Simetrías en lógicas de descripción

Giovanni Rescia

FaMAFyC

1 de Junio, 2017

Introducción

- Introducción: intuición, conceptos básicos, qué se hizo en esta tesis
- Marco teórico del trabajo
- Detalles de implementación
- Evaluación empírica y análisis de resultados obtenidos

Introducción

- Concepto principal: simetría

Introducción

- Concepto principal: simetría
- Contexto: RA / LF

Introducción

- Concepto principal: simetría
- Contexto: RA / LF
- Ejemplo: SAT

Introducción

- Concepto principal: simetría
- Contexto: RA / LF
- Ejemplo: SAT
- Simetría: permutación de variables que mantiene modelos

Introducción

- Concepto principal: simetría
- Contexto: RA / LF
- Ejemplo: SAT
- Simetría: permutación de variables que mantiene modelos

$\varphi = (p \vee q)$

v	$v(\varphi)$	$\sigma(v)$
00	0	00
01	1	10
10	1	01
11	1	11

Introducción

- Concepto principal: simetría
- Contexto: RA / LF
- Ejemplo: SAT
- Simetría: permutación de variables que mantiene modelos

- En particular aquí: lógicas de descripción (DL)

$\varphi = (p \vee q)$

v	$v(\varphi)$	$\sigma(v)$
00	0	00
01	1	10
10	1	01
11	1	11

Introducción

- Concepto principal: simetría
- Contexto: RA / LF
- Ejemplo: SAT
- Simetría: permutación de variables que mantiene modelos

$\varphi = (p \vee q)$

v	$v(\varphi)$	$\sigma(v)$
00	0	00
01	1	10
10	1	01
11	1	11

- En particular aquí: lógicas de descripción (DL)
- Estudio: detección y uso de simetrías
  - Aquí: solo detección

Introducción

- Concepto principal: simetría
- Contexto: RA / LF
- Ejemplo: SAT
- Simetría: permutación de variables que mantiene modelos

$\varphi = (p \vee q)$

v	$v(\varphi)$	$\sigma(v)$
00	0	00
01	1	10
10	1	01
11	1	11

- En particular aquí: lógicas de descripción (DL)
- Estudio: detección y uso de simetrías
  - Aquí: solo detección
- Enfoque:

Introducción

- Concepto principal: simetría
- Contexto: RA / LF
- Ejemplo: SAT
- Simetría: permutación de variables que mantiene modelos

$\varphi = (p \vee q)$

v	$v(\varphi)$	$\sigma(v)$
00	0	00
01	1	10
10	1	01
11	1	11

- En particular aquí: lógicas de descripción (DL)
- Estudio: detección y uso de simetrías
  - Aquí: solo detección
- Enfoque:
  - Todo de cero

Introducción

- Concepto principal: simetría
- Contexto: RA / LF
- Ejemplo: SAT
- Simetría: permutación de variables que mantiene modelos

$\varphi = (p \vee q)$

v	$v(\varphi)$	$\sigma(v)$
00	0	00
01	1	10
10	1	01
11	1	11

- En particular aquí: lógicas de descripción (DL)
- Estudio: detección y uso de simetrías
  - Aquí: solo detección
- Enfoque:
  - Todo de cero
  - Aprovechando nexos y estudios previos en lógica modal

Introducción

- Concepto principal: simetría
- Contexto: RA / LF
- Ejemplo: SAT
- Simetría: permutación de variables que mantiene modelos

$\varphi = (p \vee q)$

v	$v(\varphi)$	$\sigma(v)$
00	0	00
01	1	10
10	1	01
11	1	11

- En particular aquí: lógicas de descripción (DL)
- Estudio: detección y uso de simetrías
  - Aquí: solo detección
- Enfoque:
  - Todo de cero
  - Aprovechando nexos y estudios previos en lógica modal

Lógicas de descripción

- Un subconjunto decidible de LPO

Lógicas de descripción

- Un subconjunto decidible de LPO
- DLs con diferentes expresividades

Lógicas de descripción

- Un subconjunto decidible de LPO
- DLs con diferentes expresividades
- Utilizadas para la representación de conocimiento

