Computational Complexity of Description Logics: a Friendly Introduction to Some Interesting Phenomena

Uli Sattler
Warm up

Which of the following subsumptions hold?

1. \( \exists r. (A \land B) \subseteq \exists r. A \)

2. \( \exists r. A \land \forall r. B \subseteq \exists r. B \)

3. \( \forall r. (A \land \neg A) \subseteq \forall r. B \)

4. \( \exists r. (\forall r. A) \subseteq \exists r. (\exists r. (A \lor \neg A)) \)

5. \( \forall r. (A \land B) \subseteq \forall r. A \land \forall r. B \)

6. \( \exists r. B \subseteq \forall r. A \)
• we will discuss a lot of things
• but also leave out a lot
• …please ask if you have a question!
Reminder: Standard DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$,

- is $\mathcal{O}$ consistent? $\mathcal{O} \models \top \subseteq \bot$?
- is $\mathcal{O}$ coherent? is there concept name $A$ with $\mathcal{O} \models A \subseteq \bot$?
- compute concept hierarchy! for all concept names $A, B$: $\mathcal{O} \models A \subseteq B$?
- classify individuals! for all concept names $A$, individual names $b$: $\mathcal{O} \models b : B$?

Theorem 1 Let $\mathcal{O}$ be an ontology and $a$ an individual name not in $\mathcal{O}$. Then

1. $C$ is satisfiable w.r.t. $\mathcal{O}$ iff $\mathcal{O} \cup \{a : C\}$ is consistent
2. $\mathcal{O}$ is coherent iff, for each concept name $A$, $\mathcal{O} \cup \{a : A\}$ is consistent
3. $\mathcal{O} \models A \subseteq B$ iff $\mathcal{O} \cup \{a : (A \cap \neg B)\}$ is not consistent
4. $\mathcal{O} \models b : B$ iff $\mathcal{O} \cup \{b : \neg B\}$ is not consistent

⇒ a decision procedure for consistency decides all standard DL reasoning problems
• A problem is a set $P \subseteq M$
  – e.g., $M$ is the set of all $\mathcal{ALC}$ ontologies,
  – $P \subseteq M$ is the set of all consistent $\mathcal{ALC}$ ontologies
  – ...and the problem $P$ is to decide whether, for a given $m \in M$, we have $m \in P$

• An algorithm is a decision procedure for a problem $P \subseteq M$ if it is
  – sound for $P$: if it answers "$m \in P$", then $m \in P$
  – complete for $P$: if $m \in P$, then it answers "$m \in P"
  – terminating: it stops after finitely many steps on any input $m \in M$

Why does "sound and complete" not suffice for being a decision procedure?
Earlier: Anni explained a tableau algorithm for \textit{ALC}

\begin{itemize}
  \item Input: \textit{ALC} TBox $T$, \textit{ALC} concept name $C$
  \item Output: “yes” if $C$ is satisfiable wrt. $T$
  \item “no” if not
\end{itemize}

Is this algorithm

- sound?
- complete?
- terminating?
- …and how long does it run?
Lemma 1: Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ALC}$ ontology in NNF. Then

1. the algorithm terminates when applied to $\mathcal{T}$ and $\mathcal{C}$
2. if the rules generate a complete & clash-free ABox, then $\mathcal{C}$ is satisfiable wrt. $\mathcal{T}$
3. if $\mathcal{C}$ is satisfiable wrt. $\mathcal{T}$, then the rules generate a clash-free & complete ABox

Corollary 1:

1. Our tableau algorithm decides satisfiability of $\mathcal{ALC}$ concepts wrt. TBoxes.
2. Satisfiability of $\mathcal{ALC}$ concepts (no TBox!) is decidable in $\text{PSpace}$.  
3. Satisfiability of $\mathcal{ALC}$ concepts wrt. TBoxes is decidable in $\text{ExpSpace}$.  
4. $\mathcal{ALC}$ concepts have the finite model property  
   i.e., every consistent ontology has a finite model.  
5. $\mathcal{ALC}$ concepts have the tree model property  
   i.e., every consistent ontology has a tree model.
If we start the algorithm with \( \{a : C\} \) to test satisfiability of \( C \), and construct ABox in non-deterministic depth-first manner rather than constructing set of ABoxes so that we only consider a single ABox and re-use space for branches already visited, mark \( b : \exists R.C \in A \) with “todo” or “done”

we can run tableau algorithm (even without blocking) in polynomial space:

- ABox is of depth bounded by \( |C| \), and
- we keep only a single branch in memory at any time.
Regarding Corollary 1.3

If we start the algorithm with \( \{ a : C \} \) and \( T \) to test satisfiability of \( C \) wrt. \( T \), and construct ABox in non-deterministic depth-first manner rather than constructing set of ABoxes so that we only consider a single ABox

we can run tableau algorithm in exponential space:

- number of individuals in ABox is bounded by \( 2^{\#_{\text{sub}}(T)} \)

This is not optimal: consistency of \( \mathcal{ALC} \) ontologies is decidable in exponential time, in fact ExpTime-complete.
The tableau algorithm presented here

→ \textit{decides} consistency of $\mathcal{ALC}$ ontologies, and thus also
→ all other standard reasoning problems
→ uses \textit{blocking} to ensure termination, and
→ can be implemented as such or
  using a \textit{non-deterministic} alternative for the $\sqcap$-rule and backtracking.
→ uses $\text{P/Exp-Space}$
→ can be implemented in various ways,
  – order/priorities of rules
  – data structure
  – etc.
→ is amenable to optimisations...
Implementing the $\mathcal{ALC}$ Tableau Algorithm

Naive implementation of $\mathcal{ALC}$ tableau algorithm is doomed to failure:

It constructs a
- set of ABoxes,
- each ABox being of possibly exponential size, with possibly exponentially many individuals (see binary counting example)
- in the presence of a GCI such as $\top \sqsubseteq (C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)$ and exponentially many individuals, algorithm might generate double exponentially many ABoxes

$\Rightarrow$ requires double exponential space or

- use non-deterministic variant and backtracking to consider one ABox at a time

$\Rightarrow$ requires exponential space
Implementing the $\mathcal{ALC}$ Tableau Algorithm

Optimisations are crucial
corn, every aspect of the algorithm
help in “many” cases (which?)
are implemented in various DL reasoners
e.g., FaCT++, Pellet, RacerPro

In the following: a selection of some vital optimisations
Reasoners provides service “classify all concept names in $T$”, i.e., for all concept names $C, D$ in $T$, reasoner decides does $T \models C \sqsubseteq D$?

$\leadsto$ test consistency of $T \cup \{a: (C \cap \neg D)\}$

$\leadsto n^2$ consistency tests!

**Goal:** reduce number of consistency tests when classifying TBox

**Idea 1:** “trickle” new concept $C$ into hierarchy computed so far
Reasoners provides service “classify all concept names $\mathcal{T}$”, i.e., for all concept names $C, D$ in $\mathcal{T}$, reasoner decides does $\mathcal{T} \models C \sqsubseteq D$?

