UAF Model Verification Method based on Description Logic

Qian Ding¹²*, Yueli Wang³ and Guanping Cao⁴

1Institute of Military Law Studies, Academy of Military Sciences, Beijing, China
2Vocational Education Center, Special Police College, Beijing, China
3 Research Center for Assessment and Demonstration, Academy of Military Sciences, Beijing, China
4 Postgraduate School, Academy of Military Sciences, Beijing, China

*Corresponding author e-mail: dingx931@sina.com

Abstract. The grammatical structure of model based on Unified Architecture Framework (UAF) adopts formal specification, but the lack of formal and accurate description of semantics may result in information inconsistency between different views. In this paper, an UAF model verification method based on Descriptive Logic SHIQ is proposed. The knowledge base of SHIQ is constructed in formal description, with the help of Racerpro reasoning engine based on Tableau algorithm, conflicting or contradictory information in the models is reasoned and judged. The feasibility and convenience of this method are proved by verifying the "Requirement" model of the Major Disaster Relief System.

1. Introduction

Unified Architecture Framework (UAF) is an architectural standard issued by Object Management Group (OMG) in November 2017[1][4]. It can be compatible with Department of Defense Architecture Framework (DoDAF), Ministry of Defense Architecture Framework (MODAF), NATO Architecture Framework (NAF), Department of National Defense Architecture Framework (DNDAF) and other architecture frameworks. The views based on UAF covers strategic objectives, business support and technical realization, involving requirement generation, capability building, business process, organization management, resource allocation, security protection, etc. Architectural work is a complex System of Systems Engineering, there may has some information overlaps among the models of different viewpoints. UAF model is constructed by SysML, which grammatical structure adopts formal specification, but its semantics is described by natural language, lacking formal and accurate description. System Architecture, MagicDraw, IBM Rational Rhapsody and other commercialized architecture design tools supporting SysML can only provide verification at the grammatical level[5][6], but without precise semantics. Traditional verification methods are mostly based on tuples, sets or matrices[7], the complexity of algorithms increases exponentially while the scale of architecture expands, which can’t meet the requirements of consistency verification of large and complex enterprise architecture models.

This paper Proposes a method of UAF model verification based on Description Logic SHIQ. Firstly, the Domain Meta-Model (DMM) of UAF is extended adaptively, then the Extended Domain Meta-Models (DMM+) and models are formally described to SHIQ knowledge base. Then, the semantic reasoning based on SHIQ is carried out by RacerPro v2.0 following Tableau algorithm, and the ontology query language nRQL is used to construct verification rules and return the reasoning results to
architects. Finally, the validity and convenience of this verification method are proved by verifying the "Requirement" model of Major Disaster Relief System (MDRS).

2. UAF

UAF is driven by requirement throughout the whole life cycle, supported by DMM and use UML/SysML to carry out top-down object-oriented design, taking into account the elements of system, personnel, processes and resources, as well as their interrelationship. UAF grid supporting enterprise architecture modeling is as shown in Figure 1. It adopts the object-oriented method and divides the enterprise into 13 domains, including Metadata, Strategic, Operational, Services, Personnel, Resources, Security, Projects, Standards, Actual Resources, Dictionary, Summary and Overview, Requirements, etc. Horizontally, each domain can be described in details from static structure, dynamic interaction and supporting standard, including 11 viewpoints e.g. Taxonomy, Structure, Connectivity, Processes, States, Interaction Scenarios, Information, Parameters, Constraints, Roadmap, traceability, etc. Each view of UAF grid has one or a group of DMM to support and standardize the corresponding modeling activities.

![Unified Architecture Framework (UAF) grid.](image1)

**Figure 1.** Unified Architecture Framework (UAF) grid.

UAF provides a more detailed, reasonable and logical object-oriented decomposition grid, and about 80 DMMs for standardizing modeling activities. Compared with 12 groups Meta-Models of DoDAF, UAF has greatly improved both in quantity and completeness. The DMM of "Requirement" is as shown in Figure 2, describes the elements and their roles should be considered when constructing the requirement model with UAF.