Lógicas de descripción

- Un subconjunto decidible de LPO
- DLs con diferentes expresividades
- Utilizadas para la representación de conocimiento
- Una base de conocimiento es una tupla <TBox, ABox> y define una ontología

Lógicas de descripción

- Un subconjunto decidible de LPO
- DLs con diferentes expresividades
- Utilizadas para la representación de conocimiento
- Una base de conocimiento es una tupla <TBox, ABox> y define una ontología
  - Una ontología puede pensarse como conocimiento jerárquico, como una generalización de una taxonomía

Lógicas de descripción

- Un subconjunto decidible de LPO
- DLs con diferentes expresividades
- Utilizadas para la representación de conocimiento
- Una base de conocimiento es una tupla <TBox, ABox> y define una ontología
  - Una ontología puede pensarse como conocimiento jerárquico, como una generalización de una taxonomía
- Expresamos estas ontologías en una computadora a través de OWL

Lógicas de descripción

- Un subconjunto decidible de LPO
- DLs con diferentes expresividades
- Utilizadas para la representación de conocimiento
- Una base de conocimiento es una tupla <TBox, ABox> y define una ontología
  - Una ontología puede pensarse como conocimiento jerárquico, como una generalización de una taxonomía
- Expresamos estas ontologías en una computadora a través de OWL
  - OWL es un formato estandar para expresar ontologías

Lógicas de descripción (cont.)

- Un lenguaje para definiciones de conceptos

Lógicas de descripción (cont.)

- Un lenguaje para definiciones de conceptos
  - TBox

Lógicas de descripción (cont.)

- Un lenguaje para definiciones de conceptos
  - TBox
    - $Persona \equiv Mujer \sqcup Hombre$

Lógicas de descripción (cont.)

- Un lenguaje para definiciones de conceptos
  - TBox
    - $Persona \equiv Mujer \sqcup Hombre$
    - $Padre \equiv Persona \sqcap \exists\, tieneHijo.Persona$

Lógicas de descripción (cont.)

- Un lenguaje para definiciones de conceptos
  - TBox
    - $Persona \equiv Mujer \sqcup Hombre$
    - $Padre \equiv Persona \sqcap \exists\, tieneHijo.Persona$
- Un lenguaje para aserciones

Lógicas de descripción (cont.)

- Un lenguaje para definiciones de conceptos
  - TBox
    - $Persona \equiv Mujer \sqcup Hombre$
    - $Padre \equiv Persona \sqcap \exists\, tieneHijo.Persona$
- Un lenguaje para aserciones
  - ABox

Lógicas de descripción (cont.)

- Un lenguaje para definiciones de conceptos
  - TBox
    - $Persona \equiv Mujer \sqcup Hombre$
    - $Padre \equiv Persona \sqcap \exists\, tieneHijo.Persona$
- Un lenguaje para aserciones
  - ABox
    - $tieneCapital(Italia, Roma)$

Lógicas de descripción (cont.)

- Un lenguaje para definiciones de conceptos
  - TBox
    - $Persona \equiv Mujer \sqcup Hombre$
    - $Padre \equiv Persona \sqcap \exists\, tieneHijo.Persona$
- Un lenguaje para aserciones
  - ABox
    - $tieneCapital(Italia, Roma)$
    - $Músico(Mozart)$

Lógicas de descripción (cont.)

