Description Logics:
a Nice Family of Logics

Day 5: More Complexity & Justifications

ESSLLI 2016

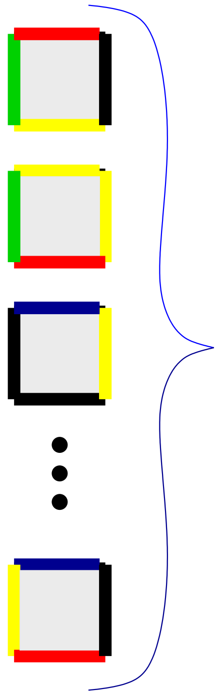
Uli Sattler and Thomas Schneider

Next

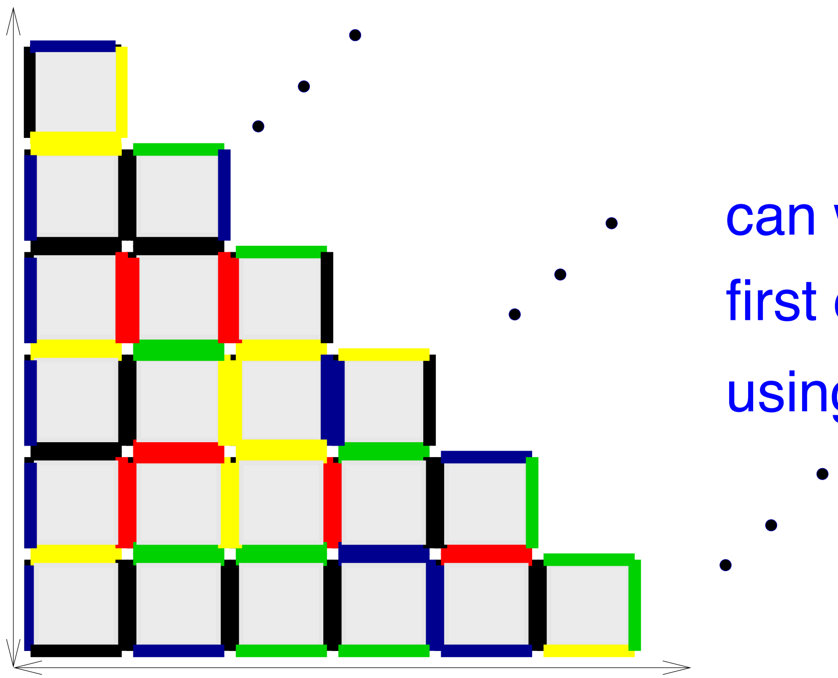
Some

- undecidability results: closing grid makes a DL undecidable
- complexity results: when 3 constructors interact badly and lead to NExpTime-hardness
- justifications: explaining and debugging entailments

The Classical Domino Problem ✔



D, a fixed set of dominoe types



can we tile the first quadrant using D?

The Classical Domino Problem ✔

Definition: A **domino system** $\mathcal{D} = (D, H, V)$

- set of domino **types** $D = \{D_1, \ldots, D_d\}$, and
- horizontal and vertical **matching conditions** $H \subseteq D \times D$ and $V \subseteq D \times D$

A **tiling** for $\mathcal{D}$ is a (total) function:

$$
\begin{aligned}
t : \mathbb{N} \times \mathbb{N} \rightarrow\ & D \text{ such that} \\
& \langle t(m,n), t(m+1,n) \rangle \in H \text{ and} \\
& \langle t(m,n), t(m,n+1) \rangle \in V
\end{aligned}
$$

Domino problem: given $\mathcal{D}$, has $\mathcal{D}$ a tiling?