$\Rightarrow$ test consistency of $\mathcal{T} \cup \{a : (C \cap \neg D)\}$

$\Rightarrow n^2$ consistency tests!

Goal: reduce number of consistency tests when classifying $TBox$

Idea 1: “trickle” new concept $C$ into hierarchy computed so far
Reasoners provides service “classify all concept names $\mathcal{T}$”, i.e., for all concept names $C, D$ in $\mathcal{T}$, reasoner decides does $\mathcal{T} \models C \sqsubseteq D$?

$\leadsto$ test consistency of $\mathcal{T} \cup \{a : (C \sqcap \neg D)\}$

$\leadsto n^2$ consistency tests!

Goal: reduce number of consistency tests when classifying TBox

Idea 1: “trickle” new concept $C$ into hierarchy computed so far
Reasoners provides service "classify all concept names $T$", i.e.,
for all concept names $C, D$ in $T$, reasoner decides does $T \models C \sqsubseteq D$?
$\iff$ test consistency of $T \cup \{ a : (C \sqcap \neg D) \}$
$\iff n^2$ consistency tests!

**Goal: reduce number of consistency tests when classifying TBox**

**Idea 2:**
- maintain graph with a node for each concept name
- edges representing subsumption, disjointness ($T \models A \sqsubseteq \neg B$), and non-subsumption
- initialise graph with all “obvious” information in $T$
- to avoid testing subsumption, exploit
  - all info in ABox during tableau algorithm to update graph
  - transitivity of subsumption and its interaction with disjointness
### Optimising the ALC Tableau Algorithm: Absorption

**Remember:** for $T = \{ C_i \sqsubseteq D_i \mid 1 \leq i \leq n \}$, each individual $x$ will have $n$ disjunctions $x: (\neg C_i \sqcup D_i)$ due to

**GCI-rule:** if $T = \{ C_i \sqsubseteq D_i \mid 1 \leq i \leq n \}$ replace $\mathcal{A}$ with $\mathcal{A} \cup \{ a: (\neg C_1 \sqcup D_1) \cap (\neg C_2 \sqcup D_2) \cap \ldots \cap (\neg C_n \sqcup D_n) \}$

**Problem:** high degree of choice and huge search space blows up set of ABoxes...we can do better:

**2GCI-rule:** if $C \sqsubseteq D \in T$, $a$ is not blocked, and if $C$ is a concept name, $a: C \in \mathcal{A}$ but $a: D \not\in \mathcal{A}$, replace $\mathcal{A}$ with $\mathcal{A} \cup \{ a: D \}$

else if $a: (\neg C \sqcup D) \not\in \mathcal{A}$ for $a$ in $\mathcal{A}$, replace $\mathcal{A}$ with $\mathcal{A} \cup \{ a: (\neg C \sqcup D) \}$

**Problem:** still possibly high degree of choice and huge search space...
Observation: many GCIs are of the form $A \sqcap \ldots \sqsubseteq C$ for concept name $A$

e.g., Human $\sqcap \ldots \sqsubseteq C$ or Device $\sqcap \ldots \sqsubseteq C$

Idea: localise GCIs to concept names by transforming

$A \sqcap X \sqsubseteq C$ into equivalent $A \sqsubseteq \neg X \sqcup C$

e.g., Human $\sqcap \exists \text{owns}.Pet \sqsubseteq C$ becomes Human $\sqsubseteq \neg \exists \text{owns}.Pet \sqcup C$

For “absorbed” $T = \{ A_i \sqsubseteq D_i \mid 1 \leq i \leq n_1 \} \cup \{ C_i \sqsubseteq D_i \mid 1 \leq i \leq n_2 \}$

the second, non-deterministic choice in GCI-rule is taken only $n_2$ times.

2GCI-rule: if $C \sqsubseteq D \in T$, $a$ is not blocked, and

if $C$ is a concept name, $a : C \in \mathcal{A}$ but $a : D \notin \mathcal{A}$,

replace $\mathcal{A}$ with $\mathcal{A} \cup \{ a : D \}$

else if $a : (\neg C \sqcup D) \notin \mathcal{A}$ for $a$ in $\mathcal{A}$,

replace $\mathcal{A}$ with $\mathcal{A} \cup \{ a : (\neg C \sqcup D) \}$
Optimising the \textit{ALC} Tableau Algorithm: Absorption

\textbf{Observation:} many GCIs are of the form $A \sqcap \ldots \sqsubseteq C$ for concept name $A$

e.g., Human $\sqcap \ldots \sqsubseteq C$ or Device $\sqcap \ldots \sqsubseteq C$

\textbf{Idea:} localise GCIs to concept names by transforming
$$A \sqcap X \sqsubseteq C$$ into equivalent $A \sqsubseteq \neg X \sqcup C$
e.g., Human $\sqcap \exists \text{owns.Pet} \sqsubseteq C$ becomes Human $\sqsubseteq \neg \exists \text{owns.Pet} \sqcup C$

\textbf{Observations:} If no GCI is absorbable, nothing changes
Each absorption saves 1 disjunction per individual outside $A_i$,
in the best case, this avoids almost all disjunctions from TBox axioms!
Remember If a clash is encountered, non-deterministic algorithm backtracks

i.e., returns to last non-deterministic choice and
tries other possibility

Example \( x: \exists R. (A \sqcap B) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R. \neg A \)
Remember If a clash is encountered, **non-deterministic algorithm backtracks**
i.e., returns to last non-deterministic choice and tries other possibility

**Example** $x: \exists R. (A \sqcap B) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R. \neg A$

```
\text{\textbf{x}} \xrightarrow{\text{\textbf{L}}(x) \cup \{C_1\}} \text{\textbf{x}} \xrightarrow{\text{\textbf{L}}(x) \cup \{C_{n-1}\}} \text{\textbf{x}} \xrightarrow{\text{\textbf{L}}(x) \cup \{C_n\}} \text{\textbf{x}} \xrightarrow{\text{\textbf{L}}(x) \cup \{\neg C_n, D_n\}} \text{\textbf{x}} \xrightarrow{\text{\textbf{L}}(x) \cup \{\neg C_2, D_2\}} \text{\textbf{x}} \xrightarrow{\text{\textbf{L}}(x) \cup \{\neg C_1, D_1\}} \text{\textbf{x}}
```

$\text{\textbf{L}}(y) = \{(A \sqcap B), \neg A, A, B\}$ $\text{\textbf{y}}$

---

Clash
Remember If a clash is encountered, non-deterministic algorithm backtracks

i.e., returns to last non-deterministic choice and
tries other possibility

Example \( x: \exists R. (A \cap B) \cap ((C_1 \cup D_1) \cap \ldots \cap (C_n \cup D_n)) \cap \forall R. \neg A \)
Optimising the $\mathcal{ALC}$ Tableau Algorithm: Backjumping

**Remember** If a clash is encountered, non-deterministic algorithm backtracks

e.g., returns to last non-deterministic choice and
tries other possibility

**Example** $x : \exists R. (A \sqcap B) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R. \neg A$
Finally: \( \mathcal{ALC} \) extends propositional logic
\[ \iff \]\, heuristics developed for SAT are relevant

Summing up: optimisations are possible at each aspect of tableau algorithm
\[ \iff \]\, can dramatically enhance performance
\[ \iff \]\, do they interact?
\[ \iff \]\, how?
\[ \iff \]\, which combination works best for which “cases”?
\[ \iff \]\, is the optimised algorithm still correct?
... check out ORE 2013 results & our “Robustness” paper at DL 2013
...now for some proper computational complexity...
We have seen 1 algorithm that runs

- in $\text{PSpace}$ without a TBox
- in non-deterministic $\text{ExpSpace}$ with a TBox

...can we do better? How can we tell? ...perhaps try much harder, think much longer?

