ulm university universität
uulm

Pavel Klinov[1] Bijan Parsia[2]

[1]University of Ulm [2]University of Manchester |

August 6, 2013

Practical Tableau-based Reasoning in Description Logics

Reminder: Reasoning Problems in DLs

$\mathcal{ALC}$ Tableau with TBoxes

Two Essential Optimizations

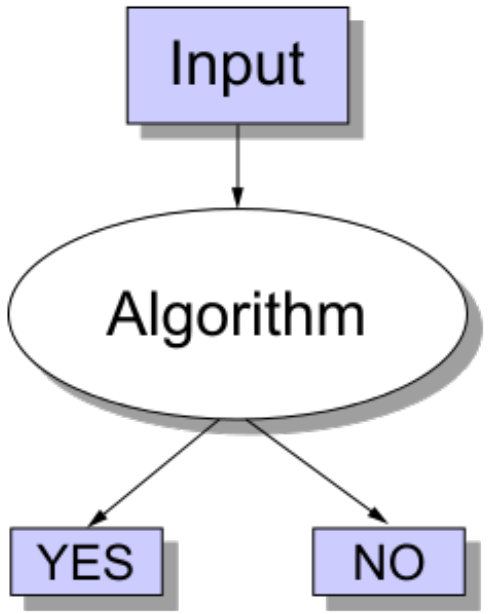
Reminder: Reasoning Problems in DLs

$\mathcal{ALC}$ Tableau with TBoxes

Two Essential Optimizations

Decision Problem and Decision Procedure

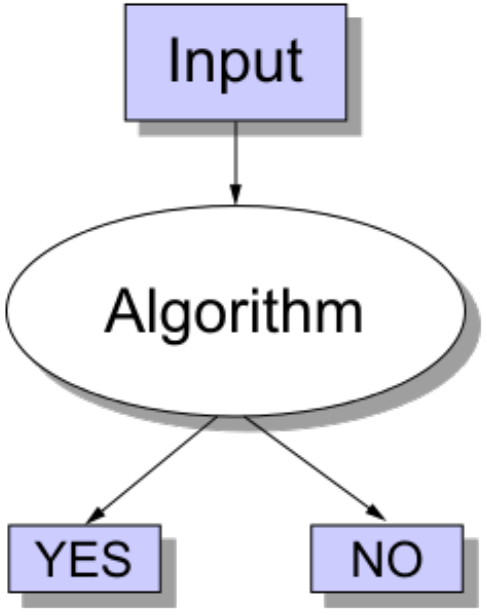
Decision problem: a well-formulated yes/no question



Decision Problem and Decision Procedure

Decision problem: a well-formulated yes/no question

Given I: all inputs and $J \subseteq I$: inputs for which the answer must be yes



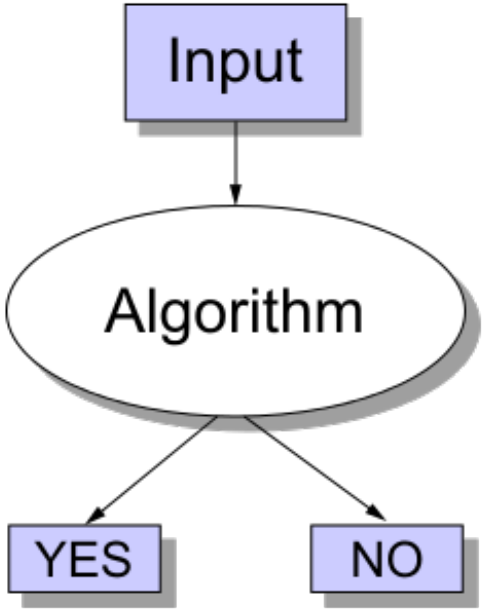
Decision Problem and Decision Procedure

Decision problem: a well-formulated yes/no question

Given I: all inputs and $J \subseteq I$: inputs for which the answer must be yes

Decision procedure is an algorithm s.t.:

- Sound: if it answers yes, then the input is in J
- Complete: if the input is in J, then it answers yes
- Terminating: it returns an answer after a finite number of steps



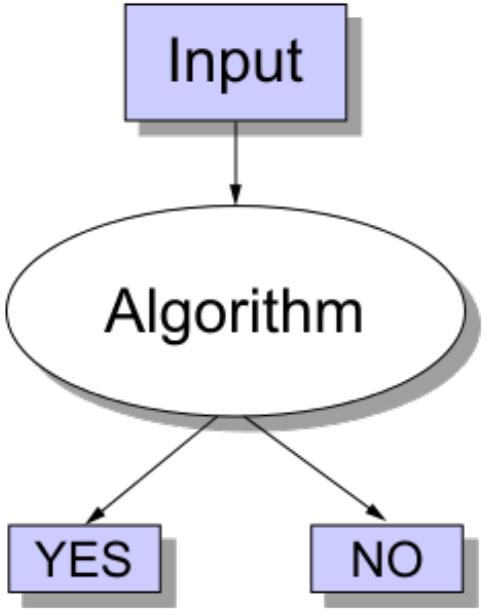
Decision Problem and Decision Procedure

Decision problem: a well-formulated yes/no question

Given I: all inputs and $J \subseteq I$: inputs for which the answer must be yes

Decision procedure is an algorithm s.t.:

- Sound: if it answers yes, then the input is in J
- Complete: if the input is in J, then it answers yes
- Terminating: it returns an answer after a finite number of steps



We only talk about decision problems in this course

Reducibility of DL Reasoning Problems

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

- is $\mathcal{O}$ consistent? $\mathcal{O} \models \top \sqsubseteq \bot$?

Reducibility of DL Reasoning Problems

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

- is $\mathcal{O}$ consistent? $\mathcal{O} \models \top \sqsubseteq \bot$?
- is $\mathcal{O}$ coherent? $\mathcal{O} \models \mathsf{A} \sqsubseteq \bot$?
  (for some concept name A)

Reducibility of DL Reasoning Problems

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

- is $\mathcal{O}$ consistent? $\mathcal{O} \models \top \sqsubseteq \bot$?
- is $\mathcal{O}$ coherent? $\mathcal{O} \models \mathsf{A} \sqsubseteq \bot$?
  (for some concept name A)
- classification $\mathcal{O} \models \mathsf{A} \sqsubseteq \mathsf{B}$?
  (for all concept names A, B)

Reducibility of DL Reasoning Problems

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

- is $\mathcal{O}$ consistent? $\mathcal{O} \models \top \sqsubseteq \bot$?
- is $\mathcal{O}$ coherent? $\mathcal{O} \models \mathsf{A} \sqsubseteq \bot$?
  (for some concept name A)
- classification $\mathcal{O} \models \mathsf{A} \sqsubseteq \mathsf{B}$?
  (for all concept names A, B)
- realization $\mathcal{O} \models b : \mathsf{B}$?
  (for all concept names A, individual names b)

Question: do we need 4 different algorithms for these?

Consistency Suffices

Theorem: Let $\mathcal{O}$ be an ontology and a a fresh individual name:

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

Answer: a decision procedure to solve consistency decides all standard DL reasoning problems

Consistency for $\mathcal{ALC}$

The way we will do it:

- Concept satisfiability for $\mathcal{ALC}$ (no ABoxes or TBoxes)
- ABox consistency (very slight generalization)
- Ontology consistency (TBoxes enter the picture)

Consistency for $\mathcal{ALC}$

The way we will do it:

- Concept satisfiability for $\mathcal{ALC}$ (no ABoxes or TBoxes)
- ABox consistency (very slight generalization)
- Ontology consistency (TBoxes enter the picture)

Optimizations along the way, two essential at the end

Consistency for $\mathcal{ALC}$

The way we will do it:

- Concept satisfiability for $\mathcal{ALC}$ (no ABoxes or TBoxes)
- ABox consistency (very slight generalization)
- Ontology consistency (TBoxes enter the picture)

Optimizations along the way, two essential at the end

We will work through examples

Tableau for Concept Satisfiability in $\mathcal{ALC}$

Given an $\mathcal{ALC}$ concept C, devise an algoritm that outputs:

- yes, if there is a model of $a : C$
- no, no such model exists

Tableau for Concept Satisfiability in $\mathcal{ALC}$

Given an $\mathcal{ALC}$ concept C, devise an algoritm that outputs:

- yes, if there is a model of $a : C$
- no, no such model exists

The idea is simple: it tries to construct a model $C^{\mathcal{I}}$:

- if successful, C is satisfiable
- otherwise, no model provably exists

Tableau for Concept Satisfiability in $\mathcal{ALC}$

Given an $\mathcal{ALC}$ concept C, devise an algoritm that outputs:

- yes, if there is a model of $a : C$
- no, no such model exists

The idea is simple: it tries to construct a model $C^{\mathcal{I}}$:

- if successful, C is satisfiable
- otherwise, no model provably exists

This approach is different from syntactic proof calculi, e.g., natural deduction (Lecture 4)

$\mathcal{ALC}$ Concept Interpretations

A reminder from yesterday: what $\mathcal{ALC}$ interpretations look like

$C^{\mathcal{I}}$ can be represented as a labeled graph $\mathcal{G}_{C^{\mathcal{I}}} = \langle V, E \rangle$, where:

- V set of domain elements where $v_0 \in C^{\mathcal{I}}$
- $C \in L(v)$ if $v \in C^{\mathcal{I}}$
- (x, y) is a R-labeled edge if $(x, y) \in \mathsf{R}^{\mathcal{I}}$

Negation Normal Form

All concepts are assumed to be in Negation Normal Form

- negation can only appear before concept names
- need this to simplify explanation

Negation Normal Form

All concepts are assumed to be in Negation Normal Form

- negation can only appear before concept names
- need this to simplify explanation
- transform all concepts in $\mathcal{O}$ into $\mathrm{NNF}(C)$ using

$$\neg(C \sqcap D) \equiv \neg C \sqcup \neg D \qquad \neg(C \sqcup D) \equiv \neg C \sqcap \neg D$$
$$\neg(\exists R.C) \equiv (\forall R.\neg C) \qquad \neg(\forall R.C) \equiv (\exists R.\neg C)$$

Negation Normal Form

All concepts are assumed to be in Negation Normal Form

- negation can only appear before concept names
- need this to simplify explanation
- transform all concepts in $\mathcal{O}$ into $\mathrm{NNF}(C)$ using

$$\neg(C \sqcap D) \equiv \neg C \sqcup \neg D \qquad \neg(C \sqcup D) \equiv \neg C \sqcap \neg D$$
$$\neg(\exists R.C) \equiv (\forall R.\neg C) \qquad \neg(\forall R.C) \equiv (\exists R.\neg C)$$

Lemma: Let C be an $\mathcal{ALC}$ concept. Then $C \equiv \mathrm{NNF}(C)$.

Negation Normal Form

All concepts are assumed to be in Negation Normal Form

- negation can only appear before concept names
- need this to simplify explanation
- transform all concepts in $\mathcal{O}$ into $\mathrm{NNF}(C)$ using

$$\begin{array}{rclrcl} \neg(C \sqcap D) & \equiv & \neg C \sqcup \neg D & \neg(C \sqcup D) & \equiv & \neg C \sqcap \neg D \\ \neg(\exists R.C) & \equiv & (\forall R.\neg C) & \neg(\forall R.C) & \equiv & (\exists R.\neg C) \end{array}$$

Lemma: Let C be an $\mathcal{ALC}$ concept. Then $C \equiv \mathrm{NNF}(C)$.

All concepts are assumed to be in NNF

$\dot{\neg} C$ denotes $\mathrm{NNF}(\neg C)$

Tableau Algorithm for Concept Satisfiability in $\mathcal{ALC}$

Tableau algorithm:

- works on sets of labeled graphs $\mathcal{S} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$

Tableau Algorithm for Concept Satisfiability in $\mathcal{ALC}$

Tableau algorithm:

- works on sets of labeled graphs $\mathcal{S} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$
- For an $\mathcal{ALC}$ concept C it starts with a singleton set $\mathcal{S} = \{\mathcal{G}_C\}$, where $\mathcal{G}_C = \langle\{a\}, \emptyset\rangle$, $L(a) = \{C\}$

Tableau Algorithm for Concept Satisfiability in $\mathcal{ALC}$

Tableau algorithm:

- works on sets of labeled graphs $\mathcal{S} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$
- For an $\mathcal{ALC}$ concept C it starts with a singleton set $\mathcal{S} = \{\mathcal{G}_C\}$, where $\mathcal{G}_C = \langle\{a\}, \emptyset\rangle$, $L(a) = \{C\}$
- applies tableau rules that infer constraints on models of $C^{\mathcal{I}}$

Tableau Algorithm for Concept Satisfiability in $\mathcal{ALC}$

Tableau algorithm:

- works on sets of labeled graphs $\mathcal{S} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$
- For an $\mathcal{ALC}$ concept C it starts with a singleton set $\mathcal{S} = \{\mathcal{G}_C\}$, where $\mathcal{G}_C = \langle\{a\}, \emptyset\rangle$, $L(a) = \{C\}$
- applies tableau rules that infer constraints on models of $C^{\mathcal{I}}$
- a rule is applied to some $\mathcal{G} \in \mathcal{S}$; its application extends $\mathcal{G}$ or replaces it with two new graphs

Tableau Algorithm for Concept Satisfiability in $\mathcal{ALC}$

Tableau algorithm:

- works on sets of labeled graphs $\mathcal{S} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$
- For an $\mathcal{ALC}$ concept C it starts with a singleton set $\mathcal{S} = \{\mathcal{G}_C\}$, where $\mathcal{G}_C = \langle\{a\}, \emptyset\rangle$, $L(a) = \{C\}$
- applies tableau rules that infer constraints on models of $C^{\mathcal{I}}$
- a rule is applied to some $\mathcal{G} \in \mathcal{S}$; its application extends $\mathcal{G}$ or replaces it with two new graphs
- answers yes, if a rule application leads to a graph $\mathcal{G}$ that is
  - complete, i.e., to which no more rules apply and
  - clash-free, i.e., $\{\mathsf{A}, \neg\mathsf{A}\} \not\subseteq L(a)$, $\bot \notin L(a)$ for all a, A

Tableau Algorithm for Concept Satisfiability in $\mathcal{ALC}$

Tableau algorithm:

- works on sets of labeled graphs $\mathcal{S} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$
- For an $\mathcal{ALC}$ concept C it starts with a singleton set $\mathcal{S} = \{\mathcal{G}_C\}$, where $\mathcal{G}_C = \langle\{a\}, \emptyset\rangle$, $L(a) = \{C\}$
- applies tableau rules that infer constraints on models of $C^{\mathcal{I}}$
- a rule is applied to some $\mathcal{G} \in \mathcal{S}$; its application extends $\mathcal{G}$ or replaces it with two new graphs
- answers yes, if a rule application leads to a graph $\mathcal{G}$ that is
  - complete, i.e., to which no more rules apply and
  - clash-free, i.e., $\{\mathsf{A}, \neg\mathsf{A}\} \not\subseteq L(a)$, $\bot \notin L(a)$ for all a, A
- answers no, if all graphs contain clashes (complete or not)

Tableau Algorithm for Concept Satisfiability in $\mathcal{ALC}$

Tableau algorithm:

- works on sets of labeled graphs $\mathcal{S} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k\}$
- For an $\mathcal{ALC}$ concept C it starts with a singleton set $\mathcal{S} = \{\mathcal{G}_C\}$, where $\mathcal{G}_C = \langle\{a\}, \emptyset\rangle$, $L(a) = \{C\}$
- applies tableau rules that infer constraints on models of $C^{\mathcal{I}}$
- a rule is applied to some $\mathcal{G} \in \mathcal{S}$; its application extends $\mathcal{G}$ or replaces it with two new graphs
- answers yes, if a rule application leads to a graph $\mathcal{G}$ that is
  - complete, i.e., to which no more rules apply and
  - clash-free, i.e., $\{\mathsf{A}, \neg\mathsf{A}\} \not\subseteq L(a)$, $\bot \notin L(a)$ for all a, A
- answers no, if all graphs contain clashes (complete or not)

Graphs computed by the procedure are called completion graphs

Tableau Expansion Rules for $\mathcal{ALC}$

$\sqcap$-rule: if $C_1 \sqcap C_2 \in L(a)$ for some a and $\{C_1, C_2\} \not\subseteq L(a)$

then add $\{C_1, C_2\}$ to $L(a)$

Tableau Expansion Rules for $\mathcal{ALC}$

$\sqcap$-rule: if $C_1 \sqcap C_2 \in L(a)$ for some a and $\{C_1, C_2\} \not\subseteq L(a)$

then add $\{C_1, C_2\}$ to $L(a)$

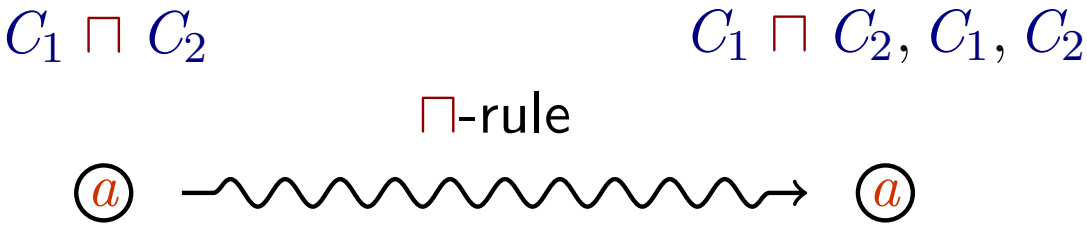


Tableau Expansion Rules for $\mathcal{ALC}$

$\sqcup$-rule: if $C_1 \sqcup C_2 \in L(a)$ for some a and $\{C_1, C_2\} \cap L(a) = \emptyset$

then replace $\mathcal{G}$ with $\mathcal{G}_1$ and $\mathcal{G}_2$ s.t.

$C_1 \in L(a)$ in $\mathcal{G}_1$ and $C_2 \in L_2(a)$ in $\mathcal{G}_2$

Tableau Expansion Rules for $\mathcal{ALC}$

$\sqcup$-rule: if $C_1 \sqcup C_2 \in L(a)$ for some a and $\{C_1, C_2\} \cap L(a) = \emptyset$

then replace $\mathcal{G}$ with $\mathcal{G}_1$ and $\mathcal{G}_2$ s.t.

$C_1 \in L(a)$ in $\mathcal{G}_1$ and $C_2 \in L_2(a)$ in $\mathcal{G}_2$

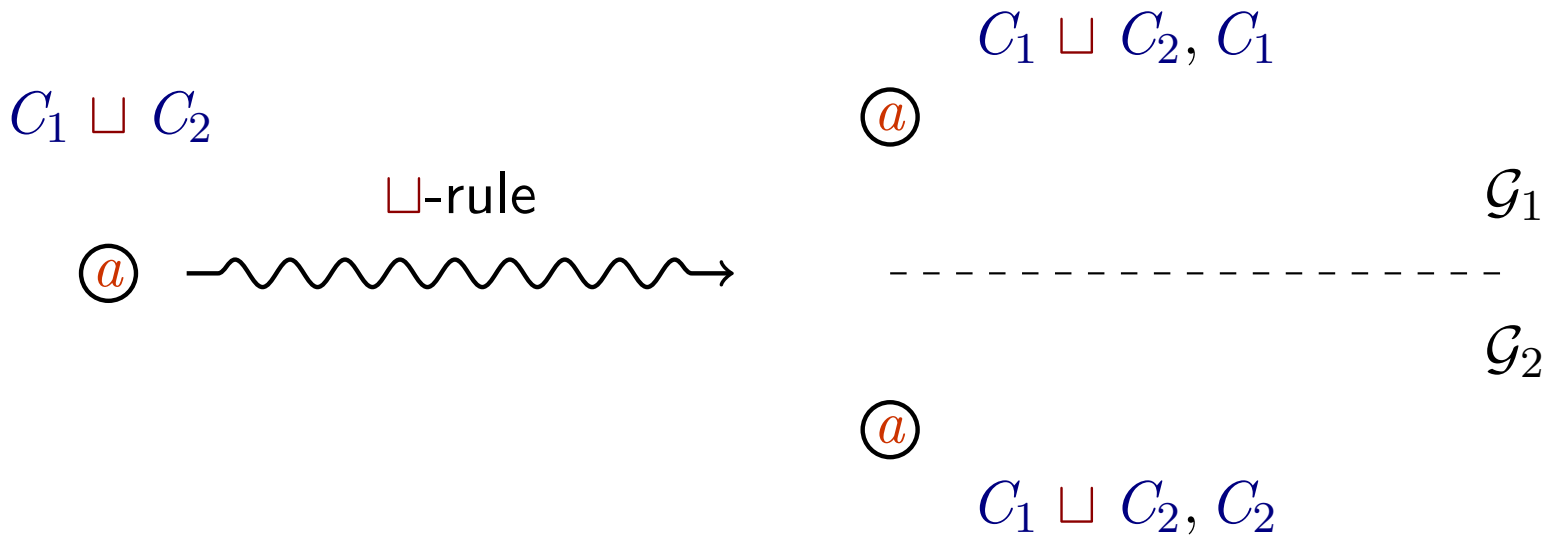
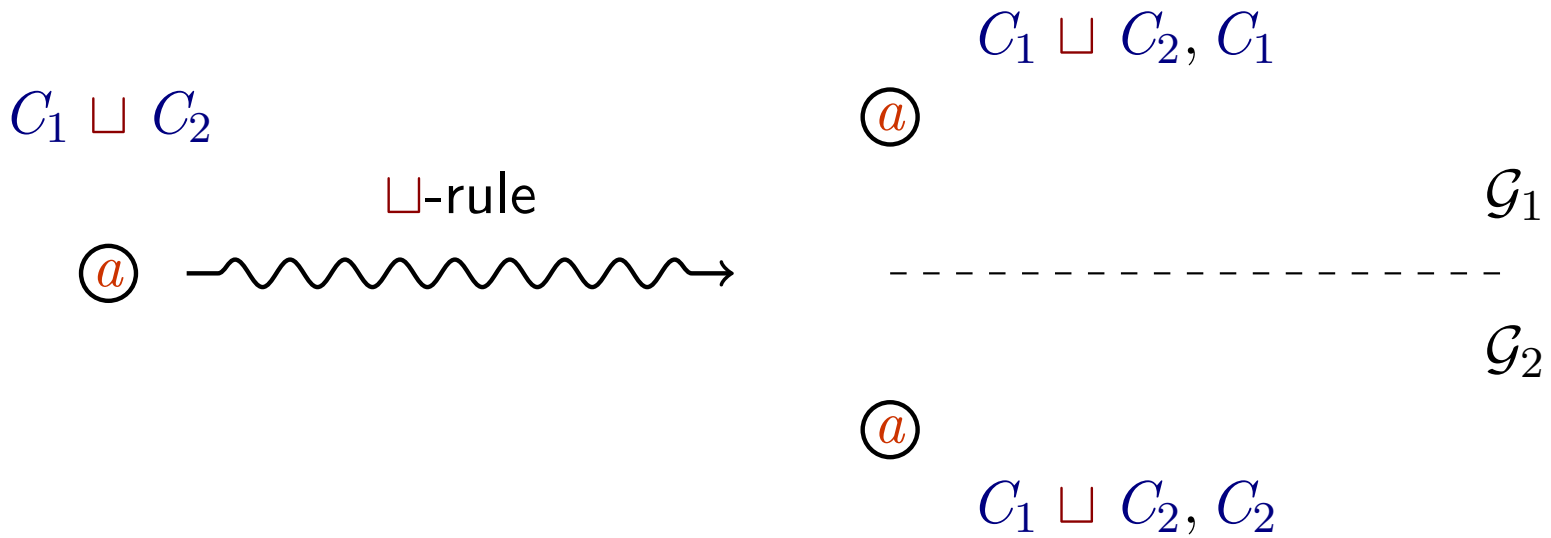


Tableau Expansion Rules for $\mathcal{ALC}$

$\sqcup$-rule: if $C_1 \sqcup C_2 \in L(a)$ for some a and $\{C_1, C_2\} \cap L(a) = \emptyset$

then replace $\mathcal{G}$ with $\mathcal{G}_1$ and $\mathcal{G}_2$ s.t.

$C_1 \in L(a)$ in $\mathcal{G}_1$ and $C_2 \in L_2(a)$ in $\mathcal{G}_2$



This is a don't know (disjunctive) non-determinism

Tableau Expansion Rules for $\mathcal{ALC}$

$\exists$-rule: if $\exists \mathsf{R}.C \in L(a)$ for some a and there is no b s.t. $C \in L(b)$ and an R-edge from a to b is in $\mathcal{G}$

then create a new node b,
add an R-edge from a to b and add C to $L(b)$

Tableau Expansion Rules for $\mathcal{ALC}$

$\exists$-rule: if $\exists \mathsf{R}.C \in L(a)$ for some a and there is no b s.t. $C \in L(b)$ and an R-edge from a to b is in $\mathcal{G}$

then create a new node b,
add an R-edge from a to b and add C to $L(b)$

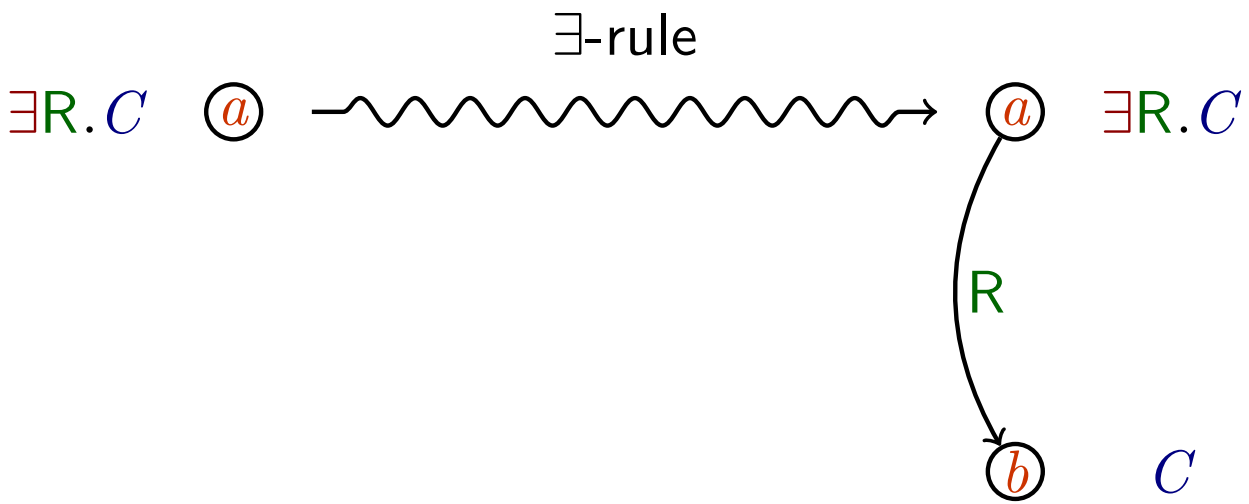
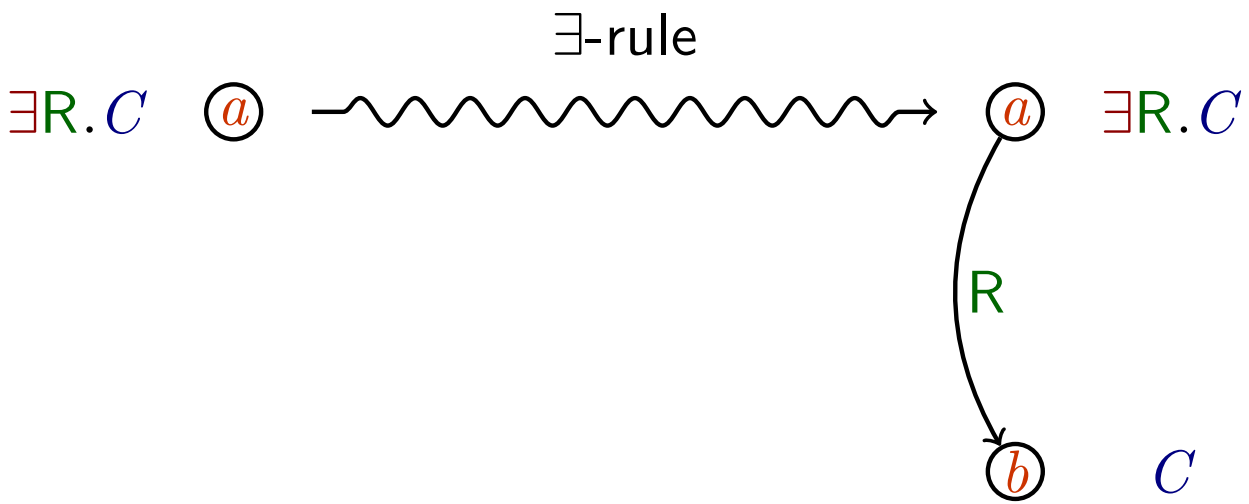


Tableau Expansion Rules for $\mathcal{ALC}$

$\exists$-rule: if $\exists \mathsf{R}.C \in L(a)$ for some a and there is no b s.t. $C \in L(b)$ and an R-edge from a to b is in $\mathcal{G}$

then create a new node b,
add an R-edge from a to b and add C to $L(b)$



This is the only rule that generates new nodes

Tableau Expansion Rules for $\mathcal{ALC}$

$\forall$-rule: if $\forall R.C \in L(a)$ for some a
and there is an R-edge from a to b and $C \notin L(b)$

then add C to $L(b)$

Tableau Expansion Rules for $\mathcal{ALC}$

$\forall$-rule: if $\forall \mathsf{R}.C \in L(a)$ for some a
and there is an R-edge from a to b and $C \notin L(b)$

then add C to $L(b)$

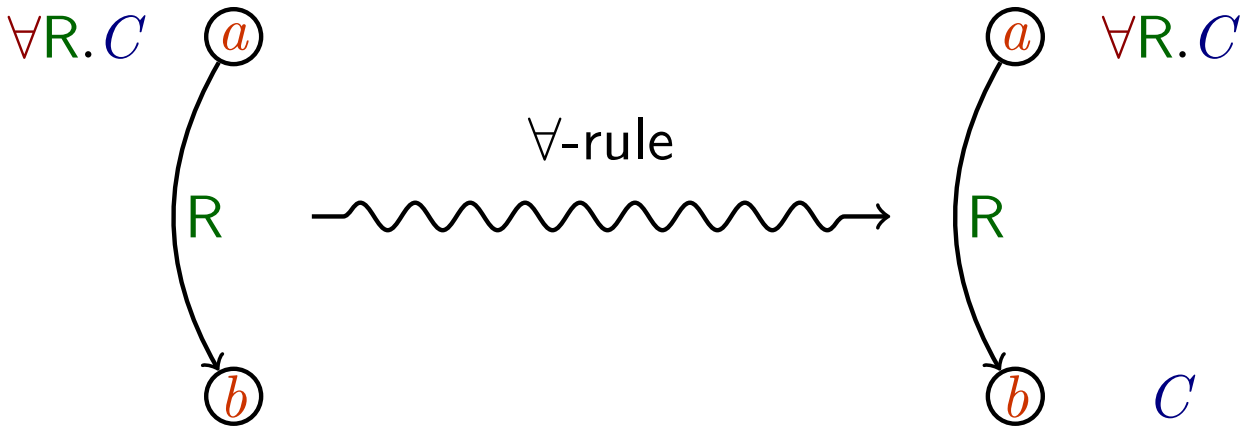


Tableau Algorithm is a Decision Procedure

Sound

- Every completion graph can be transformed into a model

Tableau Algorithm is a Decision Procedure

Sound

- Every completion graph can be transformed into a model

Complete

- If there is a model $\mathcal{I}$, then there is a run of the algorithm which constructs a clash-free $\mathcal{G}$
- If the algorithm says no, then there is no model

