Reasoning on Context-Dependent Domain Models

Stephan Böhme[1⋆] and Thomas Kühn[2⋆]

[1] Institute for Theoretical Computer Science, Technische Universität Dresden,
stephan.boehme@posteo.de
[2] Software Technology Group, Technische Universität Dresden,
thomas.kuehn3@tu-dresden.de

Abstract Modelling context-dependent domains is hard, as capturing multiple context-dependent concepts and constraints easily leads to inconsistent models or unintended restrictions. However, current semantic technologies not yet support reasoning on context-dependent domains. To remedy this, we introduced ConDL, a set of novel description logics tailored to reason on contextual knowledge, as well as JConHT, a dedicated reasoner for ConDL ontologies. ConDL enables reasoning on the consistency and satisfiability of context-dependent domain models, e.g., Compartment Role Object Models (CROM). We evaluate the suitability and efficiency of our approach by reasoning on a modelled banking application and measuring the performance on randomly generated models.

1 Introduction

Modelling current information systems is hard, as they are characterised by increased context-dependence. They not only require domain analysts to capture multiple context-dependent concepts, but also to specify the particular constraints and requirements found in each context. The latter, however, can easily lead to an inconsistent model or unintended restrictions. Thus, it becomes imperative for domain analysts to reason on context-dependent domain models to uncover implicit or unsatisfiable specifications. While reasoning on classical domain modelling languages, e.g., ER [9] and UML [21], is possible, as [4,25,1] have shown, both lack the formal semantics required to make them suitable for formal reasoning. More importantly, classical domain modelling languages are unable to capture context-dependent concepts and constraints, hence, researchers have focused on more advanced modelling languages, e.g., [13,10,18,12,14] (see [16] for detailed surveys). We focus on the *Compartment Role Object Model* (CROM) [17] that directly supports the formal specification of context-dependent domains. Although CROM provides a formal model, due to a lack of tool support this model is not amenable for reasoning. Thus, we aim at transforming a CROM to a viable logical formalism. *Description Logics* (DLs) are a well-known family of knowledge representation formalisms that have a formal semantics and allow for defining a variety of reasoning services. DLs can model application domains

⋆ Both authors were supported by the DFG in the RTG 1907 (RoSI).

in a well-structured way. Yet, classical DLs lack expressive means to formalise context-dependent domains, i.e., express context dependent knowledge. To overcome their deficits, *Contextualised DLs* (ConDLs) were introduced in [6], a set of novel description logics especially tailored for reasoning on contextual knowledge. This paper utilizes ConDL to reason on CROMs. In particular, we describe a mapping from CROM to ConDL that preserves the semantics of the modelled domain. Moreover, we introduce the first reasoner dedicated to contextualised DL ontologies, JConHT. JConHT can check the consistency and satisfiability of context-dependent domain models. As a result, we show that ConDL is not only suitable to encode CROM domain models, but also allows for efficient checking of inconsistencies and unintended restrictions. We demonstrate this by reasoning on a modelled banking application and measuring the reasoner's performance on randomly generated models.

The paper is structured accordingly. Sect. 2 introduces a running example and formal definition of both CROM and ConDL. Afterwards, Sect. 3 presents our approach to reasoning on CROMs utilizing ConDL. Accordingly, Sect. 4 introduces JConHT, the first reasoner for ConDLs. To evaluate our approach, Sect. 5 showcases its suitability and measures its performance. In conclusion, Sect. 6 discusses related approaches and Sect. 7 summarises our results.

2 Basic Notions

Running Example. Before diving into the definitions of CROM and ConDL, we model a small banking application, extracted from [17]. Fig. 1 depicts an example model of a *Bank* that employs at least one *Consultant* and provides banking services to *Customers*, who own *CheckingAccounts* and *SavingsAccounts*. They can *issue* money transferals (denoted *MoneyTransfer*), such that each transferal belongs to one customer and a customer issued an initial transferal. *Consultants advise* one or more customers. However, the advises relationship is constrained to be *irreflexive*, to prohibit self advising consultants. Besides that, *Transactions* are specified to orchestrate the transfer of money between exactly two *Accounts* by means of the roles *Source* and *Target*, such that there is a unique *Target* counterpart for each *Source*. This is ensured by the one-to-one cardinality of the *trans* relation. Additionally, the *Participants* role group with 1..1 cardinality enforces that one account cannot be *Source* and *Target* in the same *Transaction*. Finally, *Persons* can play the roles *Consultant* and *Customer*; *Companies* only *Customer*. Similarly, *Accounts* either the role of *CheckingAccount* or *SavingsAccount* in the context of a bank, as well as *Source* and *Target* in the context of transactions. Henceforth, this domain model will serve as our running example.

CROM in a Nutshell. The *Compartment Role Object Model* (CROM) was introduced in [17] to model dynamic, context-dependent domains. It introduces *compartment types* to represent a reified context, i.e., containing *role types* and *relationship types*. *Natural types*, in turn, fulfil role types in multiple compartment types.[3]

[3] A detailed ontological foundation of these kinds is provided in [17].

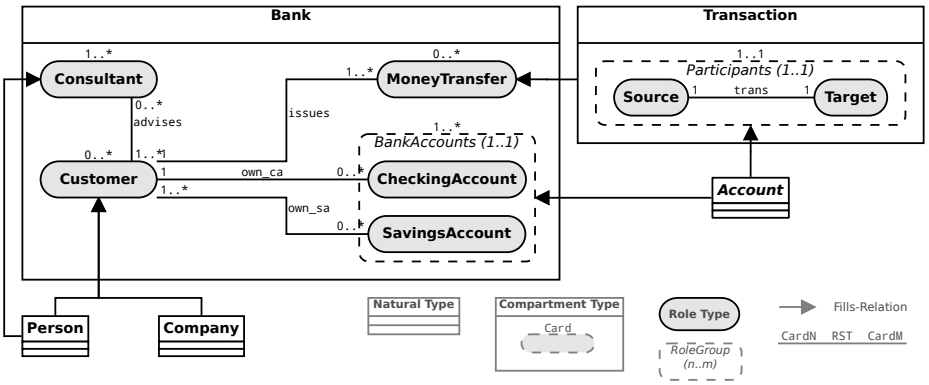


Figure 1. Bank example

Definition 1 (Compartment Role Object Model). *Let* $\mathsf{N_{NT}}$, $\mathsf{N_{RT}}$, $\mathsf{N_{CT}}$, *and* $\mathsf{N_{RST}}$ *be mutual disjoint sets of Natural Types, Role Types, Compartment Types, and Relationship Types, respectively. Then,* $\mathcal{M} = (\mathsf{N_{NT}}, \mathsf{N_{RT}}, \mathsf{N_{CT}}, \mathsf{N_{RST}}, \mathsf{fills}, \mathsf{parts}, \mathsf{rel})$ *is a* Compartment Role Object Model (CROM)*, where* $\mathsf{fills} \subseteq (\mathsf{N_{NT}} \cup \mathsf{N_{CT}}) \times \mathsf{N_{RT}}$ *is a relation,* $\mathsf{parts} : \mathsf{N_{CT}} \to \mathcal{P}(\mathsf{N_{RT}})$ *and* $\mathsf{rel} : \mathsf{N_{RST}} \to (\mathsf{N_{RT}} \times \mathsf{N_{RT}})$ *are total functions. A CROM is* well-formed *if it holds that:*

$$\forall RT \in \mathsf{N_{RT}}\ \exists T \in (\mathsf{N_{NT}} \cup \mathsf{N_{CT}}) : (T, RT) \in \mathsf{fills} \tag{1}$$

$$\forall CT \in \mathsf{N_{CT}} : \mathsf{parts}(CT) \neq \emptyset \tag{2}$$

$$\forall RT \in \mathsf{N_{RT}}\ \exists! CT \in \mathsf{N_{CT}} : RT \in \mathsf{parts}(CT) \tag{3}$$

$$\forall RST \in \mathsf{N_{RST}} : \mathsf{rel}(RST) = (RT_1, RT_2)\ \wedge\ RT_1 \neq RT_2 \tag{4}$$

$$\forall RST \in \mathsf{N_{RST}}\ \exists CT \in \mathsf{N_{CT}} : \mathsf{rel}(RST) = (RT_1, RT_2)\ \wedge\ RT_1, RT_2 \in \mathsf{parts}(CT) \tag{5}$$

In detail, fills denotes that rigid types can play roles of a certain role type, parts is a partition of the set of role types wrt. the compartment type they participate in, and rel captures the two role types at the respective ends of each relationship type. The well-formedness rules ensure that the fills relation is surjective (1); each compartment type has a nonempty, disjoint set of role types as its parts (2, 3); and rel maps each relationship type to exactly two distinct role types of the same compartment type (4, 5). Accordingly, a CROM can be constructed for the banking application, depicted in Fig. 1