It is well-known that this problem is undecidable [Berger66]

Almost Encoding the Classical Domino Problem in $\mathcal{ALC}$ ✔

For our reduction, we express various **obligations** of the domino problem in $\mathcal{ALC}$ TBox axioms:

① each element carries **exactly one domino type** D_i

$\rightsquigarrow$ use unary predicate symbol D_i for each domino type and

$$
\begin{array}{rcll}
\top & \sqsubseteq & D_1 \sqcup \ldots \sqcup D_d & \text{\% each element carries a domino type} \\
D_1 & \sqsubseteq & \neg D_2 \sqcap \ldots \sqcap \neg D_d & \text{\% but not more than one} \\
D_2 & \sqsubseteq & \neg D_3 \sqcap \ldots \sqcap \neg D_d & \text{\% ...} \\
\vdots & & \vdots & \\
D_{d-1} & \sqsubseteq & \neg D_d &
\end{array}
$$

Almost Encoding the Classical Domino Problem in $\mathcal{ALC}$ ✔

② every element has a horizontal (X-) successor and a vertical (Y-) successor

$$\top \sqsubseteq \exists X.\top \sqcap \exists Y.\top$$

③ every element satisfies D's **horizontal/vertical matching conditions**:

$$\begin{aligned}
D_1 &\sqsubseteq \bigsqcup_{(D_1,D)\in H} \forall X.D \sqcap \bigsqcup_{(D_1,D)\in V} \forall Y.D\\
D_2 &\sqsubseteq \bigsqcup_{(D_2,D)\in H} \forall X.D \sqcap \bigsqcup_{(D_2,D)\in V} \forall Y.D\\
&\vdots\\
D_d &\sqsubseteq \bigsqcup_{(D_d,D)\in H} \forall X.D \sqcap \bigsqcup_{(D_d,D)\in V} \forall Y.D
\end{aligned}$$

Does this suffice?

No: if yes, $\mathcal{ALC}$ would be undecidable!

Encoding the Classical Domino Problem in $\mathcal{ALC}$ with role chain inclusions ✔

④ for each element, its horizontal-vertical-successors **coincide** with their vertical-horizontal-successors & vice versa

$$X \circ Y \sqsubseteq Y \circ X \text{ and } Y \circ X \sqsubseteq X \circ Y$$

Lemma: Let $\mathcal{T}_D$ be the set of axioms ① to ④.
Then $\top$ is satisfiable w.r.t. $\mathcal{T}_D$ iff $\mathcal{D}$ has a tiling.

- since the domino problem is undecidable, this implies undecidability of concept satisfiability w.r.t. TBoxes of $\mathcal{ALC}$ with role chain inclusions
- due to Theorem 2, all other standard reasoning problems are undecidable, too
- Proof:
  1. show that, from a tiling for D, you can construct a model of $\mathcal{T}_D$
  2. show that, from a model $\mathcal{I}$ of $\mathcal{T}_D$, you can construct a tiling for D (tricky because elements in $\mathcal{I}$ can have several X- or Y-successors but we can simply take **the right ones**...)

Let's do this again!

We only need $\mathcal{ALC}$ for ①-③.
What other constructors can us help to express ④?

A weak form of counting: $(\leq 1 r)^{\mathcal{I}} = \{x \mid$ there is at most one y with $(x, y) \in r^{\mathcal{I}}\}$

- counting and complex roles (role chains and role intersection):

$$\top \sqsubseteq (\leq 1 X) \sqcap (\leq 1 Y) \sqcap (\exists (X \circ Y) \sqcap (Y \circ X).\top)$$

- restricted role chain inclusions (only 1 role on RHS), and counting (an **all** roles):

$$\begin{aligned} \top &\sqsubseteq (\leq 1 X) \sqcap (\leq 1 Y) \\ X \circ Y &\sqsubseteq r \\ Y \circ X &\sqsubseteq r \\ \top &\sqsubseteq (\leq 1 r) \end{aligned}$$

- various others...

Are all DLs in ExpTime?

