From Wine to Water: Optimizing Description Logic Reasoning for Nominals

Evren Sirin
Maryland Information and Network Dynamics Lab.
8400 Baltimore Av.
College Park, MD, 20740 USA
evren@cs.umd.edu

Bernardo Cuenca Grau *
Information Management Group
School of Computer Science
University of Manchester, UK
bcg@cs.man.ac.uk

Bijan Parsia
Maryland Information and Network Dynamics Lab.
8400 Baltimore Av.
College Park, MD, 20740 USA
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.

Introduction and Motivation

OWL-DL became a World Wide Web Consortium standard for representing ontologies on the Semantic Web in Feburary, 2004. As the W3C Web Ontology working 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 decidabilty as its main justification, and, secondly, the example ontology in the OWL specifications (Smith, Welty, & McGuiness 2004), the Wine Ontology, which tried to exercise every feature of OWL-DL, was not processable by any existing or anticipated reasoner. Of particular concern were the presence of a large number of *nominals*, that is, individuals appearing in concept definitions. To the best of our knowledge, at the time, there were no reasoners that could handle nominals at all, even for the subsets OWL-DL where there were known decision procedures covering nominals. In this paper, we present a suite of optimizations implemented in our OWL-DL reasoner, Pellet (Sirin *et al.* 2005), that suffice to render the Wine ontology (and most current ontologies with nominals) a solved problem. Our experiments show that without such optimizations reasoning with nominals is not practical at all.

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 $\mathcal{SHIN}(\mathcal{D})$ (Horrocks & Sattler 1999) and $\mathcal{SHON}(\mathcal{D})$ (Horrocks & Sattler 2001) 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 (Horrocks & Sattler 2005).

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* (Horrocks 2003). 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 of a decision procedure for $\mathcal{SHOIN}(\mathcal{D})$ poses new challenges:

*This author is is supported by the EU Project TONES (Thinking ONtologiES) ref: IST-007603.

[1] We refer the reader to (Horrocks & Sattler 2005) and (Patel-Schneider, Hayes, & I.Horrocks 2004) for a detailed discussion of $\mathcal{SHOIN}$ and OWL-DL respectively. We also recommend (Horrocks, Patel-Schneider, & van Harmelen 2003) for a thorough discussion on the relationship between OWL and expressive Description Logics.

- 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 "separation" between TBox and ABox that traditionally existed in the implemented 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 the state of the art DL reasoners, with the only exception of the Pellet system[2]. 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 until now.

From a logical point of view, the *nominal constructor* (Horrocks & Sattler 2001) (Schaerf 1994) 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 (Patel-Schneider, Hayes, & I.Horrocks 2004) 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 (Smith, Welty, & McGuiness 2004).

[2]Very recently a new version of FaCT++ reasoner that supports nominals (but not ABoxes) was released.

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

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

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 $\mathcal{SHOIN}(\mathcal{D})$ decision procedure presented in (Horrocks & Sattler 2005) 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.

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.
- *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.

- *Completion Graph Caching* stores the saturated tableaux expansion constructed during the initial KB consistency check and reuses it for subsequent concept satisfiability and subsumption tests .
- *Lazy Completion Graph 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.

Learning-based disjunct selection is completely independent of the DL under consideration and works for any KB that has a large number of instances. The other techniques are only effective in the presence of nominals in the KB. In what follows, we describe these techniques in detail.

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 (Horrocks 2003) 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.

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

$$WineColor \equiv \{red, rose, white\}$$
$$WineColor \sqsubseteq WineDescriptor$$

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.:

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

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:

Proposition 1 *The inclusion axiom (1) is logically equivalent to the set of TBox axioms and ABox assertions in (2)*

$$C \equiv \{a_1, \ldots, a_n\} \tag{1}$$
$$C \sqsubseteq \{a_1, \ldots, a_n\} \textit{ and } C(a_1) \textit{ and } \ldots \textit{ and } C(a_n) \tag{2}$$

This proposition lets us to replace a non-absorbable GCI into one primitive definition and a set of ABox assertions. Note that the set $C(a_1), ..., C(a_n)$ of ABox assertions is equivalent to the GCI $\{a_1, ..., a_n\} \sqsubseteq C$. In our example, the enumeration axiom would be absorbed as follows:

$$WineColor \sqsubseteq \{red, rose, white\}$$
$$WineColor(red); WineColor(rose); WineColor(white)$$

Note that, we still have a disjunction due to the presence of $\{red, rose, white\}$. however, this disjunction will only affect the instances of $WineColor$ concept instead of all the individuals. Thus, the effect of the disjunction is localized to a much smaller number of individuals.