Example 2 (Compartment Role Object Model). Let $\mathcal{B} = (\mathsf{N_{NT}}, \mathsf{N_{RT}}, \mathsf{N_{CT}}, \mathsf{N_{RST}}, \mathsf{fills}, \mathsf{parts}, \mathsf{rel})$ be the model of the bank, where the components are defined as:[4]

$$\mathsf{N_{NT}} := \{\mathsf{Person}, \mathsf{Company}, \mathsf{Account}\} \qquad \mathsf{N_{RT}} := \{\mathsf{Customer}, \mathsf{CA}, \mathsf{SA}, \mathsf{Source}, \dots\}$$

$$\mathsf{N_{CT}} := \{\mathsf{Bank}, \mathsf{Transaction}\} \qquad \mathsf{N_{RST}} := \{\mathsf{own\_ca}, \mathsf{own\_sa}, \mathsf{advises}, \mathsf{issues}, \mathsf{trans}\}$$

$$\mathsf{fills} := \{(\mathsf{Person}, \mathsf{Customer}), (\mathsf{Account}, \mathsf{Source}), (\mathsf{Transaction}, \mathsf{MoneyTransfer}), \dots\}$$

$$\mathsf{parts} := \{\mathsf{Bank} \to \{\mathsf{Consultant}, \mathsf{Customer}, \mathsf{CA}, \mathsf{SA}, \mathsf{MoneyTransfer}\}, \dots\}$$

$$\mathsf{rel} := \{\mathsf{trans} \to (\mathsf{Source}, \mathsf{Target}), \mathsf{own\_ca} \to (\mathsf{Customer}, \mathsf{CA}), \dots\}$$

[4] SA and CA are abbreviations for *SavingsAccount* and *CheckingAccount*, respectively.

Unsurprisingly, $\mathcal{B}$ is a well-formed CROM and directly encodes the context-dependent concepts of the banking domain. Likewise, a CROM instance features naturals, roles, compartments and relationships.

Definition 3 (Compartment Role Object Instance). *Let* $\mathcal{M} = (\mathsf{N_{NT}}, \mathsf{N_{RT}}, \mathsf{N_{CT}}, RST, \mathsf{fills}, \mathsf{parts}, \mathsf{rel})$ *be a well-formed CROM and* N*,* R*, and* C *be mutual disjoint sets of Naturals, Roles and Compartments, respectively. Then a* Compartment Role Object Instance (CROI) *of* $\mathcal{M}$ *is a tuple* $\mathfrak{i} = (N, R, C, \mathsf{type}, \mathsf{plays}, \mathsf{links})$*, where* $\mathsf{type} : (N \to \mathsf{N_{NT}}) \cup (R \to \mathsf{N_{RT}}) \cup (C \to \mathsf{N_{CT}})$ *is a labeling function,* $\mathsf{plays} \subseteq (N \cup C) \times C \times R$ *a relation, and* $\mathsf{links} : \mathsf{N_{RST}} \times C \to \mathcal{P}(R \times R)$ *is a total function. Moreover,* $O \coloneqq N \cup C$ *denotes the set of all objects in* $\mathfrak{i}$*. To be* compliant *to the model* $\mathcal{M}$ *the instance* $\mathfrak{i}$ *must satisfy the following conditions:*

$$\forall (o,c,r) \in \mathsf{plays} : (\mathsf{type}(o), \mathsf{type}(r)) \in \mathsf{fills} \wedge \mathsf{type}(r) \in \mathsf{parts}(\mathsf{type}(c)) \tag{6}$$

$$\forall (o,c,r), (o,c,r') \in \mathsf{plays} : r \neq r' \Rightarrow \mathsf{type}(r) \neq \mathsf{type}(r') \tag{7}$$

$$\forall r \in R \; \exists! o \in O \; \exists! c \in C : (o,c,r) \in \mathsf{plays} \tag{8}$$

$$\forall RST \in \mathsf{N_{RST}} \; \forall c \in C \; \forall (r_1, r_2) \in \mathsf{links}(RST, c) : (\_, c, r_1), (\_, c, r_2) \in \mathsf{plays} \tag{9}$$

$$\forall RST \in \mathsf{N_{RST}} \; \forall c \in C \; \forall (r_1, r_2) \in \mathsf{links}(RST, c) : \mathsf{rel}(RST) = (\mathsf{type}(r_1), \mathsf{type}(r_2)) \tag{10}$$

The type function assigns a distinct type to each instance, plays identifies the objects (either natural or compartment) playing a certain role in a specific compartment, and links captures the roles currently linked by a relationship type in a certain compartment. A compliant CROI guarantees the consistency of both the plays relation and the links function with the model $\mathcal{M}$.[5] Axiom (6), (7) and (8) restrict the plays relation, such that it is consistent to the types defined in the fills relation and the parts function, an object is prohibited to play instances of the same role type multiple times in the same compartment, and each role has one distinct player in one distinct compartment, respectively. In contrast, Axiom (9) and (10) ensure that the links function only contains those roles, which participate in the same compartment c as the relationship and whose types are consistent to the relationship's definition in the rel function.

Admittedly, neither Def. 1 nor 3 captures the context-dependent constraints, showcased in Fig. 1. Hence, we introduce three context-dependent constraints, i.e., *role groups*, *occurrence constraints* and *relationship cardinalities*.

Definition 4 (Syntax of Role Groups). *The* set of Role Groups $\mathbb{RG}$ *is the smallest set, such that (i) every role type* $RT \in \mathsf{N_{RT}}$ *is a role group, and (ii) if* B *is a role group and* $m..n \in \mathsf{Card}$*, then* $(B, n..m)$ *is also a role group, where* $\mathsf{Card} \subset \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$ *with* $i \leq j$ *(elements are written as* $i..j$*). A role group contained in another one is denoted* nested*.*

Definition 5 (Semantics of Role Groups). *Let* $\mathfrak{i} = (N, R, C, \mathsf{type}, \mathsf{plays}, \mathsf{links})$ *a CROI compliant to* $\mathcal{M}$*,* $c \in C$ *a compartment, and* $o \in O$ *an object. The* semantics *is defined by the evaluation function* $(\cdot)^{\mathcal{I}_o^c} : \mathbb{RG} \to \{0,1\}$*:* $a^{\mathcal{I}_o^c} = 1$ *iff* $a \in \mathsf{N_{RT}} \wedge \exists (o,c,r) \in \mathsf{plays}$*:* $type(r) = a$ *or* $a \equiv (B, n..m) \wedge n \leq \sum_{b \in B} b^{\mathcal{I}_o^c} \leq m$*.*

[5] In contrast to [17], our definition excludes empty counter roles ε.

Role groups constrain the set of roles an object o is allowed to play simultaneously in a certain compartment c. In case a is a role type, $rt^{\mathcal{I}_o^c}$ checks whether o plays a role of type rt in c. If a is a role group $(B, n..m)$, it checks whether the sum of the evaluations for all $b \in B$ is between n and m. Accordingly, the following role groups directly correspond to their graphical representation in Fig. 1:

$$\mathsf{BankAccounts} \coloneqq (\{\mathsf{CA}, \mathsf{SA}\}, 1..1) \qquad \mathsf{Participants} \coloneqq (\{\mathsf{Source}, \mathsf{Target}\}, 1..1)$$

Next, the *Constraint Model* is defined to collect all constraints imposed on a particular CROM $\mathcal{M}$.