Earlier, we have claimed that $\mathcal{ALCQI}$, $\mathcal{ALCQO}$, and $\mathcal{ALCIO}$ are all **ExpTime**-complete, i.e., as hard/easy as $\mathcal{ALC}$

Next, we will see that consistency of $\mathcal{ALCQIO}$ ontologies, the extension of $\mathcal{ALC}$ with

- **inverse roles** r^- with $(r^-)^\mathcal{I} = \{(y,x) \mid (x,y) \in r^\mathcal{I}\}$
- the weakest **number restrictions** $(\leq 1r)$ with $(\leq 1r)^\mathcal{I} = \{x \mid \text{there is at most 1 } y \text{ with } (x,y) \in r^\mathcal{I}\}$
- **nominals** $\{a\}$ with $(\{a\})^\mathcal{I} = \{a^\mathcal{I}\}$

$\Rightarrow$ is harder, namely **NExpTime-hard**

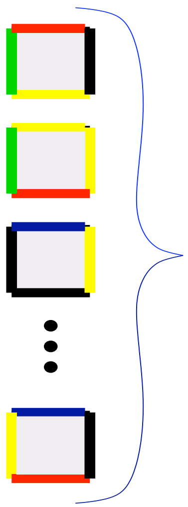
- this is typical phenomenon where
  - **combination** of otherwise harmless constructors leads to increased complexity

$\mathcal{ALCQIO}$ is NExpTime-hard

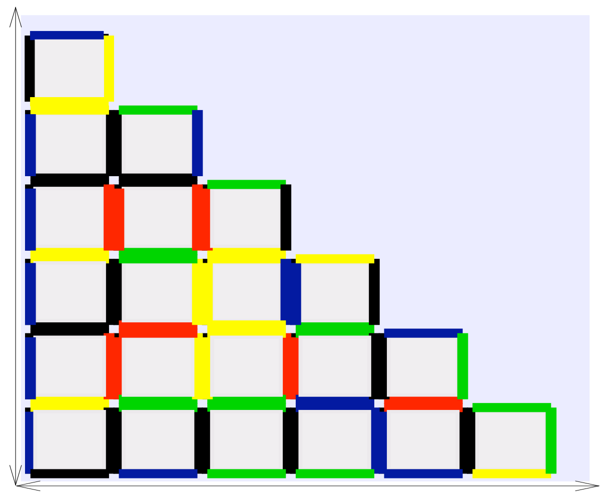
We follow hardness proof recipe:

- to show that consistency of $\mathcal{ALCQIO}$ ontologies is **NExpTime-hard**, we
  - find a suitable problem $P' \subseteq M'$ that is **known to be NExpTime-hard** and
  - a reduction from P' to P

The **NExpTime** version of the **domino problem**



D, a fixed set of dominoe types



can we tile a 2^n x 2^n squar using *D*?

Domino Problems

Definition: A **domino system** $\mathcal{D} = (D, H, V)$

- set of domino **types** $D = \{D_1, \ldots, D_d\}$, and
- horizontal and vertical **matching conditions** $H \subseteq D \times D$ and $V \subseteq D \times D$

A **tiling** for $\mathcal{D}$ is a function:

$$t : \mathbb{N} \times \mathbb{N} \to D \text{ such that } \begin{array}{l} \langle t(m,n), t(m+1,n) \rangle \in H \text{ and} \\ \langle t(m,n), t(m,n+1) \rangle \in V \end{array}$$

Domino problems: ✔ **classical** given $\mathcal{D}$, has $\mathcal{D}$ a tiling?

⇒ well-known that this problem is **undecidable** [Berger66]

☞ **NexpTime** given $\mathcal{D}$, has $\mathcal{D}$ a tiling for $2^n \times 2^n$ square?