...how do we show that our algorithm is *optimal*? And what does that mean anyway?

→ look at 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

- **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
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 $\pi$:  
- $\pi : M \longrightarrow M'$
- $m \in P$ 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 $\pi$:

• $\pi : M \rightarrow M'$ with $m \in P$ iff $\pi(m) \in P'$
Computational Complexity: Decision Problems

**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 $\pi$:
- $\pi : M \rightarrow M'$ 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 $\pi$

\(^a\)Of course only for suitably complex problems.
### Computational Complexity

Some standard complexity classes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>logarithmic space</td>
<td>graph accessibility</td>
</tr>
<tr>
<td>P</td>
<td>polynomial time</td>
<td>model checking</td>
</tr>
<tr>
<td>NP</td>
<td>nondeterministic pol. time</td>
<td>prop. logic SAT</td>
</tr>
<tr>
<td>PSpace</td>
<td>polynomial space</td>
<td>Q-SAT</td>
</tr>
<tr>
<td>ExpTime</td>
<td>exponential time</td>
<td></td>
</tr>
<tr>
<td>NExpTime</td>
<td>nondeterministic exponential time</td>
<td></td>
</tr>
<tr>
<td>ExpSpace</td>
<td>exponential space</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>undecided</td>
<td></td>
<td>FOL-SAT</td>
</tr>
</tbody>
</table>
To determine that a problem $P \subseteq M$ is

- **in** a complexity class $C$, 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 $C$ resources
  - ...this algorithm can be a reduction to a problem known to be in $C$

- **hard for** a complexity class $C$, we need to
  - find a suitable problem $P' \subseteq M'$ that is known to be hard for $C$ and
  - a reduction $\pi(.)$ from $P'$ to $P$

- **complete for** a complexity class $C$, we need to show that it is
  - in $C$ and
  - hard for $C$
Known Complexity Results so Far:

- We have seen that $\mathcal{ALC}$ concept satisfiability (no TBox) is in $\text{PSpace}$:
  - non-deterministic tableau algorithm runs in polynomial space
  - can be extended to ABoxes
- ✔️ we can’t do better: $\mathcal{ALC}$ satisfiability is $\text{PSpace-hard}$:
  - but proof is a bit cumbersome
  - via a reduction of satisfiability of quantified Boolean formulae
- We have seen that $\mathcal{ALC}$ concept satisfiability w.r.t. TBoxes is in $\text{NExpSpace}$:
  - non-deterministic tableau algorithm runs in exponential space
  - can be extended to ABoxes & ontology consistency
  - can be extended to $\mathcal{ALCQI}$, $\mathcal{ALCQO}$, and $\mathcal{ALCIO}$
- ✔️ we can do better: $\mathcal{ALC}$ satisfiability wrt. TBoxes is $\text{ExpTime-complete}$:
  - but such (optimal) algorithm takes too long for this course
  - as is lower bound/hardness proof
  (via a reduction of the halting problem of polynomial-space-bounded alternating TMs)
Understanding lower bounds/hardness of problems

• tell us when “in principle” improvement of algorithms is futile
• inform us of relationships of logics:
  – who is harder than who?
  – which are of similar difficulty?
• their proofs often
  – reveal interesting model theoretic properties:
    * tree model property: each satisfiable input has a tree-shaped model
    * finite model property: each satisfiable input has a finite model
  – use interesting translations between logics

They don’t always tell us much about “typical” performance...
Worst-case: algorithm runs, for every $m \in M$, in at most $C$ resources, e.g., like this, on all problems of size 7:
Worst-case: algorithm runs, for every $m \in M$, in at most $C$ resources, e.g., or like this, on all problems of size 7:
Worst-case: algorithm runs, for every $m \in M$, in at most $C$ resources, e.g., or like this, on all problems of size 7:
Worst-case: algorithm runs, for every $m \in M$, in at most $C$ resources, e.g., or like this, on all problems of size 7:
Earlier, we have claimed that, for $\mathcal{ALC}$,

- concept satisfiability is in PSpace, but
- concept satisfiability w.r.t. a TBox is in ExpTime

Next, we will see that, for $\mathcal{ALC}^u$, the extension of $\mathcal{ALC}$ with

- universal role $u$ with $u^I = \Delta^I \times \Delta^I$

$\Rightarrow$ concept satisfiability is as hard as reasoning w.r.t. a TBox, namely ExpTime-hard

- this is typical phenomenon where the
  - certain constructors enable us to internalise a TBox
Internalising a TBox

Remember: for $\mathcal{T} = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$, we use

$$C_\mathcal{T} = \left(\neg C_1 \sqcup D_1\right) \cap \ldots \cap \left(\neg C_n \sqcup D_n\right)$$

for the universal $\mathcal{T}$ concept that has to hold everywhere.

Reduction: for $C$ a concept and $\mathcal{T}$ a TBox, define

$$\pi(C, \mathcal{T}) = C \cap \forall u. C_\mathcal{T}$$

Lemma: 1. $C$ is satisfiable w.r.t. $\mathcal{T}$ iff the concept $\pi(C, \mathcal{T})$ is satisfiable
2. the size of $\pi(C, \mathcal{T})$ is linear in that of $C$ plus $\mathcal{T}$

Corollary: satisfiability of $\mathcal{ALC}^u$ concepts is as hard as satisfiability of $\mathcal{ALC}$ concepts w.r.t. TBoxes is, namely ExpTime-hard.
Let’s do that again!
Is concept satisfiability always easier? II

Earlier, we have claimed that, for \( \mathcal{ALC} \),

- concept satisfiability is in PSpace, but
- concept satisfiability w.r.t. a TBox is in ExpTime

Next, we will see that, for \( \mathcal{ALCIO} \), the extension of \( \mathcal{ALC} \) with

- inverse roles \( r^- \) with \( (r^-)^I = \{(y, x) \mid (x, y) \in r^I\} \) and
- nominals, i.e., individual names used as concept names

⇒ concept satisfiability is as hard as reasoning w.r.t. a TBox, namely ExpTime-hard

- this is typical phenomenon where the
  - combination of certain constructors enables us to internalise a TBox
Remember: for $\mathcal{T} = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$, we use

$$C_\mathcal{T} = (\neg C_1 \sqcup D_1) \cap \ldots \cap (\neg C_n \sqcup D_n)$$

for the universal $\mathcal{T}$ concept that has to hold everywhere.

