Ingeniería de ontologías

Extensión a lógica descriptiva: Metamodelado

Caso de uso:
rol de una ontología en un sistema de contabilidad

Noviembre 2022

Contenido de la presentación

Lo que OWL no puede representar

- Modelar tiempo, probabilidad, metamodelado
- Extensión de lógica para metamodelado

Caso de estudio

- Perspectivas de usuarios, aplicación de contabilidad
- Demo

Patrón de diseño para metamodelado

Lo que OWL (SROIQ) no puede representar

⊤, ⊥, ⊓, ⊔, ∃, ∀, ¬

Hijo = *Padre*$^-$ Trans(*compañeroDeClase*) Dis(*esAmigoDe*, *esEnemigoDe*)

tieneHijo ⊑ *tieneDescendiente* *esHermano* ∘ *esPadre* ⊑ *esTio* Irr(*tieneHijo*)

PersonaViva ≡ *Persona* ⊓ ∀*viveEn.Lugar* → Representamos una realidad en **presente**

Cómo representamos a personas mortales? Vivas hasta un momento en que no estarán vivas para siempre

PersonaMortal ≡ *PersonaViva* ⊓ ∃*viveEn.Lugar* ⊓ (*PersonaViva* $\mathcal{U}$ $\Box^+$ ¬*PersonaViva*) → Lógica temporal

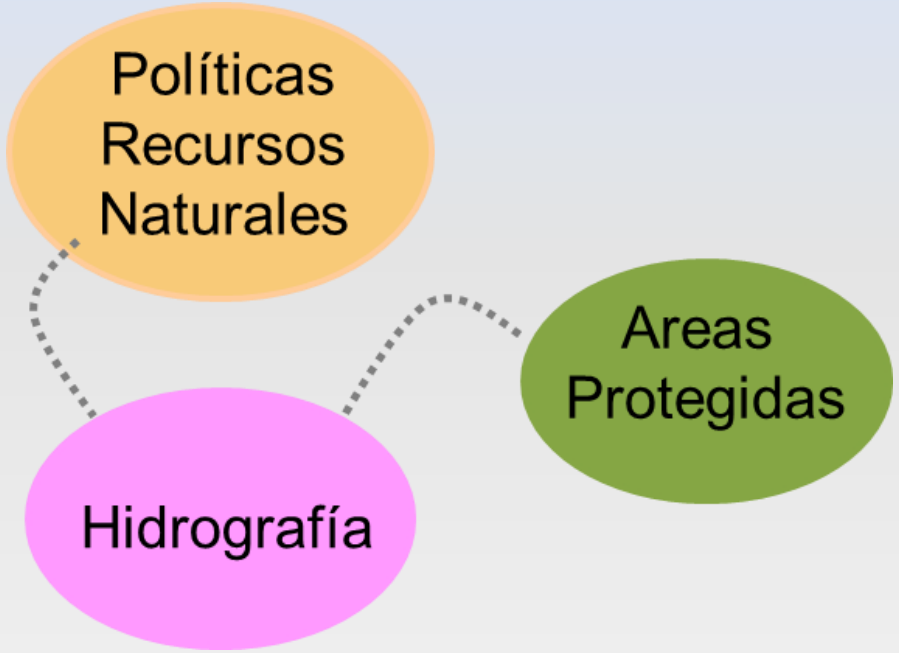
$\mathcal{U}$ $\mathcal{S}$ $\Diamond^+$ $\Diamond^-$ $\Box^+$ $\Box^-$

PersonaViva(*Juan*) ¬∃*viveEn.PaisEuropeo*(*Juan*) → Representamos una realidad **precisa** (verdadero o falso)

Cómo representamos que Juan es una persona viva con alguna probabilidad (imprecisión)