⇒ well-known that this problem is **NExpTime-hard**

Reduction of NExpTime Domino Problem to $\mathcal{ALCQIO}$ Consistency

To reduce the NExpTime domino problem to $\mathcal{ALCQIO}$ consistency, we need to

- define a mapping π from **domino problems** to $\mathcal{ALCQIO}$ **ontologies** such that
- D has an $2^n \times 2^n$ mapping iff $\pi(D)$ is consistent and
- size of $\pi(D)$ is polynomial in n

Mapping a Domino System into an $\mathcal{ALCQIO}$ Ontology

Again, we express various **obligations** of the domino problem in $\mathcal{ALC}$ axioms:

① each element carries **exactly one domino type** D_i

⇝ use unary predicate symbol D_i for each domino type and

$$
\begin{array}{rcll}
\top & \sqsubseteq & D_1 \sqcup \ldots \sqcup D_d & \text{\% each element carries a domino type} \\
D_1 & \sqsubseteq & \neg D_2 \sqcap \ldots \sqcap \neg D_d & \text{\% but not more than one} \\
D_2 & \sqsubseteq & \neg D_3 \sqcap \ldots \sqcap \neg D_d & \text{\% ...} \\
\vdots & & \vdots & \\
D_{d-1} & \sqsubseteq & \neg D_d &
\end{array}
$$

Mapping a Domino System into an $\mathcal{ALCQIO}$ Ontology

② every element has a horizontal (X-) successor and a vertical (Y-) successor

$$\top \sqsubseteq \exists X.\top \sqcap \exists Y.\top$$

③ every element satisfies D's **horizontal/vertical matching conditions**:

$$\begin{array}{lcl}
D_1 & \sqsubseteq & \bigsqcup_{(D_1,D)\in H} \forall X.D \sqcap \bigsqcup_{(D_1,D)\in V} \forall Y.D \\
D_2 & \sqsubseteq & \bigsqcup_{(D_2,D)\in H} \forall X.D \sqcap \bigsqcup_{(D_2,D)\in V} \forall Y.D \\
\vdots & \vdots & \\
D_d & \sqsubseteq & \bigsqcup_{(D_d,D)\in H} \forall X.D \sqcap \bigsqcup_{(D_d,D)\in V} \forall Y.D
\end{array}$$

Does this suffice?

I.e., does D have a $2^n \times 2^n$ tiling iff one D_i is satisfiable w.r.t. ① to ③?

- if yes, we have shown that satisfiability of $\mathcal{ALC}$ is NExpTime-hard
- so no...what is missing?

Mapping a Domino System into an $\mathcal{ALCQIO}$ Ontology

Two things are missing:

1. the model must be large enough, namely $2^n \times 2^n$ and
2. for each element, its horizontal-vertical-successors **coincide** with their vertical-horizontal-successors and vice versa

This will be addressed using a "**counting and binding together**" trick ...

④ counting and binding together

(a) use $A_1, \ldots, A_n$, $B_1, \ldots, B_n$ as "bits" for binary representation of **grid position**
e.g., (010, 011) is represented by an instance of $\neg A_3, A_2, \neg A_1, \neg B_3, B_2, B_1$

write GCI to ensure that X- and Y-successors are **incremented** correctly
e.g., X-successor of (010, 011) is (011, 011)

(b) use nominals to ensure that there is only one (111...1, 111...1)
this implies, with $\top \sqsubseteq (\leq 1\ X^-.\top) \sqcap (\leq 1\ Y^-.\top)$ **uniqueness** of grid positions

④ counting and binding together

(a) $\tilde{A}_i$ for "**bit A_i is incremented correctly**":

$$\top \sqsubseteq \tilde{A}_1 \sqcap \ldots \sqcap \tilde{A}_n$$

$$\tilde{A}_1 \sqsubseteq (A_1 \sqcap \forall X.\neg A_1) \sqcup (\neg A_1 \sqcap \forall X.A_1)$$

$$\tilde{A}_i \sqsubseteq (\bigsqcap_{\ell<i} A_\ell \sqcap ((A_i \sqcap \forall X.\neg A_i) \sqcup (\neg A_i \sqcap \forall X.A_i)) \sqcup (\neg \bigsqcap_{\ell<i} A_\ell \sqcap ((A_i \sqcap \forall X.A_i) \sqcup (\neg A_i \sqcap \forall X.\neg A_i))$$