Definition 6 (Constraint Model). *Let* $\mathcal{M} = (\mathsf{N_{NT}}, \mathsf{N_{RT}}, \mathsf{N_{CT}}, \mathsf{N_{RST}}, \mathsf{fills}, \mathsf{parts}, \mathsf{rel})$ *be a well-formed CROM. Then* $\mathcal{C} = (\mathsf{rolec}, \mathsf{card})$ *is a* Constraint Model over $\mathcal{M}$*, where* $\mathsf{rolec} : \mathsf{N_{CT}} \to \mathcal{P}(\mathsf{Card} \times \mathbb{RG})$ *and* $\mathsf{card} : \mathsf{N_{RST}} \to (\mathsf{Card} \times \mathsf{Card})$ *are total functions.*

In detail, rolec collects the set of root role groups for each compartment type combined with a cardinality limiting the occurrence of role groups in each compartment. Moreover, card assigns a cardinality to each relationship type. Notably, all these constraints are defined context-dependent, i.e., no constraint crosses the boundary of a compartment type. Similar to the CROM $\mathcal{B}$, the corresponding constraint model is easily derived, from Fig. 1:

Example 7 (Constraint Model). Let $\mathcal{B}$ be the bank model from Example 2. Then $\mathcal{C}_\mathcal{B} = (\mathsf{rolec}, \mathsf{card})$ is the constraint model with the following components:

$$\begin{aligned}\mathsf{rolec} \coloneqq \{&\mathsf{Bank} \to \{(1..\infty, \mathsf{Consultant}), (1..\infty, \mathsf{BankAccounts})\},\\ &\mathsf{Transaction} \to \{(1..1, \mathsf{Participants})\}\}\\ \mathsf{card} \coloneqq \{&\mathsf{own\_ca} \to (1..1, 0..\infty), \mathsf{own\_sa} \to (1..\infty, 0..\infty), \mathsf{issues} \to (1..1, 1..\infty),\\ &\mathsf{advises} \to (0..\infty, 1..\infty), \mathsf{trans} \to (1..1, 1..1)\}\end{aligned}$$

Finally, the validity of a given CROI is defined wrt. a constraint model.

Definition 8 (Validity). *Let* $\mathcal{M} = (\mathsf{N_{NT}}, \mathsf{N_{RT}}, \mathsf{N_{CT}}, \mathsf{N_{RST}}, \mathsf{fills}, \mathsf{parts}, \mathsf{rel})$ *be a well-formed CROM,* $\mathcal{C} = (\mathsf{rolec}, \mathsf{card})$ *a constraint model on* $\mathcal{M}$*, and* $\mathfrak{i} = (N, R, C, \mathsf{type}, \mathsf{plays}, \mathsf{links})$ *a CROI compliant to* $\mathcal{M}$*. Then* $\mathfrak{i}$ is valid with respect to $\mathcal{C}$ *iff the following conditions hold:*

$$\forall c \in C\ \forall (i..j, a) \in \mathsf{rolec}(\mathsf{type}(CT)) : i \leq \left(\sum\nolimits_{o \in O^c} a^{\mathcal{I}_o^c}\right) \leq j \tag{11}$$

$$\forall (o, c, r) \in \mathsf{plays}\ \forall (\_, a) \in \mathsf{rolec}(\mathsf{type}(c)) : \mathsf{type}(r) \in \mathsf{atoms}(a) \Rightarrow a^{\mathcal{I}_o^c} = 1 \tag{12}$$

$$\begin{aligned}\forall c \in C\ \forall RST \in \mathsf{N_{RST}} : \mathsf{rel}(RST) = (RT_1, RT_2) \wedge \mathsf{card}(RST) = (i..j, k..l) \wedge\\ \left(\forall r_2 \in R^c_{RT_2} : i \leq \left|\mathsf{pred}(RST, c, r_2)\right| \leq j\right) \wedge\\ \left(\forall r_1 \in R^c_{RT_1} : k \leq \left|\mathsf{succ}(RST, c, r_1)\right| \leq l\right)\end{aligned} \tag{13}$$

Here, $\mathsf{atoms} : \mathbb{RG} \to \mathcal{P}(\mathsf{N_{RT}})$ *recursively computes all role types within a given role group. Moreover,* $R^c_{RT} \coloneqq \{r \in R \mid (o, c, r) \in \mathsf{plays} \wedge \mathsf{type}(r) = RT\}$ *denotes the set of roles of type RT played in a compartment c. Furthermore,* $\mathsf{pred}(RST, c, r) \coloneqq \{r' \mid (r', r) \in \mathsf{links}(RST, c)\}$ *and* $\mathsf{succ}(RST, c, r) \coloneqq \{r' \mid (r, r') \in \mathsf{links}(RST, c)\}$ *collects all predecessors respectively successors of a given role r w.r.t. a given RST.*

Table 1. Syntax and Semantics of $\mathcal{SHOIQ}[\![\mathcal{SHOIQ}]\!]$

	syntax	semantics	
inverse object property	R^-	$\{(e,d) \in \Delta \times \Delta \mid (d,e) \in R^{\mathcal{I}_c}\}$	
object negation	$\neg C$	$\Delta \setminus C^{\mathcal{I}_c}$	
object conjunction	$C \sqcap D$	$C^{\mathcal{I}_c} \cap D^{\mathcal{I}_c}$	
obj. existential restriction	$\exists R.C$	$\{d \in \Delta \mid$ there is some $e \in C^{\mathcal{I}_c}$ with $(d,e) \in R^{\mathcal{I}_c}\}$	
object nominal	$\{a\}$	$\{a^{\mathcal{I}_c}\}$	
object at-most restriction	$\leqslant_n S.C$	$\{d \in \Delta \mid \sharp\{e \in C^{\mathcal{I}_c} \mid (d,e) \in S^{\mathcal{I}_c}\} \leq n\}$	
object concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}_c} \subseteq D^{\mathcal{I}_c}$	α
object concept assertion	$C(a)$	$a^{\mathcal{I}_c} \in C^{\mathcal{I}_c}$	α
object property assertion	$R(a,b)$	$(a^{\mathcal{I}_c}, b^{\mathcal{I}_c}) \in R^{\mathcal{I}_c}$	α
object property inclusion	$R \sqsubseteq S$	$R^{\mathcal{I}_c} \subseteq S^{\mathcal{I}_c}$	$\mathcal{R}_{\mathsf{O}}$
object transitivity axiom	$\mathsf{Trans}(R)$	$R^{\mathcal{I}_c}$ is transitive.	$\mathcal{R}_{\mathsf{O}}$
inverse meta property	P^-	$\{(e,d) \in \mathbb{C} \times \mathbb{C} \mid (d,e) \in P^{\mathcal{J}}\}$	
meta negation	$\neg E$	$\mathbb{C} \setminus E^{\mathcal{J}}$	
meta conjunction	$E \sqcap F$	$E^{\mathcal{J}} \cap F^{\mathcal{J}}$	
meta existential restriction	$\exists P.E$	$\{d \in \mathbb{C} \mid$ there is some $e \in E^{\mathcal{J}}$ with $(d,e) \in P^{\mathcal{J}}\}$	
meta nominal	$\{u\}$	$\{u^{\mathcal{J}}\}$	
meta at-most restriction	$\leqslant_n Q.E$	$\{d \in \mathbb{C} \mid \sharp\{e \in E^{\mathcal{J}} \mid (d,e) \in Q^{\mathcal{J}}\} \leq n\}$	
referring concept	$[\![\alpha]\!]$	$\{d \in \mathbb{C} \mid \mathcal{I}_d \models \alpha\}$	
meta concept inclusion	$E \sqsubseteq F$	$E^{\mathcal{J}} \subseteq F^{\mathcal{J}}$	$\mathcal{B}$
meta concept assertion	$E(u)$	$u^{\mathcal{J}} \in E^{\mathcal{J}}$	$\mathcal{B}$
meta property assertion	$T(u,v)$	$(u^{\mathcal{J}}, v^{\mathcal{J}}) \in T^{\mathcal{J}}$	$\mathcal{B}$
meta property inclusion	$R \sqsubseteq S$	$R^{\mathcal{I}_c} \subseteq S^{\mathcal{I}_c}$	$\mathcal{R}_{\mathsf{M}}$
meta transitivity axiom	$\mathsf{Trans}(R)$	$R^{\mathcal{I}_c}$ is transitive.	$\mathcal{R}_{\mathsf{M}}$

Each axiom verifies a particular set of constraints. Axiom (11) and (12) validate the occurrence and fulfilment of role groups, respectively. In essence, only those objects (naturals or compartments) are checked that play a corresponding role in the constrained compartment, and there are enough of such objects in that compartment. In contrast, (13) checks whether relationships respect the imposed cardinality constraints. In conclusion, the formal model easily captures the context-dependent concepts and constraints. Moreover, it allows for checking the well-formedness of CROMs and validity of CROIs. Yet, due to a lack of tool support, this formal model is not viable for verifying the consistency of a constrained CROM, thus, requiring a more suitable formalism for reasoning.

Contextualised Description Logics. The ConDLs we use in this paper were first introduced in [6]. We shortly recall the relevant definitions and refer the reader to [2] for a thorough introduction to DLs. ConDLs consist of two levels. On the meta level knowledge *about* contexts can be represented, e.g. their relation to each other, on object level knowledge *within* contexts can be stated.

