Chapter 4

Reasoning in DLs with tableau algorithms

We start with an algorithm for deciding consistency of an ABox without a TBox since this covers most of the inference problems introduced in Chapter 2:

- acyclic TBoxes can be eliminated by expansion
- satisfiability, subsumption, and the instance problem can be reduced to ABox consistency

The tableau-based consistency algorithm tries to generate a finite model for the input ABox $\mathcal{A}_0$:

- applies expansion rules to extend the ABox *one rule per constructor*
- checks for obvious contradictions (clashes)
- an ABox that is complete (no rule applies) and clash-free (no obvious contradictions) describes a model

Tableau algorithm

example

$\mathcal{T}$ $\boxed{\text{GoodStudent} \equiv \text{Smart} \sqcap \text{Studious}}$

Subsumption question:

$$\exists\text{attended.Smart} \sqcap \exists\text{attended.Studious} \sqsubseteq^{?}_{\mathcal{T}} \exists\text{attended.GoodStudent}$$

Reduction to satisfiability: is the following concept unsatisfiable w.r.t. $\mathcal{T}$?

$$\exists\text{attended.Smart} \sqcap \exists\text{attended.Studious} \sqcap \neg\exists\text{attended.GoodStudent}$$

Reduction to consistency: is the following ABox inconsistent w.r.t. $\mathcal{T}$?

$$\{\, a : (\exists\text{attended.Smart} \sqcap \exists\text{attended.Studious} \sqcap \neg\exists\text{attended.GoodStudent}) \,\}$$

Expansion: is the following ABox inconsistent?

$$\{\, a : (\exists\text{attended.Smart} \sqcap \exists\text{attended.Studious} \sqcap \neg\exists\text{attended.}(\text{Smart} \sqcap \text{Studious})) \,\}$$

Negation normal form: is the following ABox inconsistent?

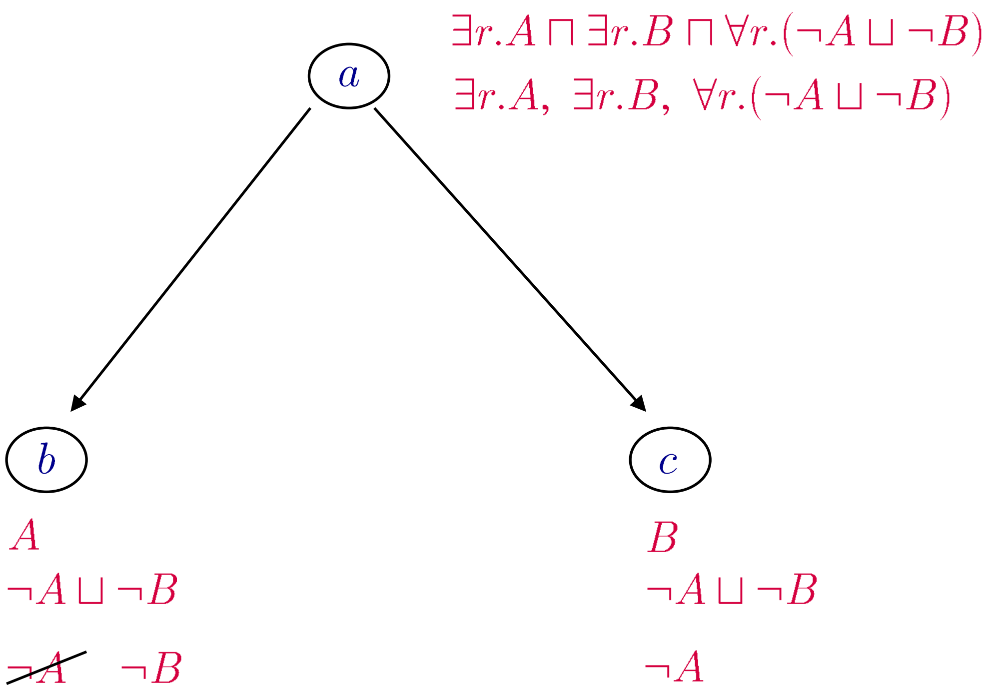
$$\{\, a : (\exists\text{attended.Smart} \sqcap \exists\text{attended.Studious} \sqcap \forall\text{attended.}(\neg\text{Smart} \sqcup \neg\text{Studious})) \,\}$$

Tableau algorithm

example continued

Is the following ABox inconsistent?

$\{ a : (\exists \text{attended}.\text{Smart} \sqcap \exists \text{attended}.\text{Studious} \sqcap \forall \text{attended}.(\neg\text{Smart} \sqcup \neg\text{Studious})) \}$



complete and clash-free ABox yields a model for the input ABox

and thus a counterexample to the subsumption relationship

Dresden

Tableau algorithm

more formal description

Input: An $\mathcal{ALC}$-ABox $\mathcal{A}_0$

Output: "yes" if $\mathcal{A}_0$ is consistent

"no" otherwise

negation only in front of concept names

Preprocessing: normalize the ABox

- transform all concept descriptions in $\mathcal{A}_0$ into negation normal form (NNF) by applying the following equivalence-preserving rules:

$$\begin{array}{rcl} \neg(C \sqcap D) & \rightsquigarrow & \neg C \sqcup \neg D \\ \neg(C \sqcup D) & \rightsquigarrow & \neg C \sqcap \neg D \\ \neg\neg C & \rightsquigarrow & C \\ \neg(\exists r.C) & \rightsquigarrow & \forall r.\neg C \\ \neg(\forall r.C) & \rightsquigarrow & \exists r.\neg C \end{array}$$

The NNF can be computed in polynomial time, and it does not change the semantics of the concept.

Exercise!

Dresden

Tableau algorithm

more formal description

Input: An $\mathcal{ALC}$-ABox $\mathcal{A}_0$

Output: "yes" if $\mathcal{A}_0$ is consistent

"no" otherwise

negation only in front of concept names

Preprocessing: normalize the ABox

- transform all concept descriptions in $\mathcal{A}_0$ into negation normal form (NNF)
- ensure that the ABox is non-empty
  by adding $a : \top$ for an arbitrary individual name a if needed
- ensure that every individual name a occurring in the ABox
  occurs in a concept assertion by adding $a : \top$ if needed

We assume in the following that the input ABox $\mathcal{A}_0$ is normalized in this sense.

Dresden

Tableau algorithm

more formal description

Application of expansion rules:

- The rules are triggered by the presence of certain assertions in the current ABox,
- and extend the ABox by new assertions.
- Deterministic rule: only one option for how to extend the ABox.
- Nondeterministic rule: several options for how to extend the ABox, where at least one of them must lead to success.

A b

$\neg A \sqcup \neg B$

~~$\neg A$~~ $\neg B$

Dresden

Tableau algorithm

more formal description

Application of expansion rules:

- The rules are triggered by the presence of certain assertions in the current ABox,
- and extend the ABox by new assertion.
- Deterministic rule: only one option for how to extend the ABox.
- Nondeterministic rule: several options for how to extend the ABox, where at least one of them must lead to success.
  - Nondeterministic algorithm: always "guesses" the "right" option.
  - Deterministic realization: try options consecutively and backtrack in case of failure.

Expansion rules

one for every constructor (except for negation)

The $\sqcap$-rule

Condition: $\mathcal{A}$ contains $a:(C \sqcap D)$, but not both $a:C$ and $a:D$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a:C, a:D\}$

The $\sqcup$-rule

Condition: $\mathcal{A}$ contains $a:(C \sqcup D)$, but neither $a:C$ nor $a:D$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a:X\}$ for some $X \in \{C, D\}$

The $\exists$-rule

Condition: $\mathcal{A}$ contains $a:(\exists r.C)$, but there is no b with $\{(a,b):r, b:C\} \subseteq \mathcal{A}$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{(a,d):r, d:C\}$ where d is new in $\mathcal{A}$

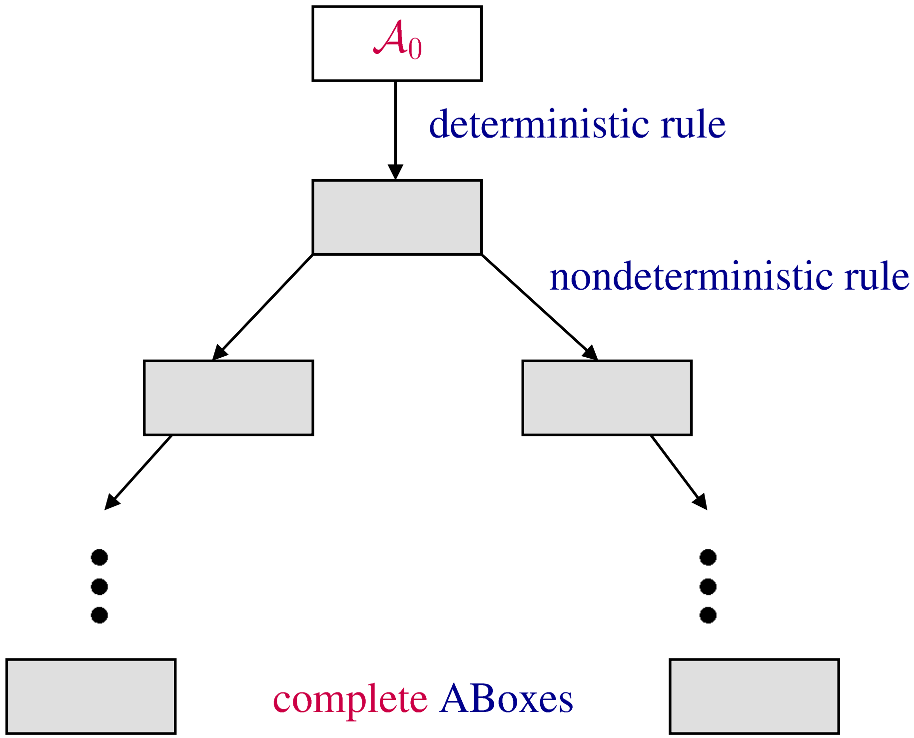
The $\forall$-rule

Condition: $\mathcal{A}$ contains $a:(\forall r.C)$ and $(a,b):r$, but not $b:C$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{b:C\}$

Tableau algorithm

How does it work?



Return "consistent" iff one of these complete ABoxes is clash-free.

Tableau algorithm

more formally

Definition 4.1 (Complete and clash-free ABox)

- An ABox $\mathcal{A}$ contains a clash if

$$\{a : C, a : \neg C\} \subseteq \mathcal{A}$$

for some individual name a, and for some concept C.

- $\mathcal{A}$ is complete if it contains a clash, or if none of the expansion rules is applicable.