(add the same for the B_is and Y)

(b) ensure uniqueness of grid positions:

$$A_1 \sqcap \ldots \sqcap A_n \sqcap B_1 \sqcap \ldots \sqcap B_n \sqsubseteq \{o\}$$ % top right $(2^n, 2^n)$ is unique

$$\top \sqsubseteq (\leq 1\; X^-.\top) \sqcap (\leq 1\; Y^-.\top)$$ % everything else is also unique

Reduction of NExpTime Domino Problem to $\mathcal{ALCQIO}$ Consistency

Since the NExpTime-domino problem is NExpTime-hard, this implies consistency of $\mathcal{ALCQIO}$ is also NExpTime-hard:

Lemma: let $\mathcal{O}_D$ be ontology consisting of all axioms mentioned in reduction of D:

- D has an $2^n \times 2^n$ tiling iff $\mathcal{O}_D$ is consistent
- size of $\mathcal{O}_D$ is polynomial (quadratic) in
  - the size of D and
  - n

Are Standard Reasoning Problems/Services Everything?

So far, we have talked a lot about **standard reasoning problems**

- consistency
- satisfiability
- entailments
- ...is this all that is relevant?

Next, we will look at 1 reasoning problem that

- cannot be polynomially reduced to any of the above standard reasoning problems
- is relevant when working with a non-trivial ontology
- ...justifications!

Imagine you are building, possibly with your colleagues, an ontology $\mathcal{O}$:

non-trivial, with say 500 axioms, or 5,000 (NCI has $\geq$ 300,000)

(S1) $\mathcal{O} \models C \sqsubseteq \bot$ and you want to know why

(S2) 27 classes C_i are unsatisfiable w.r.t. $\mathcal{O}$

– imagine $\mathcal{O}$ is coherent, but $\mathcal{O} \cup \{\alpha\}$ contains 27 unsatisfiable classes

– ...even for a very sensible, small, harmless axiom α

(S3) $\mathcal{O}$ is inconsistent

– imagine $\mathcal{O}$ is consistent, but $\mathcal{O} \cup \{\alpha\}$ is inconsistent

– ...even for a very sensible, small, harmless axiom α

? what do you do?

? how do you go about **repairing** $\mathcal{O}$?

? which tool support would help you to **repair** $\mathcal{O}$?

Building Ontologies for Real II

Imagine you are building, possibly with your colleagues, an ontology $\mathcal{O}$:

non-trivial, with say 500 axioms, or 5,000 (NCI has $\geq$ 300,000)

(S4) $\mathcal{O} \models \alpha$, and you want to know **why**

- e.g., so that you can trust $\mathcal{O}$ and α
- e.g., so that you understand how $\mathcal{O}$ models its domain

? what do you do?

? how do you go about **understanding** this entailment?

? which tool support would help you to **understand** this entailment?

? would this tool support be the same/similar to the one to support repair?

In all scenarios (Si), we clearly want to know at least the **reasons** for $\mathcal{O} \models \alpha$,

which axioms can I/should I

(S1) **change** so that C' becomes satisfiable w.r.t. $\mathcal{O}'$?

(S2) **change** so that $\mathcal{O}'$ becomes coherent?

(S3) **change** so that $\mathcal{O}'$ becomes consistent?

(S4) **look** at to understand $\mathcal{O} \models \alpha$?