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

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

Considering that there are other inclusion 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 the $Wine$ 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 equivalence:

Proposition 2 *The following two inclusion axioms are logically equivalent:*

$$\exists p.\{o\} \sqsubseteq C \tag{3}$$
$$\{o\} \sqsubseteq \forall p^{-}.C \tag{4}$$

It is very straight-forward to show that the inclusion axiom $\{o\} \sqsubseteq C$ is logically equivalent to the ABox assertion $C(o)$ (see the proof of Proposition 1 in the appendix). Using these equivalences in the previous example would yield the following ABox assertion:

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

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

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

(i) *Initialize* Create a set $G = \{C, \neg D\}$ for the inclusion axiom $C \sqsubseteq D$.

(ii) *Concept absorption* 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) **OneOf absorption** If $C \in G$ where C is an enumeration $\{o_1, \ldots, o_n\}$, add $\neg(\sqcap G)$ to each individual o_i and exit.

(iv) **HasValue absorption** If $C \in G$ where C is in the form $\exists p.\{o_1, \ldots, o_n\}$ then add $\forall p^-.\neg(\sqcap G)$ to each individual o_i and exit.

(v) *Simplification* 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) *Conjunction simplification* If $C \in G$ where C is in the form $(C_1 \sqcap \ldots \sqcap C_n)$, then set G to $G \sqcup \{C_1, \ldots, C_n\}$ and go to (ii).

(vii) *Recursion* If $C \in G$ where C is in the form $(C_1 \sqcup \ldots \sqcup C_n)$, then for every C_i try recursively absorbing $\neg C_i \cup G \setminus \{C\}$, else fail.

Figure 1: Standard absorption algorithm extended with nominal absorption. Our extensions are highlighted in bold font.

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, such as dependency-directed backjumping (Horrocks 2003).

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. Given the following three axioms

$$Process \equiv AtomicProcess \sqcup CompositeProcess \sqcup SimpleProcess$$
$$CompositeProcess \equiv \leq 1.composedOf \sqcap \geq 1.composedOf$$
$$\top \sqsubseteq \forall composedOf.ControlConstruct \sqcap \forall composedOf^-.CompositeProcess$$

the standard preprocessing steps, e.g. normalization and absorption, produce the following axiom[3]:

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

During tableaux expansion, for any *AtomicProcess* instance we will face to expand this disjunction. Obviously, there is only one right selection here ($\leq 0.composedOf$) 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 *≥-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.

```
function expand-disjunction( x, D )

int[] stats = get-statistics( D )
if stats not found then
  stats = new int[n]
  ∀i stats[i] = 0
  save-statistics( D, stats )
Pick the next untried disjunct D_k such that
  stats[k] is minimum
Add D_k to L(x) and continue tableau expansion
if there is a clash then increment stats[k]
```

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

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

Nominal-based Pseudo-model Merging

Classification of named concepts in a KB is one of the most important applications of DL reasoners. Optimization tech-

[3]There is a possibility that absorption algorithm yields different results depending on the order axioms are processed, but the nondeterminism does not have any effect on this specific example

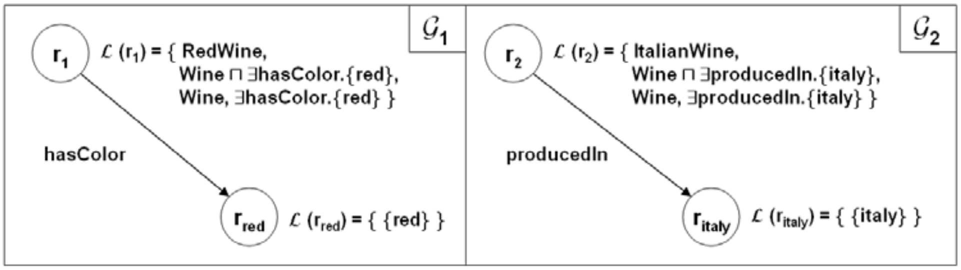

Figure 3: Completion graphs for concepts $\mathit{RedWine}$ and $\mathit{ItalianWine}$

niques 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:

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

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 completed completion graph generated to check the satisfiability of a concept. For example, checking the satisfiability of concept $RedWine$ starts by creating a completion graph that contains a root node r_1 labeled with concept $\mathit{RedWine}$ and one nominal node for each nominal occurring in the ontology. The completion graph $\mathbf{G}_1$ for concept $\mathit{RedWine}$ is schematically shown in Figure 3. The root node r_1 in $\mathbf{G}_1$ is connected to the nominal node r_{red} through a $hasColor$-labeled edge showing that $\mathit{RedWine} \sqsubseteq \exists \mathit{hasColor}.\{\mathit{red}\}$. Now let us consider Italian wines, defined as follows:

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

In the completion graph of $\mathit{ItalianWine}$ (shown as $\mathbf{G}_2$ in Figure 3), the nominal node r_{red} is not a neighbor of the concept node r_2. From this information, it is possible to infer that $\mathcal{O} \not\models \mathit{ItalianWine} \sqsubseteq \exists \mathit{hasColor}.\{\mathit{red}\}$ and thus $\mathcal{O} \not\models \mathit{ItalianWine} \sqsubseteq \mathit{RedWine}$. Note that, for transitive roles, 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:

$$DryWine \equiv Wine \sqcap \exists hasSugar.\{dry\}$$
$$NonSweetWine \equiv Wine \sqcap \exists hasSugar.\{dry, offdry\}$$

We want to test whether $\mathit{DryWine}$ is subsumed by $\mathit{NonSweetWine}$. The graphs $\mathbf{G}_1$ and G_2 in Figure 4 are valid completion graphs for $\mathit{DryWine}$ and $\mathit{NonSweetWine}$ respectively. The root node r_1 for the concept $\mathit{DryWine}$ in $\mathbf{G}_1$ is connected to the nominal node r_{dry} by a $\mathit{hasSugar}$-edge. On the other hand, in $\mathbf{G}_2$, the nominal node r_{dry} is not neighbor of the root node r_2. A naive application of nominal-based pseudo model merging would incorrectly conclude that $DryWine$ is not a subclass of $NonSweetWine$.

In this case, the subsumption holds although the edges to nominal nodes differ. The reason is that there is another valid completion graph ($\mathbf{G}_3$ in Figure 4) for $NonSweetWine$ in which the root node r_3 for concept $NonSweetWine$ *does* have a $\mathit{hasSugar}$-edge leading to the nominal node r_{dry}. Therefore, in order to infer the non-subsumption, the edge to the nominal node should be present in *every* possible completion graph for $NonSweetWine$ or, in other words, the presence of the edge should not depend on a non-deterministic choice in the execution of the tableau algorithm. For this reason, nominal-based pseudo-model merging can be used only in conjunction when dependency sets are stored for each node label and edge label. Since all the existing DL reasoners already make use of the dependency-directed backjumping optimization, this requirement does not cause an extra overhead.

Let us now describe formally how the nominal-based pseudo-model merging technique works: Let $\mathbf{G} = (V, E, \mathcal{L}, \neq)$ be a clash-free completion graph for concept A w.r.t. to an ontology $\mathcal{O}$ and $r_A \in V$ be the root node create for concept A[4] that was initialized with $\mathcal{L}(r_A) = \{A\}$. For each nominal o in $\mathcal{O}$ we are guaranteed to have a nominal node $r_o \in V$ such that $\{o\} \in \mathcal{L}(r_o)$.

Suppose that we want to test whether an ontology $\mathcal{O}$ entails the subsumption relation $D \sqsubseteq C$. Let $\mathbf{G}_C$ (respectively $\mathbf{G}_D$) be a fully expanded and clash-free tableaux expansion representing a common model of C and $\mathcal{O}$ (respectively a common model of D and $\mathcal{O}$). Then we say that $\mathcal{O} \not\models D \sqsubseteq C$ if one of the following two conditions hold:

1. There is a *simple* role p such that:

(a) The nominal node r_o is a p-neighbor of the root node r_C in $\mathbf{G}_C$ and the presence of such an edge does not depend on a non-deterministic choice, and

[4]Note that, there is a possibility that the root node r_A will not exist in the final completion graph $\mathbf{G}_A$ because it was merged into a nominal node and then pruned from the graph. In such cases we cannot apply this technique.

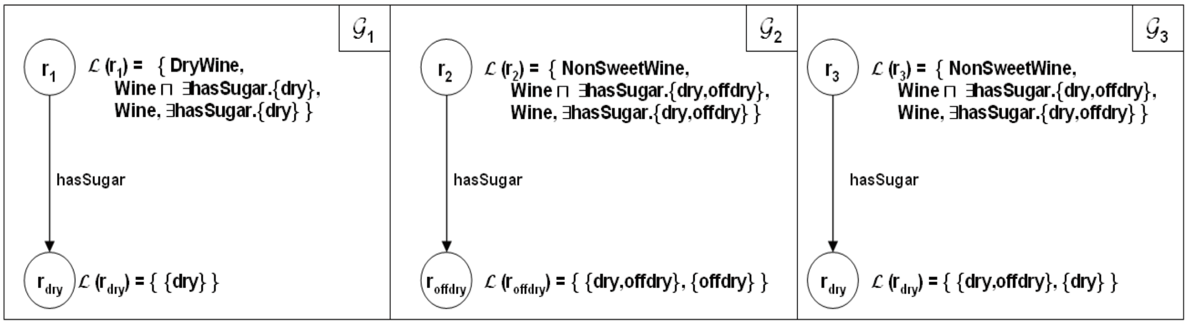

Figure 4: Different completion graphs for concepts $DryWine$ and $NonSweetWine$

(b) The nominal node r_o is not a p-neighbor of r_D in $\mathbf{G}_D$.