Dresden

Tableau algorithm

more formally

The procedure exp:

- takes as input a normalised and clash-free $\mathcal{ALC}$ ABox $\mathcal{A}$, a rule R and an assertion or pair of assertions α such that R is applicable to α in $\mathcal{A}$;
- it returns a set $\mathsf{exp}(\mathcal{A}, R, \alpha)$ containing each of the ABoxes that can result from applying R to α in $\mathcal{A}$.

Examples:

$\mathsf{exp}(\{a : \neg D, a : C \sqcup D\}, \sqcup\text{-rule}, a : C \sqcup D)$

$\mathsf{exp}(\{b : \neg D, a : \forall r.D, (a, b) : r\}, \forall\text{-rule}, (a : \forall r.D, (a, b) : r))$

Dresden

Algorithm consistent()
Input: a normalised $\mathcal{ALC}$ ABox $\mathcal{A}$
if expand($\mathcal{A}$) $\neq \emptyset$ **then**
    return "consistent"
else
    return "inconsistent"

Definition 4.2

deterministic version of the tableau algorithm

Algorithm expand()
Input: a normalised $\mathcal{ALC}$ ABox $\mathcal{A}$
if $\mathcal{A}$ is not complete **then**
    select a rule R that is applicable to $\mathcal{A}$ and an assertion or pair of assertions α in $\mathcal{A}$ to which R is applicable
    if there is $\mathcal{A}' \in \mathsf{exp}(\mathcal{A}, R, \alpha)$ with expand($\mathcal{A}'$) $\neq \emptyset$ **then**
        return expand($\mathcal{A}'$)
    else
        return $\emptyset$
else
    if $\mathcal{A}$ contains a clash **then**
        return $\emptyset$
    else
        return $\mathcal{A}$

Tableau algorithm

example

$$\mathcal{A}_{ex} = \{a : A \sqcap \exists s.F, \quad (a,b) : s,$$
$$a : \forall s.(\neg F \sqcup \neg B), \quad (a,c) : r,$$
$$b : B, \quad c : C \sqcap \exists s.D\}$$

Dresden

Expansion rules

one for every constructor (except for negation)

The $\sqcap$-rule

Condition: $\mathcal{A}$ contains $a:(C \sqcap D)$, but not both $a:C$ and $a:D$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a:C, a:D\}$

The $\sqcup$-rule

Condition: $\mathcal{A}$ contains $a:(C \sqcup D)$, but neither $a:C$ nor $a:D$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a:X\}$ for some $X \in \{C, D\}$

The $\exists$-rule

Condition: $\mathcal{A}$ contains $a:(\exists r.C)$, but there is no b with $\{(a,b):r, b:C\} \subseteq \mathcal{A}$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{(a,d):r, d:C\}$ where d is new in $\mathcal{A}$

The $\forall$-rule

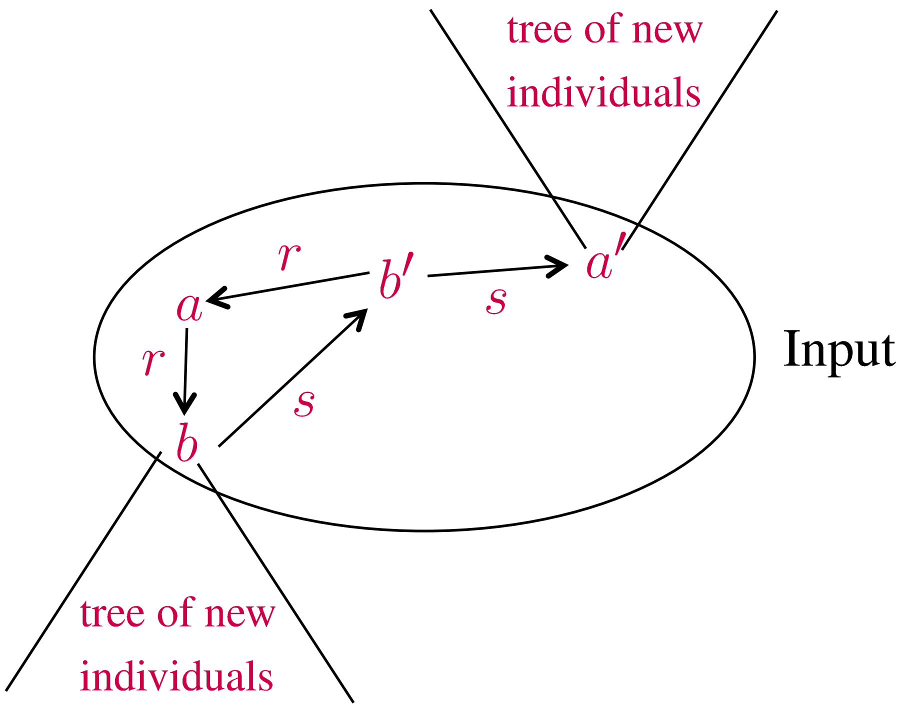
Condition: $\mathcal{A}$ contains $a:(\forall r.C)$ and $(a,b):r$, but not $b:C$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{b:C\}$

Dresden

Trees and forests

In an ABox generated by the algorithm, the individuals generated by the $\exists$-rule form a tree whose root is an individual from the input ABox.



Trees and forests

In an ABox generated by the algorithm, the individuals generated by the $\exists$-rule form a tree whose root is an individual from the input ABox.

- Root individual: individual occurring in the input ABox
- Tree individual: individual generated by the application of the $\exists$-rule
- If the $\exists$-rule adds a tree individual b and a role assertion $(a, b) : r$, then b is a (r-) successor of a and a is a predecessor of b
- We use ancestor and and descendant for the transitive closure of predecessor and successor, respectively

Note: root individuals may have successors and hence descendants, but they have no predecessor or ancestors.

Tableau algorithm

Why is it a decision procedure for consistency?

We need to show:

Termination:
consistent($\mathcal{A}$) terminates for all normalised $\mathcal{ALC}$ ABoxes $\mathcal{A}$

Soundness:
if consistent($\mathcal{A}$) returns "consistent", then $\mathcal{A}$ is consistent

Completeness:
if $\mathcal{A}$ is consistent, then consistent($\mathcal{A}$) returns "consistent"

Dresden

Termination

auxiliary definitions and results

Extend the definition of subconcept to ABoxes and to knowledge bases:

$$\mathsf{sub}(\mathcal{A}) = \bigcup_{a:C\in\mathcal{A}} \mathsf{sub}(C)$$

and for $\mathcal{K} = (\mathcal{T}, \mathcal{A})$,

$$\mathsf{sub}(\mathcal{K}) = \mathsf{sub}(\mathcal{T}) \cup \mathsf{sub}(\mathcal{A}).$$

Set of concepts occurring in a concept assertion:

$$\mathsf{con}_{\mathcal{A}}(a) = \{C \mid a : C \in \mathcal{A}\}.$$

Lemma 4.3

For each $\mathcal{ALC}$ ABox $\mathcal{A}$, we have that $|\,\mathsf{sub}(\mathcal{A})| \leq \sum_{a:C\in\mathcal{A}} \mathsf{size}(C)$.

linear in the size of $\mathcal{A}$

Dresden

Termination

Lemma 4.4 (Termination)

For each normalized $\mathcal{ALC}$ ABox $\mathcal{A}$, consistent$(\mathcal{A})$ terminates.

Proof: blackboard

Soundness

Lemma 4.5 (Soundness)

If $\mathsf{consistent}(\mathcal{A})$ returns "consistent", then $\mathcal{A}$ is consistent.

Proof. Let $\mathcal{A}'$ be the set returned by $\mathsf{expand}(\mathcal{A})$.

Since the algorithm returns "consistent", $\mathcal{A}'$ is a complete and clash-free ABox.

We use $\mathcal{A}'$ to define an interpretation $\mathcal{I}$ and show that it is a model of $\mathcal{A}'$.

$$\Delta^{\mathcal{I}} = \{a \mid a : C \in \mathcal{A}'\}$$

$a^{\mathcal{I}} = a$ for each individual name a occurring in $\mathcal{A}'$

$A^{\mathcal{I}} = \{a \mid A \in \mathsf{con}_{\mathcal{A}'}(a)\}$ for each concept name A in $\mathsf{sub}(\mathcal{A}')$

$r^{\mathcal{I}} = \{(a, b) \mid (a, b) : r \in \mathcal{A}'\}$ for each role r occurring in $\mathcal{A}'$

Since the expansion rules never delete assertions, we have $\mathcal{A} \subseteq \mathcal{A}'$,
so $\mathcal{I}$ is also a model of $\mathcal{A}$.

Dresden

Soundness

proof continued

$$
\begin{aligned}
\Delta^{\mathcal{I}} &= \{a \mid a : C \in \mathcal{A}'\} \\
a^{\mathcal{I}} &= a \text{ for each individual name } a \text{ occurring in } \mathcal{A}' \\
A^{\mathcal{I}} &= \{a \mid A \in \mathsf{con}_{\mathcal{A}'}(a)\} \text{ for each concept name } A \text{ in } \mathsf{sub}(\mathcal{A}') \\
r^{\mathcal{I}} &= \{(a,b) \mid (a,b) : r \in \mathcal{A}'\} \text{ for each role } r \text{ occurring in } \mathcal{A}'
\end{aligned}
$$

The interpretation $\mathcal{I}$ it is a model of $\mathcal{A}'$.

Proof: blackboard

Dresden

Completeness

Lemma 4.6 (Completeness)

If $\mathcal{A}$ is consistent, then $\mathsf{consistent}(\mathcal{A})$ returns "consistent".

Proof. Let $\mathcal{A}$ be consistent, and consider a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{A}$.

Since $\mathcal{A}$ is consistent, it cannot contain a clash.

Thus, if $\mathcal{A}$ is complete, then expand simply returns $\mathcal{A}$ and $\mathsf{consistent}(\mathcal{A})$ returns "consistent".

If $\mathcal{A}$ is not complete, then expand calls itself recursively until $\mathcal{A}$ is complete; each call selects a rule and applies it.

It is thus sufficient to show that rule application preserves consistency.

Proof: blackboard

Tableau algorithm

Why is it a decision procedure for consistency?

We have shown:

Termination:
$\mathsf{consistent}(\mathcal{A})$ terminates for all normalised $\mathcal{ALC}$ ABoxes $\mathcal{A}$

Soundness:
if $\mathsf{consistent}(\mathcal{A})$ returns "consistent", then $\mathcal{A}$ is consistent

Completeness:
if $\mathcal{A}$ is consistent, then $\mathsf{consistent}(\mathcal{A})$ returns "consistent"

Theorem 4.7

The tableau algorithm presented in Definition 4.2 is a decision procedure for the consistency of $\mathcal{ALC}$ ABoxes.