Definition 9 (Syntax of $\mathcal{SHOIQ}[\![\mathcal{SHOIQ}]\!]$). *Let* $\mathsf{O_C}$, $\mathsf{O_P}$, $\mathsf{O_I}$, $\mathsf{M_C}$, $\mathsf{M_P}$, $\mathsf{M_I}$ *be non-empty, pairwise disjoint sets of* concept names, property names *and* individual names *of the* object level *and the* meta level, *respectively.*

An object property[6] *is either some* $R \in \mathsf{O_P}$ *or an* inverse object property R^- *for* $R \in \mathsf{O_P}$*. An* object RBox $\mathcal{R}_\mathsf{O}$ *is a finite set of property inclusion axioms* $R \sqsubseteq S$ *and transitivity axioms* $\mathsf{Trans}(R)$*, where* R *and* S *are object properties. For* $R \in \mathsf{O_P}$*, we define* $\mathsf{Inv}(R) \coloneqq R^-$ *and* $\mathsf{Inv}(R^-) \coloneqq R$*, and assume that* $R \sqsubseteq S \in \mathcal{R}_\mathsf{O}$ *iff* $\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \in \mathcal{R}_\mathsf{O}$ *and that* $\mathsf{Trans}(R) \in \mathcal{R}_\mathsf{O}$ *iff* $\mathsf{Trans}(\mathsf{Inv}(R)) \in \mathcal{R}_\mathsf{O}$*. An object property* R *is called* simple *if* $\mathsf{Trans}(S) \notin \mathcal{R}_\mathsf{O}$ *for each* $S \sqsubseteq^{*} R$*, where* $\sqsubseteq^{*}$ *is the reflexive-transitive closure of* $\sqsubseteq$*. The set of* object concepts *is inductively defined starting from object concept names* $A \in \mathsf{O_C}$*, using the constructors in the first part of Table 1, where* $a, b \in \mathsf{O_I}$*,* $n \in \mathbb{N}$*,* R *is a object property,* S *is a simple object property and* C*,* D *are object concepts. The second part of Table 1 shows how* object axioms *are defined.*

A meta property*, the* meta RBox $\mathcal{R}_\mathsf{M}$ *and to be called* simple *are defined analogously to the object level. The set of* meta concepts *is inductively defined starting from meta property names* $P \in \mathsf{M_P}$ *and meta concept names* $B \in \mathsf{M_C}$*, using the constructors in the third part of Table 1, where* $u, v \in \mathsf{M_I}$*,* $n \in \mathbb{N}$*,* PQ *is a meta property,* Q *is a simple meta property,* E*,* F *are meta concepts and* α *is an object concept inclusion, concept assertion or property assertion. The fourth part of Table 1 shows how* meta axioms *are defined.*

A $\mathcal{SHOIQ}[\![\mathcal{SHOIQ}]\!]$ *ontology* $\mathcal{O}$ *is a triple* $\mathcal{O} = (\mathcal{B}, \mathcal{R}_\mathsf{M}, \mathcal{R}_\mathsf{O})$*, where* $\mathcal{B}$ *is a finite set of meta concept inclusions, concept assertions or property assertions,* $\mathcal{R}_\mathsf{M}$ *is a meta RBox and* $\mathcal{R}_\mathsf{O}$ *is an object RBox.*

We use the usual abbreviations for the object level: $C \sqcup D$ (disjunction) for $\neg(\neg C \sqcap \neg D)$, $\top$ (top concept) for $A \sqcup \neg A$, where $A \in \mathsf{O_C}$ is arbitrary but fixed, $\bot$ (bottom concept) for $\neg\top$, $\forall S.C$ (value restriction) for $\neg\exists S.\neg C$, $\geqslant_n S.C$ (at-least restriction) for $\neg(\leqslant_{n-1} S.C)$, and $=_n S.C$ (exact restriction) for $(\geqslant_n S.C) \sqcap (\leqslant_n S.C)$. Abbreviations for the meta level are used analogously.

To be able to express context independent knowledge, we have the sets of *rigid concepts* $\mathsf{O_{CR}} \subseteq \mathsf{O_C}$ and *rigid properties* $\mathsf{O_{PR}} \subseteq \mathsf{O_P}$ which must be interpreted the same in all contexts. Furthermore, we employ the *constant domain assumption*, i.e. all contexts speak about the same object domain, and the *rigid individual assumption*, i.e. individuals are always the same. The semantics of ConDLs are defined in a model-theoretic way.

Definition 10 (Semantics of $\mathcal{SHOIQ}[\![\mathcal{SHOIQ}]\!]$). *A* nested interpretation *is a tuple* $\mathcal{J} = (\mathbb{C}, \cdot^\mathcal{J}, \Delta, (\cdot^{\mathcal{I}_c})_{c \in \mathbb{C}})$*, where* $\mathbb{C}$ *and* Δ *are non-empty sets (called* contexts *and* (object) domain*),* $\cdot^\mathcal{J}$ *is a mapping assigning a set* $B^\mathcal{J} \subseteq \mathbb{C}$ *to every* $B \in \mathsf{M_C}$*, a binary relation* $P^\mathcal{J} \subseteq \mathbb{C} \times \mathbb{C}$ *to every* $P \in \mathsf{M_P}$ *and a context* $u^\mathcal{J} \in \mathbb{C}$ *to every* $u \in \mathsf{M_I}$*, and for every* $c \in \mathbb{C}$*,* $\cdot^{\mathcal{I}_c}$ *is a mapping assigning a set* $A^{\mathcal{I}_c} \subseteq \Delta$ *to every* $A \in \mathsf{O_C}$*, a binary relation* $R^{\mathcal{I}_c} \subseteq \Delta \times \Delta$ *to every* $R \in \mathsf{O_P}$ *and a domain element* $a^{\mathcal{I}_c} \in \Delta$ *to every* $a \in \mathsf{O_I}$ *such that for all* $c, c' \in \mathbb{C}$ *we have* $x^{\mathcal{I}_c} = x^{\mathcal{I}_{c'}}$ *for every* $x \in \mathsf{O_I} \cup \mathsf{O_{CR}} \cup \mathsf{O_{PR}}$*. The functions* $\cdot^{\mathcal{I}_c}$ *and* $\cdot^\mathcal{J}$ *are extended to object and meta properties and concepts, respectively, as shown in Table 1, where* $\sharp X$ *denotes the cardinality of the set* X*.*

[6] To avoid confusion with roles in CROM, the term *property* is used for binary relations instead.

Moreover, $\mathcal{J}$ ($\mathcal{I}_c$) satisfies *an meta axiom (object axiom), denoted by $\mathcal{J} \models \beta$ ($\mathcal{I}_c \models \alpha$), if the condition in the fourth (second) part of Table 1 holds, $\mathcal{J}$* satisfies *$\mathcal{B}$ ($\mathcal{R}_\mathsf{M}$) if $\mathcal{J}$ satisfies all axioms in $\mathcal{B}$ ($\mathcal{R}_\mathsf{M}$), $\mathcal{J}$* satisfies *$\mathcal{R}_\mathsf{O}$ if $\mathcal{I}_c$ satisfies all axioms in $\mathcal{R}_\mathsf{O}$ for all $c \in \mathbb{C}$, and $\mathcal{J}$* satisfies *$\mathcal{O}$ if it satisfies $\mathcal{B}$, $\mathcal{R}_\mathsf{M}$ and $\mathcal{R}_\mathsf{O}$. $\mathcal{O}$ is* consistent *if there exists a nested interpretation that satisfies $\mathcal{O}$. The* consistency problem *is the problem of deciding whether a given ontology is consistent.*

$\mathcal{SHOIQ}[\![\mathcal{SHOIQ}]\!]$ is a suitable candidate to encode both the context-dependent concepts and constraints of CROM. It permits, for instance to encode that in every context every role must have a player, as: $\top \sqsubseteq [\![A_{\mathsf{RT}} \sqsubseteq {=_1}\mathsf{plays}^-.\top]\!]$. Utilizing $\mathcal{SHOIQ}[\![\mathcal{SHOIQ}]\!]$, it becomes feasible to automatically map CROM domain models to a DL ontology.

3 Reasoning on Role-Based Models

To verify the consistency and satisfiability of CROM domain models, it is necessary to encode both the underlying semantics as well as the context-dependent concepts and constraints in ConDL axioms. This mapping must preserve validity, i.e. the ConDL ontology is consistent if and only if there exists a CROI that is compliant with the CROM and valid w.r.t. the constraint model. Henceforth, we highlight our encoding scheme and prove that it preserves the semantics.

