now, suppose that p is simple, $s \in \mathbf{S}$, $C \in \mathcal{L}(s)$ and $(s, o) \notin \mathcal{E}(p)$ Using 1) and 2) above, we have that $s \in C^{\mathcal{I}}$ and $(s, o) \notin p^{\mathcal{I}}$, which implies that $s \notin (\exists p.\{o\})^{\mathcal{I}}$. Consequently, \mathcal{I} is a model of \mathcal{O} that does not satisfy the axiom $C \sqsubseteq \exists p.\{o\}$, and hence a contradiction.

Suppose that p is not simple and there is no path s_0, \dots, s_k with $k > 1$, $s = s_0$, $t = s_k$ and $(s_i, s_{i+1}) \in \mathcal{E}(q)$ for $0 \leq i < k$ with $q \sqsubseteq_{\mathcal{R}}^* p$ and $\text{Trans}(q, \mathcal{R}) = \text{true}$. If $C \in \mathcal{L}(s)$, then by 1) and 2), we have that $s \in C^{\mathcal{I}}$ and $(s, o) \notin p^{\mathcal{I}}$, which again yields a contradiction.

Q.E.D

Lemma 4 Assume that every tableau $\mathbf{T} = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for C w.r.t. \mathcal{O} verifies the following:

$$\text{if } C \in \mathcal{L}(s) \text{ with } s \in \mathbf{S} \Rightarrow (s, o) \in \mathcal{E}(p)$$

With a an individual name occurring in \mathcal{O} or C .

Let $\mathcal{G} = (\mathbf{V}, \mathbf{E}, \mathbf{L})$ be a clash-free and complete completion graph for C w.r.t. \mathcal{O} and let the node $x \in \mathbf{V}$ be s.t. $C \in \mathbf{L}(x)$.

Then, the nominal node $a \in \mathbf{V}$ is a p -neighbor of x in \mathcal{G}

Proof

We will prove that from \mathcal{G} , which is clash free and complete, it is possible to construct a tableau \mathcal{T} for C w.r.t. \mathcal{O} . The way this is done is identical to the soundness proof for \mathcal{SHOIN} presented in [5].

More precisely, a *path* is a sequence of pairs of blockable nodes of \mathcal{G} of the form $\tilde{p} = (\frac{x_0}{x'_0}, \dots, \frac{x_n}{x'_n})$. For such a path we define $Tail(p) = x_n$ and $Tail'(\tilde{p}) = x'_n$. With $(\tilde{p} | \frac{x_{n+1}}{x'_{n+1}})$ we denote the path $\tilde{p} = (\frac{x_0}{x'_0}, \dots, \frac{x_n}{x'_n}, \frac{x_{n+1}}{x'_{n+1}})$. The set $Paths(\mathcal{G})$ is inductively defined as follows:

- For each blockable node x of \mathcal{G} that is a successor of a nominal node or a root node, $(\frac{x}{x}) \in Paths(\mathcal{G})$, and
- For a path $\tilde{p} \in Paths(\mathcal{G})$ and a blockable node y in \mathcal{G} :
 - If y is a successor of $Tail(\tilde{p})$ and y is not blocked, then $(p | \frac{y}{y}) \in Paths(\mathcal{G})$ and
 - If y is a successor of $Tail(\tilde{p})$ and y is blocked by y' , then $(p | \frac{y'}{y}) \in Paths(\mathcal{G})$

Due to the construction of $Paths(\mathcal{G})$, all nodes occurring in a path are blockable and for $\tilde{p} \in Paths(\mathcal{G})$ with $\tilde{p} = (\tilde{p}' | \frac{x}{x'})$, x is not blocked, x' is blocked iff $x \neq x'$ and x' is never indirectly blocked. Furthermore the blocking condition implies $\mathbf{L}(x) = \mathbf{L}(x')$. We denote by $Nom(\mathcal{G})$ the set of nominal nodes in \mathcal{G} and define a tableau $\mathcal{T} = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ from \mathcal{G} as follows:

- $\mathbf{S} = Nom(\mathcal{G}) \cup Paths(\mathcal{G})$
- $\mathcal{L}(\tilde{p}) = \mathbf{L}(Tail(\tilde{p}))$, if $\tilde{p} \in Paths(\mathcal{G})$ and $\mathbf{L}(\tilde{p})$ if $\tilde{p} \in Nom(\mathcal{G})$
- $\mathcal{E}(R) = \{(\tilde{p}, \tilde{q}) \in Paths(\mathcal{G} \times \mathcal{G}) |$
 $\tilde{q} = (p | \frac{x}{x'})$ and x' is an R-successor of $Tail(\tilde{p})$ or
 $\tilde{p} = (q | \frac{x}{x'})$ and x' is an inv(R)-successor of $Tail(\tilde{q})\} \cup$
 $\{(\tilde{p}, a) \in Paths(\mathcal{G}) \times Nom(\mathcal{G}) | a$ is an R-neighbor of $Tail(\tilde{p})\} \cup$
 $\{(a, \tilde{p}) \in Nom(\mathcal{G}) \times Paths(\mathcal{G}) | \tilde{p}$ is an R-neighbor of $a\} \cup$
 $\{(a, b) \in Nom(\mathcal{G}) \times Nom(\mathcal{G}) | b$ is an R-neighbor of $a\}$

In [5] it is proved that \mathcal{T} constructed this way is a tableau for C w.r.t. \mathcal{O} .

Now, assume that $x \in \mathbf{V}$ be s.t. $C \in \mathbf{L}(x)$ and the nominal node $o \in \mathbf{V}$ is *not* a p-neighbor of x in \mathcal{G} . We show that we then encounter a contradiction.