Tableau algorithm

What is its complexity?

We will see in Chapter 5 that the complexity of the $\mathcal{ALC}$ ABox consistency problem is PSPACE-complete.

However, the tableau algorithm as described until now needs exponential time and space for two reasons:

- Due to the nondeterministic $\sqcup$-rule, exponentially many complete ABoxes may be generated.

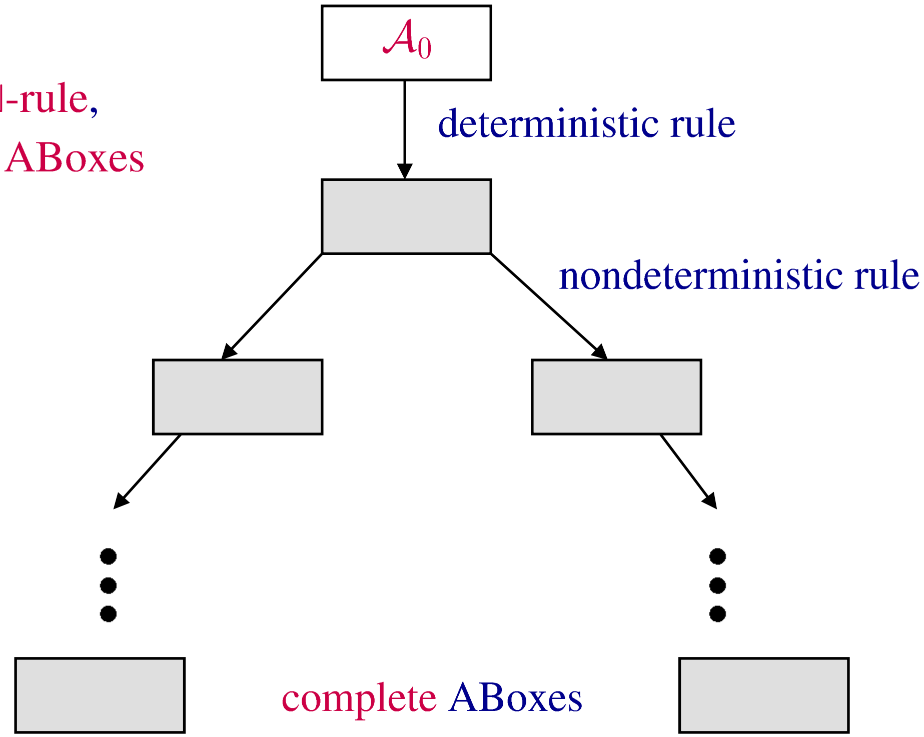


Tableau algorithm

What is its complexity?

We will see in Chapter 5 that the complexity of the $\mathcal{ALC}$ ABox consistency problem is PSPACE-complete.

However, the tableau algorithm as described until now needs exponential time and space for two reasons:

- Due to the nondeterministic $\sqcup$-rule, exponentially many complete ABoxes may be generated.
- Due to the interaction of $\forall$- and $\exists$, complete ABoxes may be exponentially large.

$$
\begin{aligned}
C_1 &:= \exists r.A \sqcap \exists r.B \\
C_{i+1} &:= \exists r.A \sqcap \exists r.B \sqcap \forall r.C_i
\end{aligned}
$$

The call consistent$(\{C_n(a)\})$ generates a single complete ABox of size exponential in n.

size of C_n is linear in n

Dresden

Tableau algorithm

What is its complexity?

The tableau algorithm can be modified such that it uses only polynomial space:

- Due to the nondeterministic $\sqcup$-rule, exponentially many complete ABoxes may be generated.
  - use a nondeterministic algorithm, which always chooses the correct alternative (if possible);
  - thus only one complete ABox is generated;
  - use Savitch's theorem, which says that PSpace = NPSpace.

Dresden

Tableau algorithm

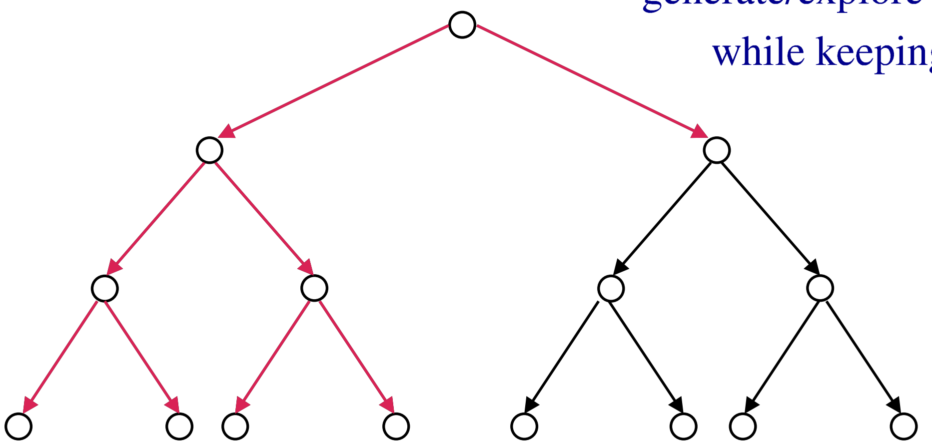
What is its complexity?

The tableau algorithm can be modified such that it uses only polynomial space:

- Due to the nondeterministic $\sqcup$-rule, exponentially many complete ABoxes may be generated.
- Due to the interaction of $\forall$- and $\exists$, complete ABoxes may be exponentially large.

Idea:

generate/explore the tree in a depth-first manner
while keeping only one path in memory



Dresden

Tableau algorithm

w.r.t. acyclic TBoxes

In principle, consistency of ABoxes w.r.t. acyclic TBoxes can be reduced to consistency of ABoxes without TBox by unfolding.

Problem: unfolding of an acyclic TBox may result in an exponential blow-up.

Idea: unfolding only "on demand" (lazy unfolding)

The $\equiv_1$-rule

Condition: $a : A \in \mathcal{A}$, $A \equiv C \in \mathcal{T}$, and $a : C \notin \mathcal{A}$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : C\}$

The $\equiv_2$-rule

Condition: $a : \neg A \in \mathcal{A}$, $A \equiv C \in \mathcal{T}$, and $a : \dot\neg C \notin \mathcal{A}$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : \dot\neg C\}$

$\dot\neg C$

Negation normal form of $\neg C$

Termination, soundness, and completeness can be shown similarly to the case without TBox (Exercise).

Dresden

Tableau algorithm

w.r.t. general TBoxes

Preprocessing: also normalize the TBox

– transform all GCIs in $\mathcal{T}$ into the form $\top \sqsubseteq E$

$$\mathcal{I} \text{ satisfies } C \sqsubseteq D \quad \text{iff} \quad \mathcal{I} \text{ satisfies } \top \sqsubseteq D \sqcup \neg C$$

– transform the right-hand sides E of GCIs $\top \sqsubseteq E$ in $\mathcal{T}$ into NNF

We assume in the following that the
input TBox $\mathcal{T}$ is normalized in this sense.

Dresden

Tableau algorithm

w.r.t. general TBoxes

Add a new expansion rule that takes the semantics of normalized GCIs into account:

The $\sqsubseteq$-rule

Condition: $\boxed{a : C} \in \mathcal{A}$, $\top \sqsubseteq D \in \mathcal{T}$, $a : D \notin \mathcal{A}$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : D\}$

Note: since the input ABox is normalized,
all individuals occur in a concept assertion.

Dresden

Tableau algorithm

w.r.t. general TBoxes

Add a new expansion rule that takes the semantics of normalized GCIs into account:

> The $\sqsubseteq$-rule
>
> *Condition:* $a : C \in \mathcal{A}$, $\top \sqsubseteq D \in \mathcal{T}$, $a : D \notin \mathcal{A}$
>
> *Action:* $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : D\}$

Soundness and completeness of the tableau algorithm extended with this rule is easy to show.

Termination? Need not hold!

Example: $(\{A \sqsubseteq \exists r.A\},\ \{a : A\})$

Dresden

Tableau algorithm

w.r.t. general TBoxes

How can we regain termination.

Definition 4.8 ($\mathcal{ALC}$ blocking)

An individual name b in an $\mathcal{ALC}$ ABox $\mathcal{A}$ is blocked by an individual name a if

- a is an ancestor of b and
- $\mathsf{con}_{\mathcal{A}}(a) \supseteq \mathsf{con}_{\mathcal{A}}(b)$.

Only tree individuals can be blocked.

An individual name b is blocked in $\mathcal{A}$ if

- it is blocked by some individual name a, or
- if one or more of its ancestors is blocked in $\mathcal{A}$.

All descendants of a blocked individual are also blocked.

When it is clear from the context, we may not mention the ABox explicitly; e.g., we may simply say that b is blocked.

Dresden

Tableau algorithm

w.r.t. general TBoxes

The tableau algorithm for $\mathcal{ALC}$ knowledge base consistency uses

- the $\sqcap$-rule, the $\sqcup$-rule, the $\forall$-rule without changes,
- the new $\sqsubseteq$-rule,
- the following modified $\exists$-rule:

The modified $\exists$-rule

Condition: $\mathcal{A}$ contains $a:(\exists r.C)$, but there is no b with $\{(a,b):r, b:C\} \subseteq \mathcal{A}$ and a is not blocked

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{(a,d):r, d:C\}$ where d is new in $\mathcal{A}$

Dresden

Definition 4.9

deterministic version of the tableau algorithm for KB consistency

Algorithm consistent()
Input: a normalised $\mathcal{ALC}$ KB $(\mathcal{T},\mathcal{A})$
if expand$(\mathcal{T},\mathcal{A}) \neq \emptyset$ **then**
    return "consistent"
else
    return "inconsistent"

Algorithm expand()
Input: a normalised $\mathcal{ALC}$ KB $(\mathcal{T},\mathcal{A})$
if $\mathcal{A}$ is not complete **then**
    select a rule R that is applicable to $\mathcal{A}$ and an assertion
    or pair of assertions α in $\mathcal{A}$ to which R is applicable
    if there is $\mathcal{A}' \in$ exp$(\mathcal{A}, R, \alpha)$ with expand$(\mathcal{T}, \mathcal{A}') \neq \emptyset$ **then**
        return expand$(\mathcal{T}, \mathcal{A}')$
    else
        return $\emptyset$
else
    if $\mathcal{A}$ contains a clash **then**
        return $\emptyset$
    else
        return $\mathcal{A}$

Termination

Lemma 4.10 (Termination)

For each normalized $\mathcal{ALC}$ KB $\mathcal{K}$, consistent$(\mathcal{K})$ terminates.