Reduction: for $C$ a concept and $\mathcal{T}$ a TBox, define

$$\pi(C, \mathcal{T}) = C \cap C_\mathcal{T} \cap \exists p. (\{o\} \cap \forall p^-. (\bigcap_r \forall r. (\exists p. \{o\} \cap C_\mathcal{T})))$$

Lemma: 1. $C$ is satisfiable w.r.t. $\mathcal{T}$ iff the concept $\pi(C, \mathcal{T})$ is satisfiable

2. The size of $\pi(C, \mathcal{T})$ is linear in that of $C$ plus $\mathcal{T}$

Corollary: satisfiability of $\textit{ALCIO}$ concepts is as hard as satisfiability of $\textit{ALCIO}$ concepts w.r.t. TBoxes, namely ExpTime-hard.
Earlier, we have claimed that \textit{ALCQI}, \textit{ALCQO}, and \textit{ALCIo} are all ExpTime-complete, i.e., as hard/easy as \textit{ALC}.

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

- inverse roles $r^{-}$ with $(r^{-})^I = \{(y, x) \mid (x, y) \in r^I\}$
- number restrictions, in fact functionality restrictions ($\leq 1r^\top$) and
- nominals, i.e., individual names used as concept names

$\Rightarrow$ is harder, namely \textit{NExpTime}-hard.

- this is typical phenomenon where
  - combination of otherwise harmless constructors
  leads to increased complexity
We follow our hardness proof recipe:

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

The $\text{NExpTime}$ version of the 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 function:

$$t : \mathbb{N} \times \mathbb{N} \rightarrow D$$

such that

$$\langle t(m, n), t(m + 1, n) \rangle \in H \quad \text{and} \quad \langle t(m, n), t(m, n + 1) \rangle \in V$$

Domino problems: classical given $\mathcal{D}$, does $\mathcal{D}$ have a tiling?

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

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

$\Rightarrow$ well-known that this problem is NExpTime-hard
To reduce the NExpTime domino problem to $\text{ALCQIO}$ consistency, we need to

- define a mapping $\pi(.)$ from domino problems to $\text{ALCQIO}$ 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 $n$
Elements in models of $\pi(D)$ will stand for points in the grid, i.e., $(m, n)$...

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

1. each element carries exactly one domino type $D_i$

   $\Rightarrow$ use concept name $D_i$ for each domino type and

   $$\top \sqsubseteq D_1 \sqcup \ldots \sqcup D_d \quad \text{% each element carries a domino type}$$
   $$D_1 \sqsubseteq \neg D_2 \sqcap \ldots \sqcap \neg D_d \quad \text{% but not more than one}$$
   $$D_2 \sqsubseteq \neg D_3 \sqcap \ldots \sqcap \neg D_d \quad \text{% ...}$$
   $$\vdots$$
   $$D_{d-1} \sqsubseteq \neg D_d$$
Mapping a Domino System into an $\textit{ALCQIO}$ Ontology

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

$$\top \subseteq \exists X.\top \cap \exists Y.\top$$

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

$$D_1 \subseteq \bigcup_{(D_1, D) \in H} \forall X. D \cap \bigcup_{(D_1, D) \in V} \forall Y. D$$

$$D_2 \subseteq \bigcup_{(D_2, D) \in H} \forall X. D \cap \bigcup_{(D_2, D) \in V} \forall Y. D$$

$$\vdots$$

$$D_d \subseteq \bigcup_{(D_d, D) \in H} \forall X. D \cap \bigcup_{(D_d, D) \in V} \forall Y. D$$

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 $\textit{ALC}$ is NExpTime-hard
- so no...what is missing?
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 ...
Mapping a Domino System into an \textit{ALCQIO} Ontology

(1) 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 \textit{incremented correctly}
e.g., $X$-successor of $(010, 011)$ is $(011, 011)$
e.g., $Y$-successor of $(010, 011)$ is $(010, 100)$

(b) use a nominal to ensure that there is only one $(111\ldots 1, 111\ldots 1)$
this implies, with $\top \sqsubseteq (\leq 1 \ X\neg. \top) \cap (\leq 1 \ Y\neg. \top)$ \textit{uniqueness} of grid positions
Mapping a Domino System into an **ALCQIO** Ontology

4. counting and binding together

(a) \(\tilde{A}_i\) for “bit \(A_i\) is incremented correctly”:

\[
\begin{align*}
\top & \subseteq \tilde{A}_1 \cap \ldots \cap \tilde{A}_n \\
\tilde{A}_1 & \subseteq (A_1 \cap \forall X.\neg A_1) \cup (\neg A_1 \cap \forall X.A_1) \\
\tilde{A}_i & \subseteq (\cap_{\ell<i} A_\ell \cap ((A_i \cap \forall X.\neg A_i) \cup (\neg A_i \cap \forall X.A_i)) \cup \\
& \quad (\neg \cap_{\ell<i} A_\ell \cap ((A_i \cap \forall X.A_i) \cup (\neg A_i \cap \forall X.\neg A_i))
\end{align*}
\]

(b) ensure uniqueness of grid positions:

\[
A_1 \cap \ldots \cap A_n \cap B_1 \cap \ldots \cap B_n \subseteq \{o\} \quad \% \text{top right } (2^n, 2^n) \text{ is unique} \\
\top \subseteq (\leq 1 X^-.\top) \cap (\leq 1 Y^-.\top) \quad \% \text{everything else is also unique}
\]
Reduction of NExpTime Domino Problem to $ALCQIO$ Consistency

Lemma: let $\pi(D)$ be ontology consisting of all axioms mentioned in ①-④:

- $D$ has an $2^n \times 2^n$ tiling iff $\pi(D)$ is consistent
- size of $\pi(D)$ is polynomial (quadratic) in
  - the size of $D$ and
  - $n$

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

if we could solve consistency of $ALCQIO$ in, say, ExpTime,
this would allow us to solve the domino problem also in ExpTime via $\pi(.)$
Let’s do this again!
Are all DLs decidable?

So far, we have extended $ALC$ 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
- $ALC$ with role chain inclusions
- $ALC$ with number restrictions on complex roles
OWL 2 supports axioms of the form

- $r \sqsubseteq s$: a model of $\mathcal{O}$ with $r \sqsubseteq s \in \mathcal{O}$ must satisfy $r^I \subseteq s^I$
- $\text{trans}(r)$: a model of $\mathcal{O}$ with $\text{trans}(r) \in \mathcal{O}$ must satisfy $r^I \circ r^I \subseteq r^I$, where $p \circ q = \{(x, z) \mid \text{there is } y : (x, y) \in p \text{ and } (y, z) \in q\}$, 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 $r^I \circ s^I \subseteq t^I$

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

Similar to hardness results, 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 1?
   - if concept satisfiability w.r.t. TBox is undecidable,
   - then so is consistency of ontology
   - then so is subsumption w.r.t. TBox
   - ...  

