Description Logics:
an Introductory Course on a Nice Family of Logics

Day 4: Computational Complexity

Uli Sattler

Complexity

We distinguish between

- **cognitive complexity:**
  - e.g., how hard is it, for a human, to determine/understand $\mathcal{O} \models^? C \sqsubseteq D$
  - interesting, little understood topic
  - relevant to provide tool support for ontology engineers
  - more tomorrow
- **computational complexity:**
  - e.g., how much time/space do we need to determine $\mathcal{O} \models^? C \sqsubseteq D$
  - well understood topic
  - loads of results thanks to relationships DL - FOL - Modal Logic
  - relevant to understand
    * trade-off between expressivity (of a DL) and complexity of reasoning
    * whether a given algorithm is optimal/can be improved

Computational Complexity: Decision Problems

Decision problem:

- is a subset $P \subseteq M$
- e.g., $P =$ the set of consistent $\mathcal{ALC}$ ontologies and
  $M =$ the set of all $\mathcal{ALC}$ ontologies
- think of it as **black box** with
  - input $m \in M$
  - output "*yes*" if $m \in P$
    "*no*" if $m \notin P$

(Polynomial) reduction from $P \subseteq M$ to $P' \subseteq M'$ is a (polynomial) function π:

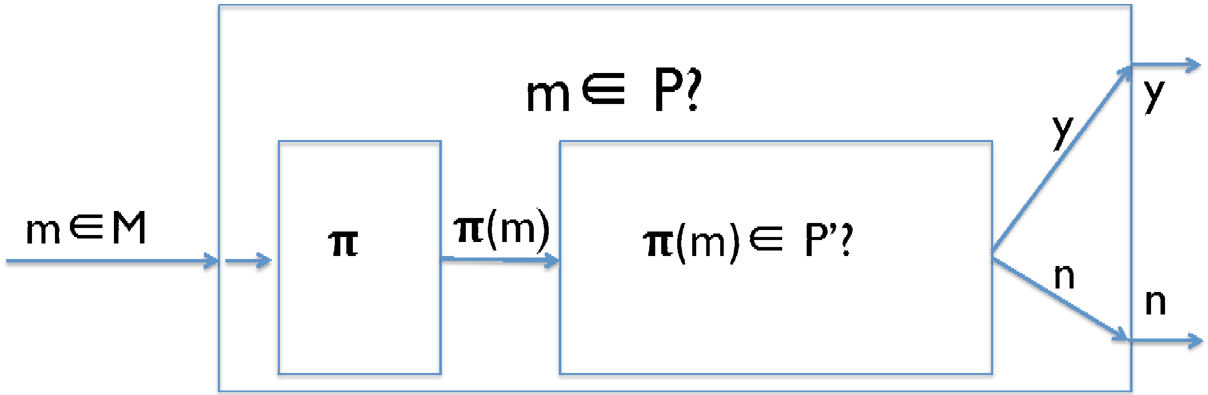
- $\pi : M \longrightarrow M'$
- $m \in P$ iff $\pi(m) \in P'$
- e.g., our translation $t()$ from $\mathcal{ALC}$ to FOL
- e.g., our reduction from subsumption to ontology consistency

Decision problem:

- is a subset $P \subseteq M$
- think of it as **black box** with
  - input $m \in M$
  - output: "*yes*" if $m \in P$, "*no*" otherwise

(Polynomial) reduction from $P \subseteq M$ to $P' \subseteq M'$ is a (polynomial) function π:

- $\pi : M \longrightarrow M'$ with $m \in P$ iff $\pi(m) \in P'$



Decision problem:

- is a subset $P \subseteq M$
- think of it as **black box** with
  - input $m \in M$
  - output: "*yes*" if $m \in P$, "*no*" otherwise

(Polynomial) reduction from $P \subseteq M$ to $P' \subseteq M'$ is a (polynomial) function π:

- $\pi : M \longrightarrow M'$ with $m \in P$ iff $\pi(m) \in P'$

Fact: if $P \subseteq M$ is reducible to $P' \subseteq M'$, then
P is at most as hard/complex[a] as P'
because P can be solved by solving P' via π

[a] Of course only for suitably complex problems.