!["Requirement" Domain Meta-Model.](image2)

**Figure 2.** "Requirement" Domain Meta-Model.
3. Descriptive Logic SHIQ

In order to make up for the defects of basic logic description language ALC in expressing knowledge in complex domains, a more expressive logic description language ALCHIQ+, abbreviated as SHIQ\(^8\), has been formed by adding characteristics of transitive role (S)\(^9\), inverse role (I)\(^10\), role hierarchy (H)\(^11\) and qualified quantitative constraint (Q) to ALC. SHIQ is a decidable subset of Descriptive Logic, has stronger expressive ability in conceptual hierarchy, transfer and quantitative description. Many automatic reasoning engines, such as Racer, Pellet and Jena, have been introduced to support automatic logical reasoning based on SHIQ.

3.1. Grammar and Semantics

Description logic has specific constructors, semantics and grammar. For example, the constructor of “conjunction” is “C \(\sqcap\) D”, The corresponding semantics is “\(C \cap D\)”, e.g. "Human \(\sqcap\) Male".

\(I=(\Delta,\bullet)\) is a semantic interpretation, consisted by a set of non-empty domains \(\Delta\) and an interpretation function \(\bullet\). \(\bullet(A)\) Maps each concept \(A\) to subset \(A\) of \(\Delta\), \(\bullet(R)\) Maps each role \(R\) to subset \(\Delta\) of \(\Delta\). The constructors, grammar and semantics of SHIQ are listed in Table 1.

| Name               | Constructor | Grammar | Semantics              |
|--------------------|-------------|---------|------------------------|
| Atomic Concept     | A           | A       | \(A \subseteq \Delta\) |
| Role               | R           | R       | \(R \subseteq \Delta\) |
| Conjunctions       | \(\sqcap\)  | \(C \sqcap D\) | \(C \cap D\)          |
| Disjunctive        | \(\sqcup\)  | \(C \sqcup D\) | \(C \cup D\)          |
| Negative           | \(\neg\)    | \(\neg C\) | \(\Delta \setminus C\) |
| Existence Constraint | \(\exists\) | \(\exists \ R.C\) | \{\(x | \exists y, <x, y> \in R \wedge y \in C\}\} |
| Global Constraint  | \(\forall\) | \(\forall \ R.C\) | \{\(x | \forall y, <x, y> \in R \Rightarrow y \in C\}\} |
| Full Concept       | T           | T       | \(\Delta\)             |
| Empty Concept      | \(\perp\)   | \(\perp\) | \(\Phi\)               |
| Transitive         | +           | R +     | \{\(x, y \in R | \exists z, <y, z> \in C \Rightarrow <x, z> \in R\}\} |
| Hierarchy          | \(\sqsubseteq\) | \(R_1 \sqsubseteq R_2\) | \(R_1 \subseteq R_2\) |
| Inverse            | -           | \(R^-\) | \{\(y, x \in R^- | y \in C\}\} |
| Quantitative       | \(\geq\)    | \(\geq n \ R \cdot C\) | \{\(x | |\{y | <x, y> \in R, y \in C\}| \geq n\}\} |
|                    | \(\leq\)    | \(\leq n \ R \cdot C\) | \{\(x | |\{y | <x, y> \in R, y \in C\}| \leq n\}\} |

3.2. SHIQ Knowledge Base

SHIQ Knowledge Base consists of Tbox and Abox, \(\Sigma=\langle Tbox, Abox \rangle\).

Tbox represents the set of axiomatic assertions and connotation knowledge, Abox represents the instances and extension knowledge.

3.2.1. Tbox. Tbox is used to describe semantics: If \(C \sqsubseteq D\), then Interpretation I satisfies the assertion \(C \sqsubseteq D\); If \(C = D\), then Interpretation I satisfies the assertion \(C = D\); Interpretation I=(\(\Delta, \bullet\)) Satisfies all assertions in Tbox.
3.2.2. Abox. Abox is used to describe facts: If $a^I \in C^I$, $C(a)$ is satisfied; If $(a^I, b^I) \in R^I$, $R(a, b)$ is satisfied; Then the assertion set A is Abox; an interpretation $I = (\Delta^I, \cdot^I)$ satisfies the role hierarchy $H$, if and only if all $R_1^I \sqsubseteq R_2^I$ has $R_1^I \subseteq R_2^I$, then I is a model of $H$, marked as $I = H$; an interpretation $I = (\Delta^I, \cdot^I)$ satisfies the terminology assertion $Tbox$, if and only if all $R_1^I \sqsubseteq R_2^I$ has $R_1^I \subseteq R_2^I$, then I is a model of $Tbox$, marked as $I = Tbox$; an interpretation $I = (\Delta^I, \cdot^I)$ satisfies the instance assertion $Abox$, if and only if (1) Conceptual assertions $C(a)$ has $a^I \in C^I$, (2) Role assertions $R(a, b)$ has $(a^I, b^I) \in R^I$, (3) Equivalent assertion $a \neq b$, has $a^I \neq b^I$. Then I is a model of Abox, marked as $I = Abox$.