- Un lenguaje para definiciones de conceptos
  - TBox
    - $Persona \equiv Mujer \sqcup Hombre$
    - $Padre \equiv Persona \sqcap \exists\, tieneHijo.Persona$
- Un lenguaje para aserciones
  - ABox
    - $tieneCapital(Italia, Roma)$
    - $Músico(Mozart)$
- Modelos relacionales

Lógicas modales

- Desarrollada en los 60

Lógicas modales

- Desarrollada en los 60
- Extiende a la lógica proposicional, incluyendo modalidades

Lógicas modales

- Desarrollada en los 60
- Extiende a la lógica proposicional, incluyendo modalidades
  - $[m_i]p \vee \langle m_j \rangle q$

Lógicas modales

- Desarrollada en los 60
- Extiende a la lógica proposicional, incluyendo modalidades
  - $[m_i]p \vee \langle m_j \rangle q$
- Utilizada para model checking

Lógicas modales

- Desarrollada en los 60
- Extiende a la lógica proposicional, incluyendo modalidades
  - $[m_i]p \vee \langle m_j \rangle q$
- Utilizada para model checking
- Un ejemplo particular de lógica modal: Lógica temporal

Lógicas modales

- Desarrollada en los 60
- Extiende a la lógica proposicional, incluyendo modalidades
  - $[m_i]p \vee \langle m_j \rangle q$
- Utilizada para model checking
- Un ejemplo particular de lógica modal: Lógica temporal
- Modelos relacionales (como en DL)

Lógicas modales

- Desarrollada en los 60
- Extiende a la lógica proposicional, incluyendo modalidades
  - $[m_i]p \vee \langle m_j \rangle q$
- Utilizada para model checking
- Un ejemplo particular de lógica modal: Lógica temporal
- Modelos relacionales (como en DL)

ML	DL
p_i	C_i, y C_i es un concepto atómico
$\neg\varphi$	$\neg C$, y C es un concepto
$\varphi \vee \Psi$	$C_j \sqcup C_i$, con Cj, C_i conceptos
$\langle R \rangle \varphi$	$\exists R.C_j$, y R es un rol atómico y C_j es un concepto

De OWL a ML

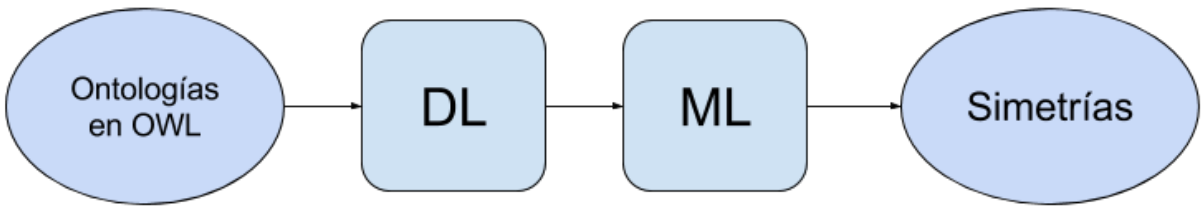
Ya vimos que existe un nexo sintáctico y semántico entre DL y ML