PersonaViva(*Juan* **.7**) ¬∃*viveEn.PaisEuropeo*(*Juan* **1**) → Lógica difusa

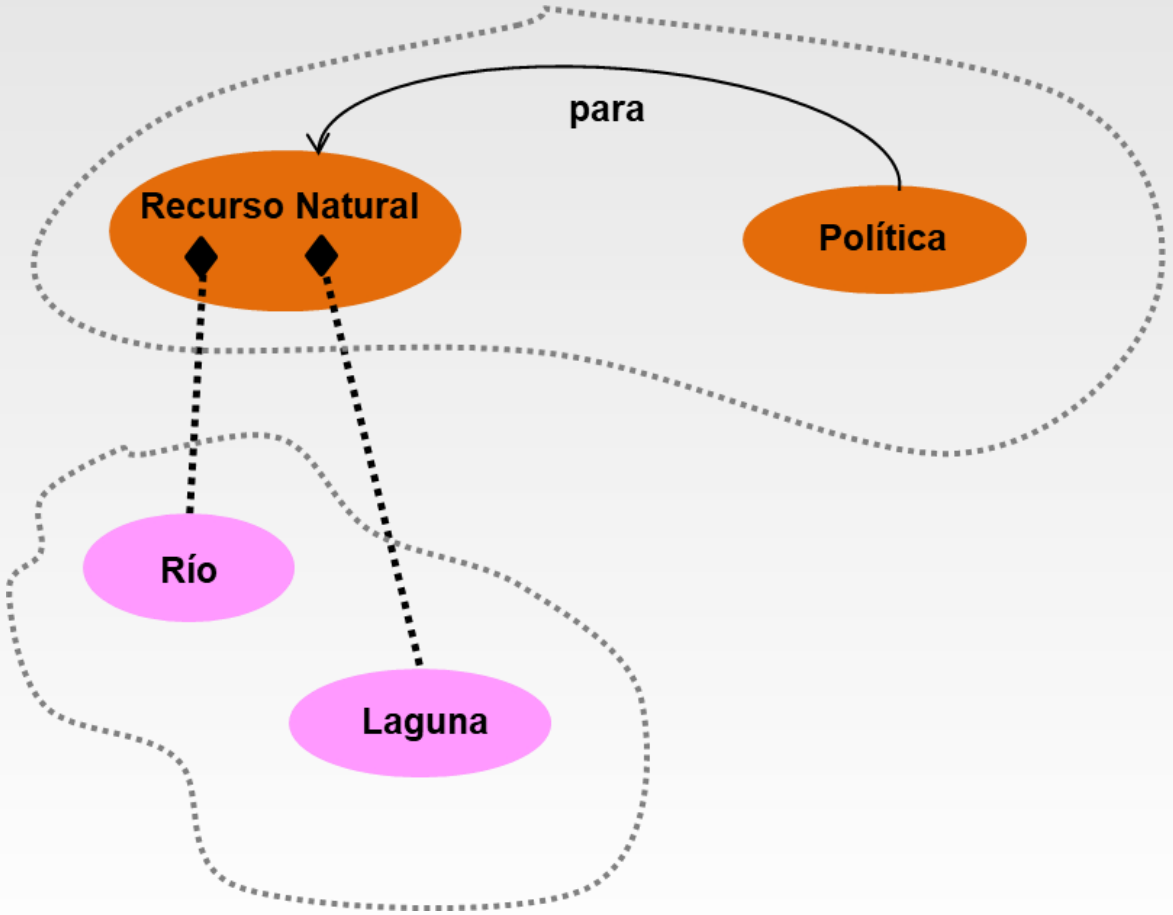
Lo que OWL no puede representar: Metamodelado