3.2.3. Knowledge Base Reasoning. The reasoning of descriptive logic SHIQ is based on Tableau algorithm[12], which is a decision-making process to solve satisfiability problems. It calculates the satisfiability of concept $C$ by constructing a Tableau tree, transforms $C$ into Negation Normal Form (NNF) by De Morgan Rule, initializes the Tableau tree which contains only one root node, then expands the Tableau tree repeatedly according to the extension rules until the branches are contradictory or irregular. The extended rules for Tableau algorithm to describe SHIQ are shown in Table 2.

| Rule | Content of rule |
|------|-----------------|
| $\sqcap$ - Rule | If (1) $C_1 \sqcap C_2 \in L(x)$, and x isn’t blocked directly (2) $\{C_1, C_2\} \not\subseteq L(x)$ Then $L(x) \rightarrow L(x) \cup \{C_1, C_2\}$ |
| $\sqcup$ - Rule | If (1) $C_1 \sqcup C_2 \in L(x)$, and x isn’t blocked directly (2) $\{C_1, C_2\} \cap L(x) = \Phi$ Then $L(x) \rightarrow L(x) \cup \{C_1\}$ or $L(x) \rightarrow L(x) \cup \{C_2\}$ |
| $\exists$ - Rule | If (1) $\exists S.C \in L(x)$, and x isn’t blocked directly (2) for x, there is no S-adjacent y, make $C \in L(y)$ valid then add a Node, assign $L(<x, y>) = S$ and $L(y) = \{C\}$ |
| $\forall$ - Rule | If (1) $\forall S.C \in L(x)$, and x isn’t blocked directly (2) for x, there is a S-adjacency, but $C \not\subseteq L(y)$ then $L(y) \rightarrow L(y) \cup \{C\}$ |
| $\forall^-$ - Rule | If (1) $\forall S.C \in L(x)$, and x isn’t blocked directly (2) for x, there is a S-adjacency, but $\forall S.C \not\in L(x)$ Then $L(y) \rightarrow L(y) \cup \{\forall S.C\}$ |
| Choice-Rule | If (1) $(\geq n S.C) \in L(x)$, and x isn’t blocked directly (2) for x, there is a S-adjacent y, make $\{C, \neg C\} \cap L(y) = \Phi$ valid then $L(y) \rightarrow L(y) \cup \{C\}$ or $L(y) \rightarrow L(y) \cup \{-C\}$ |
| $\geq$ - Rule | If (1) $\geq n S.C \in L(x)$, and x isn’t blocked directly (2) $x$ does not exist S-adjacent $y_1, \ldots, y_n$, make $C \in y_i$ valid, and $y_i \neq y_j$, for all $1 \leq i \neq j \leq n$ correct then add N nodes $y_1, \ldots, y_n$, assign $L(<x, y_i>) = \{S\}$, and $L(y_i) = C$, in which $i \neq y_j, 1 \leq i \leq n$ |
| $\leq$ - Rule | If (1) $\leq n S.C \in L(x)$, and x isn’t blocked directly (2) there is $\{u|u$ is a S-neighbor of x, and $C \in L(u)\} > n$, exist two S-neighbors of x are $y, z$, satisfy $C \in L(y), C \in L(z)$, and y isn’t the pioneer of z, and there isn’t $y \neq z$ then (1) $L(z) \rightarrow L(z) \cup \{y\}$ (2) if x is the pioneer of y, then $L(<x, y>) \rightarrow L(<x, z>) \cup \text{Inv}(L(<x, y>))$ otherwise $L(<x, z>) \rightarrow L(<x, z>) \cup \text{Inv}(L(<x, y>)) = \Phi$ (3) $L(<x, y>) \rightarrow \Phi$ (4) for all of u, if $u \neq y$, then $u \neq z$ |