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:

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

**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 $D_i$

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

\[
\top \sqsubseteq D_1 \sqcup \ldots \sqcup D_d \quad \% \text{each element carries a domino type}
\]
\[
D_1 \sqsubseteq \neg D_2 \sqcap \ldots \sqcap \neg D_d \quad \% \text{but not more than one}
\]
\[
D_2 \sqsubseteq \neg D_3 \sqcap \ldots \sqcap \neg D_d \quad \% \ldots
\]
\[
\vdots \quad \vdots
\]
\[
D_{d-1} \subseteq \neg D_d
\]
Almost Encoding the Classical Domino Problem in $\mathcal{ALC}$

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

\[ \top \subseteq \exists X. \top \land \exists Y. \top \]

3. every element satisfies $D$’s horizontal/vertical matching conditions:

\[
\begin{align*}
D_1 \subseteq & \quad \bigcup_{(D_1,D) \in H} \forall X. D \land \bigcup_{(D_1,D) \in V} \forall Y. D \\
D_2 \subseteq & \quad \bigcup_{(D_2,D) \in H} \forall X. D \land \bigcup_{(D_2,D) \in V} \forall Y. D \\
\vdots & \quad \vdots \\
D_d \subseteq & \quad \bigcup_{(D_d,D) \in H} \forall X. D \land \bigcup_{(D_d,D) \in V} \forall Y. D
\end{align*}
\]

Does this suffice?

No, we know that it doesn’t!
for each element, its horizontal-vertical-successors coincide with their vertical-horizontal-successors & vice versa

\[ X \circ Y \sqsubseteq Y \circ X \quad \text{and} \quad Y \circ X \sqsubseteq X \circ Y \]

Lemma: Let \( \mathcal{T}_D \) be the axioms from ① to ④.
Then \( \top \) is satisfiable w.r.t. \( \mathcal{T}_D \) iff \( D \) 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 1, 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!
What other constructors can us help to express obligation \( \mathfrak{O} \)?

- **counting and complex roles (role chains and role intersection):**
  \[
  \top \sqsubseteq (\leq 1x.\top) \cap (\leq 1y.\top) \cap (\exists (x \circ y) \cap (y \circ x).\top)
  \]
- **restricted role chain inclusions (only 1 role on RHS), and counting on non-simple roles:**
  \[
  \begin{align*}
  \top & \sqsubseteq (\leq 1x.\top) \cap (\leq 1y.\top) \\
  x \circ y & \sqsubseteq r \\
  y \circ x & \sqsubseteq r \\
  \top & \sqsubseteq (\leq 1r.\top)
  \end{align*}
  \]
- **various others...**
Are all DLs hard/intractable?

Let’s see a less complex DL: $\mathcal{EL}$
Thanks to Thomas Schneider, University of Bremen: he made the originals of these slides, which I borrowed and slightly modified.
Are all DLs intractable?

1. What is $\mathcal{EL}$?

2. Normalisation

3. A simple poly-time reasoning algorithm
And now . . .

1. What is $\mathcal{EL}$?

2. Normalisation

3. A simple poly-time reasoning algorithm
**Summary**

\( \mathcal{EL} \) is a restriction of \( \mathcal{ALC} \) that . . .

- allows only conjunction and existential restrictions
- is at the heart of OWL 2 EL
- whose standard reasoning problems are in PTime, i.e.,
  
  i.e., there is a worst-case polynomial-time algorithm for deciding subsumption etc.
Summary

\(\mathcal{EL}\) is a restriction of \(\mathcal{ALC}\) that . . .

- allows only conjunction and existential restrictions
- is at the heart of OWL 2 EL
- whose standard reasoning problems are in PTime, i.e.,
  i.e., there is a worst-case polynomial-time algorithm for
deciding subsumption etc.
- can be extended to \(\mathcal{EL}^{++}\) with other features, without
increase in complexity:
  \[
  \bot \\
  \text{disjoint concepts}
  \text{domain and range restrictions}
  \text{concept and role assertions}
  \text{nominals}
  \text{transitive roles, reflexive roles}
  \text{concrete domains}
  \[

Uli Sattler

DL: \(\mathcal{EL}\)
Summary

\(\mathcal{EL}\) is a restriction of \(\mathcal{ALC}\) that . . .

- allows only conjunction and existential restrictions
- is at the heart of OWL 2 EL
- whose standard reasoning problems are in PTime, i.e.,
  - i.e., there is a worst-case polynomial-time algorithm for deciding subsumption etc.
- can be extended to \(\mathcal{EL}^{++}\) with other features, without increase in complexity:
  - \(\bot\) (domain and range restrictions)
  - disjoint concepts (concept and role assertions)
  - role (chain) inclusions (nominals)
  - transitive roles, reflexive roles (concrete domains)
- whose extension with inverse roles or counting increases complexity to that of \(\mathcal{ALC}\)
Syntax and semantics of $\mathcal{EL}$

**Concepts**

For $C, D$ concepts and $R$ a role name:

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Example</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>top</td>
<td>$\top$</td>
<td></td>
<td>$\Delta^I$</td>
</tr>
<tr>
<td>conjunction</td>
<td>$C \sqcap D$</td>
<td>Human $\sqcap$ Male</td>
<td>$C^I \cap D^I$</td>
</tr>
<tr>
<td>exist. restr.</td>
<td>$\exists r.C$</td>
<td>$\exists$hasChild.Human</td>
<td>${x \mid \exists y.(x, y) \in r^I \land y \in C^I}$</td>
</tr>
</tbody>
</table>

**Axioms**

- $C \sqsubseteq D$
- $C \equiv D$ as a shortcut for “$C \sqsubseteq D, D \sqsubseteq C$”
What a tiny logic!?

✓ We can say in $\mathcal{EL}$

Hand $\sqsubseteq \exists \text{hasPart}.\text{Finger}$

✗ but we can’t say

Hand $\sqsubseteq =5 \text{hasPart}.\text{Finger}$
Finger $\sqsubseteq \exists \text{hasPart}.\text{Hand}$
What is $\mathcal{EL}$?

What a tiny logic!?

- **We can say in $\mathcal{EL}$**
  
  Hand $\sqsubseteq$ $\exists$ hasPart.Finger

- **We’d like to say, but can’t**
  
  MildFlu $\equiv$ Flu $\sqcap$ $\forall$ symptom.$\neg$Triv
  Triv $\equiv$ Cough $\sqcup$ Sneeze. $\sqcap$ Headache
  MildFlu $\equiv$ Flu $\sqcap$ $\forall$ symptom.$\neg$Fever

- **but we can’t say**
  
  Hand $\sqsubseteq$ $=$5 hasPart.Finger
  Finger $\sqsubseteq$ $\exists$ hasPart$^\neg$.Hand

- **all we can say (in $\mathcal{EL}^{++}$) is**
  
  MildFlu $\sqsubseteq$ Flu
  Cough $\sqsubseteq$ Triv, Sneeze $\sqsubseteq$ Triv, ...
  MildFlu $\sqcap$ $\exists$ symptom.Fever $\sqsubseteq$ $\bot$
What is $\mathcal{EL}$?

What a tiny logic!?