In general, compartment types are modelled as concepts on the meta level; whereas playing roles, relationships and constraints are modelled within a compartment on the object level. Thus, we introduce o-concepts for natural types and role types, as well as a special o-property plays. Accordingly, the fills relation is transformed into domain and range axioms for plays. Relationship types are intuitively modelled as o-properties between two played roles. Role groups are handled like roles with an additional axiom stating that "playing" a role group is equivalent to fulfilling the constraints specified in that role group. Furthermore, if an object plays an atom of a non-nested role group, that object must fulfill the role group. For occurrence constraints a fresh individual name $\mathsf{counter}$ and an o-property counts is introduced and each played role or fulfilled role group is connected to this counter. Thus, both occurrence constraints and relationship cardinalities can be represented as qualified number restriction. Special consideration is needed for compartments that play roles within other compartments. Even though establishing a one-to-one link between an element on the object level and one on the meta level is impossible, for consistency it is only relevant if the compartment type of the nested compartment is instantiable. Hence, we consider the o-concepts $\mathsf{N}_{\mathsf{CT}'}$, e.g. compartments which play roles on the object level, denoted as *o-compartments*, as copies of the m-concepts N_{CT}.

Ontologically, natural types would be captured as rigid concepts and their fields as rigid properties, since that information does not change within contexts. In our setting, rigidity has no influence on the consistency, and neglecting it decreases the computational complexity exponentially in the size of the input. Admittedly, the case with rigid names can be handled quite similar. In summary, we consider the following concept, property and individual names:

Table 2. Mapping for occurring types and the CROM $\mathcal{M}$.

	$\top \sqsubseteq \bigsqcup_{CT \in \mathsf{N_{CT}}} CT$	(14)
$\bigwedge_{\substack{CT_1, CT_2 \in \mathsf{N_{CT}} \\ CT_1 \neq CT_2}}$	$CT_1 \sqsubseteq \neg CT_2$	(15)
	$\top \sqsubseteq [\![A_{\mathsf{O}} \equiv \bigsqcup_{NT \in \mathsf{N_{NT}}} NT \sqcup \bigsqcup_{CT' \in \mathsf{N_{CT'}}} CT']\!]$	(16)
	$\top \sqsubseteq [\![A_{\mathsf{RT}} \equiv \bigsqcup_{RT \in \mathsf{N_{RT}}} RT]\!]$	(17)
	$\top \sqsubseteq \bigsqcap_{T_1, T_2 \in \mathsf{N_{NT}} \cup \mathsf{N_{CT'}} \cup \mathsf{N_{RT}}, T_1 \neq T_2} [\![T_1 \sqsubseteq \neg T_2]\!]$	(18)
	$\top \sqsubseteq [\![\top \sqsubseteq A_{\mathsf{O}} \sqcup A_{\mathsf{RT}} \sqcup A_{\mathsf{RG}} \sqcup \{\mathsf{counter}\}]\!]$	(19)
	$\top \sqsubseteq [\![A_{\mathsf{O}} \sqsubseteq \neg A_{\mathsf{RT}}]\!] \sqcap [\![A_{\mathsf{O}} \sqsubseteq \neg A_{\mathsf{RG}}]\!] \sqcap [\![A_{\mathsf{RT}} \sqsubseteq \neg A_{\mathsf{RG}}]\!] \sqcap [\![\neg (A_{\mathsf{O}} \sqcup A_{\mathsf{RT}} \sqcup A_{\mathsf{RG}})(\mathsf{counter})]\!]$	(20)
	$\top \sqsubseteq \bigsqcap_{RT \in \mathsf{N_{RT}}} [\![A_{\mathsf{O}} \sqsubseteq \leqslant_1 \mathsf{plays}.RT]\!]$	(21)
	$\top \sqsubseteq [\![A_{\mathsf{RT}} \sqsubseteq =_1 \mathsf{plays}^-.\top]\!]$	(22)
	$\top \sqsubseteq [\![\exists \mathsf{plays}.\top \sqsubseteq A_{\mathsf{O}}]\!]$	(23)
	$\top \sqsubseteq [\![\top \sqsubseteq \forall \mathsf{plays}.(A_{\mathsf{RT}} \sqcup A_{\mathsf{RG}})]\!]$	(24)
$\bigwedge_{CT' \in \mathsf{N_{CT'}}}$	$\neg [\![CT' \sqcap \exists \mathsf{plays}.\top \sqsubseteq \bot]\!] \sqsubseteq \exists \mathsf{nested}.CT$	(25)
	$\top \sqsubseteq \bigsqcap_{RT \in \mathsf{N_{RT}}} [\![\exists \mathsf{plays}.RT \sqsubseteq (\bigsqcup_{(T,RT) \in \mathsf{fills}} T)]\!]$	(26)
$\bigwedge_{CT \in \mathsf{N_{CT}}}$	$CT \sqsubseteq [\![A_{\mathsf{RT}} \sqsubseteq \bigsqcup_{RT \in \mathsf{parts}(CT)} RT]\!]$	(27)
$\bigwedge_{RST \in \mathsf{N_{RST}}, \mathsf{rel}(RST) = (RT_1, RT_2)}$	$\top \sqsubseteq [\![\exists RST.\top \sqsubseteq RT_1]\!] \sqcap [\![\top \sqsubseteq \forall RST.RT_2]\!]$	(28)

- $\mathsf{N_{CT}} \subseteq \mathsf{M_C}$ since every compartment type is a m-concept,
- $\mathsf{nested} \in \mathsf{M_P}$ to assure the existence of compartments that play roles,
- $\mathsf{N_{NT}} \cup \mathsf{N_{CT'}} \cup \mathsf{N_{RT}} \subseteq \mathsf{O_C}$ since every natural type, every o-compartment type and every role type is an o-concept,
- $\mathsf{plays} \in \mathsf{O_P}$ to express the plays-relation,
- $\mathsf{N_{RST}} \subseteq \mathsf{O_P}$ since every relationship type is an o-property,
- $\mathsf{counter} \in \mathsf{O_I}$ and $\mathsf{counts} \in \mathsf{O_P}$ to express the occurrence constraints, and
- $A_{\mathsf{O}}, A_{\mathsf{RT}}, A_{\mathsf{RG}} \in \mathsf{O_C}$ for, resp., all objects eligible of playing roles, i.e. naturals and o-compartments, all played roles, and all instances of role groups.

Henceforth, the mapping is trisected, first describing how types are encoded, then how fills and rel are mapped and finally how constraints are represented.

Encoding CROM Types. Table 2 (first segment) summarises the encoding of the underlying semantics of a given CROM $\mathcal{M}$. On the meta level, we assure that every context belongs to exactly one compartment type (Eq. (14), (15)). Within every context, every natural, o-compartment and role belongs to exactly one type (Eq. (16), (17), (18)). On the object level, every element is a role, a natural, an o-compartment, a role group instance or the individual counter (Eq. (19), (20)). Every natural or o-compartment can only play one RT-role in each context and each role must be played by someone (Eq. (21), (22)). We formalise a general domain and range restriction for plays. Only naturals or o-compartments can play something, and only roles or role group instances can be played (Eq. (23), (24)). Finally, if an o-compartment plays a role in some context, the o-compartment

Table 3. Mapping for constraint model $\mathcal{C}$ and for assertions.

$$\top \sqsubseteq [\![A_{\mathsf{RG}} \equiv \bigsqcup_{RG \in \mathbb{RG}(\mathcal{C})} RG]\!] \tag{29}$$

$$\top \sqsubseteq \bigsqcap_{RG_1,\, RG_2 \in \mathbb{RG}(\mathcal{C}), RG_1 \neq RG_2} [\![RG_1 \sqcap RG_2 \sqsubseteq \bot]\!] \tag{30}$$

$$\top \sqsubseteq [\![A_{\mathsf{RG}} \sqsubseteq \geqslant_1 \mathsf{plays}^- . \top \sqcap \leqslant_1 \mathsf{plays}^- . \top]\!] \tag{31}$$

$$\top \sqsubseteq \bigsqcap_{RG \in \mathbb{RG}(\mathcal{C})} [\![A_{\mathsf{O}} \sqsubseteq \leqslant_1 \mathsf{plays}.RG]\!] \tag{32}$$

$$\top \sqsubseteq \bigsqcap_{\substack{RG \in \mathbb{RG}(\mathcal{C}), \\ RG = (\{A_1, \ldots, A_n\}, k, l)}} [\![\exists \mathsf{plays}.RG \equiv (\geqslant_k \mathsf{plays}.(A_1 \sqcup \ldots \sqcup A_n)) \sqcap (\leqslant_l \mathsf{plays}.(A_1 \sqcup \ldots \sqcup A_n))]\!] \tag{33}$$

$$\top \sqsubseteq \bigsqcap_{RG \in \mathbb{RG}^\top(\mathcal{C})} [\![\exists \mathsf{plays}.\big(\bigsqcup_{RT \in \mathsf{atom}(RG)} RT\big) \sqsubseteq \exists \mathsf{plays}.RG]\!] \tag{34}$$