Tableau algorithm has been widely used in the verification of concepts and roles in various Description Logic. Many optimized Tableau algorithms have been embedded in automatic reasoning machines such as Racer, Pellet, FaCT and so on. Racerpro is selected in this paper, its query and inference functions include:

- By inclusive reasoning, analysis concept hierarchies of Tbox
- By instance reasoning, descript concepts of Abox and divide them into specific sets.
- By auxiliary reasoning, retrieve kinds of concepts, instances and roles.
4. UAF Model Verification Method

The verification method of UAF model based on SHIQ mainly includes the following aspects:

4.1. Extend Domain Meta-Model

According to the actual characteristics of the modeling object, the DMM should be extended adaptively to deepen the understanding at the Meta-Model level, to improve the quality of the models.

The Extended Domain Meta-Model (DMM+) of "Requirement" domain of MDRS is shown in Figure 3. Compared with the other Architecture Frameworks, UAF adds "Requirement" domain, which fully considers the assumptions, constraints, standards, specifications, principles and organizational guidelines of Requirement. In the DMM+ of "Requirement" domain, top-level requirements are decomposed into detailed sub-requirements, which are grounded into what capabilities an enterprise should possess, what business activities should be carried out, and what personnel and resources should be allocated, including functional requirements, non-functional requirements, business requirements and technical requirements.

![Figure 3. Extended Domain Meta-Model of "Requirement" domain.](image)

4.2. Formally description

The DMM+ and UAF model should be transformed into Descriptive Logic knowledge base \(<\text{Tbox},\text{Abox}>\) by model transform algorithm. The main process of the algorithm is as shown in figure 4:

**Algorithm Description** DMM+ and UAF model are transformed into SHIQ knowledge base

**Input** DMM+, UAF model

**Output** Descriptive Logic knowledge base \(<\text{Tbox},\text{Abox}>\)

**Step1** Transform DMM+ into Tbox

**Step2** Transform UAF model into Abox

End
4.2.1. Verify whether there is cyclic inheritance in DMM+

To verify the cyclic inheritance, in addition to using the basis semantic "role hierarchy" of SHIQ to create assertion "C₂ ⊑ C₁" in Tbox, also need to create a role "Sub-concept" of C₁ and C₂: Sub-concept(C₁,C₂). The corresponding nRQL query statement is: (retrieve ( ) (and (?C₁ ?C₂ Sub-concept) (?C₂ ?C₁ Sub-concept))).

If the returned query result is "NIL", there is no cyclic inheritance between parent concept C₁ and child concept C₂, the model is correct. If the result is "T", there is cyclic inheritance and the model is incorrect.

4.2.2. Verify the consistence of concept

All instances in the UAF model must belong to axiomatic assertion declared in Tbox. The corresponding nRQL query statement is: (retrieve ( ) (c top)).

If the returned query result is "NIL", instance "c" has no corresponding concept "C" in the Tbox, the model is incorrect. If the result is "T", "c" is valid and the model is correct.

4.2.3. Verify the consistence of role

If there is a kind of role between the elements in DMM+, the corresponding instances in the model must also have the same role. For example, if "c" and "d" are instances of "C" and "D" in Tbox, and there is role "R" between C and D, then "c" and "d" should also have role "R". The corresponding nRQL query statement is: (retrieve (( ) (c d R))).

If the returned query result is "NIL", there is no "R" between "c" and "d", the model is incorrect. If the result is "T", there is "R" between "c" and "d", the model is correct.

4.2.4. Verify the integrity of the model

If a certain kind of element exists in the DMM+, there must has a corresponding instance. That is for the concepts in Tbox, there must be corresponding instances in Abox, and for the roles, there must be corresponding instances in Abox that have such roles, otherwise the model is incomplete.

The nRQL query statement for integrity of concept is as follows: (retrieve ( ) (?x C)). If the returned query result is "NIL", there is no instance of "C", the model is incorrect; the results is "T", the model is correct. If the query is changed to: (retrieve (?x ) (?x C)), the names of all instances are returned if there has.

The nRQL query statement for integrity of role is as follows: (retrieve ( ) (?x ?y R)). If the returned query result is "NIL", there is no instance of "R", the model is incorrect; the result is "T", the model is correct. If the query is changed to: (retrieve (?x ?y) (?x ?y R)), the names of all instances are returned if there has.