De OWL a ML

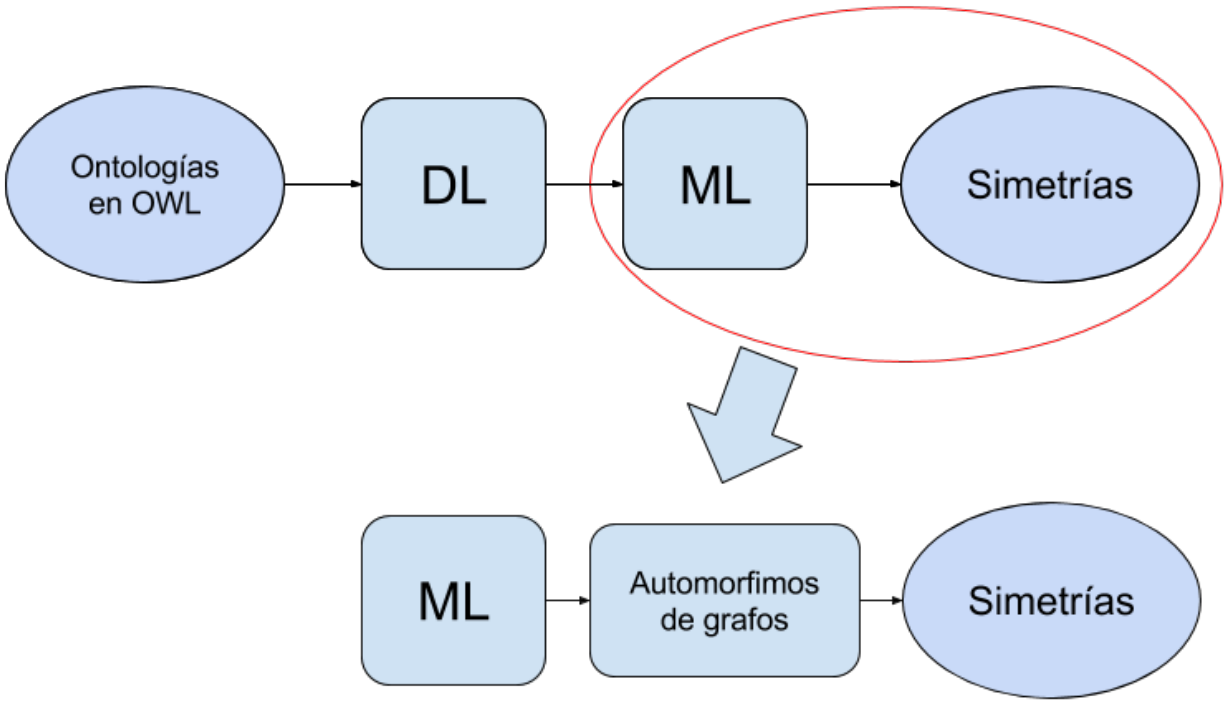
Ya vimos que existe un nexo sintáctico y semántico entre DL y ML

$$
\begin{aligned}
\Psi(\mathcal{R}\, someValuesFrom\, \mathcal{C}) &\doteq \langle \mathcal{R} \rangle \Psi(\mathcal{C}) \\
\Psi(complementOf(\mathcal{C})) &\doteq \neg \Psi(\mathcal{C}) \\
\Psi(SubClassOf(\mathcal{C}_1\, \mathcal{C}_2)) &\doteq A(\Psi(\mathcal{C}_1) \implies \Psi(\mathcal{C}_2)) \\
\Psi(EquivalentClasses(\mathcal{C}_1 \equiv \mathcal{C})) &\doteq \Psi(\mathcal{C}_1 \sqsubseteq \mathcal{C}_2) \wedge \Psi(\mathcal{C}_2 \sqsubseteq \mathcal{C}_1) \\
\Psi(unionOf(\mathcal{C}_1\, \mathcal{C}_2)) &\doteq \Psi(\mathcal{C}_1) \vee \Psi(\mathcal{C}_2)
\end{aligned}
$$