Proof: blackboard

Soundness

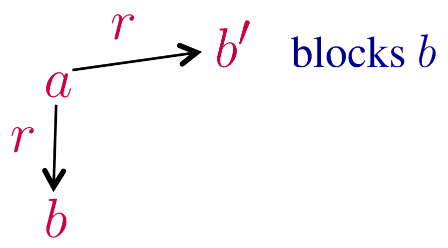
Lemma 4.11 (Soundness)

If consistent$(\mathcal{K})$ returns "consistent", then $\mathcal{K}$ is consistent.

Proof. Let $\mathcal{A}'$ be the set returned by expand$(\mathcal{K})$.

We use $\mathcal{A}'$ to construct a suitable model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{K}$ in two steps:

- Construct a new ABox $\mathcal{A}''$ that contains
  - those axioms in $\mathcal{A}'$ that do not involve blocked individual names
  - new "loop-back" role assertions:



Soundness

Lemma 4.11 (Soundness)

If $\mathsf{consistent}(\mathcal{K})$ returns "consistent", then $\mathcal{K}$ is consistent.

Proof. Let $\mathcal{A}'$ be the set returned by $\mathsf{expand}(\mathcal{K})$.

We use $\mathcal{A}'$ to construct a suitable model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{K}$ in two steps:

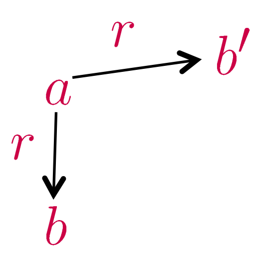
- Construct a new ABox $\mathcal{A}''$ that contains
  - those axioms in $\mathcal{A}'$ that do not involve blocked individual names
  - new "loop-back" role assertions:
- Use $\mathcal{A}''$ to construct a model of $\mathcal{K}$.

Dresden

Soundness

- Construct a new ABox $\mathcal{A}''$ that contains
  - those axioms in $\mathcal{A}'$ that do not involve blocked individual names
  - new "loop-back" role assertions:

$$\mathcal{A}'' = \{a : C \mid a : C \in \mathcal{A}' \text{ and } a \text{ is not blocked}\} \cup$$
$$\{(a, b) : r \mid (a, b) : r \in \mathcal{A}' \text{ and } b \text{ is not blocked}\} \cup$$
$$\{(a, b') : r \mid (a, b) : r \in \mathcal{A}', a \text{ is not blocked and } b \text{ is blocked by } b'\}$$



blocks b

Soundness

- Construct a new ABox $\mathcal{A}''$ that contains
  - those axioms in $\mathcal{A}'$ that do not involve blocked individual names
  - new "loop-back" role assertions:

$$\mathcal{A}'' = \{a : C \mid a : C \in \mathcal{A}' \text{ and } a \text{ is not blocked}\} \cup$$
$$\{(a,b) : r \mid (a,b) : r \in \mathcal{A}' \text{ and } b \text{ is not blocked}\} \cup$$
$$\{(a,b') : r \mid (a,b) : r \in \mathcal{A}', a \text{ is not blocked and } b \text{ is blocked by } b'\}$$

The following holds:

- $\mathcal{A} \subseteq \mathcal{A}''$ and none of the individual names occurring in $\mathcal{A}''$ is blocked
- $\mathsf{con}_{\mathcal{A}''}(a) = \mathsf{con}_{\mathcal{A}'}(a)$ for all individuals a occurring in $\mathcal{A}''$
- Since $\mathcal{A}'$ is clash-free, and complete, $\mathcal{A}''$ is also clash-free and complete

Dresden

Soundness

- Use $\mathcal{A}''$ to construct a model of $\mathcal{K}$.

We construct an interpretation $\mathcal{I}$ from $\mathcal{A}''$ exactly as in the proof of Lemma 4.5:

$$
\begin{aligned}
\Delta^{\mathcal{I}} &= \{a \mid a \text{ is an individual name occurring in } \mathcal{A}''\} \\
a^{\mathcal{I}} &= a \quad \text{for each individual name } a \text{ occurring in } \mathcal{A}'' \\
A^{\mathcal{I}} &= \{a \mid A \in \mathsf{con}_{\mathcal{A}''}(a)\} \quad \text{for each concept name } A \text{ occurring in } \mathcal{A}'' \\
r^{\mathcal{I}} &= \{(a,b) \mid (a,b) : r \in \mathcal{A}''\} \quad \text{for each role } r \text{ occurring in } \mathcal{A}''
\end{aligned}
$$

- $\mathcal{I}$ is a model of $\mathcal{A}''$ and hence of $\mathcal{A}$
- $\mathcal{I}$ is a model of $\mathcal{T}$

Proof: blackboard

Dresden

Completeness

Lemma 4.12 (Completeness)

If $\mathcal{K}$ is consistent, then $\mathsf{consistent}(\mathcal{K})$ returns "consistent".

Proof. It only remains to show that the $\sqsubseteq$-rule preserves KB consistency.

Blackboard

Theorem 4.13

The tableau algorithm presented in Definition 4.9 is a decision procedure for the consistency of $\mathcal{ALC}$ knowledge bases

Dresden

Completeness

Lemma 4.12 (Completeness)

If $\mathcal{K}$ is consistent, then $\mathsf{consistent}(\mathcal{K})$ returns "consistent".

Proof. It only remains to show that the $\sqsubseteq$-rule preserves KB consistency.

Blackboard

Theorem 4.13

The tableau algorithm presented in Definition 4.9 is a decision procedure for the consistency of $\mathcal{ALC}$ knowledge bases

Dresden

Adding number restrictions

Number restrictions: $(\geq n\, r)$, $(\leq n\, r)$ with semantics

$$(\geq n\, r)^{\mathcal{I}} \;:=\; \{d \in \Delta^{\mathcal{I}} \mid \#\{e \mid (d,e) \in r^{\mathcal{I}}\} \geq n\}$$
$$(\leq n\, r)^{\mathcal{I}} \;:=\; \{d \in \Delta^{\mathcal{I}} \mid \#\{e \mid (d,e) \in r^{\mathcal{I}}\} \leq n\}$$

Negation normal form:

$$\neg(\geq n+1\, r) \;\rightsquigarrow\; (\leq n\, r)$$
$$\neg(\geq 0\, r) \;\rightsquigarrow\; \bot$$
$$\neg(\leq n\, r) \;\rightsquigarrow\; (\geq n+1\, r)$$

Extension of algorithm:

- new rules: $\geq$-rule and $\leq$-rule
- new assertions: equality and inequality assertions of the form $x = y$, $x \neq y$ with obvious semantics $x^{\mathcal{I}} = y^{\mathcal{I}}$ and $x^{\mathcal{I}} \neq y^{\mathcal{I}}$.
- new clash

these assertions are viewed as symmetric

Dresden

Adding number restrictions

the tableau rules

The $\geq$-rule

Condition:	$\mathcal{A}$ contains $a : (\geq n\, r)$, but there are no distinct $b_1, \ldots, b_n$ with $\{(a, b_1) : r, \ldots, (a, b_n) : r\} \subseteq \mathcal{A}$
Action:	$\mathcal{A} \longrightarrow \mathcal{A} \cup \{(a, d_1) : r, \ldots, (a, d_n) : r\} \cup \{d_i \neq d_j \mid 1 \leq i < j \leq n\}$ where $d_1, \ldots, d_n$ are new individual names

The $\leq$-rule

Condition:	$\mathcal{A}$ contains $a : (\leq n\, r)$, and there are distinct $b_0, \ldots, b_n$ with $\{(a, b_0) : r, \ldots, (a, b_n) : r)\} \subseteq \mathcal{A}$
Action:	$\mathcal{A} \longrightarrow \mathcal{A}[b_j \mapsto b_i] \cup \{b_i = b_j\}$

b_j *replaced by* b_i

Adding number restrictions

the tableau rules

The $\geq$-rule

Condition:	$\mathcal{A}$ contains $a : (\geq n\, r)$, but there are no distinct $b_1, \ldots, b_n$ with $\{(a, b_1) : r, \ldots, (a, b_n) : r\} \subseteq \mathcal{A}$
Action:	$\mathcal{A} \longrightarrow \mathcal{A} \cup \{(a, d_1) : r, \ldots, (a, d_n) : r\} \cup \{d_i \neq d_j \mid 1 \leq i < j \leq n\}$ where $d_1, \ldots, d_n$ are new individual names

The $\leq$-rule

Condition:	$\mathcal{A}$ contains $a : (\leq n\, r)$, and there are distinct $b_0, \ldots, b_n$ with $\{(a, b_0) : r, \ldots, (a, b_n) : r)\} \subseteq \mathcal{A}$
Action:	$\mathcal{A} \longrightarrow \mathcal{A}[b_j \mapsto b_i] \cup \{b_i = b_j\}$ for $i \neq j$ such that, if b_j is a root individual, then so is b_i.

Dresden

New clash condition

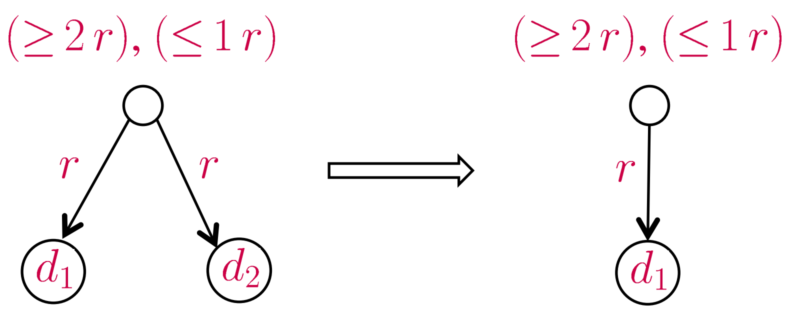
due to inequality assertions

An ABox $\mathcal{A}$ contains a clash if

$$\{a : C, a : \neg C\} \subseteq \mathcal{A} \text{ or } \{a \neq a\} \subseteq \mathcal{A}$$

for some individual name a, and for some concept C.