$$\top \sqsubseteq [\![A_{\mathsf{RT}} \sqcup A_{\mathsf{RG}} \sqsubseteq =_1 \mathsf{counts}^- . \{\mathsf{counter}\}]\!] \tag{35}$$

$$\bigwedge_{\substack{(k..l, RG) \\ \in \mathsf{occur}(CT), \\ CT \in \mathsf{N_{CT}}}} CT \sqsubseteq [\![(\geqslant_k \mathsf{counts}.RG)(\mathsf{counter})]\!] \sqcap [\![(\leqslant_l \mathsf{counts}.RG)(\mathsf{counter})]\!] \tag{36}$$

$$\top \sqsubseteq \bigsqcap_{\substack{RST \in \mathsf{N_{RST}}, \\ \mathsf{rel}(RST) = (RT_1, RT_2), \\ \mathsf{card}(RST) = (i..j, k..l)}} [\![RT_1 \sqsubseteq \geqslant_k RST . \top \sqcap \leqslant_l RST . \top]\!] \sqcap [\![RT_2 \sqsubseteq \geqslant_i RST^- . \top \sqcap \leqslant_j RST^- . \top]\!] \tag{37}$$

must also exist as context (Eq. (25)). After encoding the general knowledge about types, we succinctly map a specific CROM $\mathcal{M}$ to ConDL axioms.

Mapping CROM $\mathcal{M}$. The fills relation restricts which natural or compartment types can play which role types. Hence, a role type has as plays predecessors only naturals or o-compartments of types which fill that role type (Eq. (26)). In conjunction with Eq. (24), we know that all plays successors of naturals or o-compartments of a specific type are either instances of a role type that are filled by that type or instances of a role group. Thus, the axiom $\top \sqsubseteq \bigsqcap_{T \in \mathsf{N_{NT}} \cup \mathsf{N_{CT'}}} [\![T \sqsubseteq \forall \mathsf{plays}.(A_{\mathsf{RG}} \sqcup \bigsqcup_{(T,RT) \in \mathsf{fills}} RT)]\!]$ is entailed. Since in a compliant CROI the plays-relation respects parts, only RT-roles with $RT \in \mathsf{parts}(CT)$ exist in a CT-context (Eq. (27)). Analogous to fills restricting the domain and range of plays, the rel-function restricts these for each relationship type (Eq. (28)). Due to Eq. (14), (17), (18) and (27) as well as the fact that parts' codomain is a partition of $\mathsf{N_{RT}}$, in any context that is not in CT there are no roles of a type the participates in CT. Thus, the following axiom is entailed for all $CT \in \mathsf{N_{CT}}$: $\neg CT \sqsubseteq [\![\bigsqcup_{RT \in \mathsf{parts}(CT)} RT \sqsubseteq \bot]\!]$.

Including the Constraint Model $\mathcal{C}$. Let $\mathcal{C}$ be a constraint model, let $\mathbb{RG}(\mathcal{C})$ be the set of all complex role groups occurring in $\mathcal{C}$, and let $\mathbb{RG}^\top(\mathcal{C}) \subseteq \mathbb{RG}(\mathcal{C})$ be the subset of non-nested role groups. Analogous to roles, role groups are disjoint, every instance of a role group must be played by some object and every object can either fulfill or not fulfill a role group (Eq. (29), (30), (31), (32)). Complex role groups are treated like role types. An object "plays" an instance of a role group if it fulfills that role group (Eq. (33)). Furthermore, if an object plays a role whose type is an atom of a non-nested role group, the object must also fulfill that role group (Eq. (34)). To capture the occurrence constraints we enforce all roles and role group instances to be connected to counter via counts and state concept assertions for counter which must hold in the respective compartment type (Eq. (35), (36)). Cardinality constraints restrict the number of roles that are related to a role via a relationship type (Eq. (37)).

Preserving Semantics. After presenting the mapping we establish the main result of this section. For a CROM $\mathcal{M}$ and a constraint model $\mathcal{C}$, let $\mathcal{K}$ be the pair $(\mathcal{M}, \mathcal{C})$. Then, the ConDL ontology $\mathcal{O}_\mathcal{K}$ is the set of Axiom (14) to (37). The next theorem establishes the desired relationship between $\mathcal{K}$ and $\mathcal{O}_\mathcal{K}$.

Theorem 11. *Let $\mathcal{K}$ be the pair $(\mathcal{M}, \mathcal{C})$ with $\mathcal{M}$ being a well-formed CROM and $\mathcal{C}$ a compliant constraint model. Then, there exists a CROI that is compliant with $\mathcal{M}$ and valid w.r.t. $\mathcal{C}$ iff $\mathcal{O}_\mathcal{K}$ is consistent.*

The proof is a straight forward application of the axioms[3]. Checking consistency is quite general in the sense that many other questions can be reduced to the consistency problem. With the expressive means of ConDL assertions, for instance, we can also check whether a specific compartment type is instantiable, a certain role type is playable or two roles can be linked via some relationship type. Apart from that a reasoner that is capable of processing such ontologies is also needed to use the mapping in practice.

4 JConHT - A $\mathcal{SHOIQ}[\![\mathcal{SHOIQ}]\!]$ Reasoner

Also due to highly optimised reasoners, DLs have been successfully established. In order to reuse an existing reasoner, we convert the consistency problem in ConDL to classical reasoning tasks. In [6], a reduction into two separate decision problems is shown. Firstly, reasoning on the meta level, and secondly, checking whether the object level is consistent in each context. For several reasons we base our implementation on the hypertableau reasoner HermiT [19,11]. A model construction-based reasoner is necessary since we need information about the appearing o-axioms when reasoning on the meta level. Besides that HermiT is implemented in Java and according to the ORE Report [20] the most perfomant, model-based reasoner in the discipline *OWL DL Consistency*.

For brevity, we omit the details of the hypertableau algorithm here. The sound, complete and terminating algorithm that we construct on the basis of hypertableau is shown in Alg. 1. Here, $\mathcal{C}$ is the set of DL clauses, $\mathcal{A}$ the ABox obtained in the clausification of $\mathcal{O}$ and $\mathcal{K}_i$ is the object ontology for the world c_i which collects all o-axioms in $\mathcal{A}'$ that are asserted to hold in $c_i$and all other o-axioms as negated axioms (see Def. 11 of [6] for details). $\mathcal{K}_{\mathsf{rig}}$ is defined analogously to $\mathcal{K}_i$, but using the renaming technique as single ontology for all worlds.

The second step of the preprocessing, i.e. the *repletion*, is necessary to ensure completeness of Alg. 1. The hypertableau algorithm avoids the unnecessary non-determinism which is usually introduced by the GCI-rule in tableau algorithms [19]. But this optimisation disguises some implicit contradictions in the DL clauses. Consider $\mathcal{C}_{\mathsf{ex}} = \{[\![\neg A(a)]\!](x) \to C(x), \top \to C(x) \vee [\![A \sqsubseteq \bot]\!](x)\}$ and $\mathcal{A}_{\mathsf{ex}} = \{\neg C(s)\}$. Here, only $[\![A \sqsubseteq \bot]\!](s)$ would be derived and the ontology seems to be consistent. Let $\mathcal{J}$ be a model, then we have $s \in (\neg C)^\mathcal{J}$, $s \in [\![A \sqsubseteq \bot]\!]^\mathcal{J}$ and $s \notin [\![\neg A(a)]\!]^\mathcal{J}$ which, by the semantics of ConDL, implies $s \in [\![A(a)]\!]^\mathcal{J}$. This contradicts $[\![A \sqsubseteq \bot]\!](s)$ and $(\mathcal{C}_{\mathsf{ex}}, \mathcal{A}_{\mathsf{ex}})$ is indeed inconsistent. To make these implicitly negated o-axioms visible, we introduce the repletion of $\mathcal{C}$.