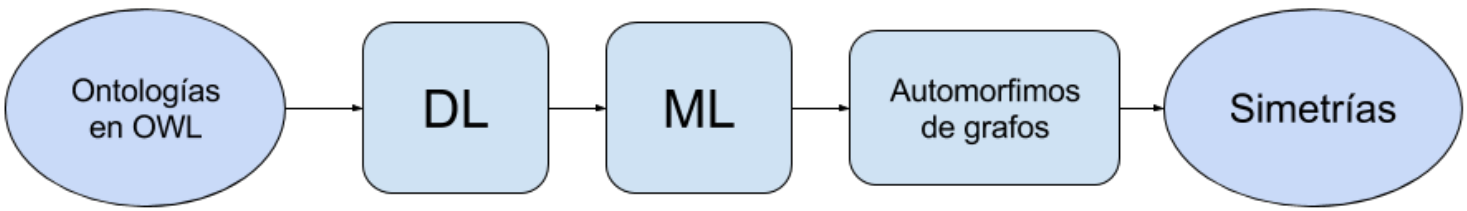
Simetrías



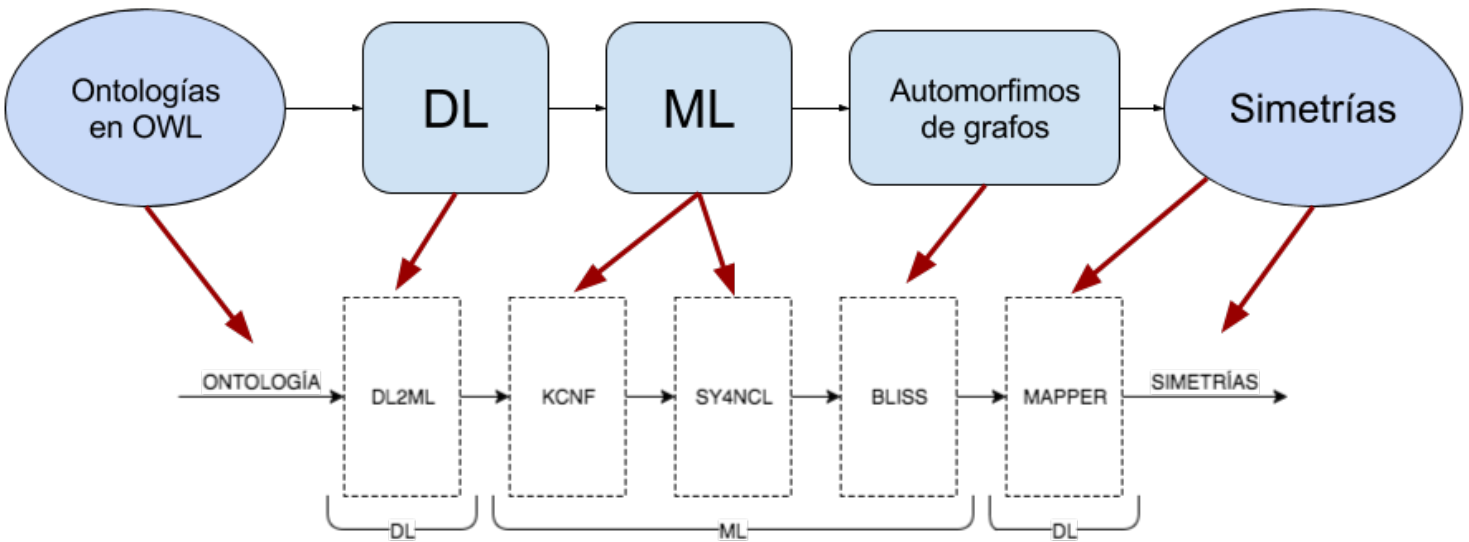
Simetrías


Ontologías en OWL
DL
ML
Simetrías
ML
Automorfimos de grafos
Simetrías

Pipeline de procesamiento



Pipeline de procesamiento



Ejemplo de base de conocimiento

TBox=

$$
\begin{aligned}
\mathit{gato} &\sqsubseteq \mathit{mamífero} \\
\mathit{perro} &\sqsubseteq \mathit{mamífero} \\
\mathit{caballo} &\sqsubseteq \mathit{mamífero} \\
\mathit{gatito} &\equiv \mathit{gato} \sqcap \exists\ \mathit{críaDe}.\mathit{gato} \\
\mathit{cachorro} &\equiv \mathit{perro} \sqcap \exists\ \mathit{críaDe}.\mathit{perro} \\
\mathit{potrillo} &\equiv \mathit{caballo} \sqcap \exists\ \mathit{críaDe}.\mathit{caballo} \\
\mathit{gato} &\sqsubseteq \mathit{mamífero} \sqcap \exists \mathit{cuadrúpedo}.\mathit{mamífero} \\
\mathit{perro} &\sqsubseteq \mathit{mamífero} \sqcap \exists \mathit{cuadrúpedo}.\mathit{mamífero} \\
\mathit{caballo} &\sqsubseteq \mathit{mamífero} \sqcap \exists \mathit{cuadrúpedo}.\mathit{mamífero}
\end{aligned}
$$

DL2ML

- Toma la ontología en OWL y devuelve la fórmula modal

DL2ML

- Toma la ontología en OWL y devuelve la fórmula modal
- Implementado para esta tesis

DL2ML

- Toma la ontología en OWL y devuelve la fórmula modal
- Implementado para esta tesis
- Programado en Scala