Prevents generate and identify loops:



$d_1 \neq d_2$

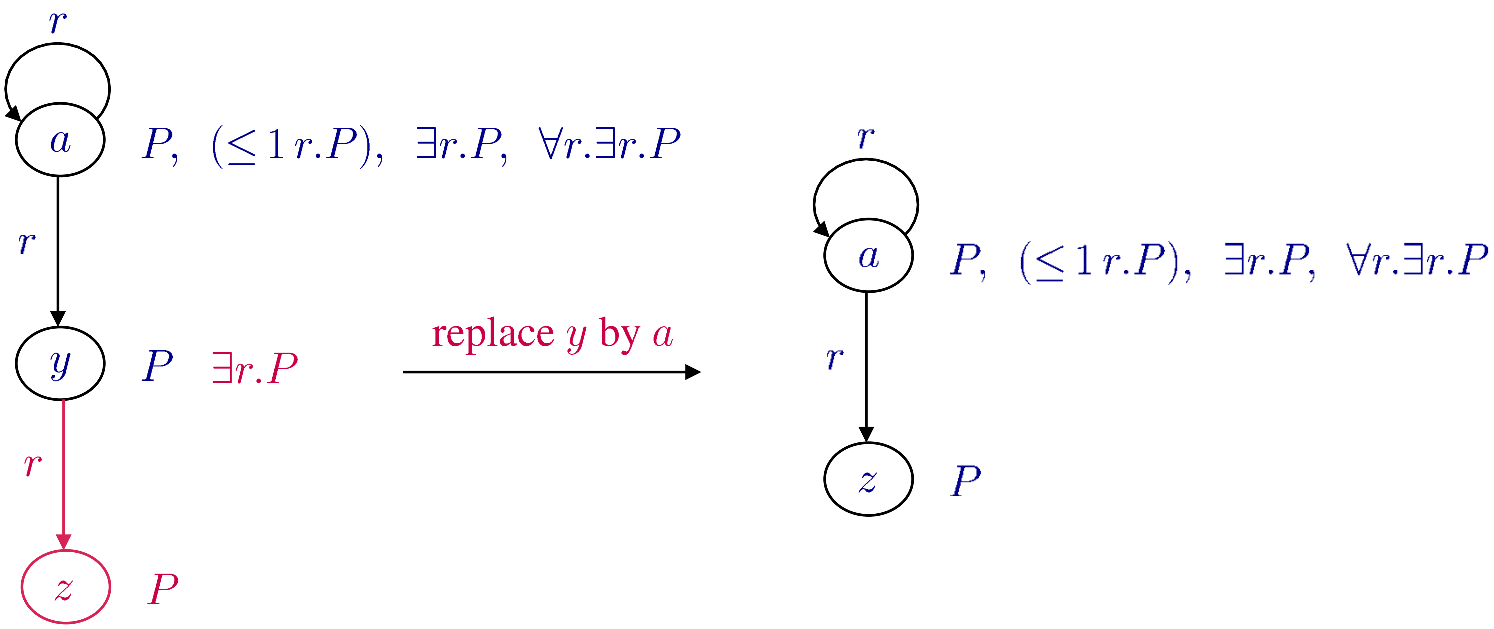
$d_1 \neq d_1$ Clash!

$d_1 = d_2$ And thus no more rules are applicable.

Dresden

Termination

need not hold even without GCIs

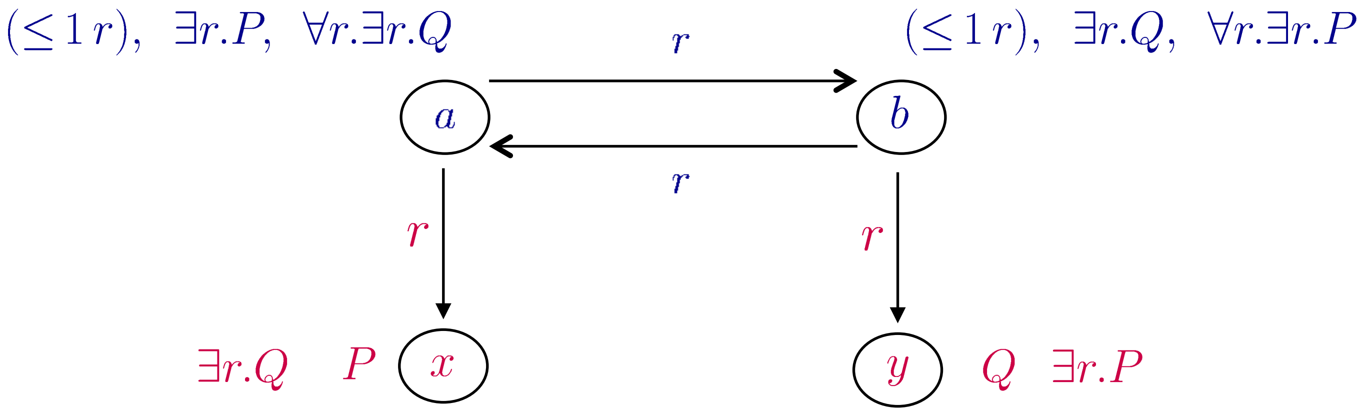


How can we solve this problem?

- In the example, the use of blocking would prevent non-termination: y is blocked by a and thus z would not be generated.
- Does blocking ensure termination in general? No!

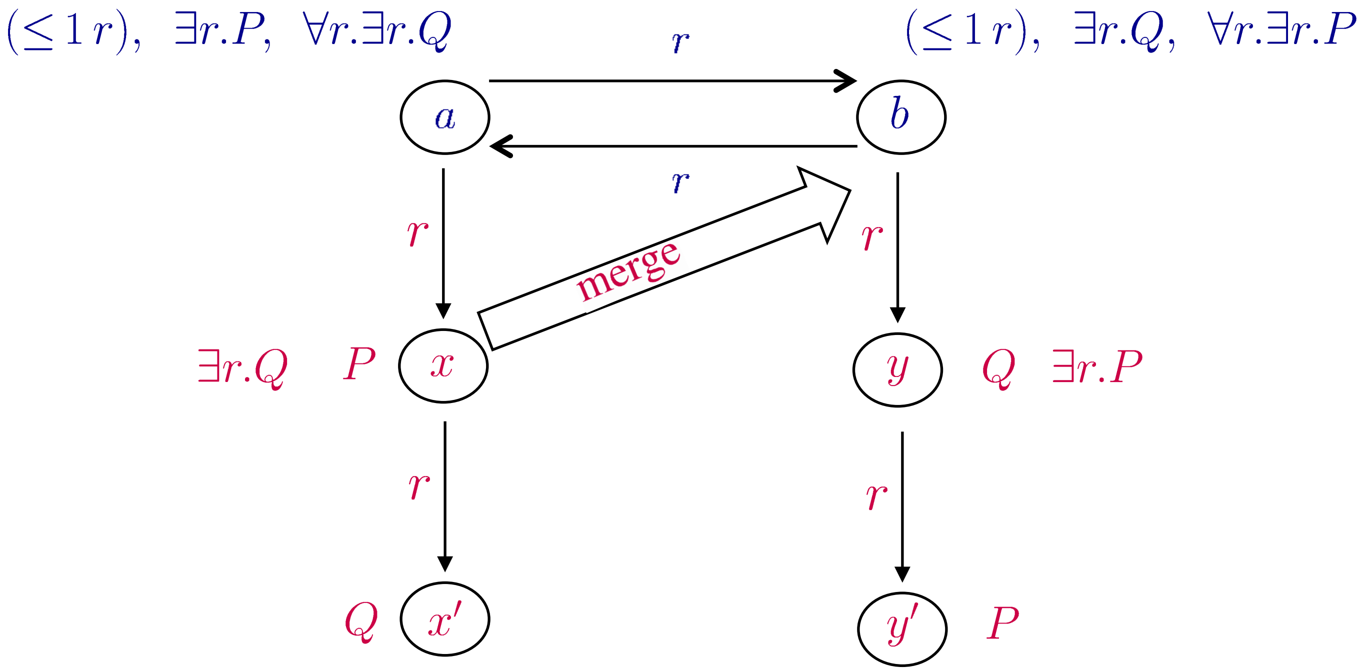
Termination

does not hold even if
blocking as in Definition 4.8 is used



Termination

does not hold even if blocking as in Definition 4.8 is used



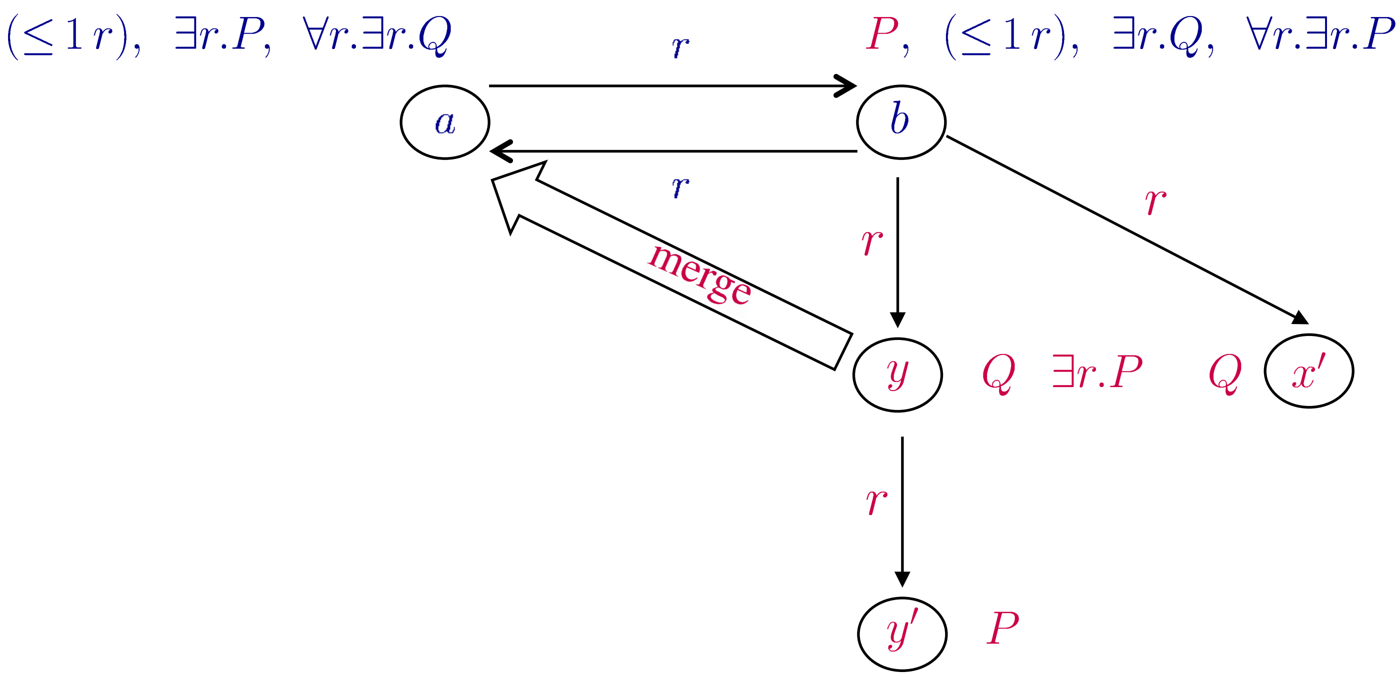
Note: x is not blocked!