Tableau Algorithm is a Decision Procedure

Sound

- Every completion graph can be transformed into a model

Complete

- If there is a model $\mathcal{I}$, then there is a run of the algorithm which constructs a clash-free $\mathcal{G}$
- If the algorithm says no, then there is no model

Terminating

- Construction is monotonic, things are only added to graphs
- Number of nodes and size of labels are bounded

Tableau Algorithm: Example 1

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.\neg A}_{X_3}$$

Tableau Algorithm: Example 1

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.\neg A}_{X_3}$$

Tableau Algorithm: Example 1

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.\neg A}_{X_3}$$

$$X_1 \sqcap X_2 \sqcap X_3$$

$$\textcircled{a_0}$$

Tableau Algorithm: Example 1

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.\neg A}_{X_3}$$

$$X_1 \sqcap X_2 \sqcap X_3, X_1, X_2, X_3$$

$$(a_0)$$

Tableau Algorithm: Example 1

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.\neg A}_{X_3}$$

$$X_1 \sqcap X_2 \sqcap X_3, X_1, X_2, X_3$$

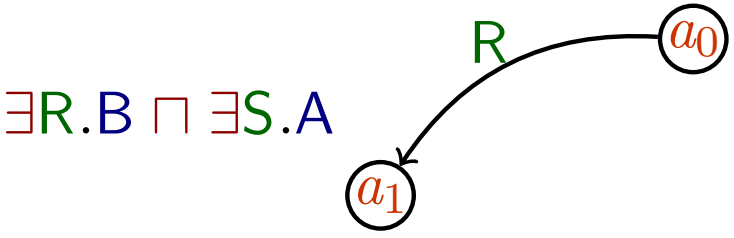


Tableau Algorithm: Example 1

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.\neg A}_{X_3}$$

$$X_1 \sqcap X_2 \sqcap X_3, X_1, X_2, X_3$$

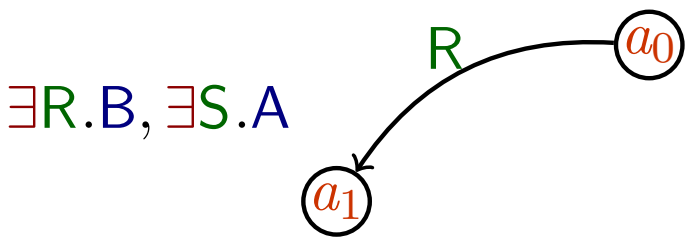


Tableau Algorithm: Example 1

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.\neg A}_{X_3}$$

$$X_1 \sqcap X_2 \sqcap X_3, X_1, X_2, X_3$$

$\exists R.B, \exists S.A,$
$\forall S.\neg A$

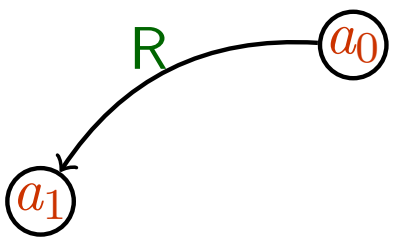


Tableau Algorithm: Example 1

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.\neg A}_{X_3}$$

$$X_1 \sqcap X_2 \sqcap X_3, X_1, X_2, X_3$$

$\exists R.B, \exists S.A,$
$\forall S.\neg A$

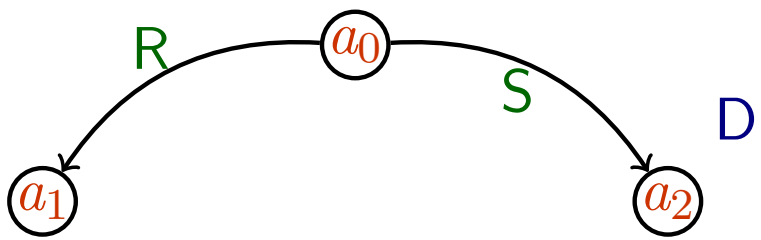


Tableau Algorithm: Example 1

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.\neg A}_{X_3}$$

$$X_1 \sqcap X_2 \sqcap X_3, X_1, X_2, X_3$$

$\exists R.B, \exists S.A,$
$\forall S.\neg A$

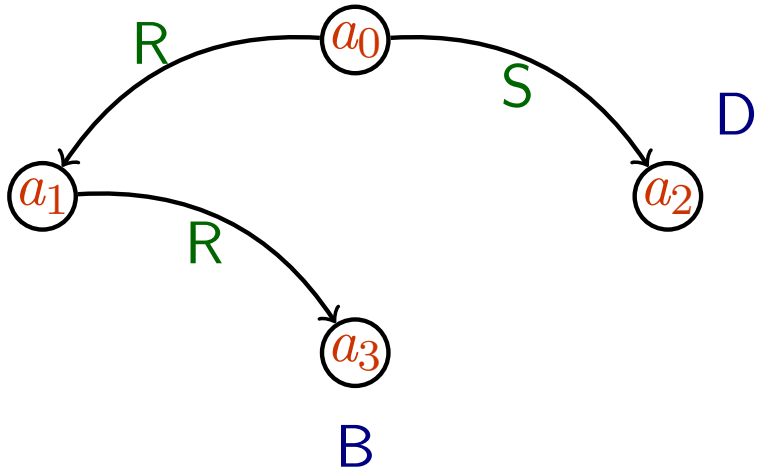


Tableau Algorithm: Example 1

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.\neg A}_{X_3}$$

$$X_1 \sqcap X_2 \sqcap X_3, X_1, X_2, X_3$$

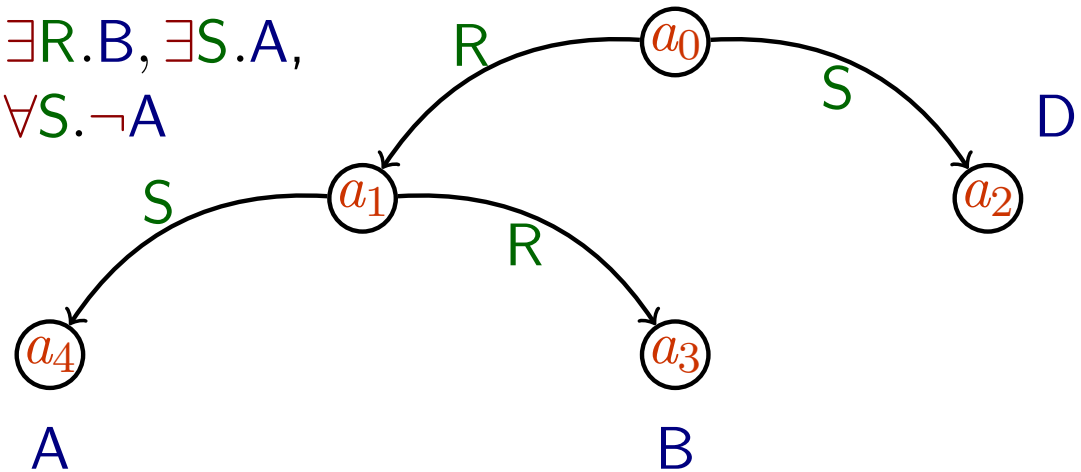


Tableau Algorithm: Example 1

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.\neg A}_{X_3}$$

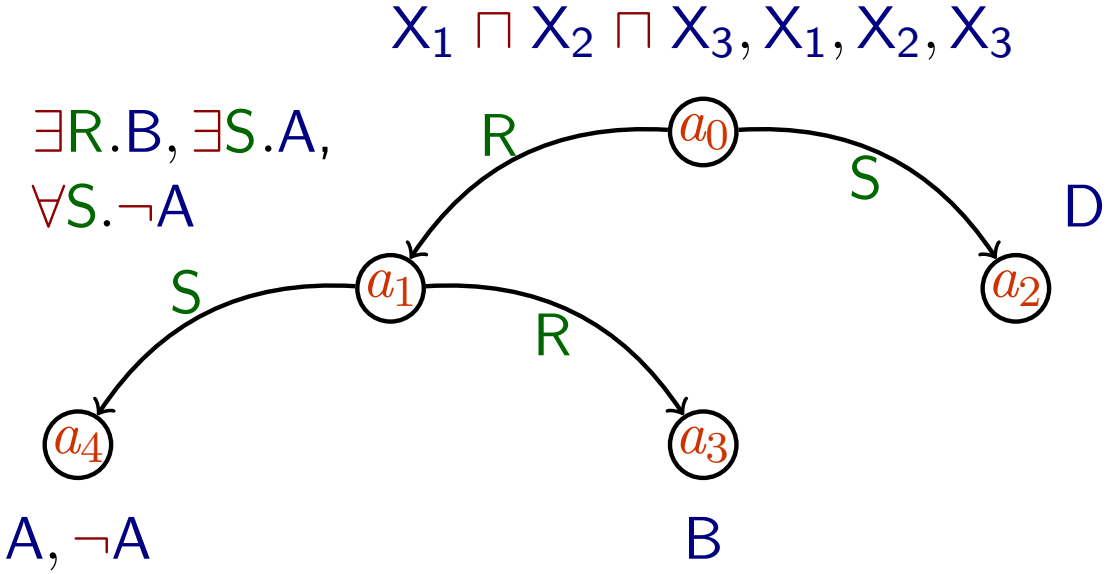


Tableau Algorithm: Example 1

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.\neg A}_{X_3}$$

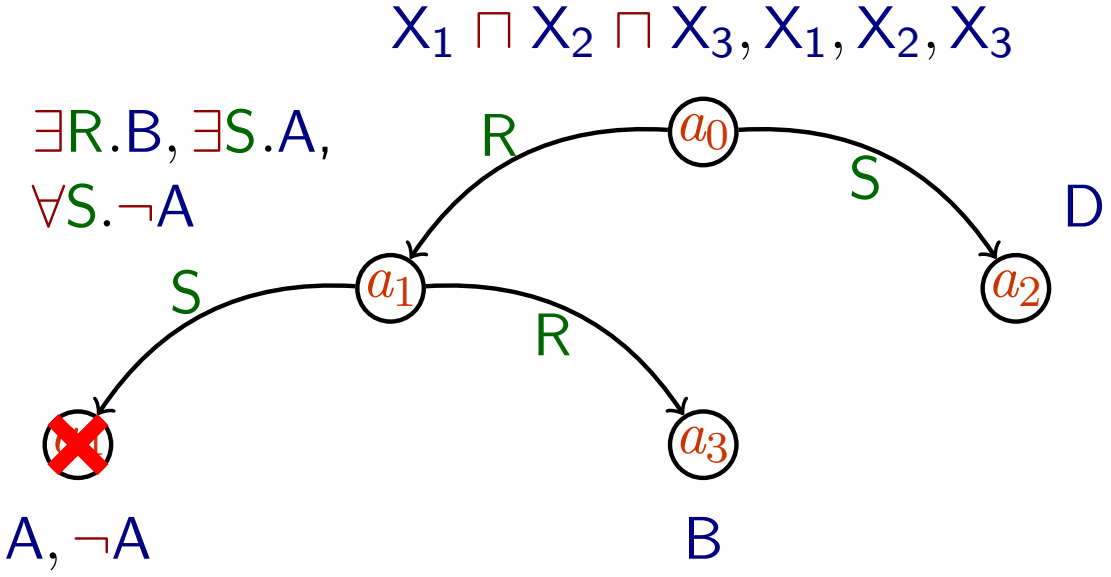
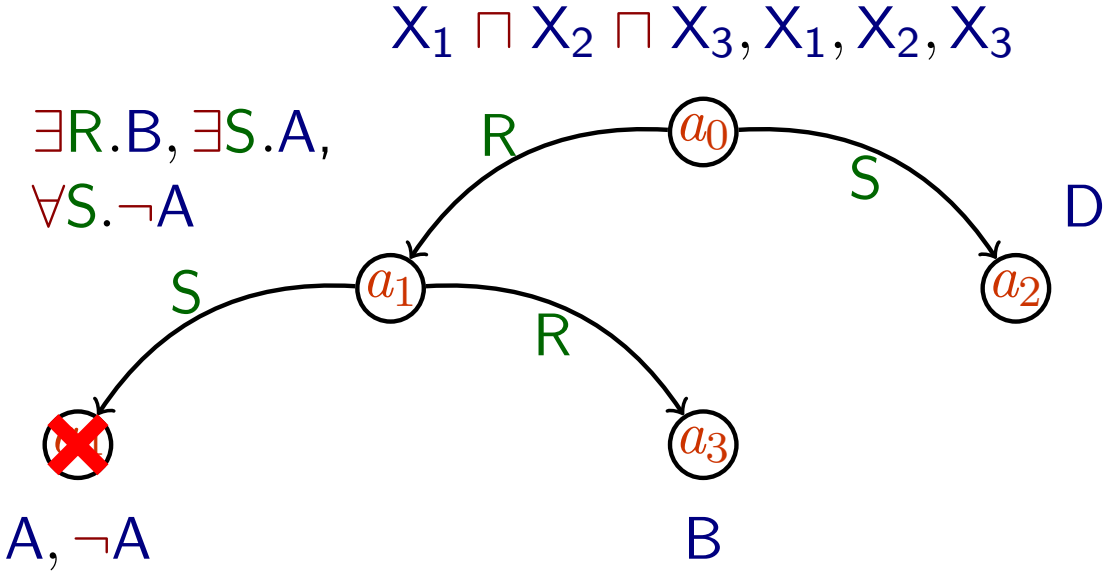


Tableau Algorithm: Example 1

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.\neg A}_{X_3}$$



$\mathcal{G}_0$ contains a clash; no other graphs in $\mathcal{S} \Rightarrow X$ is unsatisfiable

Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$

Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$

$\mathcal{G}_0$

Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$

$$X_1 \sqcap X_2 \sqcap X_3$$

$$\textcircled{a_0} \qquad \mathcal{G}_0$$

Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$

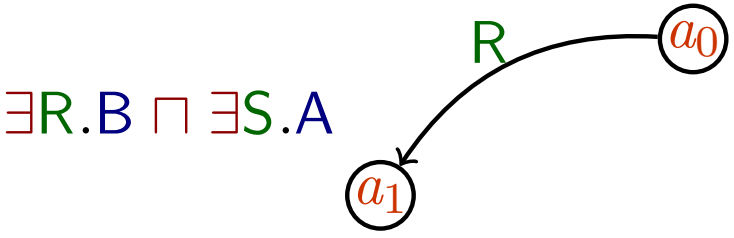
$$X_1 \sqcap X_2 \sqcap X_3, X_1, X_2, X_3$$

$$\textcircled{a_0} \qquad \mathcal{G}_0$$

Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$

$$X_1 \sqcap X_2 \sqcap X_3, X_1, X_2, X_3$$

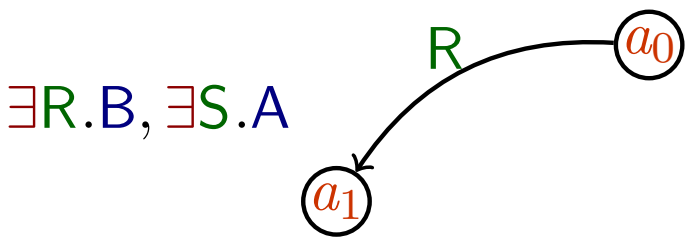


$\mathcal{G}_0$

Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$

$$X_1 \sqcap X_2 \sqcap X_3, X_1, X_2, X_3$$



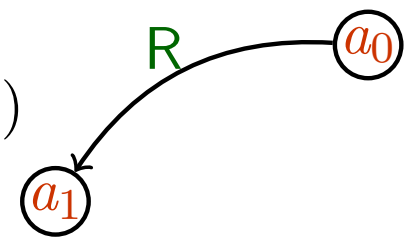
$\mathcal{G}_0$

Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$

$$X_1 \sqcap X_2 \sqcap X_3, X_1, X_2, X_3$$

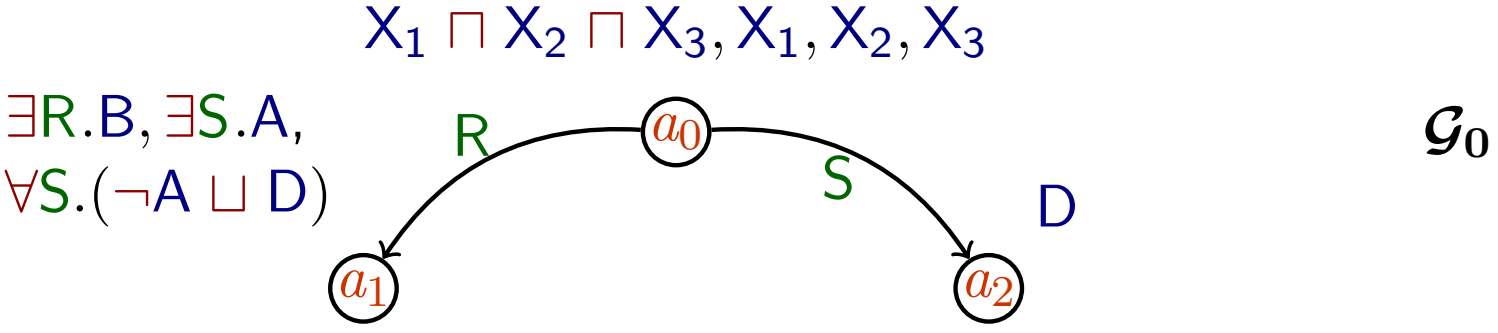
$\exists R.B, \exists S.A,$
$\forall S.(\neg A \sqcup D)$



$\mathcal{G}_0$

Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$



$\mathcal{G}_0$

Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$

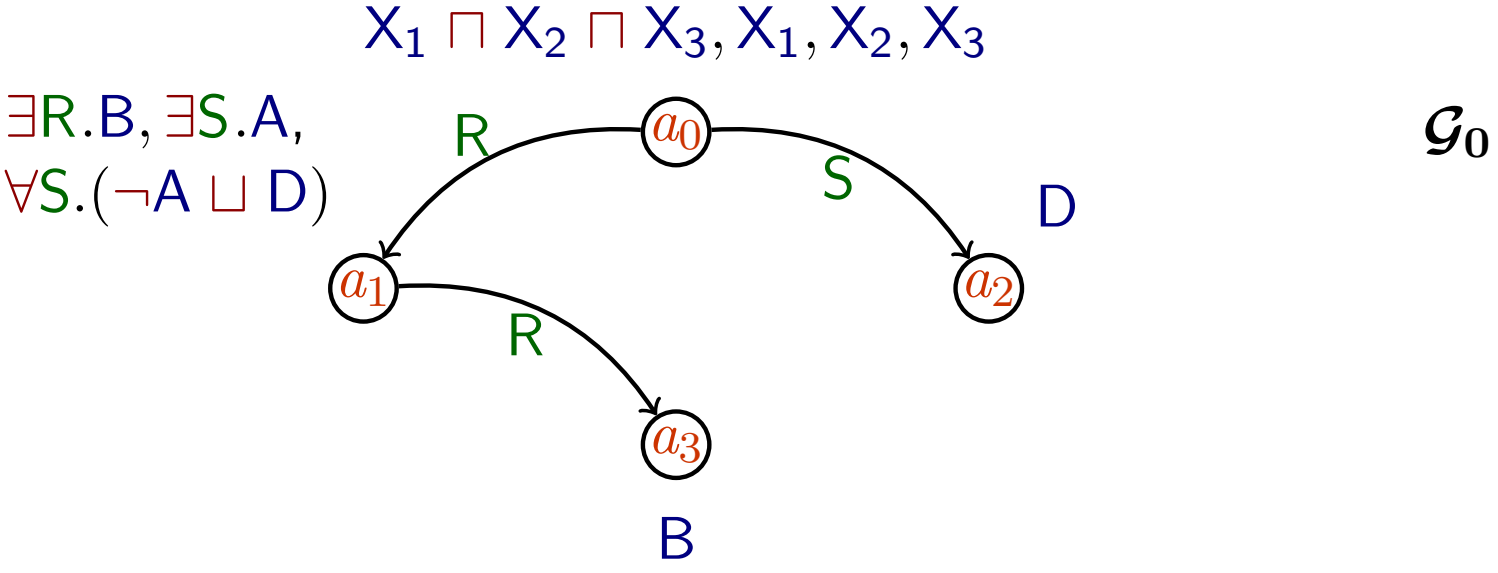
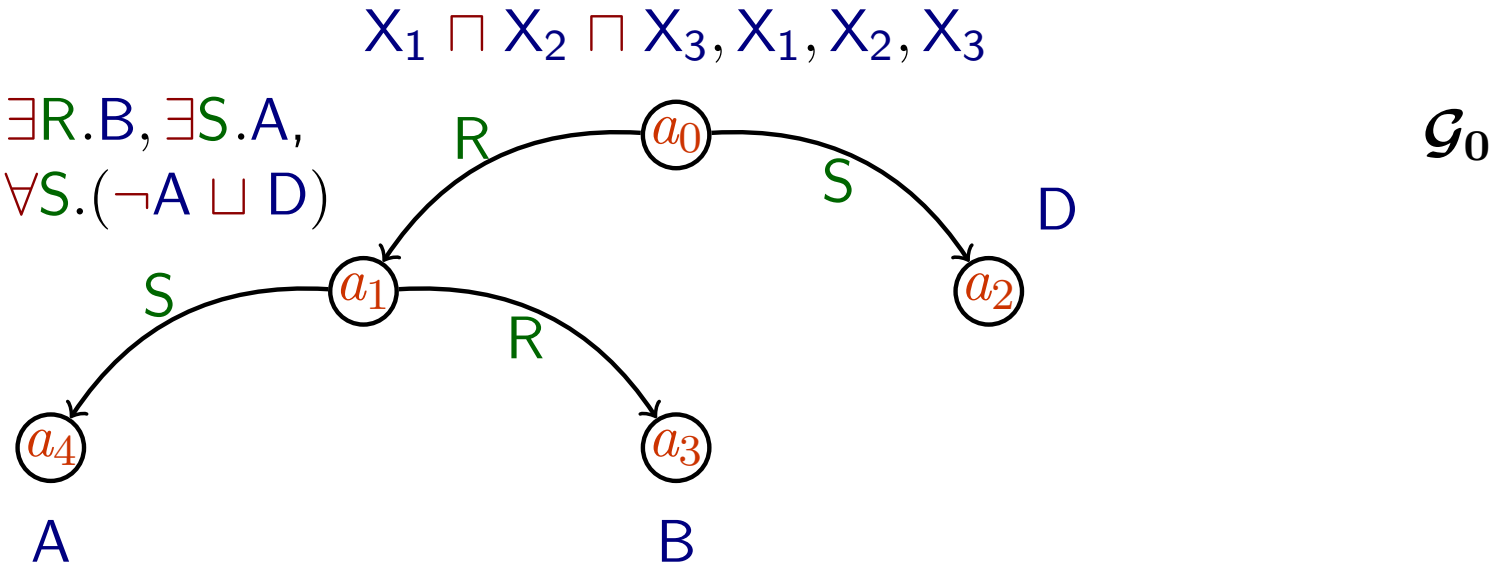


Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$



$\mathcal{G}_0$

Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$

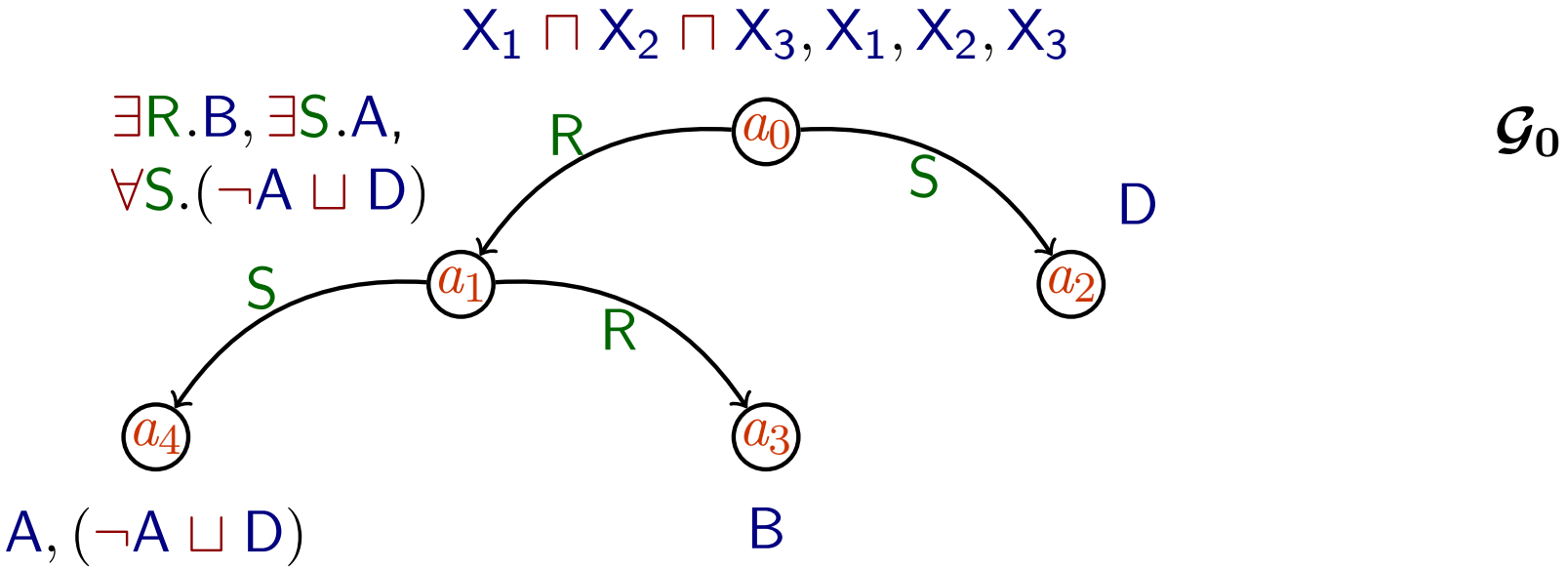


Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$

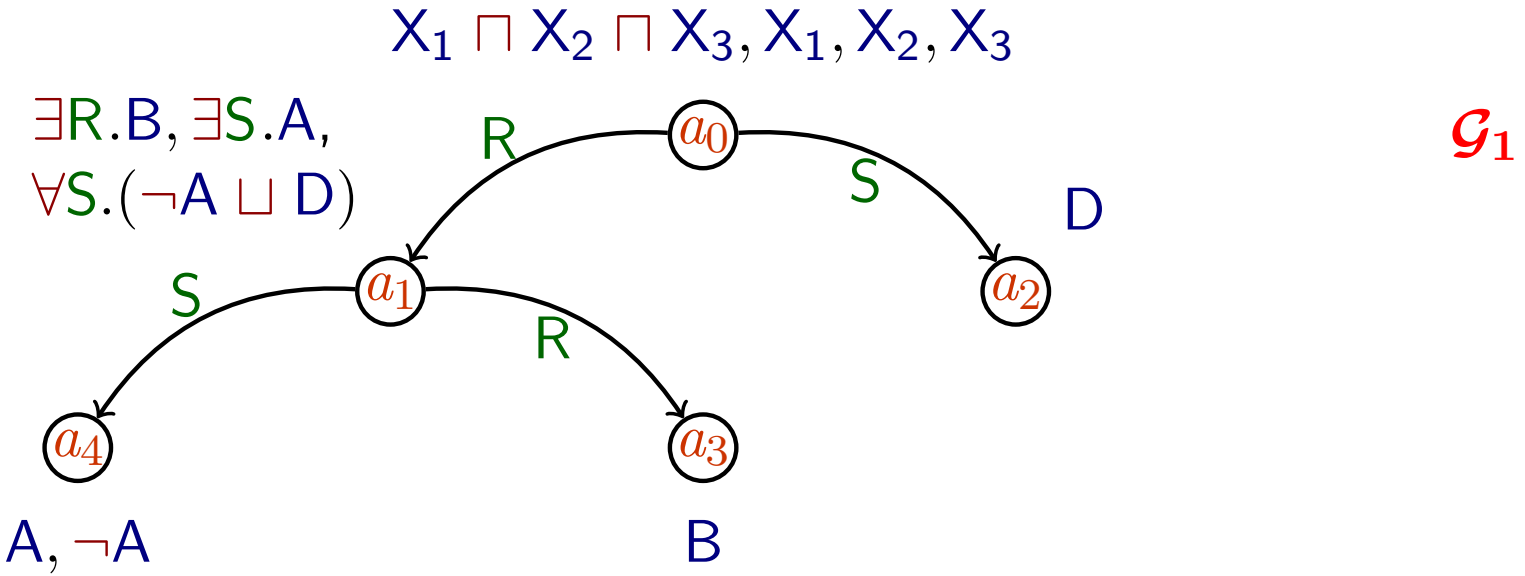


Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$

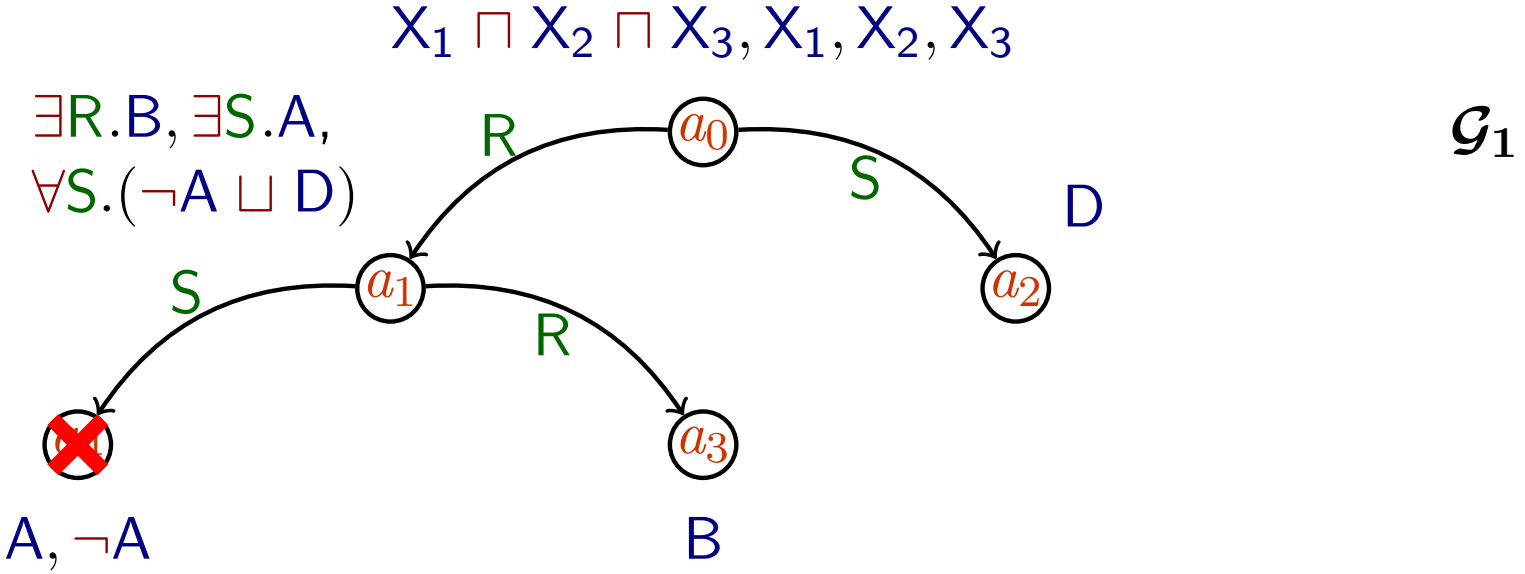


Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$

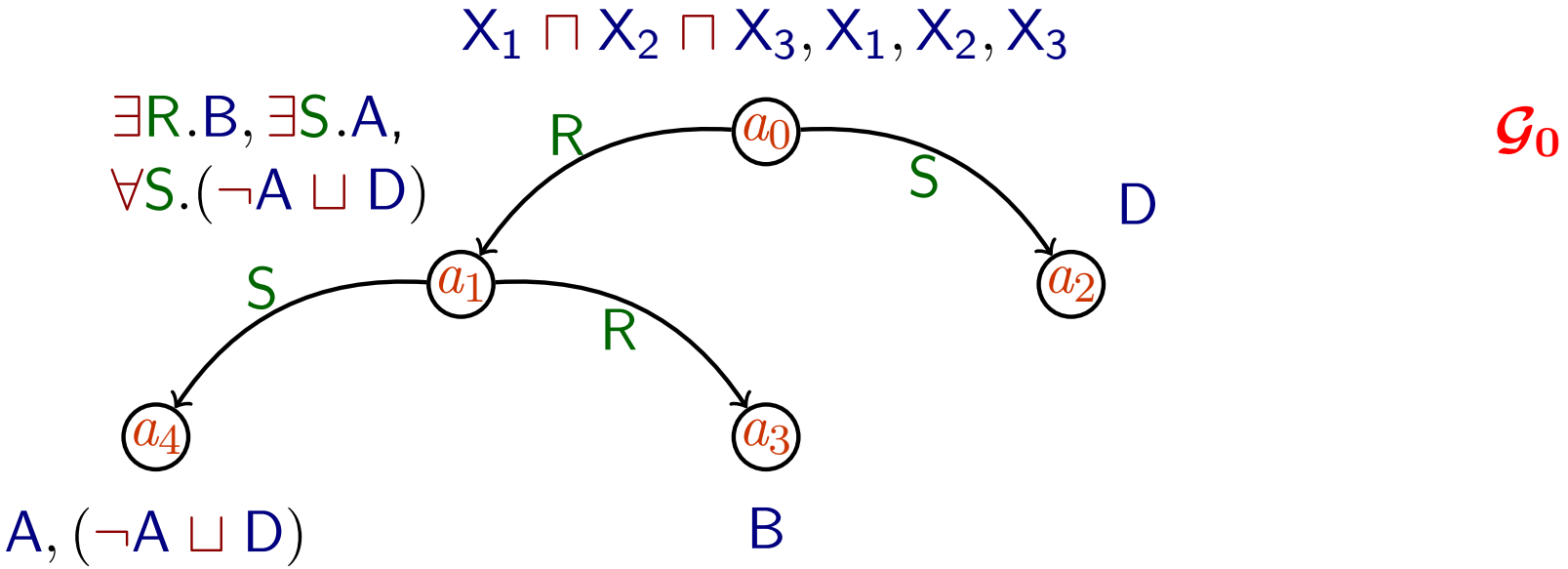
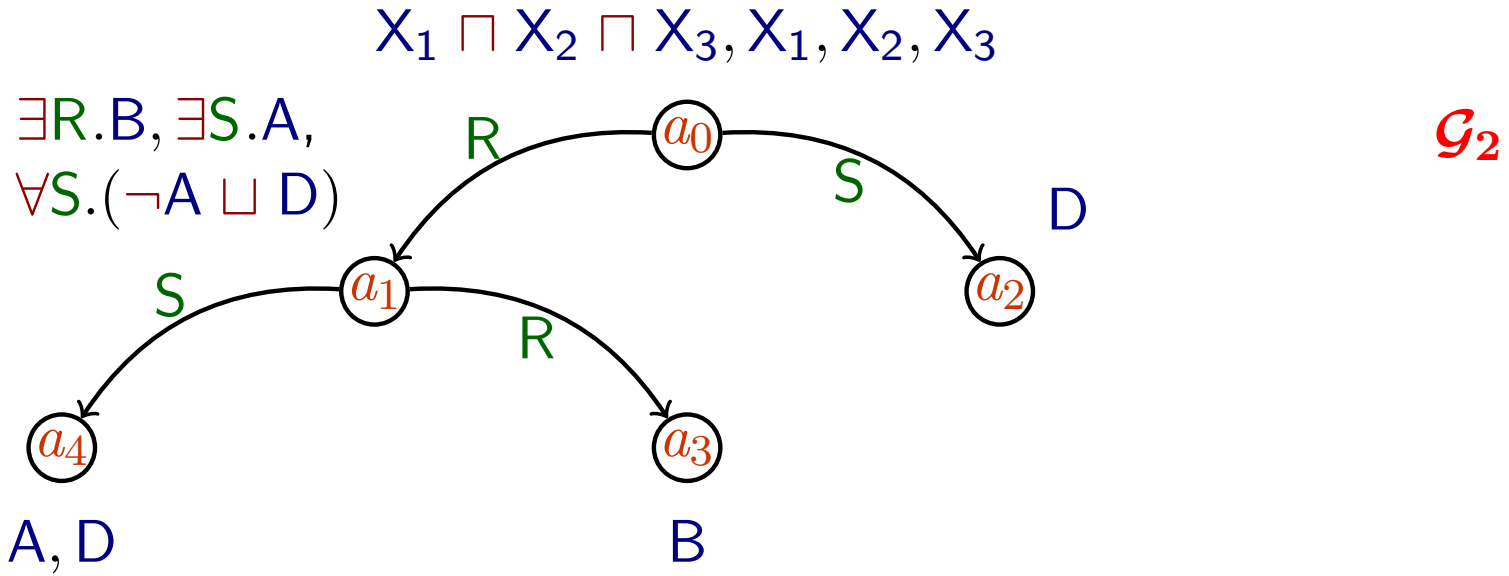


Tableau Algorithm: Example 2

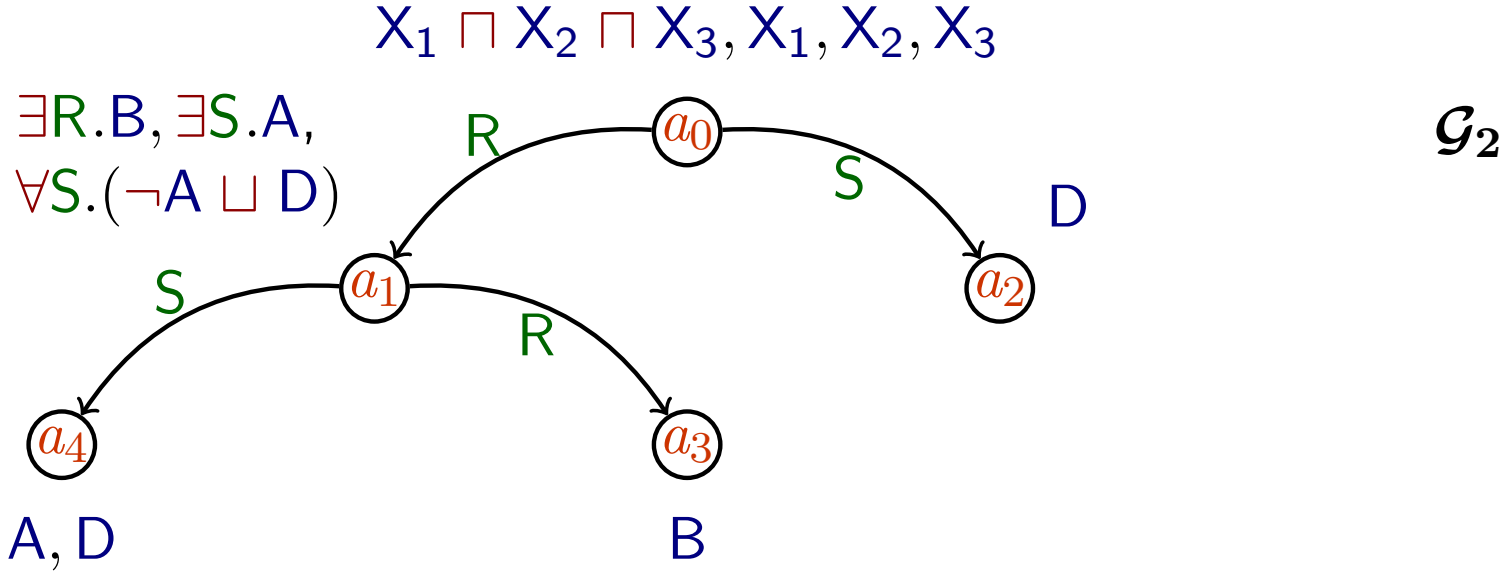
$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$



$\mathcal{G}_2$

Tableau Algorithm: Example 2

$$X \equiv \underbrace{\exists S.D}_{X_1} \sqcap \underbrace{\exists R.(\exists R.B \sqcap \exists S.A)}_{X_2} \sqcap \underbrace{\forall R.\forall S.(\neg A \sqcup D)}_{X_3}$$



$\mathcal{G}_2$ is a complete clash-free completion graph $\Rightarrow$ X is satisfiable

Tableau Algorithm for $\mathcal{ALC}$: Rule Ordering

Multiple expansion rules could be applicable

- Irrelevant for correctness
- Could be have a large impact on performance

Tableau Algorithm for $\mathcal{ALC}$: Rule Ordering

Multiple expansion rules could be applicable

- Irrelevant for correctness
- Could be have a large impact on performance

Goal: choose the order which will quickly lead to either:

- a complete completion graph
- clashes in all graphs

Tableau Algorithm for $\mathcal{ALC}$: Rule Ordering

Multiple expansion rules could be applicable

- Irrelevant for correctness
- Could be have a large impact on performance

Goal: choose the order which will quickly lead to either:

- a complete completion graph
- clashes in all graphs

General idea: apply cheap rules first and costly later
(maybe we will terminate before they're applied)

Tableau Algorithm for $\mathcal{ALC}$: Rule Ordering

Multiple expansion rules could be applicable

- Irrelevant for correctness
- Could be have a large impact on performance

Goal: choose the order which will quickly lead to either:

- a complete completion graph
- clashes in all graphs

General idea: apply cheap rules first and costly later
(maybe we will terminate before they're applied)

Question: which are cheap and which are costly?

Tableau Algorithm for $\mathcal{ALC}$: Rule Ordering

Cheap rules: only modify node labels in the current graph

$\sqcap$-rule, $\forall$-rule

Tableau Algorithm for $\mathcal{ALC}$: Rule Ordering

Cheap rules: only modify node labels in the current graph
$\sqcap$-rule, $\forall$-rule

Expensive rules:

- $\sqcup$-rule: makes a non-deterministic choice (creates new graphs)
- $\exists$-rule: expands the current graph

Tableau Algorithm for $\mathcal{ALC}$: Rule Ordering

Cheap rules: only modify node labels in the current graph
$\sqcap$-rule, $\forall$-rule

Expensive rules:

- $\sqcup$-rule: makes a non-deterministic choice (creates new graphs)
- $\exists$-rule: expands the current graph

$A \sqcap B, \neg A, \exists R.B$

$@$

Bad rule ordering:
first $\exists$-rule, then $\sqcap$-rule

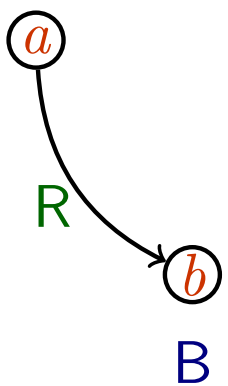
Tableau Algorithm for $\mathcal{ALC}$: Rule Ordering

Cheap rules: only modify node labels in the current graph
$\sqcap$-rule, $\forall$-rule

Expensive rules:

- $\sqcup$-rule: makes a non-deterministic choice (creates new graphs)
- $\exists$-rule: expands the current graph

$A \sqcap B, \neg A, \exists R.B$



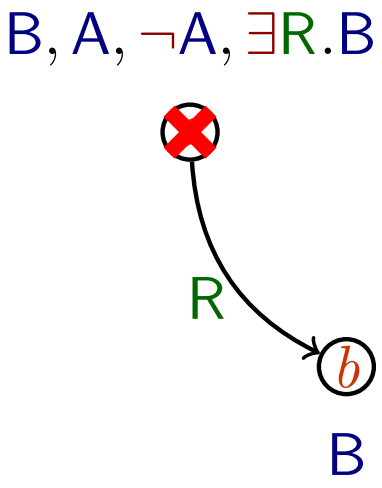
Bad rule ordering:
first $\exists$-rule, then $\sqcap$-rule

Tableau Algorithm for $\mathcal{ALC}$: Rule Ordering

Cheap rules: only modify node labels in the current graph
$\sqcap$-rule, $\forall$-rule

Expensive rules:

- $\sqcup$-rule: makes a non-deterministic choice (creates new graphs)
- $\exists$-rule: expands the current graph



Bad rule ordering:
first $\exists$-rule, then $\sqcap$-rule

Tableau Algorithm for $\mathcal{ALC}$: Rule Ordering

Cheap rules: only modify node labels in the current graph
$\sqcap$-rule, $\forall$-rule

Expensive rules:

- $\sqcup$-rule: makes a non-deterministic choice (creates new graphs)
- $\exists$-rule: expands the current graph

$A \sqcap B, \neg A, \exists R.B$

$@$

Good rule ordering:
first $\sqcap$-rule, then clash!