Desde la perspectiva del Ministerio de Ambiente:
río y laguna son instancias

Desde la perspectiva del experto en hidrografía:
Río y Laguna son conceptos

Pero la instancia río y el concepto Río son el mismo objeto del mundo real



Metamodelado

The expert defines work procedures

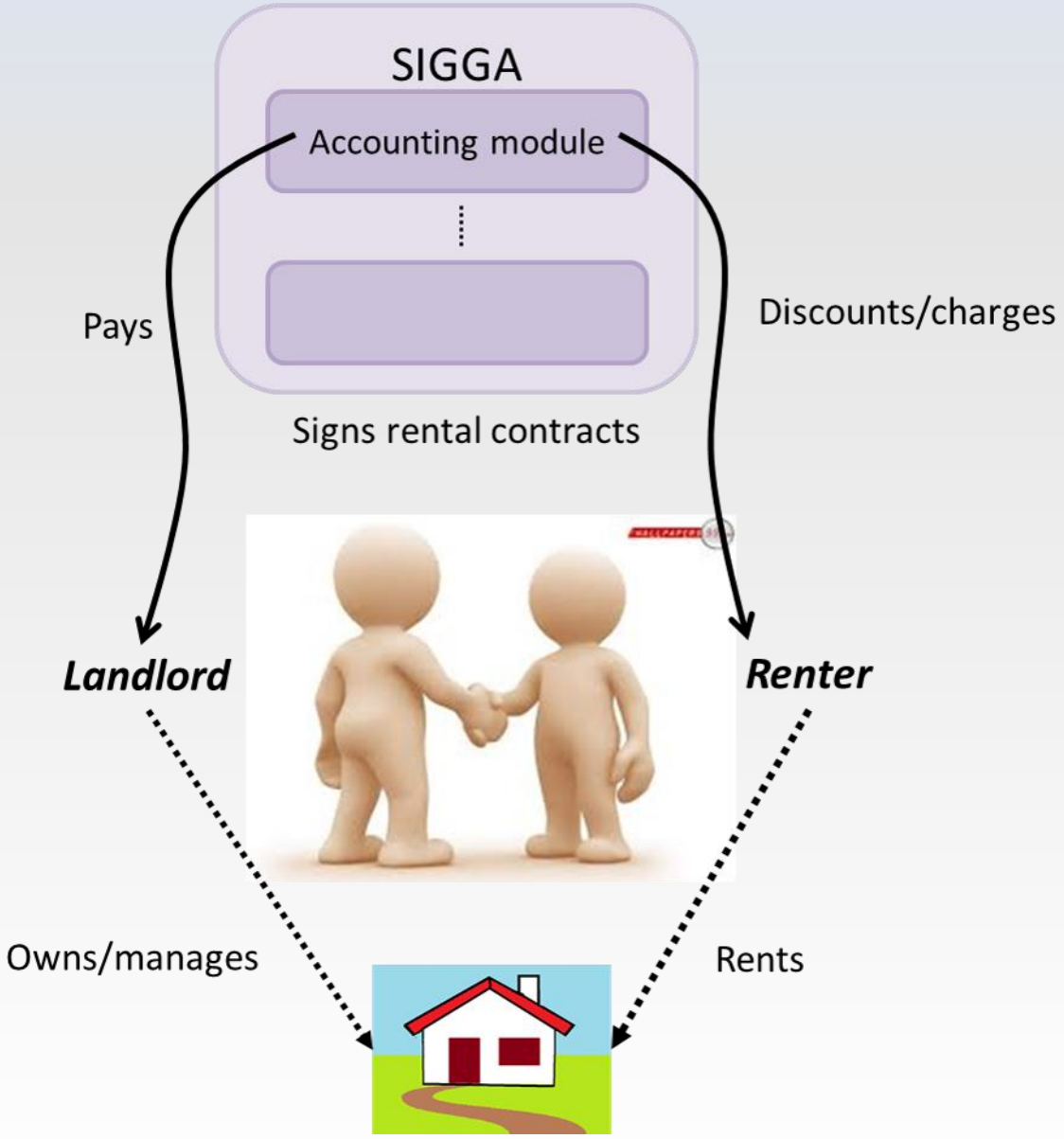
The operator executes each procedure several times

Metamodelado

The expert defines work procedures → instance

The operator executes each procedure several times → Class

Contabilidad – Garantía de alquileres



Contadores definen los tipos de asiento para registrar los movimientos contables:

- deuda de inquilino
- crédito al propietario
- pago del inquilino

Operadores registran y/o controlan asientos concretos:

- deuda de alquiler/desperfectos de Juan por $5000
- crédito a Pedro por $5000
- pago de $5000 de Juan