Definition: Let $\mathcal{O}$ be an ontology with $\mathcal{O} \models \alpha$.
Then $\mathcal{J} \subseteq \mathcal{O}$ is a **justification for** α **in** $\mathcal{O}$ if

- $\mathcal{J} \models \alpha$ and
- $\mathcal{J}$ is minimal, i.e., for each $\mathcal{J}' \subsetneq \mathcal{J}$: $\mathcal{J}' \not\models \alpha$

An Example

Consider the following ontology $\mathcal{O}$ with $\mathcal{O} \models C \sqsubseteq \bot$:

$$\begin{aligned}
\mathcal{O} := \{C &\sqsubseteq D \sqcap E & (1)\\
D &\sqsubseteq A \sqcap \exists r.B_1 & (2)\\
E &\sqsubseteq A \sqcap \forall r.B_2 & (3)\\
B_1 &\sqsubseteq \neg B_2 & (4)\\
D &\sqsubseteq \neg E & (5)\\
G &\sqsubseteq B \sqcap \exists s.C\} & (6)
\end{aligned}$$

Find a justification for $C \sqsubseteq \bot$ in $\mathcal{O}$.
How many justifications are there?

More about Justifications

Facts:

1. for each entailment of $\mathcal{O}$, there exists at least one justification
2. one entailment can have several justifications in $\mathcal{O}$
3. justifications can overlap
4. let $\mathcal{O}'$ be obtained as follows from $\mathcal{O}$ with $\mathcal{O} \models \alpha$:
   - for each justification $\mathcal{J}_i$ of the n justifications for α in $\mathcal{O}$, pick some $\beta_i \in \mathcal{J}_i$
   - set $\mathcal{O}' := \mathcal{O} \setminus \{\beta_1, \ldots, \beta_n\}$

   then $\mathcal{O}' \not\models \alpha$, i.e., $\mathcal{O}'$ is a **repair** of $\mathcal{O}$.
5. if $\mathcal{J}$ is a justification for α and $\mathcal{O}' \supseteq \mathcal{J}$, then $\mathcal{O}' \models \alpha$.
   Hence any repair of α must touch **all** justifications.
6. if $\mathcal{O} \models \alpha$, $\mathcal{O} \models \beta$, and
   $\forall$ justification $\mathcal{J}$ for α $\exists$ a justification $\mathcal{J}'$ for β with $\mathcal{J}' \subseteq \mathcal{J}$,
   then repairing β repairs α.

A Naive Black-Box Algorithm to Compute Justifications

Let $\mathcal{O} = \{\beta_1, \ldots, \beta_m\}$ be an ontology with $\mathcal{O} \models \alpha$.

Get1Just($\mathcal{O}$, α)
Set $\mathcal{J} := \mathcal{O}$ and **Out** $:= \emptyset$
For each $\beta \in \mathcal{O}$
    If $\mathcal{J} \setminus \{\beta\} \models \alpha$ then
        Set $\mathcal{J} := \mathcal{J} \setminus \{\beta\}$ and **Out** $:=$ **Out** $\cup \{\beta\}$
Return $\mathcal{J}$

Claim:

- loop invariants: $\mathcal{J} \models \alpha$ and $\mathcal{O} = \mathcal{J} \cup$ **Out**
- Get1Just(,) returns 1 justification for α in $\mathcal{O}$
- it requires m entailment tests

Other approaches to computing justifications exists, more performant, glass-box (inside reasoner) and black-box (outside).

Linking Justifications to our Scenarios

$(\mathrm{S4})$ 1 justification suffices, but which? A good, easy one...how to find?

$(\mathrm{S1\text{-}S3})$ require the computation of **all** justifications, possibly for several entailments

- even for one entailment, search space is exponential

$(\mathrm{S2})$ requires even more:

- who wants to look at $x \times 27$ justifications? Where to start?

$\Rightarrow$ A justification $\mathcal{J}$ (for α) is **root** if there is no justification $\mathcal{J}'$ with $\mathcal{J}' \subsetneq \mathcal{J}$

- **start** with root justifications, remove/change axioms in them and
- **reclassify**: you might have repaired several unsatisfiabilities at once!