2. There is a non-simple role p such that:

(a) There is a path of nodes $z_0, \ldots, z_k$ in $\mathbf{G}_C$ with $k \geq 1, r_C = z_0, r_o = z_k$ and z_i a q-neighbor of z_{i-1} for $0 \leq i < k$ for some q a sub-role of p. Moreover, the presence of such a path does not depend on a non-deterministic choice, and

(b) There is no such path in $\mathbf{G}_D$ (with or without dependencies) from r_D to the nominal node r_o.

Intuitively, conditions (1a) and (2a) imply that the concept C is subsumed by $\exists p.\{o\}$ and conditions (1b) and (2b) imply that that concept D is not subsumed by $\exists p.\{o\}$. The correctness of this technique is proved in the appendix.

Completion Graph Caching

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.

The main idea underlying the completion graph 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 tests. Expanding the nominal nodes from its initialization state may involve the application of a large number of expansion rules. By using cached graph we avoid repeating the process for different concept satisfiability tests, which causes a significant computational overhead.

For the initial KB consistency test, we create all the nominal nodes and apply all the expansion rules. For any subsequent consistency check, we use the already expanded graph as the initial graph so that already applied expansion rules will not be repeated.

One needs to be careful when reusing an earlier completion graph because there might be some edges or node labels dependent on a non-deterministic choice. If there is a clash due to such an edge or a node label, the backtracking must be done accordingly. In order to backtrack correctly, we need to cache not only the nodes and edges, but also the information about dependency sets for node and edge labels plus the history of merge operations so that nodes can be restored after backjumping. Although caching this information affects memory consumption, the overhead is not critical and pays off in terms of significant speed-up in subsequent concept satisfiability and subsumption tests, as will be discussed in emprical results section.

Lazy Completion Graph Generation

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 and the nominals do not influence their satisfiability. For these concepts, generating the nominal nodes corresponding to the ABox individuals results in an unnecessary overhead. Even if we use the cached completion graph for these individuals, maintaining extra nodes (copying, checking if a rule is applicable, etc.) can be costly.

Since the KB is consistent, the rules triggered by the presence of the initial graph of nominal nodes will never yield to a clash in the tableau expansion for A.

Lazy completion graph generation avoids such a computational burden by *not* including the nominal nodes in the initial completion graph when checking concept satisfiability. If the nominal rule is triggered during tableau expansion, then all the nominal nodes are added to the completion graph. This simple technique may yield a dramatic performance improvement, as discussed later on in empirical results section.

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

The reader may have noted that lazy completion graph 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 *subset* of the nominal nodes; second, the generation of the completion graph may not always be required after backjumping. This provides room for further improvements in the near future.

Empirical Results

We have integrated the optimization techniques presented in this paper into the $\mathcal{SHOIN}(\mathcal{D})$ 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 (Smith, Welty, & McGuiness 2004), 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 in 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.

	Wine			**OWL-S**		
Options	Consist.	Classif.	Real.	Consist.	Classif.	Real.
OHDMLC	772.0	16911.4	2154.3	377.6	2422.5	1021.5
_HDMLC	16608.9	N/A	N/A	407.6	2634.7	1141.8
O_DMLC	21748.2	64463.7	61412.4	387.4	2500.7	1062.5
__DMLC	230463.5	N/A	N/A	388.7	2488.4	1083.6
OH_MLC	3184.3	27182.1	35246.7	18006.8	2052.0	1059.5
OHD_LC	766.0	32294.3	9852.3	391.4	2461.7	1089.3
OHDM_C	779.1	20973.1	2155.4	387.3	45669.9	1113.4
OHDM__	793.2	N/A	N/A	389.4	72805.7	1116.6

	AKT Portal			**3SAT**		
Options	Consist.	Classif.	Real.	Consist.	Classif.	Real.
OHDMLC	6.0	399.6	47.0	1651.5	3.0	1.0
_HDMLC	7.0	2647.0	785.1	11478.5	3.0	6498.3
O_DMLC	2.0	374.6	41.1	1542.1	2.0	2.1
__DMLC	6.0	2606.7	786.3	8072.5	1.0	18493.7
OH_MLC	1.0	1607.2	49.1	1471.1	3.0	1.0
OHD_LC	3.0	382.6	43.2	920.5	1.0	0.0
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 absoption on OneOf (O) and hasValue (H), Learning-based Disjunct Selection (D), Nominal-based Pseudo-Model Merging (M), Lazy Completion Graph Generation (L), Completion Graph 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. Realization time shows how long it took to find the most specific type for each individual.