**✓** We can say in $\mathcal{EL}$

Hand $\sqsubseteq \exists \text{hasPart}.\text{Finger}$

**✗** We’d like to say, but can’t

MildFlu $\equiv \text{Flu} \sqcap \forall \text{symptom}.\neg \text{Triv}$

Triv $\equiv \text{Cough} \sqcup \text{Sneeze}. \sqcup \text{Headache}$

MildFlu $\equiv \text{Flu} \sqcap \forall \text{symptom}.\neg \text{Fever}$

**✗** but we can’t say

Hand $\sqsubseteq =5 \text{hasPart}.\text{Finger}$

Finger $\sqsubseteq \exists \text{hasPart}^{-}.\text{Hand}$

**✓** all we can say (in $\mathcal{EL}^{++}$) is

MildFlu $\sqsubseteq \text{Flu}$

Cough $\sqsubseteq \text{Triv}, \text{Sneeze} \sqsubseteq \text{Triv}, ...$

MildFlu $\sqcap \exists \text{symptom}.\text{Fever} \sqsubseteq \bot$

$\mathcal{EL}^{++}$ is used in some large-scale ontologies, e.g., SNOMED
EL is not so tiny – an example ontology

- Endocardium ⊑ Tissue ⊓ ∃cont-in.HeartWall
  ⊓ ∃cont-in.HeartValve

- HeartWall ⊑ BodyWall ⊓ ∃part-of.Heart

- HeartValve ⊑ BodyValve ⊓ ∃part-of.Heart

- Endocarditis ⊑ Inflammation ⊓ ∃has-loc.Endocardium

- Inflammation ⊑ Disease ⊓ ∃acts-on.Tissue

- Heartdisease ⊓ ∃has-loc.HeartValve ⊑ CriticalDisease

- Heartdisease ⊑ Disease ⊓ ∃has-loc.Heart

Taken from [Baader et al. 2006]
\( \mathcal{EL}(+) \) is not so tiny – an example ontology

\[
\begin{align*}
\text{Endocardium} & \sqsubseteq \text{Tissue} \sqcap \exists \text{cont-in. HeartWall} \sqcap \\
& \quad \exists \text{cont-in. HeartValve} \\\n\text{HeartWall} & \sqsubseteq \text{BodyWall} \sqcap \exists \text{part-of. Heart} \\\n\text{HeartValve} & \sqsubseteq \text{BodyValve} \sqcap \exists \text{part-of. Heart} \\\n\text{Endocarditis} & \sqsubseteq \text{Inflammation} \sqcap \exists \text{has-loc. Endocardium} \\\n\text{Inflammation} & \sqsubseteq \text{Disease} \sqcap \exists \text{acts-on. Tissue} \\\n\text{Heartdisease} & \sqsubseteq \exists \text{has-loc. HeartValve} \sqcap \text{CriticalDisease} \\\n\text{Heartdisease} & \equiv \text{Disease} \sqcap \exists \text{has-loc. Heart} \\\n\end{align*}
\]

\( \mathcal{EL}^+ \) \bigg\{
\begin{align*}
\text{part-of} & \circ \text{part-of} \sqsubseteq \text{part-of} \\
\text{part-of} & \sqsubseteq \text{cont-in} \\
\text{has-loc} & \circ \text{cont-in} \sqsubseteq \text{has-loc}
\end{align*}
\bigg\}

Taken from [Baader et al. 2006]
Satisfiability and subsumption

**Satisfiability + coherence are trivial:** every $\mathcal{EL}$-TBox is coherent

because ?
Satisfiability and subsumption

**Satisfiability + coherence are trivial:** every \( \mathcal{EL} \)-TBox is coherent

- \( \mathcal{I} \) with \( A^\mathcal{I} = \Delta^\mathcal{I} \) and \( r^\mathcal{I} = \Delta^\mathcal{I} \times \Delta^\mathcal{I} \), for all concept names \( A \) and role names \( r \), satisfies every \( \mathcal{EL} \) axiom

- \( \mathcal{I} \) with \( A^\mathcal{I} = r^\mathcal{I} = \emptyset \) doesn’t – **why?**
Satisfiability and subsumption

Satisfiability + coherence are trivial: every $\mathcal{EL}$-TBox is coherent

- $\mathcal{I}$ with $A^\mathcal{I} = \Delta^\mathcal{I}$ and $r^\mathcal{I} = \Delta^\mathcal{I} \times \Delta^\mathcal{I}$,
  for all concept names $A$ and role names $r$,
  satisfies every $\mathcal{EL}$ axiom

- ($\mathcal{I}$ with $A^\mathcal{I} = r^\mathcal{I} = \emptyset$ doesn’t – why?)

Subsumption ?
Satisfiability and subsumption

Satisfiability + coherence are trivial: every $\mathcal{EL}$-TBox is coherent

- $\mathcal{I}$ with $A^\mathcal{I} = \Delta^\mathcal{I}$ and $r^\mathcal{I} = \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, for all concept names $A$ and role names $r$,
satisfies every $\mathcal{EL}$ axiom
- $(\mathcal{I}$ with $A^\mathcal{I} = r^\mathcal{I} = \emptyset$ doesn’t – why?)

Subsumption isn’t:
does the following TBox entail $A \sqsubseteq B$? $A' \sqsubseteq B'$?

$$\exists r.A \sqsubseteq \exists r.B$$
$$A' \equiv \exists r.\exists r.A$$
$$B' \equiv \exists r.\exists r.B$$
Satisfiability and subsumption

Satisfiability + coherence are trivial: every $\mathcal{EL}$-TBox is coherent
- $\mathcal{I}$ with $A^\mathcal{I} = \Delta^\mathcal{I}$ and $r^\mathcal{I} = \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, for all concept names $A$ and role names $r$, satisfies every $\mathcal{EL}$ axiom
- $(\mathcal{I}$ with $A^\mathcal{I} = r^\mathcal{I} = \emptyset$ doesn’t – why?)

Subsumption isn’t:
does the following TBox entail $A \sqsubseteq B$? $A' \sqsubseteq B'$?

$$\exists r. A \sqsubseteq \exists r. B$$

$$A' \equiv \exists r. \exists r. A$$

$$B' \equiv \exists r. \exists r. B$$

Without negation, they are not interreducible: Theorem 1 fails!
An Algorithm for $\mathcal{EL}$ subsumption

Goal: present a decision procedure for subsumption in $\mathcal{EL}$

Outline:
1. Normalisation procedure
2. Decision procedure
   (simple, naïve, without optimisations)
And now . . .