Contabilidad – Garantía de alquileres

EntryDefs

entryDef	description
10	Renter payment
20	Monthly calculation of rent
30	Home damage expenses

DetDefs

entryDef	detailDef	account	D/C
10	1	11111	'D'
10	2	11112	'D'
10	3	11211	'C'
10	4	11311	'C'
20	1	11211	'D'
20	2	11311	'D'
20	3	21111	'C'
30	1	12222	'D'
30	2	21111	'C'

Accounts

account	description
11111	Cash
11112	Bank
11211	Renter Debt
21111	Landlords
12222	Damage Expenses
11311	Renter Fee

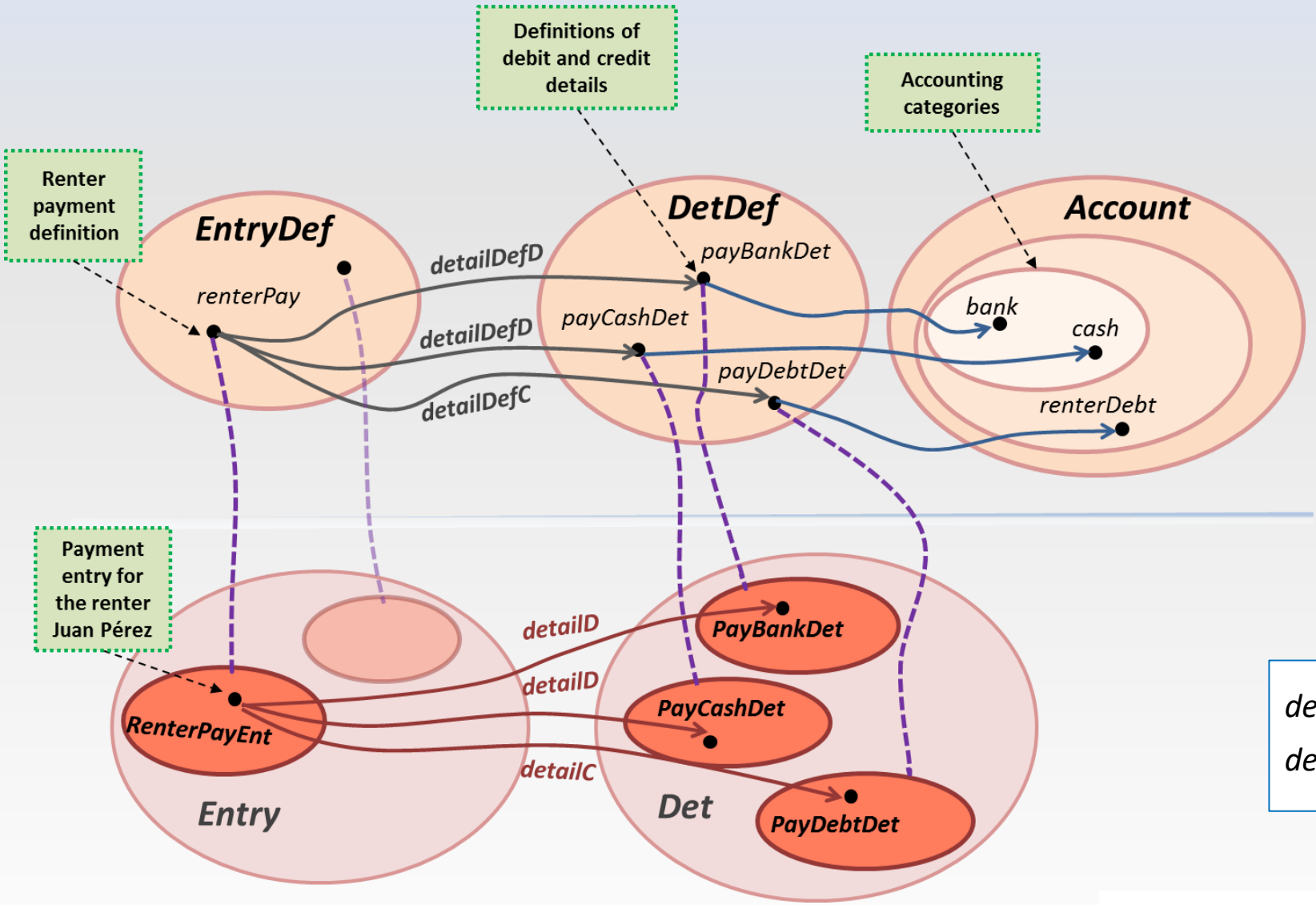
Entries

entry	entryDef	date	Observations
250	20	01/12/2017	Juan Pérez calc. of rent
251	10	13/12/2017	María García payment
252	10	14/12/2017	Juan Pérez payment

Dets

entry	detail	entryDef	detailDef	amount
250	1	20	1	4,500
250	2	20	2	1,500