Some standard complexity classes:

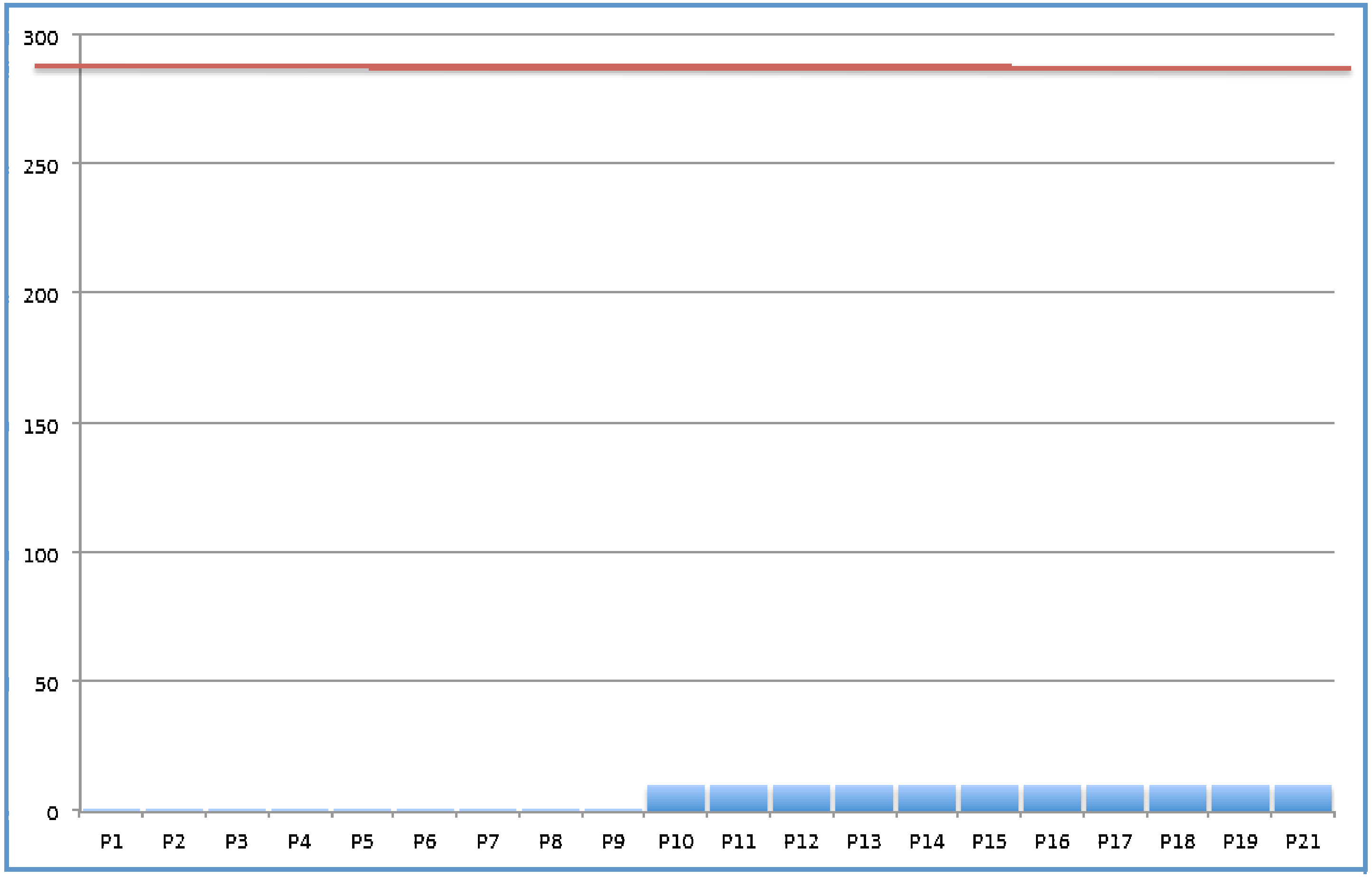
Name	Meaning	Examples
L	logarithmic space	graph accessibility
P	polynomial time	model checking
NP	nondeterministic pol. time	prop. logic SAT
PSpace	polynomial space	Q-SAT
ExpTime	exponential time	
NExpTime	nondeterministic exponential time	
ExpSpace	exponential space	
...	...	
	undecidable	FOL-SAT