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 nominal absorption Pellet cannot classify the ontology in the specified time limit and consistency time increases by three orders of magnitude.

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 completion graph generation and graph 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 (Masuoka, Parsia, & Labrou 2003) 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 graph generation, graph 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 graph 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*. Since traditionally DL reasoners do not support reasoning with nominals, the *pseudo-nominal* approach tries to approximate the enumerated class definitions by replacing each nominal $\{o\}$ with a fresh atomic concept P_o and adding the assertion $P_o(o)$ to the ABox. Reasoners such as Racer and KAON2 adopt this technique and are not complete w.r.t. nominals.

We have run 10 independent experiments with all the optimizations enabled to classify and realize the modified Wine ontology containing *pseudo-nominals*. We have obtained the following results: 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 can be more costly, especially when nominals are used heavily in the ontology.

Very recently a new version of FaCT++ reasoner supporting nominals was released. FaCT++ version 1.0.0 supports the DL $\mathcal{SHOIQ}$(D). However, this version of FaCT++ does not support ordinary ABox assertions[5] so it was not possible to run some of the above experiments or measure consistency checking and realization times separately. For this reason, we have only tried one experiment: classifying Wine ontology using FaCT++ 1.0.0. We have used a timeout of 30 minutes and classification was not completed in any experiment in the allowed time frame. This result also supports our hypothesis that without specific optimizations, reasoning with nominals is not practical.

We can summarize our results as follows:

1. It is not practical to reason with nominals without having special optimizations, especially when the ontology uses nominals heavily.
2. 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.
3. Learning-based disjunct selection is particularly effective in the presence of individuals with similar characteristics, as shown in the OWL-S case.
4. Nominal-based pseudo-model merging is only useful on ontologies with *hasValue* restrictions [6] and affects primarily classification and realization times.
5. Lazy graph generation and graph caching can have a dramatic influence on concept satisfiability and subsumption tests.
6. The pseudo-nominal approximation is not only unsound, but may actually degrade the reasoner's performance.

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 DL community, we have proved that reasoning with "real" nominals can be 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.

[5]Theoretically, ABox individuals can be encoded as nominals and ABox assertions can be turned into inclusion axioms but such an automated transformation was not available

[6]Wine is the only ontology in our experiments that contains hasValue restrictions

Acknowledgments

This work, conducted at the Maryland Information and Network Dynamics Laboratory Semantic Web Agents Project, was funded in part by Fujitsu Laboratories of America – College Park, Lockheed Martin Advanced Technology Laboratory, NTT Corp., Kevric Corp., SAIC, the National Science Foundation (NSF), the National Geospatial-Intelligence Agency, Northrop Grumman Electronic Systems, Defense Advanced Research Projects Agency (DARPA), US Army Research Laboratory, the National Institute of Standards and Technology (NIST), and other DoD sources.

References

Horrocks, I., and Sattler, U. 1999. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation* 9(3):385–410.

Horrocks, I., and Sattler, U. 2001. Ontology reasoning in the $\mathcal{SHOQ}(\mathcal{D})$ description logic. In Nebel, B., ed., *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, 199–204. Morgan Kaufmann.

Horrocks, I., and Sattler, U. 2005. A tableaux decision procedure for SHOIQ. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*. Morgan Kaufman.

Horrocks, I.; Patel-Schneider, P. F.; and van Harmelen, F. 2003. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics* 1(1):7–26.

Horrocks, I. 2003. Implementation and optimisation techniques. In Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press. 306–346.

Masuoka, R.; Parsia, B.; and Labrou, Y. 2003. Task computing - the semantic web meets pervasive computing -. In *Proceedings of 2nd International Semantic Web Conference (ISWC2003)*.

Patel-Schneider, P.; Hayes, P.; and I.Horrocks. 2004. Web ontology language OWL Abstract Syntax and Semantics. *W3C Recommendation*.

Schaerf, A. 1994. Reasoning with individuals in concept languages. *Data and Knowledge Engineering* 13(2):141–176.

Sirin, E.; Parsia, B.; Grau, B. C.; Kalyanpur, A.; and Katz, Y. 2005. Pellet: A practical owl-dl reasoner. Technical report, University of Maryland Institute for Advanced Computes Studies (UMIACS), 2005-68. Available online at http://www.mindswap.org/papers/PelletDemo.pdf.

Smith, M.; Welty, C.; and McGuiness, D. 2004. OWL Web Ontology Language Guide. *W3C Recommendation*.