Note: y is not blocked!

- a does not satisfy superset condition.
- b is not an ancestor.

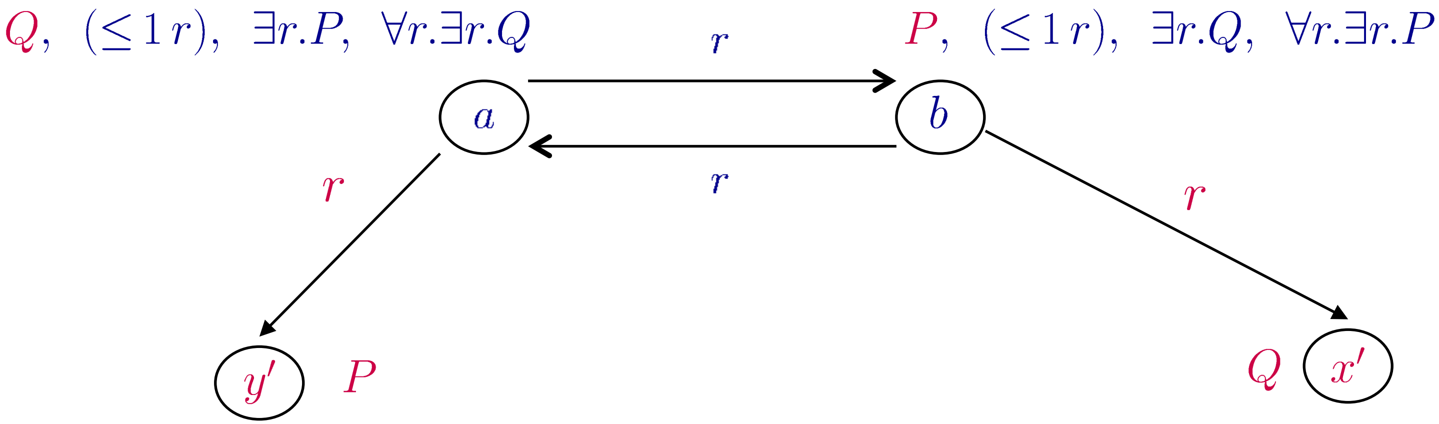
Termination

does not hold even if
blocking as in Definition 4.8 is used



Termination

does not hold even if
blocking as in Definition 4.8 is used



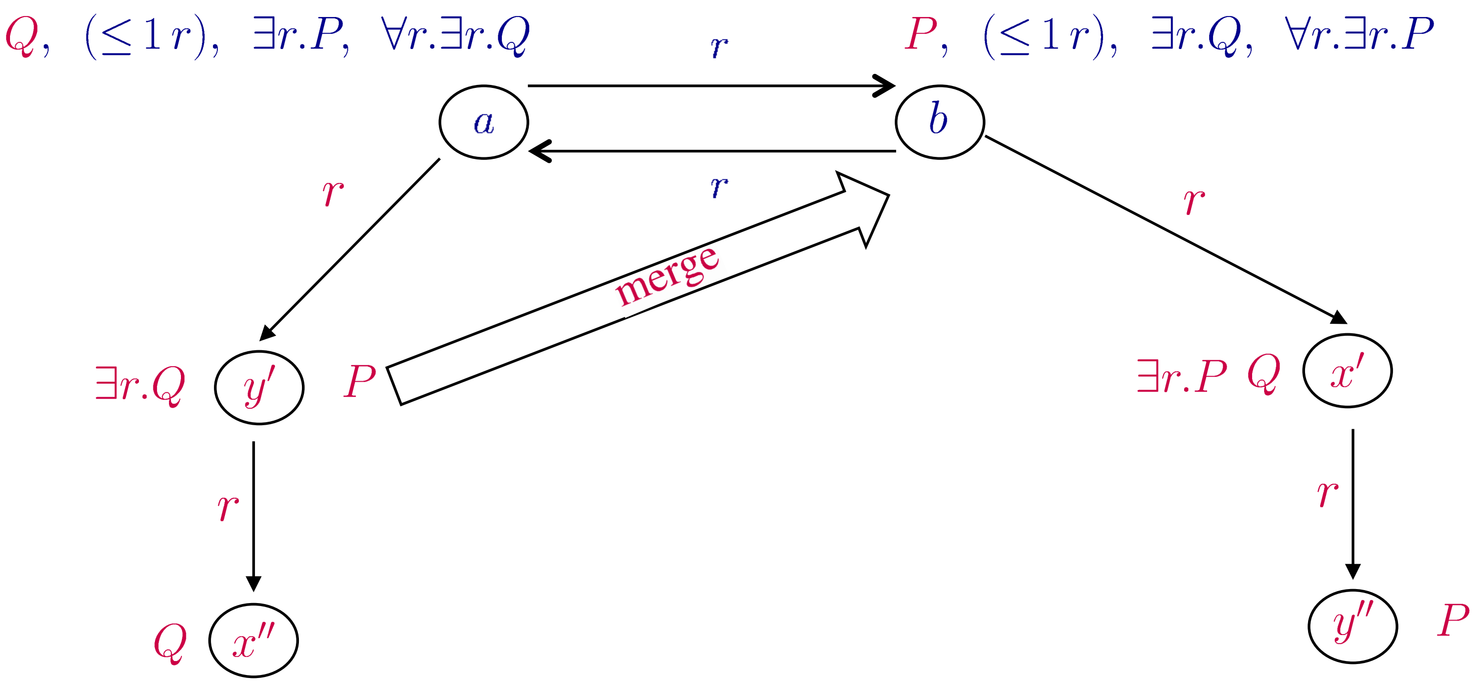
This looks almost like an ABox we have encountered before,
but now $a : Q$ and $b : P$ have been added.

We can now use the same strategy as before to
reproduce the present ABox up to renaming of tree individuals.

Dresden

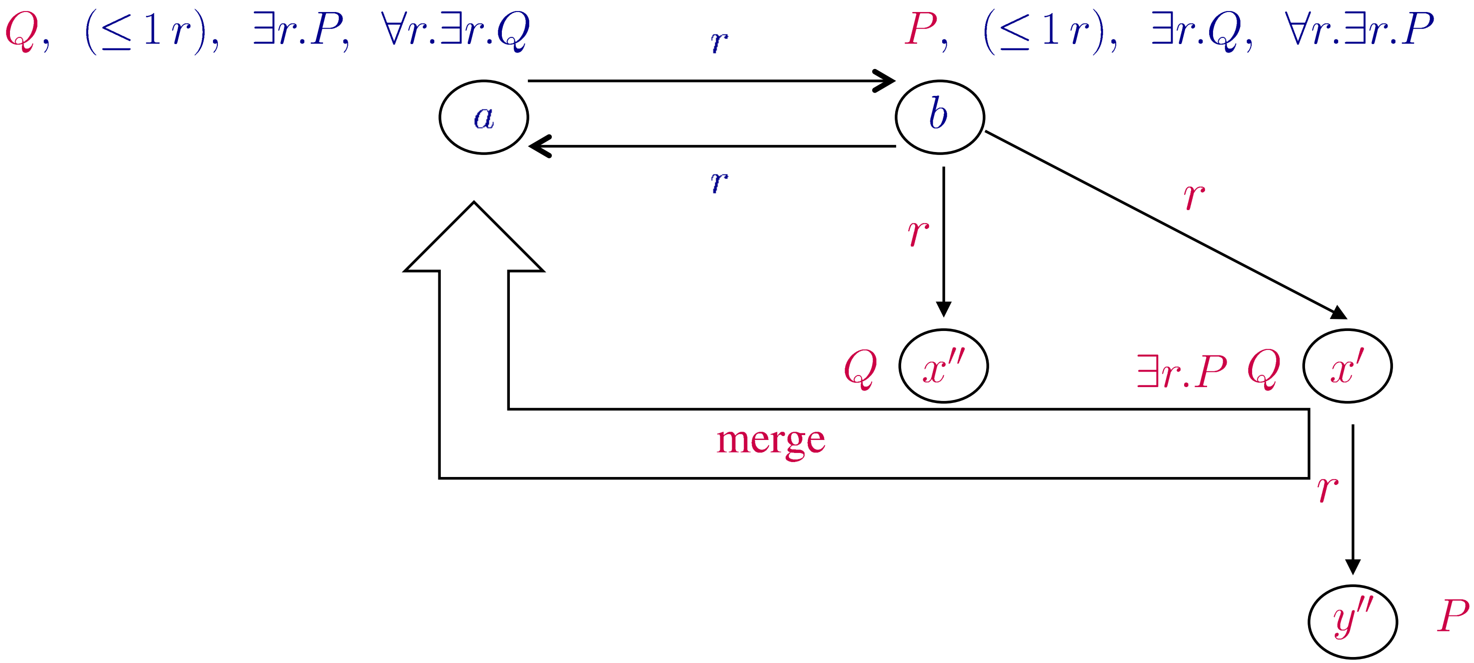
Termination

does not hold even if
blocking as in Definition 4.8 is used



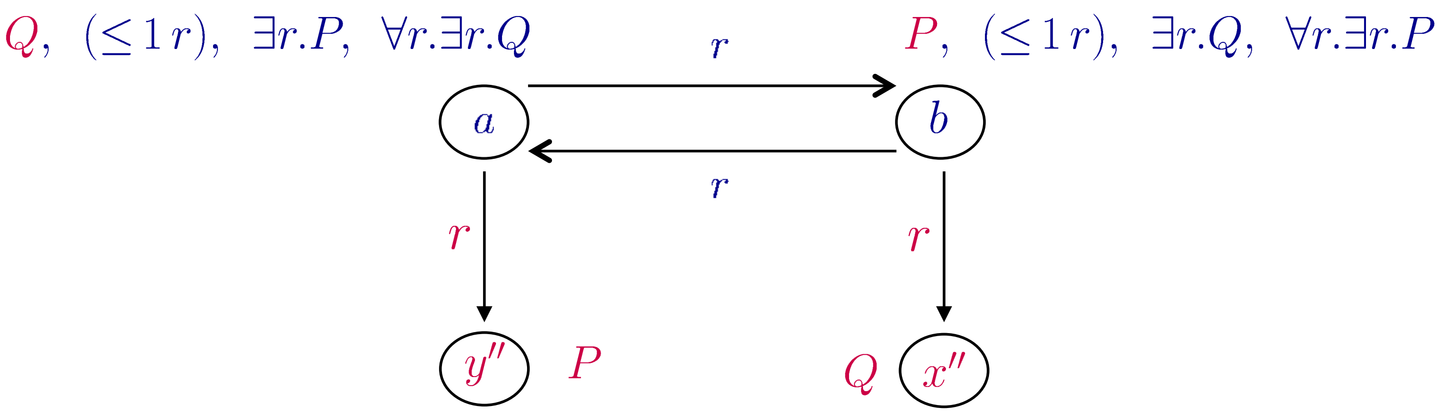
Termination

does not hold even if blocking as in Definition 4.8 is used



Termination

does not hold even if blocking as in Definition 4.8 is used



Up to the names of the tree individuals, this is an ABox we have reached already in a previous stage of the computation.