To determine that a problem $P \subseteq M$ is

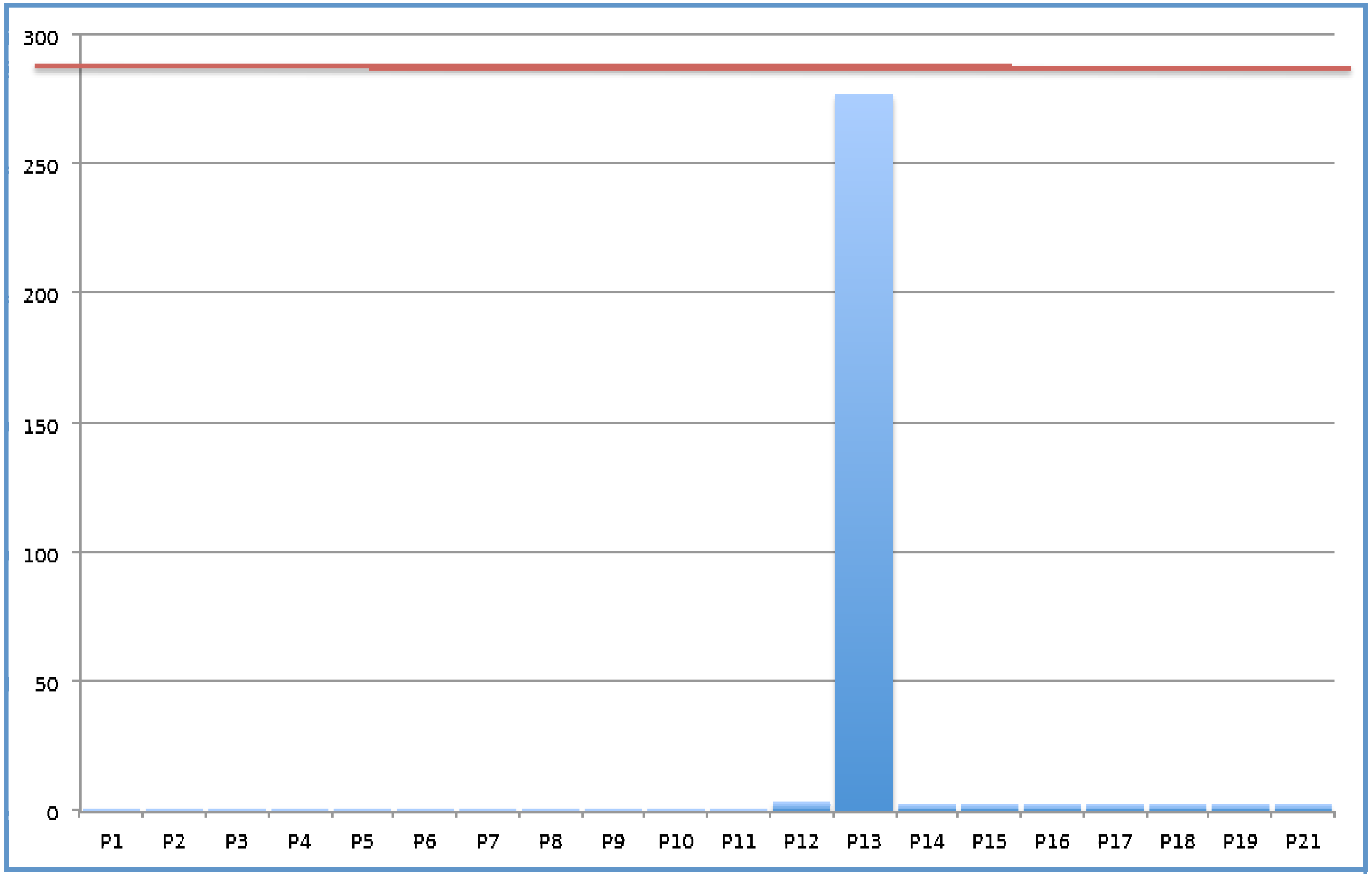
- **in** a complexity class $\mathcal{C}$, it suffices to
  - design/find an algorithm
  - show that it is sound, complete, and terminating, and
  - show that this algorithm runs, for every $m \in M$, in **at most** $\mathcal{C}$ resources
  - ...this algorithm can be a reduction to a problem known to be in $\mathcal{C}$
- **hard for** a complexity class $\mathcal{C}$, we need to
  - find a suitable problem $P' \subseteq M'$ that is **known to be hard** for $\mathcal{C}$ and
  - a reduction from P' to P
- **complete for** a complexity class $\mathcal{C}$, we need to show that it is
  - in $\mathcal{C}$ and
  - hard for $\mathcal{C}$