We have three possibilities:

1. x is not blocked and is not a nominal node in \mathcal{G}

2. x is blocked and is not a nominal node in \mathcal{G}
3. x is a nominal node in \mathcal{G}

Suppose x is not a nominal node in \mathcal{G} , it is not blocked and $C \in \mathbf{L}(x)$. Since x is not a nominal node and it is not blocked then there is a path \tilde{p} in \mathcal{G} s.t. $Tail(\tilde{p}) = Tail'(\tilde{p}) = x$. By construction of \mathcal{T} , $\tilde{p} \in \mathbf{S}$ and $C \in \mathcal{L}(\tilde{p})$. By assumption of the Lemma, $(\tilde{p}, o) \in \mathcal{E}(p)$. However, we also know that a is not a p-neighbor of $x = Tail'(\tilde{p})$ in \mathcal{G} and by construction of \mathcal{T} , $(\tilde{p}, o) \notin \mathcal{E}(p)$ and hence the contradiction.

Suppose x is not a nominal node in \mathcal{G} , it is blocked by y and $C \in \mathbf{L}(x)$. Since x is not a nominal node and it is blocked then there is a path \tilde{p} in \mathcal{G} s.t. $Tail(\tilde{p}) = y$ and $Tail'(\tilde{p}) = x$, with $\mathbf{L}(x) = \mathbf{L}(y)$. By construction of \mathcal{T} , $\tilde{p} \in \mathbf{S}$ and $C \in \mathcal{L}(\tilde{p})$. By assumption of the Lemma, $(\tilde{p}, o) \in \mathcal{E}(p)$. However, we also know that $x = Tail'(\tilde{p})$ is not a p-neighbor of a in \mathcal{G} and by construction of \mathcal{T} , $(\tilde{p}, o) \notin \mathcal{E}(p)$ and hence the contradiction again.

Finally, suppose that x is a nominal node in \mathcal{G} and $C \in \mathbf{L}(x)$. Since x is a nominal node then $x \in \mathbf{S}$ and $C \in \mathcal{L}(x)$, by construction of \mathcal{T} . By assumption of the Lemma, $(x, o) \in \mathcal{E}(p)$. However, we also know that a is not a p-neighbor of x in \mathcal{G} and by construction of \mathcal{T} , $(\tilde{p}, o) \notin \mathcal{E}(p)$ and hence the contradiction again.

Q.E.D

Lemma 5 *Suppose that for every tableau $\mathbf{T} = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for C w.r.t. \mathcal{O} the following condition holds:*

If $C \in \mathcal{L}(s)$, then there exists a role $q \sqsubseteq_{\mathcal{R}}^ p$ with $Trans(q, \mathcal{R}) = true$ and a path s_0, \dots, s_k s.t. $k > 1$, $s = s_0$, $t = s_k$ and $(s_i, s_{i+1}) \in \mathcal{E}(q)$ for $0 \leq i < k$.*

Let $\mathcal{G} = (\mathbf{V}, \mathbf{E}, \mathbf{L})$ be a clash-free and complete completion graph for C w.r.t. \mathcal{O} and let the node $x \in \mathbf{V}$ be a node with $C \in \mathbf{L}(x)$, then there exists a path z_0, \dots, z_k in \mathcal{G} with $k > 1$, $x = z_0$, $o = z_k$ and z_i a q -neighbor of z_{i-1} for $0 \leq i < k$ and $q \sqsubseteq_{\mathcal{R}^} p$.*

Proof

Let x be a node with $C \in \mathbf{L}(x)$, and assume that there is no path z_0, \dots, z_k in \mathcal{G} with $k > 1$, $x = z_0$, $o = z_k$ and z_i a q -neighbor of z_{i-1} for $0 \leq i < k$ and $q \sqsubseteq_{\mathcal{R}^*} p$.

Identically to the proof of Lemma 4, we can construct a tableau $\mathbf{T} = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ from \mathcal{G} . By construction of \mathbf{T} , $C \in \mathcal{L}(\tilde{p})$, where $Tail(\tilde{p}) = x$. We have two possibilities:

- x is not an ancestor of o in \mathcal{G} .
- x is an ancestor of o , but there exists a pair of nodes y_1, y_2 s.t. x is an ancestor of y_1 , y_2 is an ancestor of o and y_2 is a successor of y_1 , but y_2 is not a q -neighbor of y_1

In the first case, we obviously encounter a contradiction, because x and o are not even connected in \mathcal{G} . The second case reduces to the proof of Lemma 4. Let \tilde{p}, \tilde{q} be paths in \mathcal{G} (according to the definition of the set $Paths(\mathcal{G})$ in Lemma 4) with $Tail'(\tilde{p}) = y_1$ and $Tail'(\tilde{q}) = y_2$ then $(\tilde{p}, \tilde{q}) \notin \mathcal{E}(q)$ (note that by construction $\tilde{p}, \tilde{q} \in \mathbf{S}$) and hence we find a contradiction.

Q.E.D

Theorem 2 *Let $\mathcal{O} \models C \sqsubseteq \exists p.\{o\}$ with C satisfiable w.r.t. \mathcal{O} , then in every clash-free and complete graph \mathcal{G} for C w.r.t. \mathcal{O} there must exist a blockable node x with no predecessors (i.e. a root) that verifies the following:*

- *If p is simple then the nominal node o must be a p -neighbor of x in \mathcal{G}*
- *If p is not simple, then there must exist a path z_0, \dots, z_k in \mathcal{G} with $k > 1, x = z_0, o = z_k$ and z_i a q -neighbor of z_{i-1} for $0 \leq i < k$ and $q \sqsubseteq_{\mathcal{R}^*} p$.*

Proof

It is a straightforward consequence of the above lemmas.

Q.E.D