Tableau Algorithm for $\mathcal{ALC}$: Rule Ordering

Cheap rules: only modify node labels in the current graph
$\sqcap$-rule, $\forall$-rule

Expensive rules:

- $\sqcup$-rule: makes a non-deterministic choice (creates new graphs)
- $\exists$-rule: expands the current graph

$B, A, \neg A, \exists R.B$

Good rule ordering:
first $\sqcap$-rule, then clash!

Disjunctive Non-Determinism

The $\sqcup$-rule is the only one to create two graphs out of one

Observations:

- copying isn't necessary
  sufficient to consider only a single graph at a time
  (the reason the algorithm is in PSPACE)
- makes sense to apply this rule last

Disjunctive Non-Determinism

The $\sqcup$-rule is the only one to create two graphs out of one

Observations:

- copying isn't necessary
  sufficient to consider only a single graph at a time
  (the reason the algorithm is in PSPACE)
- makes sense to apply this rule last

Example: $\exists R.B \sqcap (C \sqcup \neg D)$

$\exists R.B \sqcap (C \sqcup \neg D)$

$@$

Good order: $\sqcup$ after $\sqcap$, $\exists$

Disjunctive Non-Determinism

The $\sqcup$-rule is the only one to create two graphs out of one

Observations:

- copying isn't necessary
  sufficient to consider only a single graph at a time
  (the reason the algorithm is in PSPACE)
- makes sense to apply this rule last

Example: $\exists R.B \sqcap (C \sqcup \neg D)$

$\exists R.B, C \sqcup \neg D$

$@$

Good order: $\sqcup$ after $\sqcap, \exists$

Disjunctive Non-Determinism

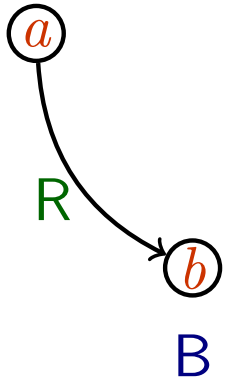
The $\sqcup$-rule is the only one to create two graphs out of one

Observations:

- copying isn't necessary
  sufficient to consider only a single graph at a time
  (the reason the algorithm is in PSPACE)
- makes sense to apply this rule last

Example: $\exists R.B \sqcap (C \sqcup \neg D)$

$\exists R.B, C \sqcup \neg D$



Good order: $\sqcup$ after $\sqcap$, $\exists$

Disjunctive Non-Determinism

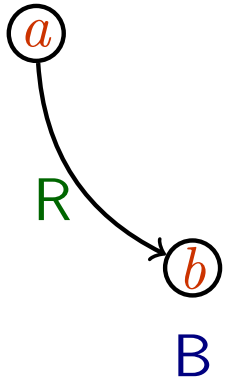
The $\sqcup$-rule is the only one to create two graphs out of one

Observations:

- copying isn't necessary
  sufficient to consider only a single graph at a time
  (the reason the algorithm is in PSPACE)
- makes sense to apply this rule last

Example: $\exists R.B \sqcap (C \sqcup \neg D)$

$\exists R.B, C$ (or $\neg D$)



Good order: $\sqcup$ after $\sqcap$, $\exists$

$\sqcap$, $\exists$, $\sqcup$

3 rule applications

Disjunctive Non-Determinism

The $\sqcup$-rule is the only one to create two graphs out of one

Observations:

- copying isn't necessary
  sufficient to consider only a single graph at a time
  (the reason the algorithm is in PSPACE)
- makes sense to apply this rule last

Example: $\exists R.B \sqcap (C \sqcup \neg D)$

$\exists R.B \sqcap (C \sqcup \neg D)$

a

Bad order: $\sqcup$ before $\exists$

Disjunctive Non-Determinism

The $\sqcup$-rule is the only one to create two graphs out of one

Observations:

- copying isn't necessary
  sufficient to consider only a single graph at a time
  (the reason the algorithm is in PSPACE)
- makes sense to apply this rule last

Example: $\exists R.B \sqcap (C \sqcup \neg D)$

$\exists R.B, C \sqcup \neg D$

$@$

Bad order: $\sqcup$ before $\exists$

Disjunctive Non-Determinism

The $\sqcup$-rule is the only one to create two graphs out of one

Observations:

- copying isn't necessary
  sufficient to consider only a single graph at a time
  (the reason the algorithm is in PSPACE)
- makes sense to apply this rule last

Example: $\exists R.B \sqcap (C \sqcup \neg D)$

Bad order: $\sqcup$ before $\exists$

$\exists R.B, C$

$@$ $\mathcal{G}_1$

Disjunctive Non-Determinism

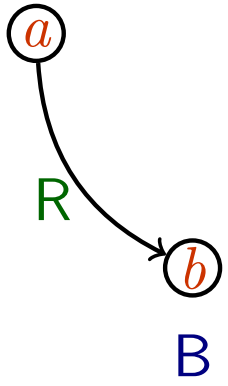
The $\sqcup$-rule is the only one to create two graphs out of one

Observations:

- copying isn't necessary
  sufficient to consider only a single graph at a time
  (the reason the algorithm is in PSPACE)
- makes sense to apply this rule last

Example: $\exists R.B \sqcap (C \sqcup \neg D)$ Bad order: $\sqcup$ before $\exists$

$\exists R.B, C$



$\mathcal{G}_1$

Disjunctive Non-Determinism

The $\sqcup$-rule is the only one to create two graphs out of one

Observations:

- copying isn't necessary
  sufficient to consider only a single graph at a time
  (the reason the algorithm is in PSPACE)
- makes sense to apply this rule last

Example: $\exists R.B \sqcap (C \sqcup \neg D)$

$\exists R.B, \neg D$

$@a$ $\mathcal{G}_2$

Bad order: $\sqcup$ before $\exists$

Disjunctive Non-Determinism

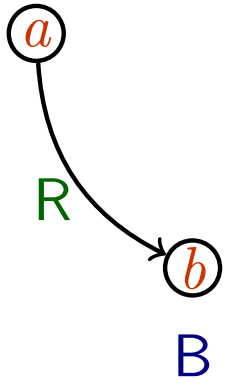
The $\sqcup$-rule is the only one to create two graphs out of one

Observations:

- copying isn't necessary
  sufficient to consider only a single graph at a time
  (the reason the algorithm is in PSPACE)
- makes sense to apply this rule last

Example: $\exists R.B \sqcap (C \sqcup \neg D)$

$\exists R.B, \neg D$



$\mathcal{G}_2$

Bad order: $\sqcup$ before $\exists$

$\sqcap$, $\sqcup$, $\exists$ $(\mathcal{G}_1)$, $\exists$ $(\mathcal{G}_2)$

4 rule applications

No reason to do the same work twice

Tableau Algorithm for ABox Consistency

ABox: $\mathcal{A}$ is a finite set of assertions $a : C$ or $(a, b) : \mathsf{R}$

Consistency: is there interpretation $\mathcal{I}$ which satisfies all assertions?

Tableau Algorithm for ABox Consistency

ABox: $\mathcal{A}$ is a finite set of assertions $a\colon C$ or $(a, b)\colon \mathsf{R}$

Consistency: is there interpretation $\mathcal{I}$ which satisfies all assertions?

A slight generalization of the concept satisfiability algorithm:

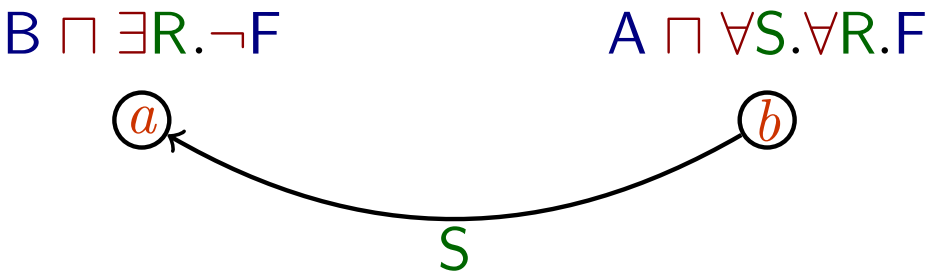
- Start with $\mathcal{G}_{\mathcal{A}}$ such that:
  - each individual in $\mathcal{A}$ corresponds to its own node in $\mathcal{G}_{\mathcal{A}}$
  - $a\colon C \in \mathcal{A}$, then $C \in L(a)$
  - $(a, b)\colon \mathsf{R} \in \mathcal{A}$, then an R-edge from a to b is in $\mathcal{G}_{\mathcal{A}}$
- Apply the tableau expansion rule exhaustively
  - clashes in all graphs $\Rightarrow$ inconsistent
  - otherwise $\Rightarrow$ consistent

ABox Consistency: Example

$$\mathcal{A} = \{a\colon (\mathsf{B} \sqcap \exists \mathsf{R}.\neg \mathsf{F}),\quad b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F}),\quad (b, a)\colon \mathsf{S}\}$$

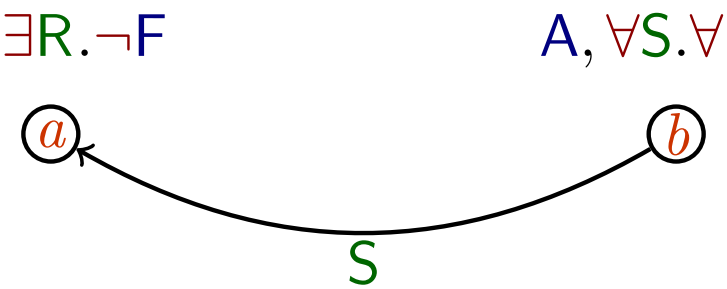
ABox Consistency: Example

$\mathcal{A} = \{a\colon (\mathsf{B} \sqcap \exists \mathsf{R}.\neg \mathsf{F}),\quad b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F}),\quad (b, a)\colon \mathsf{S}\}$



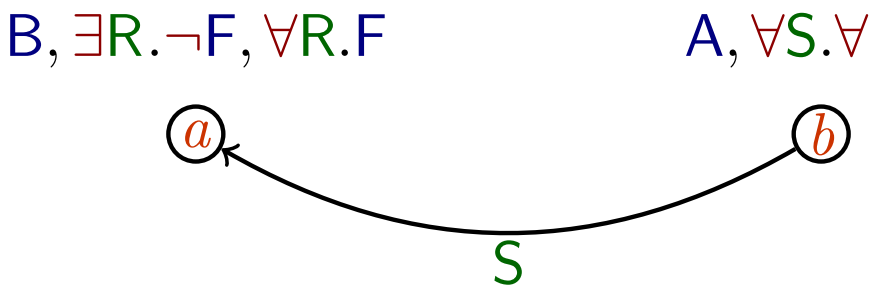
ABox Consistency: Example

$\mathcal{A} = \{a\colon (\mathsf{B} \sqcap \exists \mathsf{R}.\neg \mathsf{F}),\quad b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F}),\quad (b, a)\colon \mathsf{S}\}$



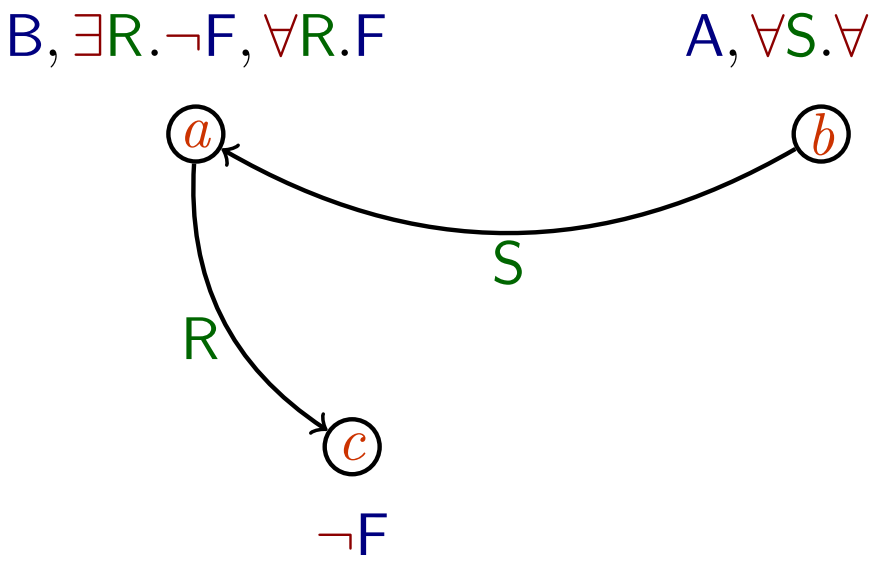
ABox Consistency: Example

$\mathcal{A} = \{a\colon (\mathsf{B} \sqcap \exists \mathsf{R}.\neg \mathsf{F}),\quad b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F}),\quad (b, a)\colon \mathsf{S}\}$



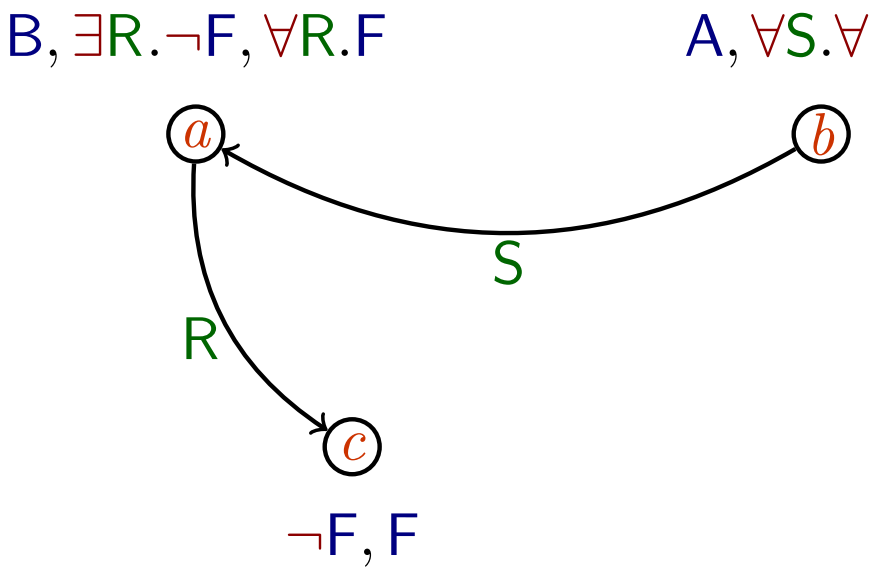
ABox Consistency: Example

$\mathcal{A} = \{a\colon (\mathsf{B} \sqcap \exists \mathsf{R}.\neg \mathsf{F}),\quad b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F}),\quad (b, a)\colon \mathsf{S}\}$