1. What is $\mathcal{EL}$?

2. Normalisation

3. A simple poly-time reasoning algorithm
Normal form

...keeps the reasoning procedure simple

**Definition**
An $\mathcal{E}L$ ontology is in **normal form** if all axioms have these forms:

\[
A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \\
A \sqsubseteq \exists r. B \\
\exists r. A \sqsubseteq B
\]

$A_{(i)}, B$: concepts **names**  \quad \quad r: \text{role}  \quad \quad n \geq 1$ is an integer
The normalisation procedure

- ... applies **normalisation rules** to axioms in a given TBox $\mathcal{T}$
- each rule transforms an axiom into one or several shorter ones
- old axiom is removed from $\mathcal{T}$; new axioms are added
- $\leadsto$ results in an “equivalent” TBox $\mathcal{T}'$
The normalisation rules

NF1
Input: \( C \equiv D \)
Output: \( C \sqsubseteq D \), \( D \sqsubseteq C \)

NF2
Input: \( C \sqsubseteq D \)
Output: \( C \sqsubseteq A \), \( A \sqsubseteq D \)

\( C(i) \) \( D \) arbitrary concepts
\( C(i) \) \( D \) complex concepts
\( B \) concept name
\( A \) fresh concept name
The normalisation rules

**NF1**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C \equiv D$</td>
<td>$C \sqsubseteq D$, $D \sqsubseteq C$</td>
</tr>
</tbody>
</table>

**NF2**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C \sqsubseteq D$</td>
<td>$C \sqsubseteq A$, $A \sqsubseteq D$</td>
</tr>
</tbody>
</table>

**NF3**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\exists r. C \sqsubseteq D$</td>
<td>$C \sqsubseteq A$, $\exists r. A \sqsubseteq D$</td>
</tr>
</tbody>
</table>

**NF4**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1 \sqcap \ldots \sqcap C_i \sqcap \ldots \sqcap C_n \sqsubseteq D$</td>
<td>$C_i \sqsubseteq A$, $C_1 \sqcap \ldots \sqcap A \sqcap \ldots \sqcap C_n \sqsubseteq D$</td>
</tr>
</tbody>
</table>

**NF5**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B \sqsubseteq \exists r. C$</td>
<td>$B \sqsubseteq \exists r. A \sqsubseteq C$</td>
</tr>
</tbody>
</table>

**NF6**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$</td>
<td>$B \sqsubseteq C_1 \ldots \sqcap B \sqsubseteq C_n$</td>
</tr>
</tbody>
</table>

**Abbreviations**

- **C** (i) arbitrary concepts
- **C(i) D** complex concepts
- **B** concept name
- **A** fresh concept name

---

*Uli Sattler*  
*DL: EL*  
*13*
The normalisation rules

**NF1**

**Input**  \( C \equiv D \)

**Output**  \( C \sqsubseteq D \quad D \sqsubseteq C \)

**NF2**

**Input**  \( C \sqsubseteq D \)

**Output**  \( C \sqsubseteq A \quad A \sqsubseteq D \)

**NF3**

**Input**  \( \exists r. C \sqsubseteq D \)

**Output**  \( C \sqsubseteq A \quad \exists r. A \sqsubseteq D \)

**NF4**

**Input**  \( C_1 \sqcap \ldots \sqcap C_i \sqcap \ldots \sqcap C_n \sqsubseteq D \)

**Output**  \( C_i \sqsubseteq A \quad C_1 \sqcap \ldots \sqcap A \sqcap \ldots \sqcap C_n \sqsubseteq D \)

**NF5**

**Input**  \( B \sqsubseteq \exists r. C \)

**Output**  \( B \sqsubseteq \exists r. A \quad A \sqsubseteq C \)

**NF6**

**Input**  \( B \sqsubseteq C_1 \sqcap \ldots \sqcap C_n \)

**Output**  \( B \sqsubseteq C_1 \quad \ldots \quad B \sqsubseteq C_n \)
The normalisation procedure

Given TBox $\mathcal{T}$, apply NF1–NF7 axiom-wise until none can be applied
The normalisation procedure

Given TBox $\mathcal{T}$, apply NF1–NF7 axiom-wise until none can be applied

The result $\mathcal{T}'$

- contains new concept names $A_1, \ldots, A_k$
- is of size linear in the size of $\mathcal{T}$
- is “equivalent” to $\mathcal{T}$
The normalisation procedure

Given TBox $\mathcal{T}$, apply NF1–NF7 axiom-wise until none can be applied.

The result $\mathcal{T}'$ contains new concept names $A_1, \ldots, A_k$.

- is of size linear in the size of $\mathcal{T}$.
- is “equivalent” to $\mathcal{T}$ ...

Lemma

- For every model $\mathcal{I} \models \mathcal{T}$, there is a model $\mathcal{J} \models \mathcal{T}'$
  such that $X^\mathcal{J} = X^\mathcal{I}$ for all $X \notin \{A_1, \ldots, A_k\}$.
- For every model $\mathcal{J} \models \mathcal{T}'$, it holds that $\mathcal{I} \models \mathcal{T}$. 

Consequence: $\mathcal{T}'$ is equivalent to $\mathcal{T}$ w.r.t. subsumption: $\mathcal{T} \models C \sqsubseteq D$ iff $\mathcal{T}' \models C \sqsubseteq D$ for all $C, D$ that don’t use the $A_i$.

Details and Example: see [Suntisrivaraporn 2005, pg. 37–39]
The normalisation procedure

Given TBox $\mathcal{T}$, apply NF1–NF7 axiom-wise until none can be applied

The result $\mathcal{T}'$ contains new concept names $A_1, \ldots, A_k$

is of size linear in the size of $\mathcal{T}$

is “equivalent” to $\mathcal{T}$ ...

Lemma

• For every model $\mathcal{I} \models \mathcal{T}$, there is a model $\mathcal{J} \models \mathcal{T}'$ such that $X^\mathcal{J} = X^\mathcal{I}$ for all $X \notin \{A_1, \ldots, A_k\}$.

• For every model $\mathcal{J} \models \mathcal{T}'$, it holds that $\mathcal{I} \models \mathcal{T}$.

Consequence: $\mathcal{T}'$ is equivalent to $\mathcal{T}$ w.r.t. subsumption:

$\mathcal{I} \models C \sqsubseteq D$ iff $\mathcal{T}' \models C \sqsubseteq D$

for all $C, D$ that don’t use the $A_i$
The normalisation procedure

Given TBox $\mathcal{T}$, apply NF1–NF7 axiom-wise until none can be applied

The result $\mathcal{T}'$
- contains new concept names $A_1, \ldots, A_k$
- is of size linear in the size of $\mathcal{T}$
- is “equivalent” to $\mathcal{T}$ ...

Lemma

- For every model $\mathcal{I} \models \mathcal{T}$, there is a model $\mathcal{J} \models \mathcal{T}'$ such that $X^\mathcal{J} = X^\mathcal{I}$ for all $X \notin \{A_1, \ldots, A_k\}$.
- For every model $\mathcal{J} \models \mathcal{T}'$, it holds that $\mathcal{I} \models \mathcal{T}$.

Consequence: $\mathcal{T}'$ is equivalent to $\mathcal{T}$ w.r.t. subsumption:

$\mathcal{T} \models C \sqsubseteq D$ iff $\mathcal{T}' \models C \sqsubseteq D$

for all $C, D$ that don’t use the $A_i$.
And now ...

1. What is $\mathcal{EL}$?

2. Normalisation

3. A simple poly-time reasoning algorithm
What is $\mathcal{EL}$?

Normalisation

Reasoning

Initial assumptions

Input: TBox $\mathcal{T}$, concept names $A, B$

Question: does $\mathcal{T} \models A \sqsubseteq B$ hold?