250	3	20	3	6,000
252	1	10	1	6,000
252	2	10	3	4,500
252	3	10	4	1,500

Contabilidad – Garantía de alquileres



$renterPay =_m RenterPayEnt$

$payCashDet =_m PayCashDet$

MetaRule(*detailDefD, detailD*)

detailDefD(*renterPay, payBankDet*)

detailDefD(*renterPay, payCaskDet*)

$RenterPayEnt \sqsubseteq \forall detailD.(PayBankDet \sqcup PayCashDet)$

Metamodelado - Semántica

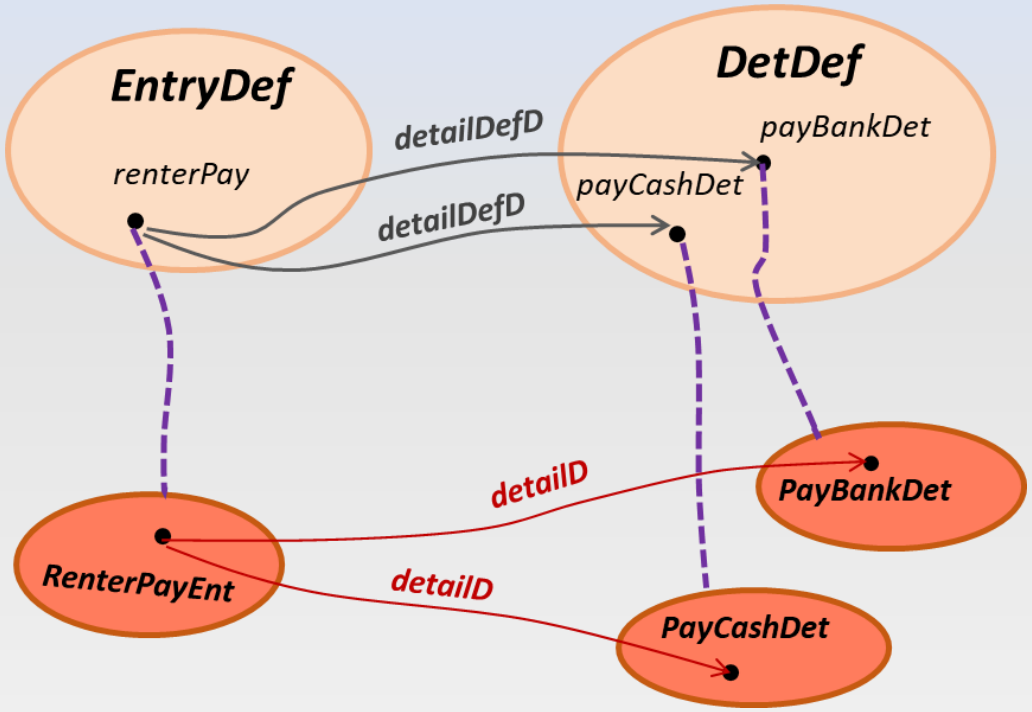
renterPay* =$_m$ *RenterPayEnt

payCashDet* =$_m$ *PayCashDet

$renterPay^I = RenterPayEnt^I$

$payCashDet^I = PayCashDet^I$

Si *renterPay* = *payCashDet* → $RenterPayEnt \equiv PayCashDet$



MetaRule(*detailDefD, detailD*) → $RenterPayEnt \sqsubseteq \forall detailD.(PayBankDet \sqcup PayCashDet)$