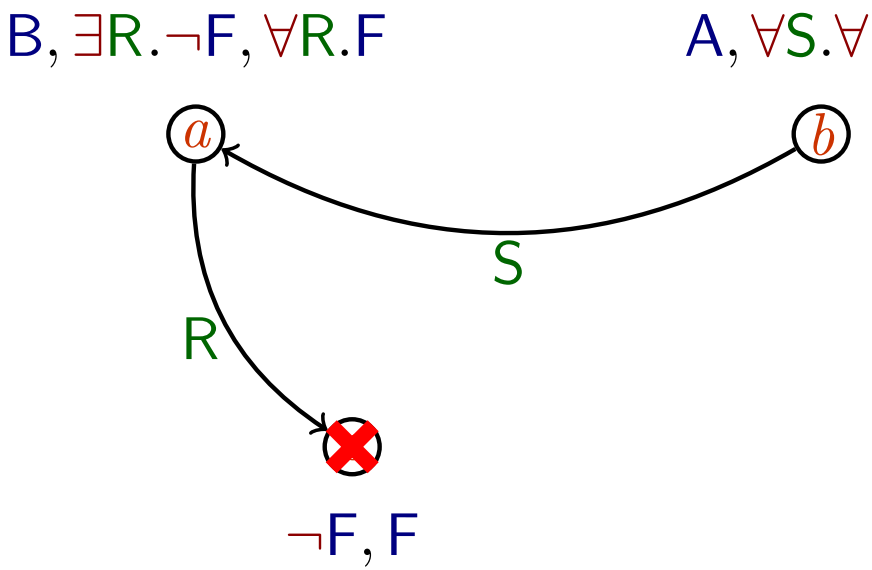
ABox Consistency: Example

$\mathcal{A} = \{a\colon (\mathsf{B} \sqcap \exists \mathsf{R}.\neg \mathsf{F}),\quad b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F}),\quad (b, a)\colon \mathsf{S}\}$



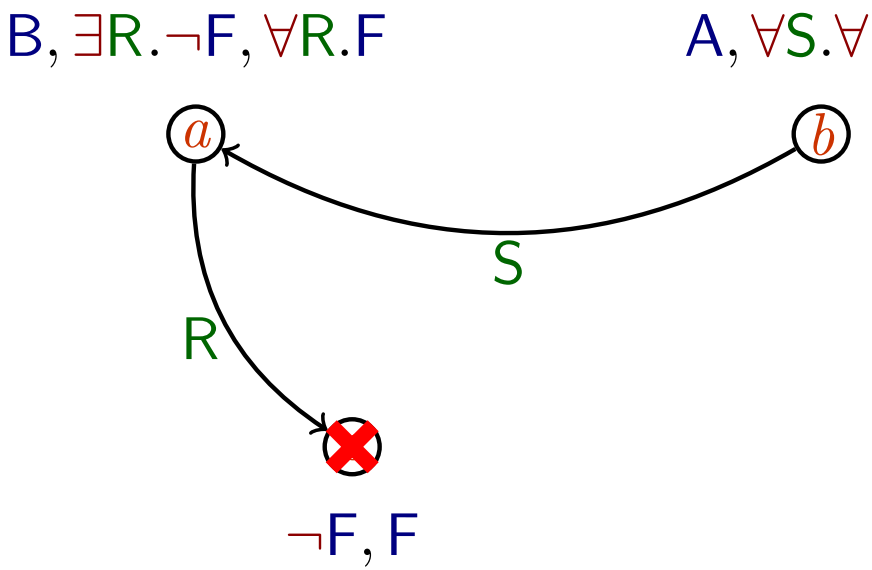
ABox Consistency: Example

$\mathcal{A} = \{a\colon (\mathsf{B} \sqcap \exists \mathsf{R}.\neg \mathsf{F}),\quad b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F}),\quad (b, a)\colon \mathsf{S}\}$



ABox Consistency: Example

$\mathcal{A} = \{a\colon (\mathsf{B} \sqcap \exists \mathsf{R}.\neg \mathsf{F}),\quad b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F}),\quad (b, a)\colon \mathsf{S}\}$



The same optimizations work

The same theoretical properties hold (soundness, completeness, termination, complexity)

Reminder: Reasoning Problems in DLs

$\mathcal{ALC}$ Tableau with TBoxes

Two Essential Optimizations

TBoxes Enter the Picture

TBox: $\mathcal{T}$ is a finite set of concept subsumption axioms $C \sqsubseteq D$

Problem: decide consistency of ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$
a.k.a. $\mathcal{A}$ consistency w.r.t. $\mathcal{T}$
"is data consistent w.r.t. schema?"

TBoxes Enter the Picture

TBox: $\mathcal{T}$ is a finite set of concept subsumption axioms $C \sqsubseteq D$

Problem: decide consistency of ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$
a.k.a. $\mathcal{A}$ consistency w.r.t. $\mathcal{T}$
"is data consistent w.r.t. schema?"

We consider cases:

- Acyclic simple TBoxes (easy)
- General TBoxes (not-so-easy)

Simple TBoxes and Reachibility

Axioms are restricted to: $\mathsf{A} \sqsubseteq C$ and $\mathsf{A} \equiv C$, where

A is a concept name and C is a concept expression

Each A (defined concept) occurs only once on the left

Simple TBoxes and Reachibility

Axioms are restricted to: A $\sqsubseteq C$ and A $\equiv C$, where

A is a concept name and C is a concept expression

Each A (defined concept) occurs only once on the left

A directly uses B if B (syntactically) occurs in C

A uses B if A directly uses B^* and B^* uses B
(B can be reached from A)

Simple TBoxes and Reachibility

Axioms are restricted to: $A \sqsubseteq C$ and $A \equiv C$, where

A is a concept name and C is a concept expression

Each A (defined concept) occurs only once on the left

A directly uses B if B (syntactically) occurs in C

A uses B if A directly uses B^* and B^* uses B
(B can be reached from A)

$\mathcal{T} = \{A \sqsubseteq \exists R.C \sqcap D, \quad C \equiv D \sqcup \neg F, \quad F \sqsubseteq E\}$

- A uses C, D

Simple TBoxes and Reachibility

Axioms are restricted to: $A \sqsubseteq C$ and $A \equiv C$, where

A is a concept name and C is a concept expression

Each A (defined concept) occurs only once on the left

A directly uses B if B (syntactically) occurs in C

A uses B if A directly uses B^* and B^* uses B
(B can be reached from A)

$\mathcal{T} = \{A \sqsubseteq \exists R.C \sqcap D, \quad C \equiv D \sqcup \neg F, \quad F \sqsubseteq E\}$

- A uses C, D
- C uses D, F

Simple TBoxes and Reachibility

Axioms are restricted to: $A \sqsubseteq C$ and $A \equiv C$, where

A is a concept name and C is a concept expression

Each A (defined concept) occurs only once on the left

A directly uses B if B (syntactically) occurs in C

A uses B if A directly uses B^* and B^* uses B
(B can be reached from A)

$\mathcal{T} = \{A \sqsubseteq \exists R.C \sqcap D, \quad C \equiv D \sqcup \neg F, \quad F \sqsubseteq E\}$

- A uses C, D, F
- C uses D, F

Simple TBoxes and Reachibility

Axioms are restricted to: $A \sqsubseteq C$ and $A \equiv C$, where

A is a concept name and C is a concept expression

Each A (defined concept) occurs only once on the left

A directly uses B if B (syntactically) occurs in C

A uses B if A directly uses B^* and B^* uses B
(B can be reached from A)

$\mathcal{T} = \{A \sqsubseteq \exists R.C \sqcap D, \quad C \equiv D \sqcup \neg F, \quad F \sqsubseteq E\}$

- A uses C, D, F
- C uses D, F
- F uses E

Simple TBoxes and Reachibility

Axioms are restricted to: $A \sqsubseteq C$ and $A \equiv C$, where

A is a concept name and C is a concept expression

Each A (defined concept) occurs only once on the left

A directly uses B if B (syntactically) occurs in C

A uses B if A directly uses B^* and B^* uses B
(B can be reached from A)

$\mathcal{T} = \{A \sqsubseteq \exists R.C \sqcap D, \quad C \equiv D \sqcup \neg F, \quad F \sqsubseteq E\}$

- A uses C, D, F
- C uses D, F, E
- F uses E

Simple TBoxes and Reachibility

Axioms are restricted to: $A \sqsubseteq C$ and $A \equiv C$, where

A is a concept name and C is a concept expression

Each A (defined concept) occurs only once on the left

A directly uses B if B (syntactically) occurs in C

A uses B if A directly uses B^* and B^* uses B
(B can be reached from A)

$\mathcal{T} = \{A \sqsubseteq \exists R.C \sqcap D, \quad C \equiv D \sqcup \neg F, \quad F \sqsubseteq E\}$

- A uses C, D, F, E
- C uses D, F, E
- F uses E

Acyclic Simple TBoxes

Acyclicity: No concept name uses itself

Acyclic Simple TBoxes

Acyclicity: No concept name uses itself

Acyclic simple TBox:
$\mathcal{T} = \{A \sqsubseteq \exists R.C \sqcap D, \quad C \equiv D \sqcup \neg F, \quad F \sqsubseteq E\}$

Acyclic Simple TBoxes

Acyclicity: No concept name uses itself

Acyclic simple TBox:
$\mathcal{T} = \{A \sqsubseteq \exists R.C \sqcap D, \quad C \equiv D \sqcup \neg F, \quad F \sqsubseteq E\}$

Cyclic simple axiom: $Human \sqsubseteq \forall hasChild.Human$

Acyclic Simple TBoxes

Acyclicity: No concept name uses itself

Acyclic simple TBox:
$\mathcal{T} = \{A \sqsubseteq \exists R.C \sqcap D, \quad C \equiv D \sqcup \neg F, \quad F \sqsubseteq E\}$

Cyclic simple axiom: $Human \sqsubseteq \forall hasChild.Human$

Question: how to check acyclicity?

Unfolding

Ontology consistency can be reduced to ABox consistency
if TBox is acyclic and simple via the procedure called unfolding

Unfolding

Ontology consistency can be reduced to ABox consistency if TBox is acyclic and simple via the procedure called unfolding

Recursively unfold each defined concept in ABox untill there are no concepts defined in TBox

- $\mathsf{A} \equiv C$: replace A by C
- $\mathsf{A} \sqsubseteq C$: replace A by $C \sqcap \mathsf{A}^*$
  (since $\mathsf{A} \sqsubseteq C$ is equivalent to saying $\mathsf{A} \equiv C \sqcap \mathsf{A}^*$ for fresh A^*)

Unfolding

Ontology consistency can be reduced to ABox consistency if TBox is acyclic and simple via the procedure called unfolding

Recursively unfold each defined concept in ABox untill there are no concepts defined in TBox

- $\mathsf{A} \equiv C$: replace A by C
- $\mathsf{A} \sqsubseteq C$: replace A by $C \sqcap \mathsf{A}^*$
  (since $\mathsf{A} \sqsubseteq C$ is equivalent to saying $\mathsf{A} \equiv C \sqcap \mathsf{A}^*$ for fresh A^*)

Unfolding is a completely syntactic procedure

Unfolding: Example

$\mathcal{A} = \{a : (\mathsf{B} \sqcap \exists \mathsf{R}.\neg \mathsf{F}), \quad b : (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F})\}$

$\mathcal{T} = \{\mathsf{B} \equiv \exists \mathsf{S}.\mathsf{D}, \quad \mathsf{F} \sqsubseteq \mathsf{B} \sqcap \mathsf{C}\}$

Unfolding: Example

$\mathcal{A} = \{a\colon (\mathsf{B} \sqcap \exists \mathsf{R}.\neg \mathsf{F}), \quad b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F})\}$
$\mathcal{T} = \{\mathsf{B} \equiv \exists \mathsf{S}.\mathsf{D}, \quad \mathsf{F} \sqsubseteq \mathsf{B} \sqcap \mathsf{C}\}$

$\mathcal{T} \rightsquigarrow \{\mathsf{B} \equiv \exists \mathsf{S}.\mathsf{D}, \quad \mathsf{F} \equiv \mathsf{B} \sqcap \mathsf{C} \sqcap \mathsf{F}^*\}$

Unfolding: Example

$\mathcal{A} = \{a\colon (\mathsf{B} \sqcap \exists \mathsf{R}.\neg \mathsf{F}), \quad b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F})\}$
$\mathcal{T} = \{\mathsf{B} \equiv \exists \mathsf{S}.\mathsf{D}, \quad \mathsf{F} \sqsubseteq \mathsf{B} \sqcap \mathsf{C}\}$

$\mathcal{T} \rightsquigarrow \{\mathsf{B} \equiv \exists \mathsf{S}.\mathsf{D}, \quad \mathsf{F} \equiv \mathsf{B} \sqcap \mathsf{C} \sqcap \mathsf{F}^*\}$

$\mathcal{A} \rightsquigarrow \{a\colon (\exists \mathsf{S}.\mathsf{D} \sqcap \exists \mathsf{R}.\neg \mathsf{F}, \quad \mathsf{B}$ unfolded
$b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F})\}$

Unfolding: Example

$\mathcal{A} = \{a\colon (\mathsf{B} \sqcap \exists \mathsf{R}.\neg \mathsf{F}), \quad b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F})\}$
$\mathcal{T} = \{\mathsf{B} \equiv \exists \mathsf{S}.\mathsf{D}, \quad \mathsf{F} \sqsubseteq \mathsf{B} \sqcap \mathsf{C}\}$

$\mathcal{T} \rightsquigarrow \{\mathsf{B} \equiv \exists \mathsf{S}.\mathsf{D}, \quad \mathsf{F} \equiv \mathsf{B} \sqcap \mathsf{C} \sqcap \mathsf{F}^*\}$

$\mathcal{A} \rightsquigarrow \{a\colon (\exists \mathsf{S}.\mathsf{D} \sqcap \exists \mathsf{R}.\neg \mathsf{F}, \quad \mathsf{B}$ unfolded
$b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F})\}$

$\mathcal{A} \rightsquigarrow \{a\colon (\exists \mathsf{S}.\mathsf{D} \sqcap \exists \mathsf{R}.\neg(\mathsf{B} \sqcap \mathsf{C} \sqcap \mathsf{F}^*)), \quad \mathsf{F}$ unfolded
$b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.(\mathsf{B} \sqcap \mathsf{C} \sqcap \mathsf{F}^*))\}$

Unfolding: Example

$\mathcal{A} = \{a\colon (\mathsf{B} \sqcap \exists \mathsf{R}.\neg \mathsf{F}), \quad b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F})\}$
$\mathcal{T} = \{\mathsf{B} \equiv \exists \mathsf{S}.\mathsf{D}, \quad \mathsf{F} \sqsubseteq \mathsf{B} \sqcap \mathsf{C}\}$

$\mathcal{T} \rightsquigarrow \{\mathsf{B} \equiv \exists \mathsf{S}.\mathsf{D}, \quad \mathsf{F} \equiv \mathsf{B} \sqcap \mathsf{C} \sqcap \mathsf{F}^*\}$

$\mathcal{A} \rightsquigarrow \{a\colon (\exists \mathsf{S}.\mathsf{D} \sqcap \exists \mathsf{R}.\neg \mathsf{F}, \quad$ B unfolded
$b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F})\}$