Worst-Case Complexity

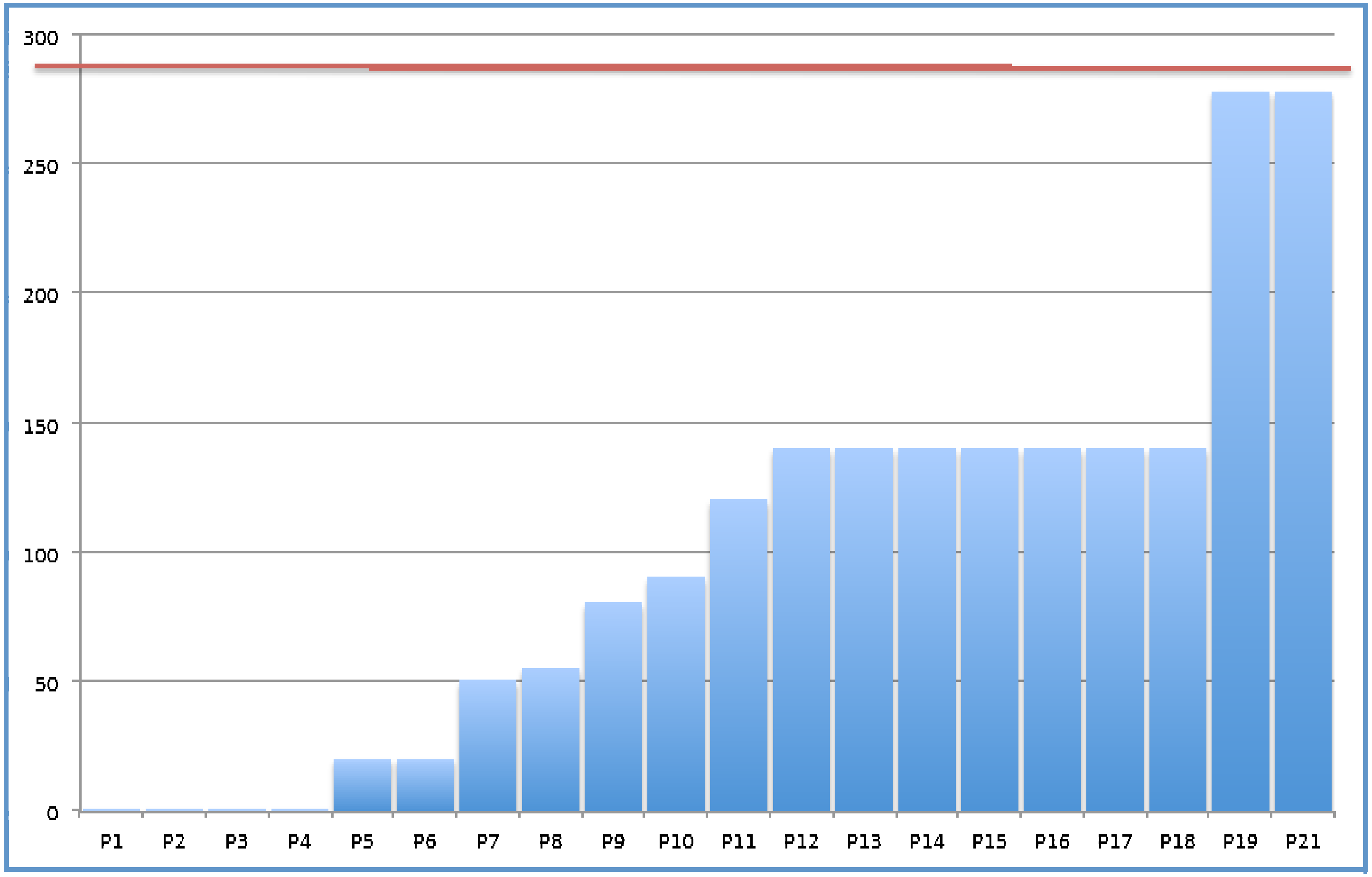
Worst-case: algorithm runs, for every $m \in M$, in **at most** $\mathcal{C}$ resources,
e.g., like this, on all problems of size 7:



Worst-case: algorithm runs, for every $m \in M$, in **at most** $\mathcal{C}$ resources,
e.g., or like this, on all problems of size 7:

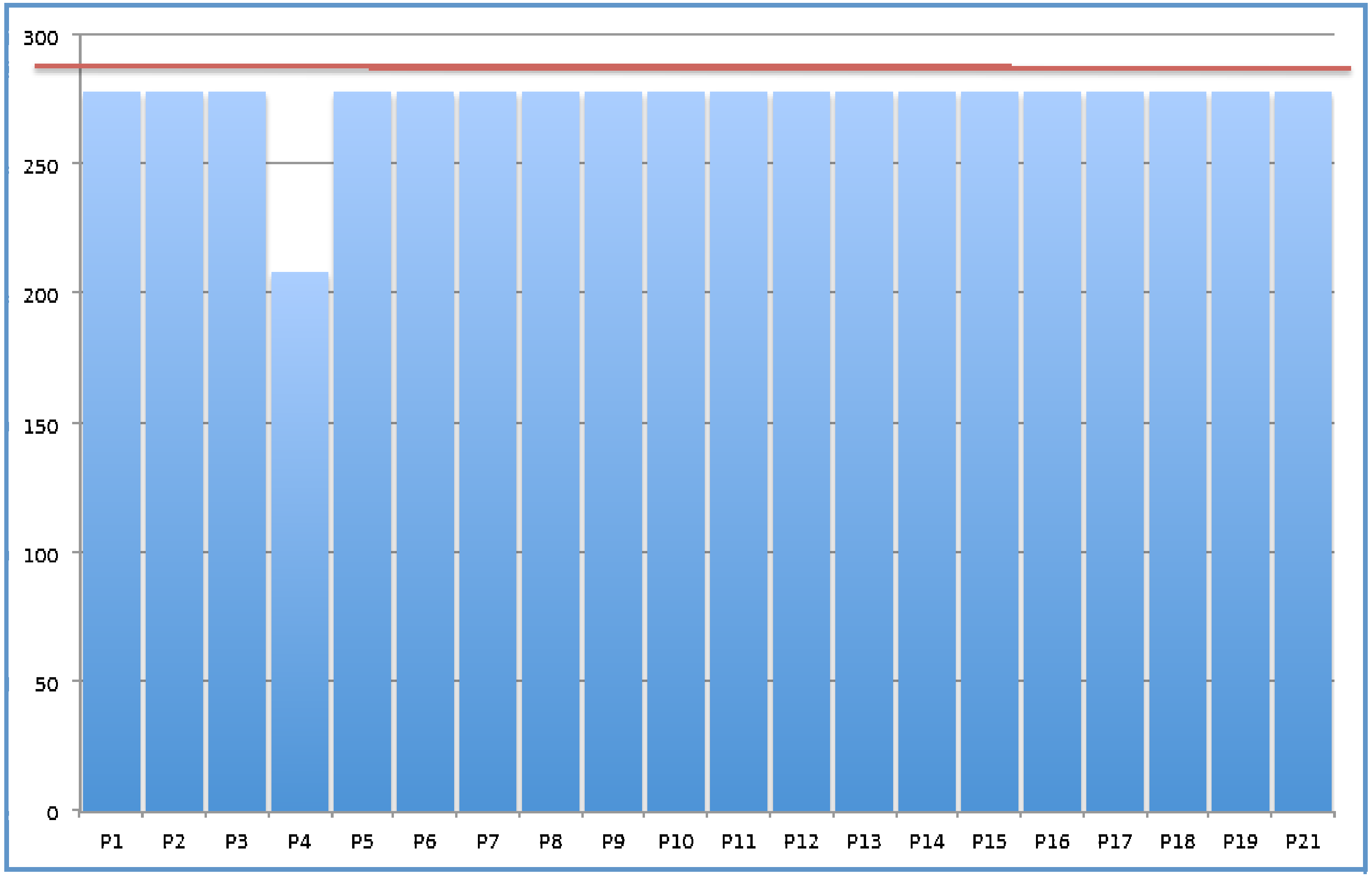


Worst-case: algorithm runs, for every $m \in M$, in **at most** $\mathcal{C}$ resources,
e.g., or like this, on all problems of size 7:



Worst-Case Complexity

Worst-case: algorithm runs, for every $m \in M$, in **at most** $\mathcal{C}$ resources, e.g., or like this, on all problems of size 7:



Known Complexity Results from Days 1-3

- Yesterday, we have seen that $\mathcal{ALCI}$ satisfiability w.r.t. TBoxes is **in ExpTime**:
  - automata-based approach runs in (best & worst case) exponential time
  - can be extended to ABoxes & ontology consistency

✔ we can't do better: already $\mathcal{ALC}$ satisfiability w.r.t. TBoxes is **ExpTime-hard**:
  - but proof is cumbersome
  - via a reduction of the halting problem of a polynomial-space-bounded alternating TM

- on Tuesday, we "saw" that $\mathcal{ALCI}$ satisfiability (no TBoxes) is **in PSpace**:
  - non-deterministic tableau algorithm runs in polynomial space
  - can be extended to ABoxes & ontology consistency

✔ we can't do better: already $\mathcal{ALC}$ satisfiability is **PSpace-hard**:
  - but proof is a bit cumbersome
  - via a reduction of satisfiability of quanitified Boolean formulae

Are all DLs in ExpTime?

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

- **number restrictions**, in fact functionality restrictions $(\leq 1 r\ \top)$ and
- **nominals**, i.e., individual names used as concept names

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

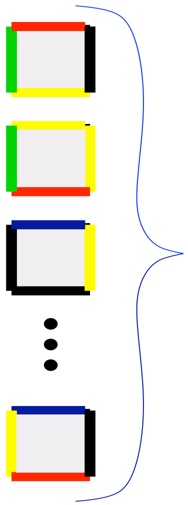
- this is typical phenomenon where
  - **combination** of otherwise harmless constructors
  - leads to increased complexity

$\mathcal{ALCQIO}$ is NExpTime-hard

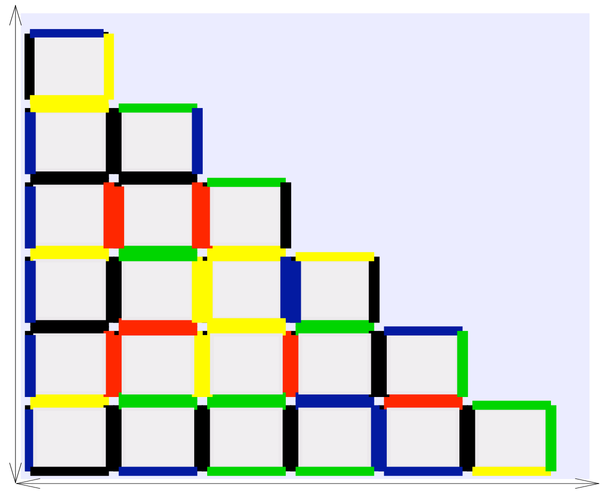
We follow hardness proof recipe:

- 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:

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

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

$\Rightarrow$ well-known that this problem is **undecidable** [Berger66]

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

$\Rightarrow$ 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

We can express various **obligations** of the domino problem in $\mathcal{ALC}$ TBox 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**:

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

Does this suffice?
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 $A_1, \ldots, A_n$, $B_1, \ldots, B_2$ 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**":

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

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

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

(add the same for the B_is)

(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

Let's do this again!

Are all DLs decidable?