$RenterPayEnt^I \subseteq (\forall detailD.(PayBankDet \sqcup PayCashDet)^I$

Se extiende razonador Pellet para verificar consistencia con ***renterPay* =$_m$ *RenterPayEnt***

Se extiende razonador Hermit para verificar consistencia con ***renterPay* =$_m$ *RenterPayEnt y*** **MetaRule(*detailDefD, detailD*)**

Caso de uso contabilidad

Prototipo implementado usando la librería Java Swing

Usando la OWL API:

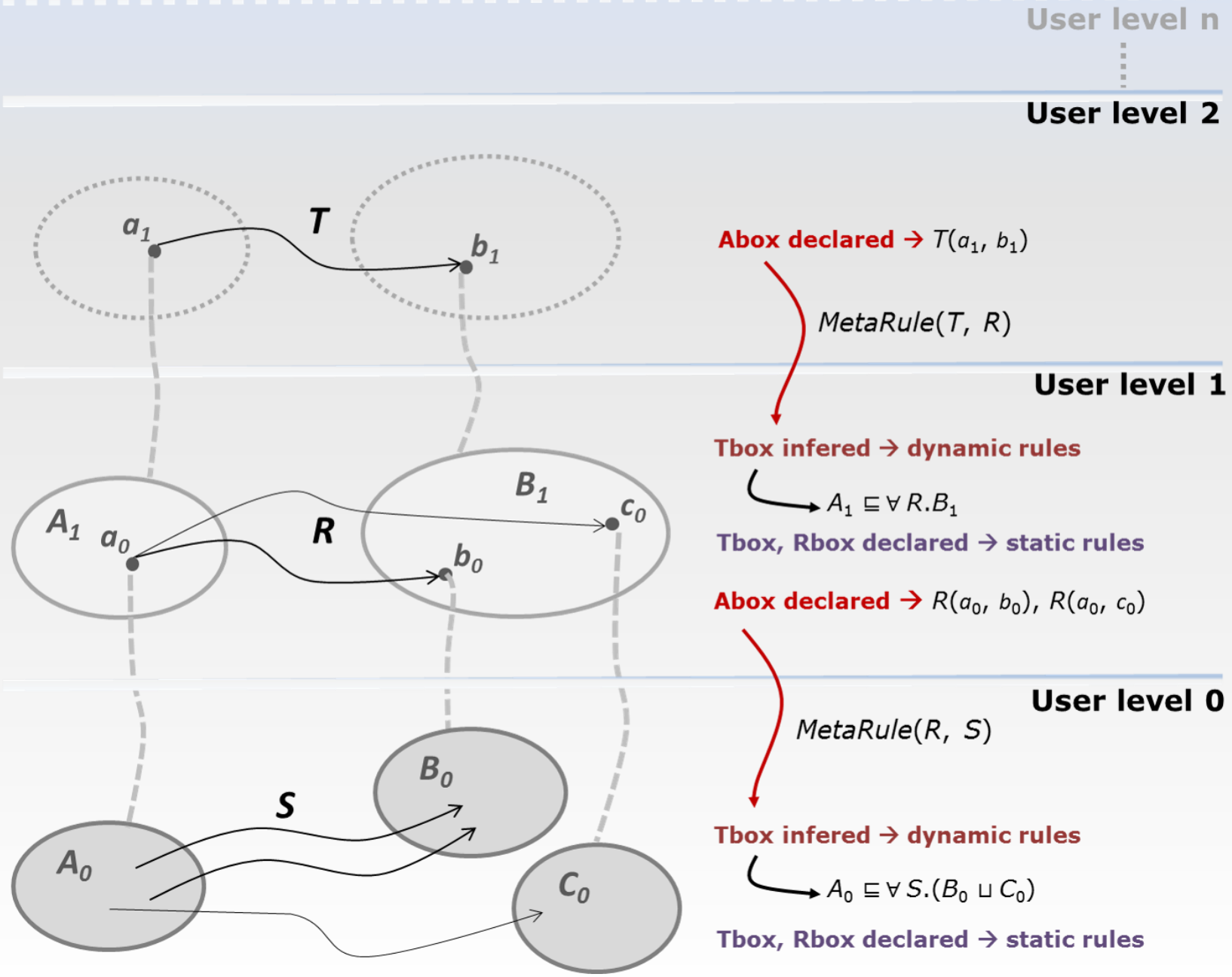
- Se carga la ontología de definiciones de asientos: Tbox
- Se carga la ontología de asientos: clases Entry, Det y las propiedades detailD y detailC de la Tbox
- Se cargan axiomas MetaRule
- A partir de los datos ingresados en la interfaz de definiciones:
  - Se puebla la ontología de definiciones con instancias
  - Se crean las subclases de Entry y Det y se ingresan axiomas =m que vinculan instancias de definiciones con subclases de Entry y Det
- Se ejecuta el razonador