DL2ML

- Toma la ontología en OWL y devuelve la fórmula modal
- Implementado para esta tesis
- Programado en Scala
  - Con la API de Scowl (wrapper de la API de OWL)

DL2ML

- Toma la ontología en OWL y devuelve la fórmula modal
- Implementado para esta tesis
- Programado en Scala
  - Con la API de Scowl (wrapper de la API de OWL)
- Trabaja solo sobre la TBox de la ontología

DL2ML

- Toma la ontología en OWL y devuelve la fórmula modal
- Implementado para esta tesis
- Programado en Scala
  - Con la API de Scowl (wrapper de la API de OWL)
- Trabaja solo sobre la TBox de la ontología

```
<!-- http://www.famaf.unc.edu.ar/giovanni/test.owl#cachorro -->

<Class rdf:about="http://www.famaf.unc.edu.ar/giovanni/test.owl#cachorro">
    <equivalentClass>
        <Class>
            <intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="http://www.famaf.unc.edu.ar/giovanni/test.owl#perro"/>
                <Restriction>
                    <onProperty rdf:resource="http://www.famaf.unc.edu.ar/giovanni/test.owl#criaDe"/>
                    <someValuesFrom rdf:resource="http://www.famaf.unc.edu.ar/giovanni/test.owl#perro"/>
                </Restriction>
            </intersectionOf>
        </Class>
    </equivalentClass>
</Class>
```

```
begin (A(P8 --> P11));
((A(P5 --> (P8 ^ (<R1>P8)))) ^ (A((P8 ^ (<R1>P8)) --> P5)));
((A(P8 --> (P11 ^ (<R2>P11)))) ^ (A((P11 ^ (<R2>P11)) --> P8)));
...
((A(P3 --> (P11 ^ (<R2>P11)))) ^ (A((P11 ^ (<R2>P11)) --> P3)))
end
```

KCNF

- Retorna la CNF de una fórmula modal

KCNF

- Retorna la CNF de una fórmula modal
- Desarrollado en Haskell

KCNF

- Retorna la CNF de una fórmula modal
- Desarrollado en Haskell
- Adaptado para este proyecto

KCNF

- Retorna la CNF de una fórmula modal
- Desarrollado en Haskell
- Adaptado para este proyecto

```
begin (A(P8 --> P11));
((A(P5 --> (P8 ^ (<R1>P8)))) ^ (A((P8 ^ (<R1>P8)) --> P5)));
((A(P8 --> (P11 ^ (<R2>P11)))) ^ (A((P11 ^ (<R2>P11)) --> P8)));
...
((A(P3 --> (P11 ^ (<R2>P11)))) ^ (A((P11 ^ (<R2>P11)) --> P3)))
end
```

```
begin
A (P3 v -P17);
A (P7 v P11);
...
A (-P18 v -[R1]-P10);
A (-P19 v -[R2]-P11)
end
```

SY4NCL

- Crea el grafo de una fórmula en CNF

SY4NCL

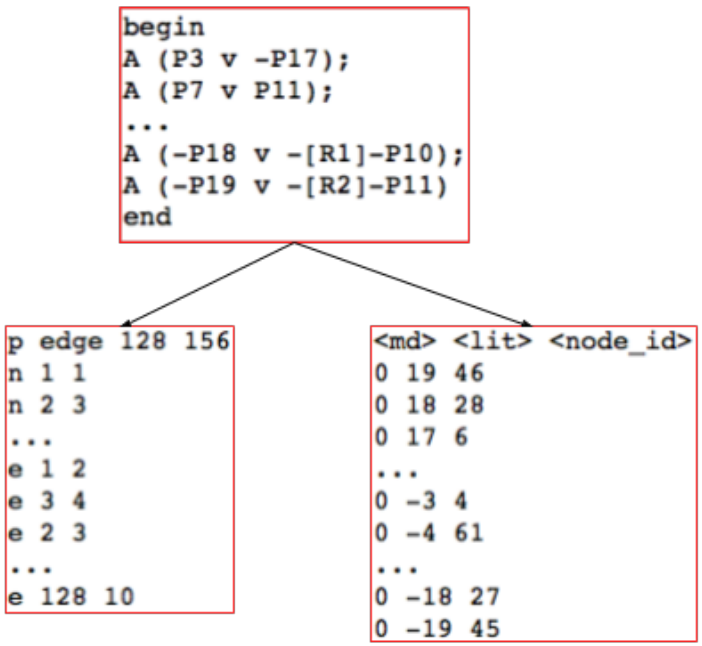
- Crea el grafo de una fórmula en CNF
- Desarrollado en Haskell