If there exists an instance "c" of role R(C,D) in the model, verify whether there has the instance of "D", the corresponding nRQL query statement is: (retrieve ( ) (and (c C) (?y D) (c ?y R))). If the returned query result is "NIL", there is no instance of "D" and the model is incorrect; the result is "T", the model is correct. If the query is changed to: (retrieve (?y ) (and (c C) (?y D) (c ?y R))), the names of all instances are returned if there has.
4.2.5. Verify the validity of roles

If there is no role between two elements in DMM+, there is no such role between the corresponding instances in the model. The corresponding nRQL query statement is: (retrieve ( ) (and (c C) (d D) (c d R))). If the returned query result is "NIL", there is no role "R" between instance "c" and "d" and the model is correct; the result is "T", the model is incorrect.

Above are some general verification rules, we also could define inference query rules flexibly according to requirements of specific application. For example, if capability c₁ depends on cₙ, and the realization of c₁ and cₙ is supported by operations o₁ and oₙ, so oₙ take precedence over o₁, for this particular requirement, define the nRQL query statement: (retrieve (?o₁ ?oₙ) (and (?c₁ ?cₙ DependOn) (?o₁ ?c₁ support) (?oₙ ?cₙ support) (?o₁ ?oₙ Precedent)))

5. Instance

This paper takes the "requirements" model of MDRS as an example (Parts of the model is as shown in figure 5), to verify the validity and feasibility of "UAF Model Verification Method based on Description Logic" from the aspects of concept and role integrity.

![Figure 5. Parts of the "requirements" model of MDRS.](image)

### 5.1. Generate Knowledge Base

Generate knowledge base <Tbox,Abox> of "requirements" model of MDRS according to Racerpro syntax.

According to the "requirements" DMM+, generate Tbox of MDRS. The Concepts include: Requirement, Sub-Requirement, Personnel, Organization, Post, Responsibility, Person, Operation, Resource, Information Resource, etc. The roles include: Depend, Verify, Satisfy, Refine, Trace, Inherit, etc. the hierarchy of concepts and roles are shown in figure 6 and figure 7.
Parts of the RacerPro code is shown as follows:

\[
\text{Tbox concept hierarchies}
\]

\[
\text{(implies (or Personnel Requirement Sub-Requirement Capability Goal Operation Resource Evolution AnalysisMethod Service TechnicalStandard PoliticalRegulation Rule) *top*)}
\]

\[
\text{(implies (or Person Post Organization Responsibility Personnel Requirement Sub-Requirement Capability MissionGoal TaskGoal SoftGoal Goal Operation InformationResource NaturalResource SoftwareandHardwareResource Resource Roadmap Optimization Evolution AnalysisMethod Service TechnicalStandard PoliticalRegulation Rule) *top*)}
\]

\[
\text{roles(Depend Verify Satisfy Refine Trace Inherit) )}
\]

\[
\text{Tbox concept hierarchies}
\]

\[
\text{(implies (or Person Post Organization Responsibility Personnel Requirement Sub-Requirement Capability MissionGoal TaskGoal SoftGoal Goal Operation InformationResource NaturalResource SoftwareandHardwareResource Resource Roadmap Optimization Evolution AnalysisMethod Service TechnicalStandard PoliticalRegulation Rule) *top*)}
\]

\[
\text{(implies (or Person Post Organization Responsibility Personnel Requirement Sub-Requirement Capability MissionGoal TaskGoal SoftGoal Goal Operation InformationResource NaturalResource SoftwareandHardwareResource Resource Roadmap Optimization Evolution AnalysisMethod Service TechnicalStandard PoliticalRegulation Rule) *top*)}
\]

\[
\text{related Capability Sub-Requirement Trace)
\]

\[
\text{related Post Sub-Requirement Satisfy)
\]

\[
\text{related Person Sub-Requirement Satisfy )
\]

\[
\text{related Operation Sub-Requirement Refine)
\]
(transitive Satisfy)
(transitive Trace)
(transitive Inherit)

;;; Tbox R(C,D)
(implies Capability(all Trace Sub-Requirement))
(implies Capability(at-least Trace Sub-Requirement))
(implies Sub-Requirement(all (inv Trace) Capability))
(implies Sub-Requirement(exactly 1 (inv Trace) Capability))

;;; Abox instance assertion
(instance InternationalDisasterReliefLaw Requirement)
(instance RenderAssistance Sub-Requirement)
(instance ProvideTreatment Capability)
(instance Communicator Post)

(related ProvideTreatment RenderAssistance Trace)
(related Communicator RenderAssistance Satisfy)
(related RescueVictims RenderAssistance Satisfy)

5.2. Generate Knowledge Base
Based on the "requirements" knowledge base of MDRS, use nRQL query language and Racer reasoning engine to verify the consistency of the model.