Algorithm 1: Algorithm for checking consistency with hypertableau

Input : $\mathcal{SHOIQ}[\![\mathcal{SHOIQ}]\!]$-ontology $\mathcal{O}$
Output: true if $\mathcal{O}$ is consistent, false otherwise
Preprocessing (results in $(\mathcal{C}, \mathcal{A})$):
1. Elimination of transitivity axioms, normalisation, clausification
2. Repletion of DL-clauses

Let (T, λ) be any derivation for $(\mathcal{C}, \mathcal{A})$.
$\mathfrak{A} \coloneqq \{\mathcal{A}' \mid$ there exists a leaf node in (T, λ) that is labelled with $\mathcal{A}'\}$
for $\mathcal{A}' \in \mathfrak{A}$ **do**
  if $\mathcal{A}'$ *is clash-free* **then**
    if $\mathcal{O}$ *contains rigid names* **then**
      if $\mathcal{K}_{\mathsf{rig}} \coloneqq (\mathcal{O}_{\mathcal{A}'}, \mathcal{R}_{\mathsf{O}}{}')$ *is consistent* **then**
        return true
    else
      Let $\{c_1, \ldots, c_k\}$ be the individuals occurring in $\mathcal{A}'$
      if $\mathcal{K}_i \coloneqq (\mathcal{O}_{c_i}, \mathcal{R}_{\mathsf{O}})$ *is consistent for all* $1 \leq i \leq k$ **then**
        return true
return *false*

Definition 12 (Repletion of DL-Clauses). *Let $\mathcal{C}$ be a set of DL-clauses. The* repletion *of $\mathcal{C}$ is obtained from $\mathcal{C}$ by adding the DL-clause $\top \to [\![\alpha]\!](x) \vee [\![\neg\alpha]\!](x)$ for each o-axiom $[\![\alpha]\!]$ occurring in $\mathcal{C}$.*

A drawback of the repletion is the high amount of non-determinism it introduces, but it is only necessary if o-axioms occur in the antecedent of a DL-clause. Apparently only Ax. (25), i.e. only if compartments play roles, introduces such o-axioms in the antecedent. Therefore, only then the repletion is necessary when reasoning on CROMs. Arguably, when constraints are omitted, CROM can be mapped to a less expressive ConDL, which further reduces the reasoning time.

5 Case Studies

Implicit Knowledge in the Banking Domain. Let us consider our running example. Instead of writing down all axioms of the respective ontology $\mathcal{O}_{\mathsf{Bank}}$, we will rather point out those inferences that uncover hidden restrictions. In detail, we first inspect the Bank compartment type and its internal role types, role groups, and relationship types. Omitting general axioms, the Axioms (38) to (43) are contained in the ontology. Consequently, in any interpretation $\mathcal{J}$ that satisfies $\mathcal{O}_{\mathsf{Bank}}$ with $c \in \mathsf{Bank}^{\mathcal{J}}$, Ax. (38) and (39) entail the existence of an element in $\mathsf{CheckingAccount}^{\mathcal{I}_c}$ or $\mathsf{SavingsAccount}^{\mathcal{I}_c}$. Due to (40) and (41), there must be some element "owning a CA or SA", which, by (42) and (43), must be in $\mathsf{Customer}^{\mathcal{I}_c}$. As a result, Ax. 44 is entailed, i.e. the occurrence constraint for Customer is essentially $1..*$. Similarly, when investigating the Transaction compartment type, we infer Ax. (45) to (48). For instance, assume there is a $d \in \mathsf{Transaction}^{\mathcal{J}}$. By (45) and (46), we infer the existence of exactly one element in $\mathsf{Source}^{\mathcal{I}_d}$ or $\mathsf{Target}^{\mathcal{I}_d}$. However, due to (47) and (48),

Table 4. ConDL axioms of the banking example

$$\mathsf{Bank} \sqsubseteq [\![(\geqslant_1 \mathsf{counts}.\mathsf{BankAccounts})(\mathsf{counter})]\!] \tag{38}$$

$$\top \sqsubseteq [\![\exists \mathsf{plays}.\mathsf{BankAccounts} \equiv =_1 \mathsf{plays}.(\mathsf{CheckingAccount} \sqcup \mathsf{SavingsAccount})]\!] \tag{39}$$

$$\top \sqsubseteq [\![\mathsf{SavingsAccount} \sqsubseteq \geqslant_1 \mathsf{own\_sa}^{-}.\top]\!] \tag{40}$$

$$\top \sqsubseteq [\![\mathsf{CheckingAccount} \sqsubseteq \geqslant_1 \mathsf{own\_ca}^{-}.\top]\!] \tag{41}$$

$$\top \sqsubseteq [\![\exists \mathsf{own\_sa}.\top \sqsubseteq \mathsf{Customer}]\!] \tag{42}$$

$$\top \sqsubseteq [\![\exists \mathsf{own\_ca}.\top \sqsubseteq \mathsf{Customer}]\!] \tag{43}$$

$$\mathsf{Bank} \sqsubseteq [\![(\geqslant_1 \mathsf{counts}.\mathsf{Customer})(\mathsf{counter})]\!] \tag{44}$$

$$\mathsf{Transaction} \sqsubseteq [\![(=_1 \mathsf{counts}.\mathsf{Participants})(\mathsf{counter})]\!] \tag{45}$$

$$\top \sqsubseteq [\![\exists \mathsf{plays}.\mathsf{Participants} \equiv =_1 \mathsf{plays}.(\mathsf{Source} \sqcup \mathsf{Target})]\!] \tag{46}$$

$$\top \sqsubseteq [\![\mathsf{Source} \sqsubseteq =_1 \mathsf{trans}.\top]\!] \sqcap [\![\mathsf{Target} \sqsubseteq =_1 \mathsf{trans}^{-}.\top]\!] \tag{47}$$

$$\top \sqsubseteq [\![\exists \mathsf{trans}.\top \sqsubseteq \mathsf{Source}]\!] \sqcap [\![\exists \mathsf{trans}^{-}.\top \sqsubseteq \mathsf{Target}]\!] \tag{48}$$

$$\mathsf{Transaction} \sqsubseteq \bot \tag{49}$$

$$\top \sqsubseteq [\![\mathsf{Customer} \sqsubseteq \geqslant_1 \mathsf{issues}.\top]\!] \tag{50}$$

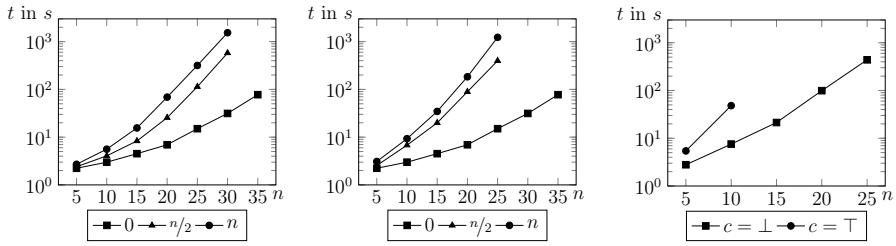
$$\top \sqsubseteq [\![\exists \mathsf{issues}^{-}.\top \sqsubseteq \mathsf{MoneyTransfer}]\!] \tag{51}$$

$$\top \sqsubseteq [\![\exists \mathsf{plays}.\mathsf{MoneyTransfer} \sqsubseteq \mathsf{Transaction}']\!] \tag{52}$$

$$\neg [\![\mathsf{Transaction}' \sqcap \exists \mathsf{plays}.\top \sqsubseteq \bot]\!] \sqsubseteq \exists \mathsf{nested}.\mathsf{Transaction} \tag{53}$$

there must also be an element in $\mathsf{Target}^{\mathcal{I}_d}$ or $\mathsf{Source}^{\mathcal{I}_d}$, respectively. It follows that $\mathsf{Participants}^{\mathcal{I}_d}$ contains two elements, which contradicts (45). In conclusion, Ax. 49 is entailed, i.e. Transaction is not instantiable, due to the occurrence constraint of the Participants role group. With this knowledge, we can further reason on the Bank compartment type. Due to (44) and (50) to (52), there must be an element in $\mathsf{MoneyTransfer}^{\mathcal{I}_c}$ playing an element in $\mathsf{Transaction}'^{\mathcal{I}_c}$. Thus, by (53), there must be a context c_2 connected to c via nested with $c_2 \in \mathsf{Transaction}$. Yet, this contradicts (49). In consequence, the banking domain model is indeed inconsistent, due to a small modelling error in an occurrence constraints.

Performance Evaluation of JConHT. To investigate the performance of our approach, we conducted a set of benchmarks to test both the translation to a contextualised DL ontology (Sect. 3) as well as the subsequent reasoning with JConHT (Sect: 4). Hence, we developed a generator for CROM to create pseudo-random domain models of increasing complexity. Then, these models are transformed to the corresponding OWL ontology, and finally tested for consistency using our reasoner. Notably though, we focus on the execution time of JConHT, as the transformation time is polynomial bounded in the input size (i.e. $\mathcal{O}(n^2)$) and negligible small. We investigated in the impact of three variables on the performance: (a) number of relationship types defined and constrained per compartment type, (b) number of role groups introduced per compartment type, and (c) a Boolean indicating whether compartment types can play roles. Our experiments generates random CROM models of stepwise increased complexity, varying each of these variables. The generator itself utilises a pseudo-



(a) Variation of ♯ RSTs. (b) Variation of ♯ RGs. (c) Flat vs. nested CTs.