Appendix: Proofs

Nominal Absorption

Proposition 1 *The inclusion axiom (1) is logically equivalent to the set of axiom and assertions in (2)*

$$C \equiv \{a_1, \dots, a_n\} \tag{1}$$

$$C \sqsubseteq \{a_1, \dots, a_n\} \textit{ and } C(a_1) \textit{ and } \dots \textit{ and } C(a_n) \tag{2}$$

Proof The axiom (1) is equivalent to the combination of following two axioms

$$C \sqsubseteq \{a_1, \dots, a_n\} \tag{3}$$

$$\{a_1, \dots, a_n\} \sqsubseteq C \tag{4}$$

By the definition of enumerations, axiom (4) is equivalent to:

$$\{a_1\} \sqcup \dots \sqcup \{a_n\} \sqsubseteq C \tag{5}$$

We can rewrite axiom (5) as the following n separate axioms:

$$\{a_1\} \sqsubseteq C \text{ and } \dots \text{ and } \{a_n\} \sqsubseteq C \tag{6}$$

which is obviously valid based on the semantics

$$(\{a_1\} \sqcup \dots \sqcup \{a_n\})^{\mathcal{I}} \subseteq C^{\mathcal{I}} \iff \{a_1\}^{\mathcal{I}} \subseteq C^{\mathcal{I}} \text{ and } \dots \text{ and } \{a_n\}^{\mathcal{I}} \subseteq C^{\mathcal{I}}$$

Axiom (6) is equivalent to the following set of assertions:

$$C(a_1) \text{ and } \dots \text{ and } C(a_n) \tag{7}$$

because for each i we have

$$\{a_i\}^{\mathcal{I}} \subseteq C^{\mathcal{I}} \iff (a_i)^{\mathcal{I}} \in C^{\mathcal{I}}$$

since $\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$.

Thus, we have shown that axiom (1) is transformed into the combination of (3) and (7) which is equivalent to (2). □

Proposition 2 *The following two inclusion axioms are logically equivalent:*

$$\exists p.\{o\} \sqsubseteq C \tag{8}$$

$$\{o\} \sqsubseteq \forall p^{-}.C \tag{9}$$

Proof Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a model of (8) s.t. it does not satisfy (9). Since $\mathcal{I}$ does not satisfy (9), then $o^{\mathcal{I}} \notin (\forall p^{-}.C)^{\mathcal{I}}$ which implies that $o^{\mathcal{I}} \in (\exists p^{-}.\neg C)^{\mathcal{I}}$. Thus, there exists an object $x \in \Delta^{\mathcal{I}}$ s.t. $(x, o^{\mathcal{I}}) \in p^{\mathcal{I}}$ and $x \in (\neg C)^{\mathcal{I}}$. On the other hand, since $\mathcal{I}$satisfies (8) and $x \in (\exists p.\{o\})^{\mathcal{I}}$, then $x \in C^{\mathcal{I}}$, which yields a contradiction.

Let $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ be a model of (9) s.t. it does not satisfy (8). Since $\mathcal{J}$ does not satisfy (8), there exists an $x \in \Delta^{\mathcal{J}}$ s.t. $(x, o^{\mathcal{J}}) \in p^{\mathcal{J}}$ and $x \notin C^{\mathcal{J}}$. On the other hand, since $\mathcal{J}$ satisfies (9), $o^{\mathcal{J}} \in (\forall p^{-}.C)^{\mathcal{J}}$ and, since $(o^{\mathcal{J}}, x) \in (p^{-})^{\mathcal{J}}$, then $x \in C^{\mathcal{J}}$, which again yields a contradiction. □

Nominal-Based Pseudo-Model Merging

Theorem 1 *Let $\mathbf{G}' = (V', E', \mathcal{L}', \neq)$ be the initial completion graph for the concept C w.r.t the ontology $\mathcal{O}$ such that $V' = \{r_C, r_{o_1}, \dots, r_{o_m}\}$ where r_C is the root node for concept C and r_{o_i} is the nominal node corresponding to nominal o_i. $\mathcal{L}'$ is initialized such that $\mathcal{L}(r_C) = \{C\}$ and $\mathcal{L}(r_{o_i}) = \{o_i\}$ for $1 \le i \le m$.*

Let $\mathcal{G}$ be the set of all complete and clash free graphs for C w.r.t. $\mathcal{O}$ that can be obtained from $\mathbf{G}'$ through the application of the expansion rules. If there is a role p s.t. for every $\mathbf{G} = (V, E, \mathcal{L}, \neq)$ in $\mathcal{G}$ there exists an edge $\langle r_C, r_o\rangle \in E$ with $p \in \mathcal{L}(\langle r_C, r_o\rangle)$, then, $\mathcal{O} \models C \sqsubseteq \exists p.\{o\}$.