Caso de uso contabilidad

DEMO

$\exists detailDefD.(\exists account.Availability) \sqsubseteq \exists detailDefC.(\exists account.(Asset \sqcap \neg Availability))$

Patrón de metamodelado de ontologías



Patrón de metamodelado de ontologías

Nombre del patrón	*Patrón de metamodelado de ontologías*
Propósito	Conceptualizar un dominio modelando dos o más niveles de conocimiento asociados a perspectivas de usuarios. Por cada nivel, existen requerimientos estáticos y requerimientos dinámicos que dependen del nivel inmediato superior.
Motivación	Escenarios: - Contabilidad (contadores, operadores) - Educación (universidad, docentes, estudiantes)
Aplicación	El patrón de metamodelado de ontologías se aplica en lugar de los patrones de modelado para un único nivel de conocimiento, en el caso que se requiera conceptualizar dos más niveles de conocimiento asociados a perspectivas de usuarios. Es una posible solución para representar las reglas de negocio en los diferentes niveles, que pueden ser estáticas (más estructurales, pocos cambios) o dinámicas (más sensibles a cambios en el negocio como la situación económica, nuevos líderes).
Solución propuesta	Dado el nivel de conocimiento *n* (figura anterior): • Las reglas estáticas se representan en la Tbox y Rbox del nivel *n*. • Las reglas dinámicas (restricciones en la relación entre clases A y B a través de la propiedad *S*) se representan ○ Axiomas de Abox en el nivel *n + 1*: *R*(*a*, *b*) ○ Axiomas de metamodelado: $a =_m A$, $b =_m B$, MetaRule(*R, S*) desde el nivel *n+1* al nivel *n*.
Consecuencias	Restricciones en el nivel *n,* ej. $\top \sqsubseteq \leq 1 detailDefD.(\exists account.Availaility)$, son reglas sobre las reglas del nivel *n -1*.
Limitaciones	El patrón de metamodelado de ontologías no resuelven reglas dinámicas en un nivel *n* diferentes de las definidas como relaciones entre individuos en el nivel n+1, que infieren restricciones en clases del nivel n.

Bibliografía

Regina Motz, Edelweis Rohrer, Paula Severi. *The Description Logic SHIQ Extended with a Flexible Meta-Modelling Hierarchy*. Journal of Web Semantics. Elsevier (2015).

Regina Motz, Edelweis Rohrer, Paula Severi and Ignacio Vidal. *OWL extended with metamodelling.* 3rd Workshop on Semantic Web and Open Linked Data., México 2015.

Paula Severi, Edelweis Rohrer, Regina Motz. *A description logic for unifying different points of view*. 1st Iberoamerican Knowledge Graphs and Semantic Web Conference, Villa Clara, Cuba, 2019.

Edelweis Rohrer, Paula Severi, Regina Motz. *Meta-modelling ontology design pattern*. 1st Iberoamerican Knowledge Graphs and Semantic Web Conference, Villa Clara, Cuba, 2019.

B. Di Bello. Extensión de Hermit para bases de conocimiento con metamodelado y aplicación al dominio de contabilidad. Proyecto de grado, 2021.