5.2.1. Concept Verification
For a concept in Tbox, there must be an instance in Abox. The query statement to verify the existence of instance of "Requirement" is: (retrieve () (?x Requirement)).

The result returned is "T" is shown in figure 8, so the concept is integrated.

![Figure 8. Result of concept integrity.](image)

When the "Requirement" element in the model is deleted as shown in figure 9, the query result is "NIL" as shown in figure 10, which means the concept non-integrity of the changed model is verified.

![Figure 9. The concept changed "requirements" model of MDRS.](image)
5.2.2. Role Verification

The query statement to verify whether there is “Capability” trace to "Render assistant" is as follows:
(retrieve ( ), and(?x Capability) (Renderassistant Sub-Requirement) (?x renderassistant Trace)).

The result of role Verification "Requirements" model is shown in figure 11. As the result is "T" so
the role in the model is integrated.

Figure 10. Result of changed concept integrity.

Deletes "Trace" between capabilities and sub-requirements as shown in figure 12, the result of the
integrity verification of the changed model is shown in figure 13, which means the role is incomplete.

Figure 11. Result of role integrity.

Figure 12. The role changed "requirements" model of MDRS.
6. Conclusion

In order to solve the problems of information omission, redundancy and contradiction among UAF models, this paper proposes an UAF model verification method based on SHIQ. The semi-formal model is formally expressed, convert into a knowledge base, and the automatic reasoning engine based on Tableau algorithm is adopted to verify the UAF model based on precise semantics. This method will has better performance in larger scale model verification.

References

[1] Object Management Group. Unified Architecture Framework Profile (UAFP) Version 1.0[S]. http://www.omg.org/spec/UAF/20170516/Class-Library-UAF.xmi, 2017.

[2] Object Management Group. Unified Architecture Framework (UAF) The Domain Metamodel Version 1.0[S]. http://www.omg.org/spec/UAF/20170515/UAFP_Profile.xmi, 2017.

[3] Object Management Group. Unified Architecture Framework (UAF) Traceability between Framework Views and Elements Version 1.0[S]. http://www.omg.org/spec/UAF/20170516/Class-Library-UAF.xmi, 2017.

[4] Object Management Group. Unified Architecture Framework (UAF) Sample Problem[S]. http://www.omg.org/spec/UAF/20170516/Class-Library-UAF.xmi, 2017.

[5] Telelogic Incorporation. Telelogic System Architect for DoDAF(ABM)[EB/OL].[2008-8-20]. http://www.telelogic.com/product/sys2temarchitect/system2architect2support2for2dodaf2abm.cfm.

[6] Metastorm Incorporation. Metastorm provion, enterprise architecture[EB/OL].[2008-8-20]. http://www.metastorm.com/products/mpea.asp.

[7] Huang L, Chen H H, Luo X S. Consistent Maintaining Method of Products Applied in Design Tool for C4ISR Architecture[J]. SystemsEngineering and Electronics, 2010,32(3):540-543.

[8] Horrocks I, Sattler U. A Description Logic with Transitive and Converse Roles, Role Hierarchies and Qualifying Number Restriction[EB/OIL]. (2012-07-21). http://dl.acm.org/citation.cfm?id=891255.

[9] Sattler U. A Concept Language Extend with Different Kinds of Transitive Roles[EB/OIL]. (2012-07-21). http://lat.inf.tudresden.de/reseurch/papers/1996/Sattler-KI-96.ps.gz.

[10] Horrocks I, Sattler U. A Description Logic with Transitive and Inverse Roles and Role Hierarchies[J]. Logic and Computation, 1999,9(3):385-410.

[11] Horrocks I, Gough G. Description Logics with Transitive Roles[EB/OIL]. (2012-07-21). http://dl.kr.org/dl97/proceeding/horrocks.ps.gz.

[12] Baader F, Nutt W. Handbook of Description Logic, the second chapter. Cambridge University Press, 2013 :472-494.