Figure 2. Average execution times of JConHT for benchmark ontologies.

random number generator (initialised with a given seed s), to create CROM models of size n, i.e., a domain model with n natural types and n compartment types with n role types each, such that each role type is filled by two player types (either natural or compartment type). Additional parameters determine the number of relationship types m between two distinct role types for each compartment (a), the number of role groups k of two random role types for each compartment (b), and a Boolean c indicating that compartment types are eligible as player types (c). The constraint model is generated accordingly, by assigning random cardinalities to the occurrence of role types and role groups, to role groups, as well as to the ends of relationships. Notably though, the set of cardinalities is limited $\mathsf{Card} \coloneqq \{0..0, 0..1, 0..\infty, 1..1, 1..\infty\}$. Utilizing this generator, we can individually test the performance impact of each of the variables.

We generated CROM domain models with $k \in \{0, n/2, n\}$ relationship types for (a) and $m \in \{0, n/2, n\}$ role groups for (b). To investigate nested compartment types (c), CROMs were created with $c \in \{\bot, \top\}$ whereas $k = n/2$ relationship types and $m = n/2$ role group. In each case, we generated, transformed, and verified CROM domain models with $n = 5, 10, 15, \ldots$ until the reasoner threw an out-of-memory exception. We repeated this process for each configuration, i.e., generating 100 models for each configuration with seed s, $1 \leq s \leq 100$, and calculate the average execution time of the reasoner to decide consistency.[7]

Fig. 2 sums up the impact of each variable on JConHT's performance. As ConDL's reasoning time is exponential in the size of n, the time axis is logarithmic. In fact, Fig. 2a and Fig. 2b illustrates the impact of constrained relationships and role groups, respectively. The baseline for both is a configuration without relationship cardinalities and role groups, i.e., only occurrence constraints. We found that the number of constrained relationships has a lower performance impact than the number of role groups. This is unsurprising, as role groups can represent arbitrary propositional logic formulae [17]. Thus, role groups become unsatisfiable easily. In turn, Fig. 2c indicates a significant performance penalty of nested compartment types are present. While reasoning on

[7] The tests were performed on a 3.3GHz i5-2500 quad core with 12GB heap size dedicated to an openjdk-8-jre (build 1.8.0) running on an Ubuntu 16.04.

CROMs with $n/2$ role groups and relationship types is tractable, permitting compartment types to play roles quickly leads to a state space explosion, i.e. only models of size $n \leq 10$ could be checked. This is due to Axiom (25), requiring the addition of repletion clauses, and thus, introducing a large amount of nondeterminism. In sum, our performance evaluation indicates that variable (c), nested compartment, has the highest performance impact, whereas the inclusion of role groups (b) and constrained relationships (a) has comparatively small impact. Besides that, we identified the heap size as a limiting factor, especially, when reasoning on more complex or nested models. When dealing with an average number of relationship types and role groups without nested compartments, however, reasoning was feasible for domain models of size $n \leq 25$.

6 Related Work

In the past, several approaches for formal frameworks to reason on UML arose, e.g. [7,24,4,25,1], from which we adopted some ideas, e.g. to model attributes of a UML class with DL properties and multiplicities of associations with counters [7]. In general, UML lacks expressive power to model context-dependent domains and while some approaches extended UML in this regard [23,10,12], there semantics is usually more ambiguous. In contrast, CROM has both a well-defined and formal semantics [17]. As classical DLs cannot properly formalise contextual knowledge, many different approaches and extensions of DLs have been proposed, e.g. [22,3,15,8]. Yet, many were tailored to different goals, e.g., to support context-specific reuse of ontologies. On one side, in most cases they have a different understanding of contexts, e.g. defined as set of attribute-value declarations for given *dimensions* [22]. Consequently, except a coverage relation, one can hardly express any other knowledge about contexts, such as their relational structure, rendering them insufficient to model CROMs. Similarily, in [5] a multidimensional data model is introduced with the same restrictions. On the other side, $\mathcal{ALC}_{\mathcal{ALC}}$ [15] formalizes contexts as formal objects with properties and relational structure, resulting in a similar two-dimensional DL that permits object knowledge to transcend through contexts. Yet, this leads to a double exponential time complexity and, in the presence of rigid roles, to undecidability.

7 Conclusion

To cope with context-dependent knowledge of today's Information Systems, advanced domain models permitting formal validation are indispensable. We enabled reasoning on context-dependent domain models by mapping CROM to the contextualized description logic $\mathcal{SHOIQ}[\![\mathcal{SHOIQ}]\!]$ for which the dedicated reasoner JConHT can efficiently decide consistency. We showcased the semantically correctness, suitability and performance of our approach. In future, we will map additional constraints and further optimize both our mapping and JConHT.

References

1. Ahmad, M.A., Nadeem, A.: Consistency checking of UML models using description logics: A critical review. In: Proc. ICET'10. pp. 310–315 (2010)
2. Baader, F., et al. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2nd edn. (2007)
3. Baader, F., et al.: Context-dependent views to axioms and consequences of semantic web ontologies. Journal of Web Semantics 12, 22–40 (2012)
4. Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML class diagrams. Artificial Intelligence 168(1-2), 70–118 (2005)
5. Bertossi, L.E., Milani, M.: The ontological multidimensional data model in quality data specification and extraction. In: Proc. BICOD'17. pp. 126–130 (2017)
6. Böhme, S., Lippmann, M.: Decidable description logics of context with rigid roles. In: Proc. FroCoS'15. pp. 17–32 (2015)
7. Calì, A., Calvanese, D., De Giacomo, G., Lenzerini, M.: A formal framework for reasoning on UML class diagrams. In: Proc. ISMIS'02. pp. 503–513 (2002)
8. Ceylan, I.I., Peñaloza, R.: The bayesian ontology language $\mathcal{BEL}$. Journal of Automated Reasoning 58(1), 67–95 (2017)
9. Chen, P.P.S.: The entity-relationship model toward a unified view of data. ACM Transactions on Database Systems (TODS) 1(1), 9–36 (1976)
10. Genovese, V.: A meta-model for roles: Introducing sessions. In: Proc. Ws. on Roles and Relationships in OOP, Multiagent Systems, and Ontologies. pp. 27–38 (2007)
11. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: An OWL 2 reasoner. Journal of Automated Reasoning 53(3), 245–269 (2014)
12. Guizzardi, G., Wagner, G.: Conceptual simulation modeling with Onto-UML. In: Proc. Winter Simulation Conference. p. 5 (2012)
13. Halpin, T.: Object-role modeling (ORM/NIAM). In: Handbook on architectures of information systems, pp. 81–103. Springer (2006)
14. Hennicker, R., Klarl, A.: Foundations for ensemble modeling – the Helena approach. In: Specification, Algebra, and Software, pp. 359–381. Springer (2014)
15. Klarman, S., Gutiérrez-Basulto, V.: Description logics of context. Journal of Logic and Computation 26(3), 817–854 (2016)
16. Kühn, T., et al.: A metamodel family for role-based modeling and programming languages. In: Proc. SLE'14. pp. 141–160 (2014)
17. Kühn, T., et al.: A combined formal model for relational context-dependent roles. In: Proc. SLE'15. pp. 113–124 (2015)
18. Liu, M., Hu, J.: Information networking model. In: ER'09, pp. 131–144 (2009)
19. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. Journal of Artificial Intelligence Research 36, 165–228 (2009)
20. Parsia, B., et al.: The OWL reasoner evaluation (ORE) 2015 competition report. In: Proc. SSWS'15, co-located with ISWC'15. pp. 2–15 (2015)
21. Rumbaugh, J., Jacobson, R., Booch, G.: The Unified Modelling Language Reference Manual. Addison-Wesley, 1st edn. (1999)
22. Serafini, L., Homola, M.: Contextualized knowledge repositories for the semantic web. Journal of Web Semantics 12, 64–87 (2012)
23. Sheng, Q.Z., et al.: Contextuml: A UML-based modeling language for model-driven development of context-aware web services. In: Proc. ICMB'05. pp. 206–212 (2005)
24. Simmonds, J., Straeten, R.V.D., Jonckers, V., Mens, T.: Maintaining consistency between UML models using description logic. L'Objet 10(2-3), 231–244 (2004)
25. Simmonds, J., et al.: A tool based on DL for UML model consistency checking. Int. J. of Software Eng. and Knowledge Eng. 18(6), 713–735 (2008)