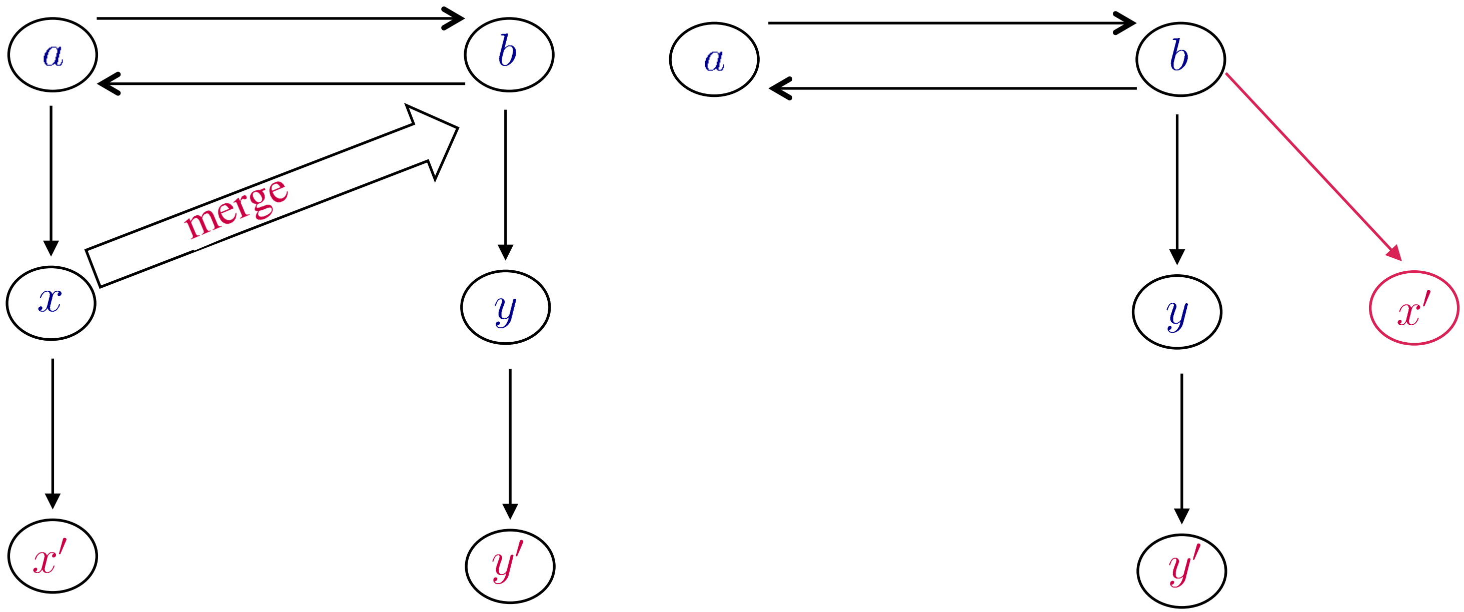
Thus, the algorithm has run into a cycle, which shows that it does not terminate.

Termination

How can it be regained?

The termination problem stems from the fact that an individual

- not only obtains successors by applications of the ∃- and ≥-rule,
- but may also inherit successors from individuals that are merged into it.



Dresden

Termination

How can it be regained?

The termination problem stems from the fact that an individual

- not only obtains successors by applications of the $\exists$- and $\geq$-rule,
- but may also inherit successors from individuals that are merged into it.

To avoid this problem, we remove the descendants of an individual before it is merged into another one:

prune$(\mathcal{A}, b_j)$ removes all the decendants of b_j from the ABox $\mathcal{A}$.

↖ only tree individuals are removed

The $\leq$-rule with pruning

Condition: $\mathcal{A}$ contains $a : (\leq n\, r)$, and there are distinct $b_0, \ldots, b_n$ with $\{(a, b_0) : r, \ldots, (a, b_n) : r)\} \subseteq \mathcal{A}$

Action: $\mathcal{A} \longrightarrow \mathsf{prune}(\mathcal{A}, b_j)[b_j \mapsto b_i] \cup \{b_i = b_j\}$ for $i \neq j$ such that, if b_j is a root individual, then so is b_i.

Tableau algorithm

The tableau algorithm for $\mathcal{ALCN}$ knowledge base consistency uses

- the $\sqcap$-rule, the $\sqcup$-rule, the $\forall$-rule,
- the following modified $\sqsubseteq$-rule:

The modified $\sqsubseteq$-rule

Condition: $a : C \in \mathcal{A}$ or $(b, a) : r \in \mathcal{A}$, $\top \sqsubseteq D \in \mathcal{T}$, $a : D \notin \mathcal{A}$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a : D\}$

The $\geq$-rule introduces individuals without concept assertion!

Tableau algorithm

The tableau algorithm for $\mathcal{ALCN}$ knowledge base consistency uses

- the $\sqcap$-rule, the $\sqcup$-rule, the $\forall$-rule,
- the modified $\sqsubseteq$-rule,
- the modified $\exists$-rule,
- the $\leq$-rule with pruning,
- the following modified $\geq$-rule:

The modified $\geq$-rule

Condition:	$\mathcal{A}$ contains $a:(\geq n\, r)$, but there are no distinct $b_1, \ldots, b_n$ with $\{(a,b_1):r, \ldots, (a,b_n):r\} \subseteq \mathcal{A}$, and a is not blocked
Action:	$\mathcal{A} \longrightarrow \mathcal{A} \cup \{(a,d_1):r, \ldots, (a,d_n):r\} \cup \{d_i \neq d_j \mid 1 \leq i < j \leq n\}$ where $d_1, \ldots, d_n$ are new individual names

Termination

How can it be shown in a formal way?

$\mathcal{A} \to \mathcal{A}'$ $\mathcal{A}'$ is obtained from $\mathcal{A}$ by application of an expansion rule

A partial order $(M, \succ)$ is called well-founded if there is no infinite descending chain

$$m_0 \succ m_1 \succ m_2 \succ m_3 \succ \ldots$$

Termination obviously holds if we can find a mapping μ from ABoxes into a well-founded partial order $(M, \succ)$ such that

$$\mathcal{A} \to \mathcal{A}' \text{ implies } \mu(\mathcal{A}) \succ \mu(\mathcal{A}')$$

Proof.

$\mathcal{A} \to \mathcal{A}_1 \to \mathcal{A}_2 \to \mathcal{A}_3 \to \ldots$

implies

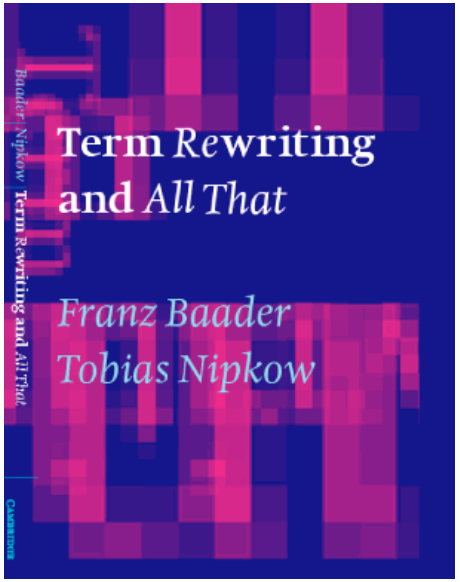
$\mu(\mathcal{A}) \succ \mu(\mathcal{A}_1) \succ \mu(\mathcal{A}_2) \succ \mu(\mathcal{A}_3) \succ \ldots$

How do we get an appropriate well-founded partial order $\succ$?

Dresden

Well-founded orders

More details in



$(\mathbb{N}, >)$ is obviously well-founded.

New well-founded orders can be obtained by using the lexicographic product and the multiset order.

Given two partial orders $(A, >_A)$ and $(B, >_B)$, the lexicographic product $>_{A\times B}$ on $A \times B$ is defined by

$$(x, y) >_{A\times B} (x', y') \;:\Leftrightarrow\; (x >_A x') \vee (x = x' \wedge y >_B y').$$

Theorem (Theorem 2.4.2 in TRAT)

The lexicographic product of two well-founded partial orders is again a well-founded partial order.

Multisets

Multisets are "sets with repeated elements": $\{a, a, b\}$, $\{a, b, b\}$ $\{a, b\}$

Definition (Definition 2.5.1 in TRAT)

1. A multiset M over a set A is a function $M : A \to \mathbb{N}$.
2. M is finite if there are only finitely many x such that $M(x) > 0$.
   $\mathcal{M}(A)$ denotes the set of all finite multisets over A.

- $x \in M \;:\Leftrightarrow\; M(x) > 0$.
- $M \subseteq N \;:\Leftrightarrow\; \forall x \in A.\, M(x) \leq N(x)$.
- $(M \cup N)(x) := M(x) + N(x)$.
- $(M - N)(x) := M(x) \dot{-} N(x)$

$$m \dot{-} n := \begin{cases} m - n & \text{if } m \geq n \\ 0 & \text{otherwise} \end{cases}$$

Dresden

The multiset order

Definition (Definition 2.5.3 in TRAT)

Given a partial order $>$ on a set A, we define the corresponding multiset order $>_{mul}$ on $\mathcal{M}(A)$ as follows:

$M >_{mul} N$ iff there exist $X, Y \in \mathcal{M}(A)$ such that
1. $\emptyset \neq X \subseteq M$ and
2. $N = (M - X) \cup Y$ and
3. $\forall y \in Y.\, \exists x \in X.\, x > y$.