Proof Let us assume that $\mathcal{O} \not\models C \sqsubseteq \exists p.\{o\}$. This means there should be an interpretation where there is an element that belongs to both concept C and $\forall p.\neg\{o\}$ (which is the negation normal form of $\neg(\exists p.\{o\})$. Then we should be able to build a clash free and complete completion graph starting with the initial graph $\mathbf{G}'' = (V'', E'', \mathcal{L}'', \neq)$ where $\mathcal{L}(r'') = \{C, \forall p.\neg\{o\}\}$. Since the graph $\mathbf{G}''$ is same as $\mathbf{G}'$ with one additional element in $\mathcal{L}(r)$, all the tableau rules applicable to $\mathbf{G}'$ will still be applicable to $\mathbf{G}''$. This means, every possible application of tableau expansion rules to $\mathbf{G}''$ will yield a member of $\mathcal{G}$ (with the additional element $\forall p.\neg\{o\}$ in $\mathcal{L}''(x)$). Then, by the assumption of the lemma, we know that $p \in \mathcal{L}''(\langle r, r_o\rangle)$ would hold. Therefore, the application of the $\forall$-rule would create a clash in $\mathbf{G}''$ since it would add $\neg\{o\}$ to the label of r_o node. Hence we conclude no such clash free completion graph exists and $\mathcal{O} \models C \sqsubseteq \exists p.\{o\}$. □

Lemma 1 *Let $\mathcal{O} \models C \sqsubseteq \exists p.\{o\}$. Let $\mathbf{T} = (\mathbf{S}, \mathbf{L}, \mathbf{E})$ be a tableau for C w.r.t. $\mathcal{O}$. Then:*

1. *If p is a simple role, then, for any $s \in \mathbf{S}$ with $C \in \mathbf{L}(s)$ we have $\langle s, o\rangle \in \mathbf{E}(p)$*
2. *If p is not simple, there exists a role $q \sqsubseteq^*_{\mathcal{R}} p$, $Trans(q) = true$ and a path $s_0, \dots, s_k$ s.t. $k \ge 1$, $s = s_0$, $o = s_k$ and $(s_i, s_{i+1}) \in \mathbf{E}(q)$ for $0 \le i < k$*

Proof In (Horrocks & Sattler 2005) it is shown that the interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ defined from $\mathbf{T}$ as follows:

- $\Delta^{\mathcal{I}} = \mathbf{S}$
- $A^{\mathcal{I}} = \{s \mid A \in \mathcal{L}(s)$ for all atomic concepts A occurring in C or $\mathcal{O}\}$
- $p^{\mathcal{I}} = \begin{cases} \mathbf{E}(p)^{+} & \text{if Trans(p) = true} \\ \mathbf{E}(p) \cup \bigcup_{\forall q, q \sqsubseteq^*_{\mathcal{R}} p} q^{\mathcal{I}} & \text{otherwise} \end{cases}$

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

1. If $D \in \mathbf{L}(s)$ then $s \in D^{\mathcal{I}}$
2. $\langle s, t\rangle \in \mathbf{E}(p)$ iff $\langle s, t\rangle \in p^{\mathcal{I}}$ or there exists a role $q \sqsubseteq^*_{\mathcal{R}} p$ with $Trans(q) = true$ and a path $s_0, \dots, s_k$ with $k \ge 1, s = s_0, t = s_k$ and $\langle s_i, s_{i+1}\rangle \in \mathbf{E}(q)$ for $0 \le i < k$. Moreover, if p is simple, $p^{\mathcal{I}} = \mathbf{E}(p)$

Now, suppose that p is simple, $s \in \mathbf{S}, C \in \mathbf{L}(s)$ and $\langle s, o\rangle \notin \mathbf{E}(p)$ Using (1) and (2) above, we have that $s \in C^{\mathcal{I}}$ and $\langle s, o\rangle \notin p^{\mathcal{I}}$, 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 $k \ge 1, s = s_0, o = s_k$ and $\langle s_i, s_{i+1}\rangle \in \mathbf{E}(q)$ for $0 \le i < k$ with $q \sqsubseteq^*_{\mathcal{R}} p$ and $Trans(q) = true$. If $C \in \mathbf{L}(s)$, then by (1) and (2), we have that $s \in C^{\mathcal{I}}$ and $\langle s, o\rangle \notin p^{\mathcal{I}}$, which again yields a contradiction. □

Lemma 2 *Assume that there is a* simple *role p s.t. in every tableau $\mathbf{T} = (\mathbf{S}, \mathbf{L}, \mathbf{E})$ for C w.r.t. $\mathcal{O}$ if $C \in \mathbf{L}(s)$ for $s \in \mathbf{S}$ then $\langle s, o\rangle \in \mathbf{E}(p)$ where o is a nominal occurring in $\mathcal{O}$.*

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

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

Proof We will prove that from $\mathbf{G}$, which is clash free and complete, it is possible to construct a tableau $\mathbf{T}$ for C w.r.t. $\mathcal{O}$. The way this is done is identical to the soundness proof for $\mathcal{SHOIN}$ presented in (Horrocks & Sattler 2005).