- Check example on slide 6: both justifications for $C \sqsubseteq \bot$ are root, contained in 2 non-root justifications for $G \sqsubseteq \bot$
- repairing $C \sqsubseteq \bot$ repairs $G \sqsubseteq \bot$

More About Justifications

BOs: NCBO BioPortal, a repository of 250 ontologies, very varied, not cherry-picked

- recent, optimised implementation of GetAllJust($\mathcal{O}$, α)
  - behave well in practise
  - can compute one justification for all atomic entailments of **BO**s
  - can compute (almost) all justifications for (almost) all atomic entailments of **BO**s
- recent surveys show that **BO**s have entailments
  - with **large** justifications, e.g., with 37 axioms and
  - with **numerous** justifications, e.g., one entailment had 837 justifications
  - for which justifications can often be understood well by **domain experts**
  - ...for more, see Horridge's dissertation

- some justification contain **superfluous parts**
  - that distract the user
  - see example on slide 6
  - identifying these can help user to focus on the **relevant** parts
  - this has led to investigation of **laconic and precise justifications**

- there are still some **hard** justifications that need further explanation
  - e.g., consider $O = \{$

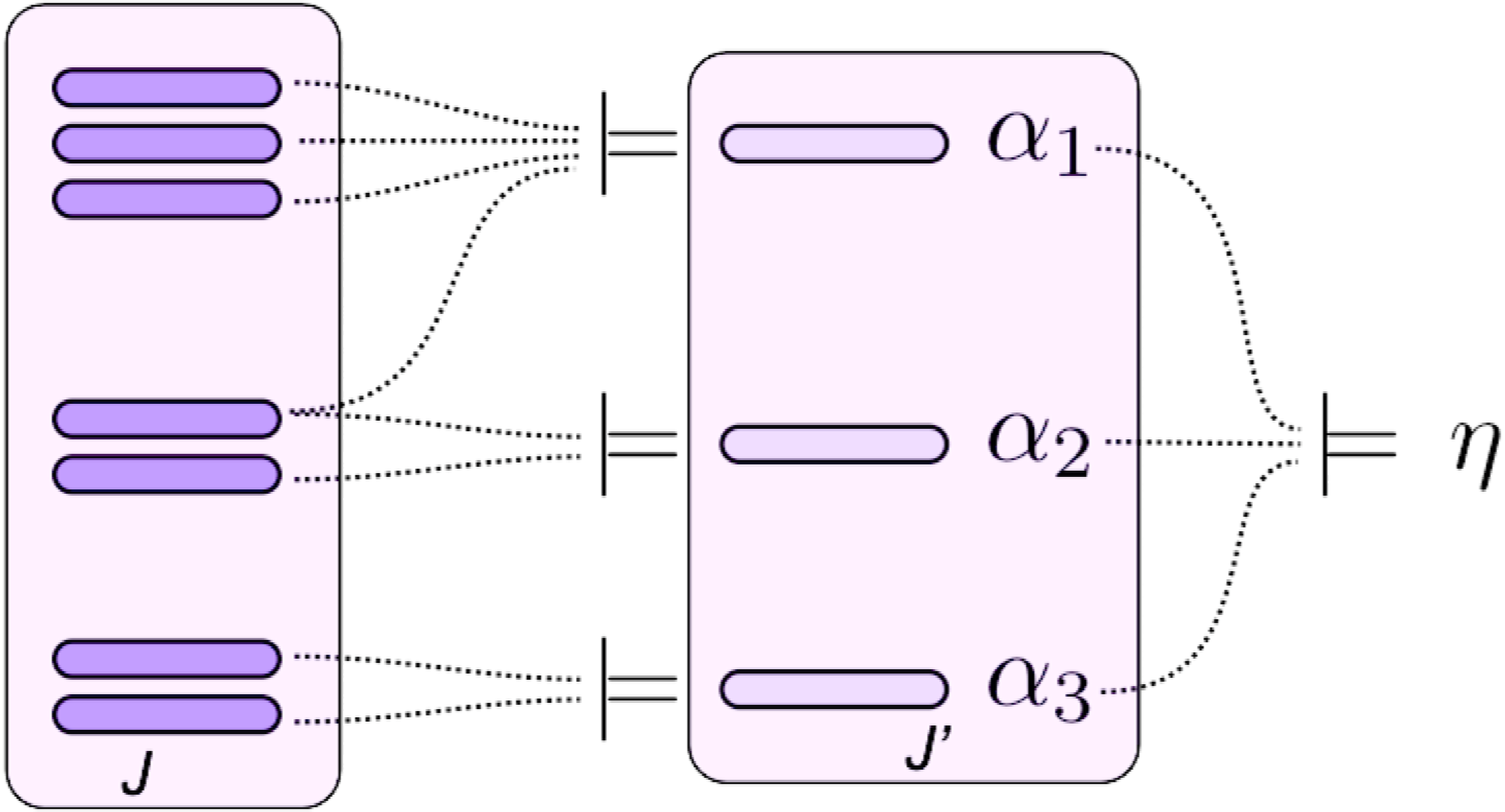
$$\begin{aligned} P &\sqsubseteq \neg M \\ RR &\sqsubseteq CM \\ CM &\sqsubseteq M \\ RR &\equiv \exists h.TS \sqcup \forall v.H \\ \exists v.\top &\sqsubseteq M\} \end{aligned}$$