N is obtained from M by removing a non-empty subset X and adding only elements that are smaller than some element in X.

$$\{5,3,1,1\} >_{mul} \{4,3,3,1\} >_{mul} \{4,3,2,2,2,2,2,1\} >_{mul} \{4,3,2,2\}$$

Dresden

The multiset order

Definition (Definition 2.5.3 in TRAT)

Given a partial order $>$ on a set A, we define the corresponding multiset order $>_{mul}$ on $\mathcal{M}(A)$ as follows:

$M >_{mul} N$ iff there exist $X, Y \in \mathcal{M}(A)$ such that

1. $\emptyset \neq X \subseteq M$ and
2. $N = (M - X) \cup Y$ and
3. $\forall y \in Y. \exists x \in X. x > y$.

Theorem (Theorem 2.5.5 in TRAT)

If $>$ is a well-founded partial order on A, then $>_{mul}$ is a well-found partial order on $\mathcal{M}(A)$.

The mapping

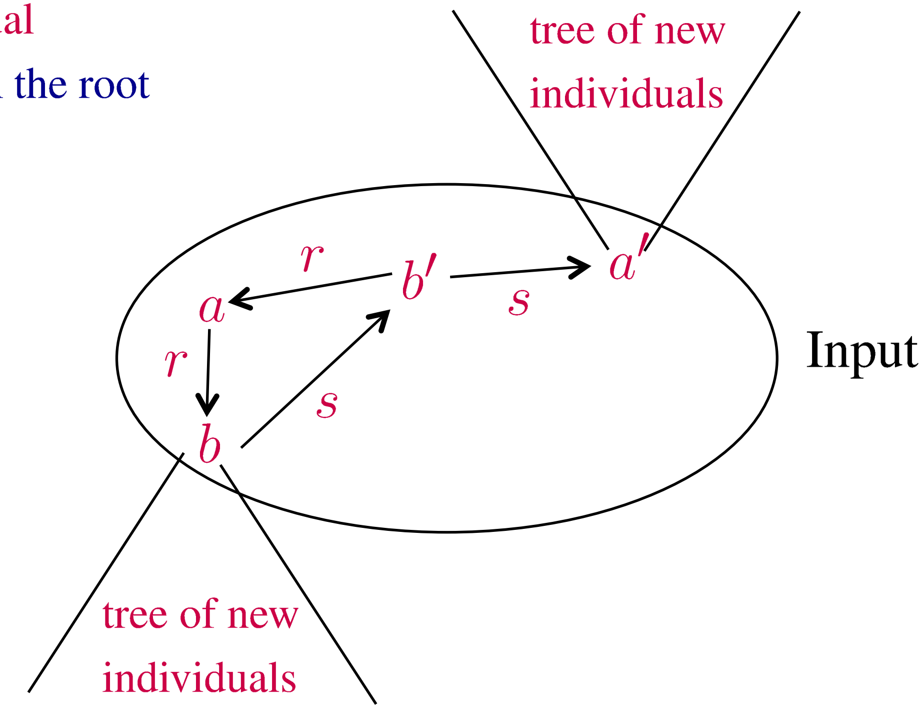
from ABoxes into a well-founded partial order

Consider an ABox $\mathcal{A}$ obtained during a run of the algorithm on input $(\mathcal{T}_0, \mathcal{A}_0)$.

The depth of an indivdual in $\mathcal{A}$ is defined as follows:

$d_{\mathcal{A}}(a) = 0$ if a is a root individual

$d_{\mathcal{A}}(x) = n$ if x is a tree individual with distance n from the root



The mapping

from ABoxes into a well-founded partial order

Consider an ABox $\mathcal{A}$ obtained during a run of the algorithm on input $(\mathcal{T}_0, \mathcal{A}_0)$.

The depth of an indivdual in $\mathcal{A}$ is defined as follows:

$d_{\mathcal{A}}(a) = 0$ if a is a root individual

$d_{\mathcal{A}}(x) = n$ if x is a tree individual with distance n from the root

Lemma 4.14

Let $m = \mathsf{size}(\mathcal{T}_0, \mathcal{A}_0)$ and $\widehat{m} = 2^m$.

- The use of blocking ensures that $d_{\mathcal{A}}(x) \leq \widehat{m}$ for all individuals x.
- $|\mathsf{con}_{\mathcal{A}}(x)| \leq m$ for all individuals x.

Proof.

See proofs of Lemma 4.4 and Lemma 4.10.

Dresden

The mapping

from ABoxes into a well-founded partial order

Consider an ABox $\mathcal{A}$ obtained during a run of the algorithm on input $(\mathcal{T}_0, \mathcal{A}_0)$.

Each individual x occurring in a concept and role assertion in $\mathcal{A}$ is mapped to a triple of natural numbers $\mu_\mathcal{A}(x) := (n_1, n_2, n_3)$:

$$n_1 := \widehat{m} - d_\mathcal{A}(x)$$

$$n_2 := m - |\mathsf{con}_\mathcal{A}(x)|$$

natural numbers due to Lemma 4.14

$$n_3 := \#\{a : \exists r.C \in \mathcal{A} \mid \text{ there is no } b \text{ with } \{(a,b):r, b:C\} \subseteq \mathcal{A}\} + \#\{a : (\geqslant n\, r) \in \mathcal{A} \mid \text{ there are no } b_1, \ldots, b_n \text{ with } \{(a,b_1):r, \ldots, (a,b_n):r\} \cup \{b_i \neq b_j \mid 1 \leq i < j \leq n\} \subseteq \mathcal{A}\}$$

Order on triples: lexicographic product of order $>$ on natural numbers

The ABox $\mathcal{A}$ is mapped to the multiset $\mu(\mathcal{A})$ of these triples.

Order $\succ$ on mulitsets: multiset extension of order on triples

Dresden

Termination

$\mu_{\mathcal{A}}(x) := (n_1, n_2, n_3)$:

$n_1 := \widehat{m} - d_{\mathcal{A}}(x)$

$n_2 := m - |\mathsf{con}_{\mathcal{A}}(x)|$

$n_3 := \#\{a : \exists r.C \in \mathcal{A} \mid$ there is no b with $\{(a,b) : r, b : C\} \subseteq \mathcal{A}\} +$
$\#\{a : (\geqslant n\, r) \in \mathcal{A} \mid$ there are no $b_1, \ldots, b_n$ with
$\{(a,b_1) : r, \ldots, (a,b_n) : r\} \cup \{b_i \neq b_j \mid 1 \leq i < j \leq n\} \subseteq \mathcal{A}\}$

Lemma 4.15

$\mathcal{A} \to \mathcal{A}'$ and $\mathcal{A}'$ clash-free implies $\mu(\mathcal{A}) \succ \mu(\mathcal{A}')$

Proof: blackboard

Dresden

Termination

$\mu_{\mathcal{A}}(x) := (n_1, n_2, n_3)$:

$n_1 := \widehat{m} - d_{\mathcal{A}}(x)$

$n_2 := m - |\mathsf{con}_{\mathcal{A}}(x)|$

$n_3 := \#\{a : \exists r.C \in \mathcal{A} \mid$ there is no b with $\{(a,b) : r, b : C\} \subseteq \mathcal{A}\} +$
$\#\{a : (\geqslant n\, r) \in \mathcal{A} \mid$ there are no $b_1, \ldots, b_n$ with
$\{(a,b_1) : r, \ldots, (a,b_n) : r\} \cup \{b_i \neq b_j \mid 1 \leq i < j \leq n\} \subseteq \mathcal{A}\}$

Lemma 4.15

$\mathcal{A} \to \mathcal{A}'$ and $\mathcal{A}'$ clash-free implies $\mu(\mathcal{A}) \succ \mu(\mathcal{A}')$

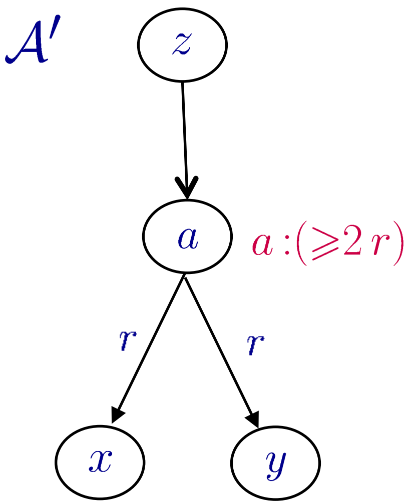
Lemma 4.16 (Termination)

For each normalized $\mathcal{ALCN}$ KB $\mathcal{K}$, consistent$(\mathcal{K})$ terminates.

Soundness

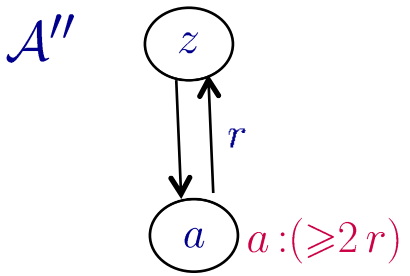
Soundness can be shown similarly to the proof of Lemma 4.11.

However, the construction of $\mathcal{A}''$ needs to be modified in order to obtain a complete and clash-free ABox:



$x \neq y \in \mathcal{A}'$

x and y blocked by z



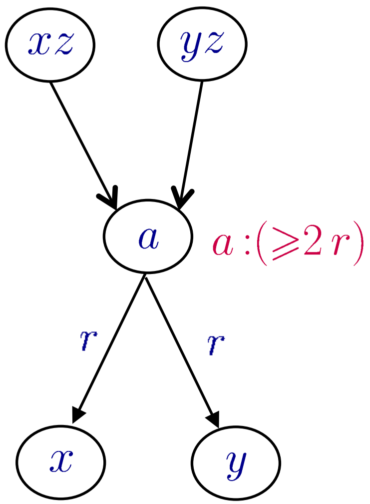
$\mathcal{A}''$ is not complete!

Idea: create copies of blocking individual for each individual it blocks.

Soundness

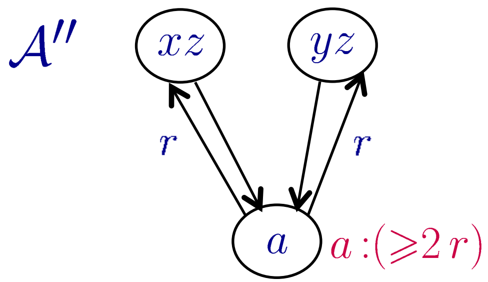
Soundness can be shown similarly to the proof of Lemma 4.11.

However, the construction of $\mathcal{A}''$ needs to be modified in order to obtain a complete and clash-free ABox:



$x \neq y \in \mathcal{A}'$

x and y blocked by z



$\mathcal{A}''$ is complete!

Note: in general we may need the equality assertions in $\mathcal{A}'$ to turn the model of $\mathcal{A}''$ into a model of $\mathcal{A}$.

Dresden

Completeness

It only remains to show that the $\leq$-rule and the $\geq$-rule preserve KB consistency.

Exercise!