So far, we have extended $\mathcal{ALC}$ with

- inverse role and
- number restrictions
- ...which resulted in logics whose reasoning problems are **decidable**
- ...we even discussed **decision procedures** for these extensions

Next, we will discuss some undecidable extension

- $\mathcal{ALC}$ with role chain inclusions
- $\mathcal{ALC}$ with number restrictions on complex roles

An undecidable DL: $\mathcal{ALC}$ with role chain inclusions

OWL 2 supports axioms of the form

- $r \sqsubseteq s$: a model of $\mathcal{O}$ with $r \sqsubseteq s \in \mathcal{O}$ must satisfy $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
- **trans**(r): a model of $\mathcal{O}$ with **trans**$(r) \in \mathcal{O}$ must satisfy $r^{\mathcal{I}} \circ r^{\mathcal{I}} \subseteq r^{\mathcal{I}}$,

  where $p \circ q = \{(x, z) \mid$ there is $y : (x, y) \in p$ and $(y, z) \in q\}$,

  i.e., a model $\mathcal{I}$ of $\mathcal{O}$ must interpret r as a transitive relation
- $r \circ s \sqsubseteq t$: a model of $\mathcal{O}$ with $r \circ s \sqsubseteq t \in \mathcal{O}$ must satisfy $r^{\mathcal{I}} \circ s^{\mathcal{I}} \subseteq t^{\mathcal{I}}$

subject to some complex restrictions

...**why do we need restrictions?**

...because axioms of this form lead to **loss of tree model property** and **undecidability**

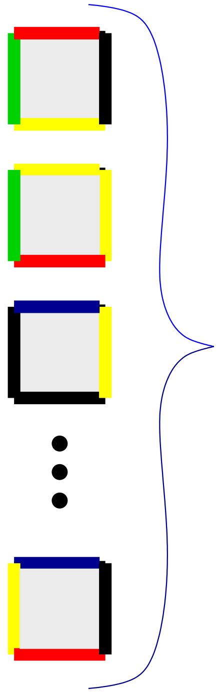
How to prove undecidability of a DL

Often, we prove undecidability of a DL as follows:

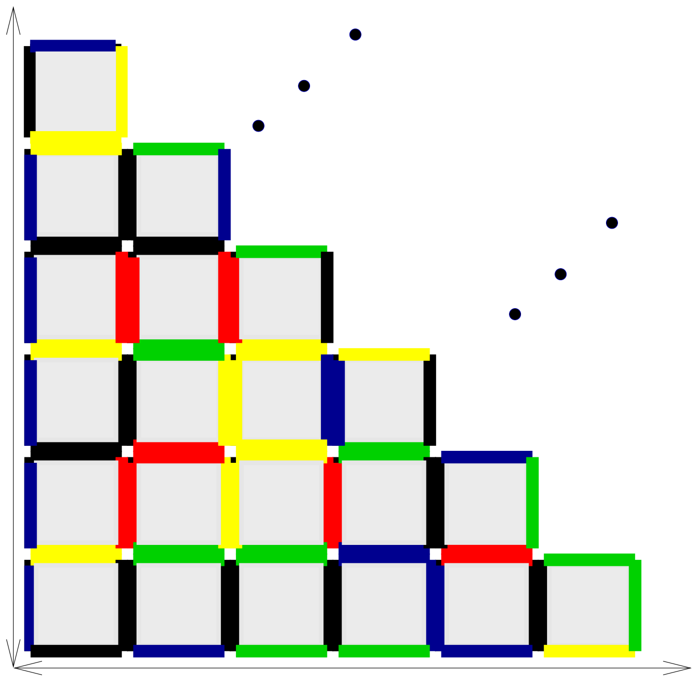
1. **fix reasoning problem**, e.g., satisfiability of a concept w.r.t. a TBox
   - remember Theorem 2?
   - if concept satisfiability w.r.t. TBox is undecidable,
   - then so is consistency of ontology
   - then so is subsumption w.r.t. an ontology
   - ...
2. **pick a decision problem** known to be undecidable, e.g., **the domino problem**
3. **provide a (computable) mapping** $\pi(\cdot)$ that
   - takes an instance D of the domino problem and
   - turns it into a concept A_D and a TBox $\mathcal{T}_D$ such that
   - D has a tiling **if and only if** A_D is satisfiable w.r.t. $\mathcal{T}_D$

   i.e., a decision procedure of concept satisfiability w.r.t. TBoxes could be used as a decision procedure for the domino problem

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:

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

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 $\mathcal{ALC}$

We have already see how to 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**: ✔

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

Does this suffice?

No, we know that it doesn't!

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 axioms from 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!

Let's do this again!

What other constructors can us help to express ④?

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

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

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

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

- various others...

Over to Thomas for ~~easy~~ fast DLs!