with $\mathcal{O} \models P \sqsubseteq \bot$

  - this has led to investigation of **lemmatised justifications** (see next slide) with work in **cognitive complexity** of justifications

Lemmatised Justifications: a picture

Compute $J' = \{\alpha_1, \alpha_2, \alpha_3\}$ so that



$Complexity(J,\eta) > Complexity(J', \eta)$

Cognitive Complexity of Justifications: snapshot of a survey

Syntax: ○ Manchester Syntax ⦿ DL Syntax

Set

$C1 \sqsubseteq C3$

    $C3 \sqsubseteq C4$

$C1 \sqsubseteq \exists prop1.C5$

    $prop1 \in R^{+}$

    $C5 \sqsubseteq \exists prop1.C6$

$C4 \sqcap (\exists prop1.C6) \sqsubseteq C2$

Does the above set of axioms entail the following axiom?

$C1 \sqsubseteq C2$

- ○ Yes
- ○ Yes, but not sure
- ○ Not sure
- ○ No, but not sure
- ○ No

Next >>

Page 18 of 21[1]

[1] See http://tinyurl.com/owlsurvey2012

Lemmatised Justifications: an example

bold: axioms in $\mathcal{J}$; normal: axioms entailed by $\mathcal{J}$; example from [Horridge Dissertation]

$$\text{Entailment}: \ \mathsf{Person} \sqsubseteq \bot$$

$\mathbf{Person} \sqsubseteq \neg\mathbf{Movie}$

$\top \sqsubseteq \mathsf{Movie}$

    $\forall\mathsf{hasViolenceLevel}.\bot \sqsubseteq \mathsf{Movie}$

        $\forall\mathsf{hasViolenceLevel}.\bot \sqsubseteq \mathsf{RRated}$

            $\mathbf{RRated} \equiv (\exists\mathbf{hasScript}.\mathbf{ThrillerScript}) \sqcup (\forall\mathbf{hasViolenceLevel}.\mathbf{High})$

        $\mathsf{RRated} \sqsubseteq \mathsf{Movie}$

            $\mathbf{RRated} \sqsubseteq \mathbf{CatMovie}$

            $\mathbf{CatMovie} \sqsubseteq \mathbf{Movie}$

    $\exists\mathsf{hasViolenceLevel}.\top \sqsubseteq \mathsf{Movie}$

        $\mathbf{Domain}(\mathbf{hasViolenceLevel}, \mathbf{Movie})$