More precisely, a *path* is a sequence of pairs of blockable nodes of $\mathbf{G}$ of the form $\tilde{p} = (\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n})$. 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 $\tilde{p} = (\frac{x_0}{x'_0}, ..., \frac{x_n}{x'_n}, \frac{x_{n+1}}{x'_{n+1}})$. The set $Paths(\mathbf{G})$ is inductively defined as follows:

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

Due to the construction of $Paths(\mathbf{G})$, all nodes occurring in a path are blockable and for $\tilde{p} \in Paths(\mathbf{G})$ 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 $\mathcal{L}(x) = \mathcal{L}(x')$. We denote by $Nom(\mathbf{G})$ the set of nominal nodes in $\mathbf{G}$ and define a tableau $\mathbf{T} = (\mathbf{S}, \mathbf{L}, \mathbf{E})$ from $\mathbf{G}$ as follows:

- $\mathbf{S} = Nom(\mathbf{G}) \cup Paths(\mathbf{G})$
- $\mathbf{L}(\tilde{p}) = \begin{cases} \mathcal{L}(Tail(\tilde{p})) & \text{if } \tilde{p} \in Paths(\mathbf{G}) \\ \mathcal{L}(\tilde{p}) & \text{if } \tilde{p} \in Nom(\mathbf{G}) \end{cases}$
- $\mathbf{E}(R) = \{\langle \tilde{p}, \tilde{q} \rangle \in Paths(\mathbf{G} \times \mathbf{G}) \mid$
  $\tilde{q} = (p|\frac{x}{x'})$ 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 $Tail(\tilde{q})\} \cup$
  $\{\langle \tilde{p}, a \rangle \in Paths(\mathbf{G}) \times Nom(\mathbf{G}) |\ a$ is an R-neighbor of $Tail(\tilde{p})\} \cup$
  $\{\langle a, \tilde{p} \rangle \in Nom(\mathbf{G}) \times Paths(\mathbf{G}) |\ \tilde{p}$ is an R-neighbor of $a\} \cup$
  $\{\langle a, b \rangle \in Nom(\mathbf{G}) \times Nom(\mathbf{G}) \mid b$ is an R-neighbor of $a\}$

In (Horrocks & Sattler 2005) it is proved that $\mathbf{T}$ constructed this way is a tableau for C w.r.t. $\mathcal{O}$.

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

There are three possibilities:

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

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

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

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

Lemma 3 *Assume that there is a* non-simple *role p s.t. in every tableau $\mathbf{T} = (\mathbf{S}, \mathbf{L}, \mathbf{E})$ for C w.r.t. $\mathcal{O}$ if $C \in \mathbf{L}(s)$, then there exists a role $q \sqsubseteq^*_{\mathcal{R}} p$ with $Trans(q) = true$ and a path $s_0, \ldots, s_k$ s.t. $k \geq 1$, $s = s_0$, $t = s_k$ and $(s_i, s_{i+1}) \in \mathbf{E}(q)$ for $0 \leq i < k$.*

Let $\mathbf{G} = (V, E, \mathcal{L}, \neq)$ be a clash-free and complete completion graph for C w.r.t. $\mathcal{O}$ and let the node $x \in V$ be a node with $C \in \mathcal{L}(x)$, then there exists a path $z_0, \ldots, z_k$ in $\mathbf{G}$ with $k \geq 1, x = z_0, o = z_k$ 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 $C \in \mathcal{L}(x)$, and assume that there is no path $z_0, \ldots, z_k$ in $\mathbf{G}$ with $k \geq 1, x = z_0, o = z_k$ 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 2, we can construct a tableau $\mathbf{T} = (\mathbf{S}, \mathbf{L}, \mathbf{E})$ from $\mathbf{G}$. 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 $\mathbf{G}$.
- 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 $\mathbf{G}$. The second case reduces to the proof of Lemma 4. Let $\tilde{p}$, $\tilde{q}$ be paths in $\mathbf{G}$ (according to the definition of the set $Paths(\mathbf{G})$ in Lemma 4) with $Tail'(\tilde{p}) = y_1$ and $Tail'(\tilde{p}) = y_2$ then $(\tilde{p}, \tilde{q}) \notin \mathbf{E}(q)$ (note that by construction $\tilde{p}, \tilde{q} \in \mathbf{S}$) and hence we find a contradiction. ☐

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 $\mathbf{G}$ 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 $\mathbf{G}$*
- *If p is not simple, then there must exist a path $z_0, \ldots, z_k$ in $\mathbf{G}$ with $k \geq 1, x = z_0, o = z_k$ 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. ☐