Assumption of $A, B$ being concept names is no real restriction:

$$
\mathcal{T} \models C \sqsubseteq D
$$

$$
\uparrow
$$

$$
\mathcal{T} \cup \{ A \equiv C, \ B \equiv D \} \models A \sqsubseteq B
$$
Deciding subsumptions via subsumer sets

**Subsumer** of $A$: a concept name $B$ (or $\top$) with $\mathcal{T} \models A \sqsubseteq B$

**Subsumer set** $S(A)$: set that contains subsumers of $A$
Deciding subsumptions via subsumer sets

**Subsumer** of $A$: a concept name $B$ (or $\top$) with $\mathcal{T} \models A \sqsubseteq B$

**Subsumer set** $S(A)$: set that contains subsumers of $A$

**Representation of subsumer sets:** in a labelled graph $G(\mathcal{T})$

- Nodes of $G(\mathcal{T}) =$ concept names (or $\top$) in $\mathcal{T}$
- Label of node $A$: $S(A)$
  
  $B \in S(A)$ means $\mathcal{T} \models A \sqsubseteq B$
- Label of edge $(A, B)$: set $R(A, B)$ of roles
  
  $r \in R(A, B)$ means $\mathcal{T} \models A \sqsubseteq \exists r.B$
Deciding subsumptions via subsumer sets

**Subsumer** of $A$: a concept name $B$ (or $\top$) with $\mathcal{T} \models A \sqsubseteq B$

**Subsumer set** $S(A)$: set that contains subsumers of $A$

**Representation of subsumer sets:** in a labelled graph $G(\mathcal{T})$

- Nodes of $G(\mathcal{T}) = \text{concept names (or } \top \text{)} \text{ in } \mathcal{T}$
- Label of node $A$: $S(A)$
  
  $B \in S(A)$ means $\mathcal{T} \models A \sqsubseteq B$

- Label of edge $(A, B)$: set $R(A, B)$ of roles
  
  $r \in R(A, B)$ means $\mathcal{T} \models A \sqsubseteq \exists r.B$

**Outline of the procedure:**

1. Set $S(A) = \{A, \top\}$ for every $A$

2. Monotonically build $G(\mathcal{T})$
   by exhaustively applying completion rules

3. Check whether $B \in S(A)$ to determine whether $\mathcal{T} \models A \sqsubseteq B$
The completion rules

R1 If $A_1 \cap \ldots \cap A_n \subseteq B \in \mathcal{T}$ and $A_1, \ldots, A_n \in S(X)$ but $B \not\in S(X)$ then add $B$ to $S(X)$
The completion rules

R1  If $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \in \mathcal{T}$
and $A_1, \ldots, A_n \in S(X)$ but $B \notin S(X)$
then add $B$ to $S(X)$

R2  If $A \sqsubseteq \exists r. B \in \mathcal{T}$
and $A \in S(X)$ but $r \notin R(X, B)$
then add $r$ to $R(X, B)$
The completion rules

R1  If \( A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \in \mathcal{T} \)
and \( A_1, \ldots, A_n \in S(X) \) but \( B \not\in S(X) \)
then add \( B \) to \( S(X) \)

R2  If \( A \sqsubseteq \exists r.B \in \mathcal{T} \)
and \( A \in S(X) \) but \( r \not\in R(X, B) \)
then add \( r \) to \( R(X, B) \)

R3  If \( \exists r.A \sqsubseteq B \in \mathcal{T} \)
and \( r \in R(X, Y) \) and \( A \in S(Y) \) but \( B \not\in S(X) \)
then add \( B \) to \( S(X) \)
The “naïve” subsumption algorithm [Baader et al. 2006]

Algorithm 1

Input: $\mathcal{EL}$ ontology $\mathcal{T}$
Output: $S(.)$ such that $\mathcal{T} \models A \sqsubseteq B$ iff $B \in S(A)$

$\mathcal{T}^{' } := \text{Normalise}(\mathcal{T})$ \hspace{1cm} \% by applying NF1 - NF6 exhaustively

Initialise graph for $\mathcal{T}^{' }$:

For each concept name $A$ in $\mathcal{T}^{' }$ (or $\top$)
create a node $A$ with $S(A) := \{A, \top\}$
set all edge labels $R(X, Y) := \emptyset$

Exhaustively apply rules R1-R3 to graph
Output resulting graph
Exercise

Let’s apply the normalisation procedure to the TBox

\[ T = \{ A \sqsubseteq B \sqcap \exists r.C, \\
C \sqsubseteq \exists s.D, \\
\exists r.\exists s.T \sqcap B \sqsubseteq D \} \]

and then check whether it entails

\[ A \sqsubseteq D. \]
Summary

Algorithm 1 . . .

- terminates in time **polynomial** in the size of $\mathcal{T}$
- constructs a **canonical model** of $\mathcal{T}$
- is **sound** and **complete**: outputs yes iff $\mathcal{T} \models A \sqsubseteq B$
- is **one pass** (all subsumptions in 1 pass)
- is still slow for big ontologies:
  ...search for applicable rules over 100K concept names/nodes

Smarter versions of Algorithm 1 . . .

- are goal-oriented, “one-pass”
- are implemented in the reasoners CEL, JCEL, ... for the
  extension $\mathcal{EL}^{++}$
- can be extended even to the Horn fragment of $\mathcal{SHIQ}$

For details see [Baader et al. 2005, Baader et al. 2006, Kazakov 2009].
References: links

Bio-medical ontologies

- **SNOMED**, the systematized nomenclature of human and veterinary medicine
- **Galen**
  [http://www.opengalen.org](http://www.opengalen.org)
- **Go**, the Gene Ontology
References: articles (1)

F. Baader.
Terminological cycles in a description logic with existential restrictions.
http://lat.inf.tu-dresden.de/research/papers.html#2003

F. Baader, S. Brandt, and C. Lutz.
Pushing the $\mathcal{EL}$ envelope.

Efficient reasoning in $\mathcal{EL}^+$.
S. Brandt.
Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and – what else?
http://www.cs.man.ac.uk/~sbrandt/papers.html

Y. Kazakov:
Consequence-Driven Reasoning for Horn $SHIQ$ Ontologies.

B. Suntisrivaraporn.
Optimization and Implementation of Subsumption Algorithms for the Description Logic $EL$ with Cyclic TBoxes and General Concept Inclusion Axioms.
http://lat.inf.tu-dresden.de/research/papers.html#2005
What has been left out

- Loads of complexity results
- Other complexity measures
  - data complexity, relevant for OBDA – see Misha’s course on Thursday!
  - average case
- Other (reasoner) performance considerations
  - what makes reasoning hard: size, tree-width
  - robustness
  - robustness under (small) changes to $\mathcal{O}$ & performance homo/heterogeneity
- Other reasoning problems
  - module extraction and inseparability
  - decomposition of ontologies
  - entailment explanation and justifications

Ask us for pointers, or look at Thomas Schneider & my ESSLLI 2012 course notes
Thank you for your attention!