SY4NCL

- Crea el grafo de una fórmula en CNF
- Desarrollado en Haskell
- Adaptado para este proyecto

SY4NCL

- Crea el grafo de una fórmula en CNF
- Desarrollado en Haskell
- Adaptado para este proyecto



BLISS

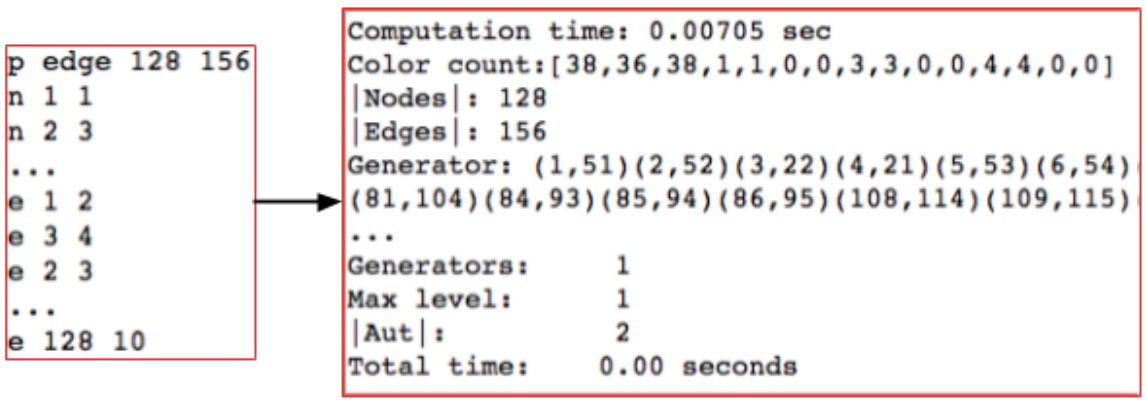
- Desarrollado en C++

BLISS

- Desarrollado en C++
- Automorfismos en grafos

BLISS

- Desarrollado en C++
- Automorfismos en grafos



MAPPER

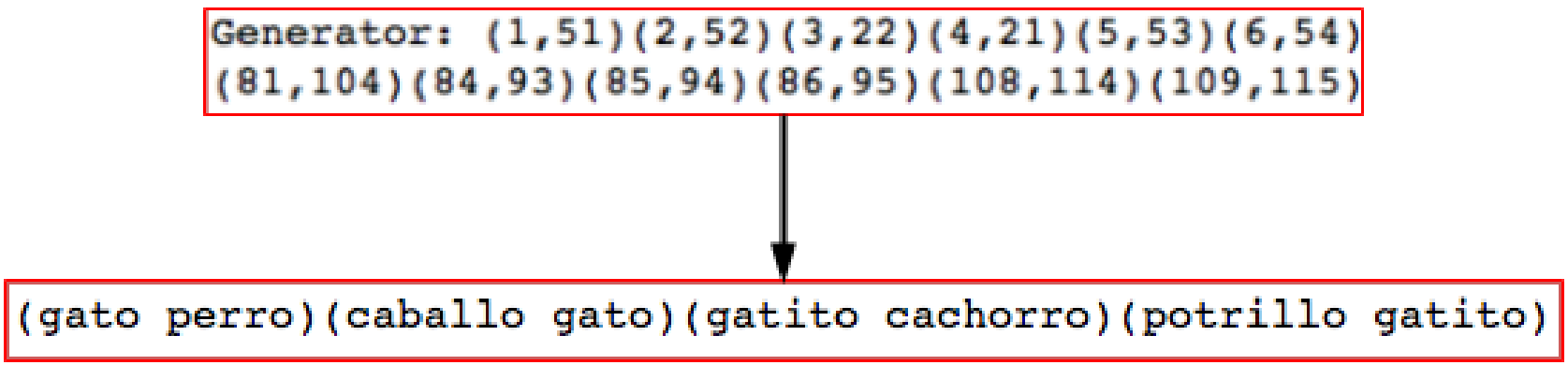
- Python, adaptado

MAPPER

- Python, adaptado
- Nodos del grafo -> variables proposicionales -> conceptos de la ontología

MAPPER

- Python, adaptado
- Nodos del grafo -> variables proposicionales -> conceptos de la ontología



Set de datos

- +20 Ontologías obtenidas en la web

Set de datos

- +20 Ontologías obtenidas en la web
- La mayoría utilizadas por razonadores

Set de datos

- +20 Ontologías obtenidas en la web
- La mayoría utilizadas por razonadores
- Diferentes expresividades

Set de datos

- +20 Ontologías obtenidas en la web
- La mayoría utilizadas por razonadores
- Diferentes expresividades
- Algunas muy conocidas

Set de datos

- +20 Ontologías obtenidas en la web
- La mayoría utilizadas por razonadores
- Diferentes expresividades
- Algunas muy conocidas
  - Go: vocabulario de términos sobre material bioquímico

Set de datos

- +20 Ontologías obtenidas en la web
- La mayoría utilizadas por razonadores
- Diferentes expresividades
- Algunas muy conocidas
  - Go: vocabulario de términos sobre material bioquímico
  - NCIT: vocabulario para la atención clínica, información pública, etc.

Set de datos

- +20 Ontologías obtenidas en la web
- La mayoría utilizadas por razonadores
- Diferentes expresividades
- Algunas muy conocidas
  - Go: vocabulario de términos sobre material bioquímico
  - NCIT: vocabulario para la atención clínica, información pública, etc.
  - Uberon: ontología de anatomía para representar partes del cuerpo

Resultados

- Go
  - Tamaño de la TBox: 104783
  - Tamaño del grafo: Nodos: 432315, Aristas: 528394
  - Generadores detectados: 8496
  - Tiempo para la detección de automorfismos: 57.93 seg.

Resultados

- Go
  - Tamaño de la TBox: 104783
  - Tamaño del grafo: Nodos: 432315, Aristas: 528394
  - Generadores detectados: 8496
  - Tiempo para la detección de automorfismos: 57.93 seg.
- NCIT
  - Tamaño de la TBox: 46940
  - Tamaño del grafo: Nodos: 164267, Aristas: 183694
  - Generadores detectados: 10218
  - Tiempo para la detección de automorfismos: 37.24 seg.

Resultados

- Go
  - Tamaño de la TBox: 104783
  - Tamaño del grafo: Nodos: 432315, Aristas: 528394
  - Generadores detectados: 8496
  - Tiempo para la detección de automorfismos: 57.93 seg.
- NCIT
  - Tamaño de la TBox: 46940
  - Tamaño del grafo: Nodos: 164267, Aristas: 183694
  - Generadores detectados: 10218
  - Tiempo para la detección de automorfismos: 37.24 seg.
- Uberon
  - Tamaño de la TBox: 25683
  - Tamaño del grafo: Nodos: 107903, Aristas: 135288
  - Generadores detectados: 818
  - Tiempo para la detección de automorfismos: 6.18 seg.

Análisis

- Simetrías en todas las ontologías utilizadas
- Tiempo despreciable
- Memoria despreciable

Trabajo futuro

- Solo trabajamos con la TBox de cada ontología
  - Extender Ψ para que traduzca la ABox también
- Implementar las simetrías en los razonadores para en cuánto mejora el rendimiento de los mismos

Outro

Preguntas?