$\mathcal{A} \rightsquigarrow \{a\colon (\exists \mathsf{S}.\mathsf{D} \sqcap \exists \mathsf{R}.\neg(\mathsf{B} \sqcap \mathsf{C} \sqcap \mathsf{F}^*)), \quad$ F unfolded
$b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.(\mathsf{B} \sqcap \mathsf{C} \sqcap \mathsf{F}^*))\}$

$\mathcal{A} \rightsquigarrow \{a\colon (\exists \mathsf{S}.\mathsf{D} \sqcap \exists \mathsf{R}.\neg(\exists \mathsf{S}.\mathsf{D} \sqcap \mathsf{C} \sqcap \mathsf{F}^*)), \quad$ B further unfolded
$b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.(\exists \mathsf{S}.(\mathsf{D} \sqcap \mathsf{C} \sqcap \mathsf{F}^*)))\}$

Unfolding: Example

$\mathcal{A} = \{a\colon (\mathsf{B} \sqcap \exists \mathsf{R}.\neg \mathsf{F}), \quad b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F})\}$
$\mathcal{T} = \{\mathsf{B} \equiv \exists \mathsf{S}.\mathsf{D}, \quad \mathsf{F} \sqsubseteq \mathsf{B} \sqcap \mathsf{C}\}$

$\mathcal{T} \rightsquigarrow \{\mathsf{B} \equiv \exists \mathsf{S}.\mathsf{D}, \quad \mathsf{F} \equiv \mathsf{B} \sqcap \mathsf{C} \sqcap \mathsf{F}^*\}$

$\mathcal{A} \rightsquigarrow \{a\colon (\exists \mathsf{S}.\mathsf{D} \sqcap \exists \mathsf{R}.\neg \mathsf{F}, \quad$ B unfolded
$b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.\mathsf{F})\}$

$\mathcal{A} \rightsquigarrow \{a\colon (\exists \mathsf{S}.\mathsf{D} \sqcap \exists \mathsf{R}.\neg(\mathsf{B} \sqcap \mathsf{C} \sqcap \mathsf{F}^*)), \quad$ F unfolded
$b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.(\mathsf{B} \sqcap \mathsf{C} \sqcap \mathsf{F}^*))\}$

$\mathcal{A} \rightsquigarrow \{a\colon (\exists \mathsf{S}.\mathsf{D} \sqcap \exists \mathsf{R}.\neg(\exists \mathsf{S}.\mathsf{D} \sqcap \mathsf{C} \sqcap \mathsf{F}^*)), \quad$ B further unfolded
$b\colon (\mathsf{A} \sqcap \forall \mathsf{S}.\forall \mathsf{R}.(\exists \mathsf{S}.(\mathsf{D} \sqcap \mathsf{C} \sqcap \mathsf{F}^*)))\}$

Now $(\mathcal{T}, \mathcal{A})$ is consistent if and only if $\mathcal{A}$ is consistent
(same technique works for concept satisfiability, subsumption, etc.)

Unfolding: Problems

How much bigger can the unfolded concepts be?

Unfolding: Problems

How much bigger can the unfolded concepts be?

$A_1 \equiv \exists R_1.A_0 \sqcap \exists R_2.A_0$,
$A_2 \equiv \exists R_1.A_1 \sqcap \exists R_2.A_1$,
$A_3 \equiv \exists R_1.A_2 \sqcap \exists R_2.A_2$
...

See what happens when A_3 is unfolded:

Unfolding: Problems

How much bigger can the unfolded concepts be?

$A_1 \equiv \exists R_1.A_0 \sqcap \exists R_2.A_0$,
$A_2 \equiv \exists R_1.A_1 \sqcap \exists R_2.A_1$,
$A_3 \equiv \exists R_1.A_2 \sqcap \exists R_2.A_2$
...

See what happens when A_3 is unfolded:

$A_3 \equiv \exists R_1.(\exists R_1.A_1 \sqcap \exists R_2.A_1) \sqcap \exists R_2.(\exists R_1.A_1 \sqcap \exists R_2.A_1)$

Unfolding: Problems

How much bigger can the unfolded concepts be?

$A_1 \equiv \exists R_1.A_0 \sqcap \exists R_2.A_0$,
$A_2 \equiv \exists R_1.A_1 \sqcap \exists R_2.A_1$,
$A_3 \equiv \exists R_1.A_2 \sqcap \exists R_2.A_2$
...

See what happens when A_3 is unfolded:

$$A_3 \equiv \exists R_1.(\exists R_1.A_1 \sqcap \exists R_2.A_1) \sqcap \exists R_2.(\exists R_1.A_1 \sqcap \exists R_2.A_1)$$

$$A_3 \equiv \exists R_1.(\exists R_1.(\exists R_1.A_0 \sqcap \exists R_2.A_0) \sqcap \exists R_2.(\exists R_1.A_0 \sqcap \exists R_2.A_0)) \sqcap \exists R_2.(\exists R_1.(\exists R_1.A_0 \sqcap \exists R_2.A_0) \sqcap \exists R_2.(\exists R_1.A_0 \sqcap \exists R_2.A_0))$$

Unfolding: Problems

How much bigger can the unfolded concepts be?

$A_1 \equiv \exists R_1.A_0 \sqcap \exists R_2.A_0$,
$A_2 \equiv \exists R_1.A_1 \sqcap \exists R_2.A_1$,
$A_3 \equiv \exists R_1.A_2 \sqcap \exists R_2.A_2$
...

See what happens when A_3 is unfolded:

$$A_3 \equiv \exists R_1.(\exists R_1.A_1 \sqcap \exists R_2.A_1) \sqcap \exists R_2.(\exists R_1.A_1 \sqcap \exists R_2.A_1)$$

$$A_3 \equiv \exists R_1.(\exists R_1.(\exists R_1.A_0 \sqcap \exists R_2.A_0) \sqcap \exists R_2.(\exists R_1.A_0 \sqcap \exists R_2.A_0)) \sqcap \exists R_2.(\exists R_1.(\exists R_1.A_0 \sqcap \exists R_2.A_0) \sqcap \exists R_2.(\exists R_1.A_0 \sqcap \exists R_2.A_0))$$

Unfolding can be done lazily
B unfolded only when B or $\neg$B is in $L(v)$ for some v

General TBoxes

TBox is a set of general concept inclusion axioms (GCIs) $C \sqsubseteq D$

$C \equiv D$ is a syntactic sugar for $\{C \sqsubseteq D,\ D \sqsubseteq C\}$

(C, D are any $\mathcal{ALC}$ concept expressions)

General TBoxes

TBox is a set of general concept inclusion axioms (GCIs) $C \sqsubseteq D$
$C \equiv D$ is a syntactic sugar for $\{C \sqsubseteq D,\ D \sqsubseteq C\}$
(C, D are any $\mathcal{ALC}$ concept expressions)

Consistency of $(\mathcal{T}, \mathcal{A})$ no longer reducible to consistency of $\mathcal{A}$
need to take $\mathcal{T}$ into account when expanding the tableau

General TBoxes

TBox is a set of general concept inclusion axioms (GCIs) $C \sqsubseteq D$
$C \equiv D$ is a syntactic sugar for $\{C \sqsubseteq D,\ D \sqsubseteq C\}$
(C, D are any $\mathcal{ALC}$ concept expressions)

Consistency of $(\mathcal{T}, \mathcal{A})$ no longer reducible to consistency of $\mathcal{A}$
need to take $\mathcal{T}$ into account when expanding the tableau

Question: would the following rule be sufficient?

GCI_0-rule: if $C \in L(a)$ for some a and $C \sqsubseteq D \in \mathcal{T}$
and $D \notin L(a)$
then add D to $L(a)$

General TBoxes

TBox is a set of general concept inclusion axioms (GCIs) $C \sqsubseteq D$
$C \equiv D$ is a syntactic sugar for $\{C \sqsubseteq D,\ D \sqsubseteq C\}$
(C, D are any $\mathcal{ALC}$ concept expressions)

Consistency of $(\mathcal{T}, \mathcal{A})$ no longer reducible to consistency of $\mathcal{A}$
need to take $\mathcal{T}$ into account when expanding the tableau

Question: would the following rule be sufficient?

GCI$_0$-rule: if $C \in L(a)$ for some a and $C \sqsubseteq D \in \mathcal{T}$
and $D \notin L(a)$
then add D to $L(a)$

i.e. can we view GCIs as IF-THEN rules?

Local GCI Rule: Problem 1

$\mathcal{T} \equiv \{\mathsf{A} \sqcap \mathsf{B} \sqsubseteq \mathsf{C}\}$
$\mathcal{A} \equiv \{a\colon \mathsf{A}, \quad a\colon \mathsf{B}\}$

$(\mathcal{T}, \mathcal{A}) \models a\colon \mathsf{C}$, can prove inconsistency of $(\mathcal{T}, \mathcal{A} \cup \{a\colon \neg\mathsf{C}\})$?

Local GCI Rule: Problem 1

$\mathcal{T} \equiv \{\mathsf{A} \sqcap \mathsf{B} \sqsubseteq \mathsf{C}\}$
$\mathcal{A} \equiv \{a\colon \mathsf{A}, \quad a\colon \mathsf{B}\}$

$(\mathcal{T}, \mathcal{A}) \models a\colon \mathsf{C}$, can prove inconsistency of $(\mathcal{T}, \mathcal{A} \cup \{a\colon \neg\mathsf{C}\})$?

The problem: $L(a)$ contains $\mathsf{A}, \mathsf{B}, \neg\mathsf{C}$ but no $\mathsf{A} \sqcap \mathsf{B}$
so we can't apply the GCI_0-rule: and get the $\{\mathsf{C}, \neg\mathsf{C}\}$ clash

Local GCI Rule: Problem 1

$\mathcal{T} \equiv \{\mathsf{A} \sqcap \mathsf{B} \sqsubseteq \mathsf{C}\}$
$\mathcal{A} \equiv \{a\colon \mathsf{A}, \quad a\colon \mathsf{B}\}$

$(\mathcal{T}, \mathcal{A}) \models a\colon \mathsf{C}$, can prove inconsistency of $(\mathcal{T}, \mathcal{A} \cup \{a\colon \neg\mathsf{C}\})$?

The problem: $L(a)$ contains $\mathsf{A}, \mathsf{B}, \neg\mathsf{C}$ but no $\mathsf{A} \sqcap \mathsf{B}$
so we can't apply the GCI_0-rule: and get the $\{\mathsf{C}, \neg\mathsf{C}\}$ clash

OK, one can "construct" conjunctions:
$\sqcap^+$-rule: $\{A, B\} \subseteq L(a)$, $A \sqcap B \notin L(a)$, then add $A \sqcap B$ to $L(a)$

Local GCI Rule: Problem 1

$\mathcal{T} \equiv \{\mathsf{A} \sqcap \mathsf{B} \sqsubseteq \mathsf{C}\}$
$\mathcal{A} \equiv \{a\colon \mathsf{A}, \quad a\colon \mathsf{B}\}$

$(\mathcal{T}, \mathcal{A}) \models a\colon \mathsf{C}$, can prove inconsistency of $(\mathcal{T}, \mathcal{A} \cup \{a\colon \neg\mathsf{C}\})$?

The problem: $L(a)$ contains $\mathsf{A}, \mathsf{B}, \neg\mathsf{C}$ but no $\mathsf{A} \sqcap \mathsf{B}$
so we can't apply the GCI_0-rule: and get the $\{\mathsf{C}, \neg\mathsf{C}\}$ clash

OK, one can "construct" conjunctions:
$\sqcap^+$-rule: $\{A, B\} \subseteq L(a)$, $A \sqcap B \notin L(a)$, then add $A \sqcap B$ to $L(a)$

- Fixes this sort of problems
- Messy, many useless conjunctions
- Particularly messy for disjunctions (what would the rule be?)

Local GCI Rule: Bigger Problem

$\mathcal{T} \equiv \{C \sqsubseteq \mathsf{D}, \quad \mathsf{D} \sqsubseteq \mathsf{F}, \quad \neg C \sqcup \mathsf{F} \sqsubseteq \mathsf{X}\}$
$\mathcal{A} \equiv \{a : \mathsf{A}\}$

Question: are there any entailments for a?

Local GCI Rule: Bigger Problem

$\mathcal{T} \equiv \{C \sqsubseteq \mathsf{D}, \quad \mathsf{D} \sqsubseteq \mathsf{F}, \quad \neg C \sqcup \mathsf{F} \sqsubseteq \mathsf{X}\}$
$\mathcal{A} \equiv \{a : \mathsf{A}\}$

Question: are there any entailments for a?

$(\mathcal{T}, \mathcal{A}) \models a : \mathsf{X}$, can prove inconsistency of $(\mathcal{T}, \mathcal{A} \cup \{a : \neg\mathsf{X}\})$?

Local GCI Rule: Bigger Problem

$\mathcal{T} \equiv \{C \sqsubseteq \mathsf{D}, \quad \mathsf{D} \sqsubseteq \mathsf{F}, \quad \neg C \sqcup \mathsf{F} \sqsubseteq \mathsf{X}\}$
$\mathcal{A} \equiv \{a : \mathsf{A}\}$

Question: are there any entailments for a?

$(\mathcal{T}, \mathcal{A}) \models a : \mathsf{X}$, can prove inconsistency of $(\mathcal{T}, \mathcal{A} \cup \{a : \neg\mathsf{X}\})$?

Not really, stuck with $L(a) = \{\mathsf{A}\}$

Local GCI Rule: Bigger Problem

$\mathcal{T} \equiv \{C \sqsubseteq \mathsf{D}, \quad \mathsf{D} \sqsubseteq \mathsf{F}, \quad \neg C \sqcup \mathsf{F} \sqsubseteq \mathsf{X}\}$
$\mathcal{A} \equiv \{a : \mathsf{A}\}$

Question: are there any entailments for a?

$(\mathcal{T}, \mathcal{A}) \models a : \mathsf{X}$, can prove inconsistency of $(\mathcal{T}, \mathcal{A} \cup \{a : \neg\mathsf{X}\})$?

Not really, stuck with $L(a) = \{\mathsf{A}\}$

Reasoning by cases: in every $\mathcal{I}$, $a^{\mathcal{I}} \in C^{\mathcal{I}}$ or $a^{\mathcal{I}} \in (\neg C)^{\mathcal{I}}$

Local GCI Rule: Bigger Problem

$\mathcal{T} \equiv \{C \sqsubseteq \mathsf{D}, \quad \mathsf{D} \sqsubseteq \mathsf{F}, \quad \neg C \sqcup \mathsf{F} \sqsubseteq \mathsf{X}\}$
$\mathcal{A} \equiv \{a : \mathsf{A}\}$

Question: are there any entailments for a?

$(\mathcal{T}, \mathcal{A}) \models a : \mathsf{X}$, can prove inconsistency of $(\mathcal{T}, \mathcal{A} \cup \{a : \neg\mathsf{X}\})$?

Not really, stuck with $L(a) = \{\mathsf{A}\}$

Reasoning by cases: in every $\mathcal{I}$, $a^{\mathcal{I}} \in C^{\mathcal{I}}$ or $a^{\mathcal{I}} \in (\neg C)^{\mathcal{I}}$

I: $a^{\mathcal{I}} \in C^{\mathcal{I}}$, thus $a^{\mathcal{I}} \in \mathsf{D}^{\mathcal{I}}$, thus $a^{\mathcal{I}} \in \mathsf{F}^{\mathcal{I}}$, thus $a^{\mathcal{I}} \in \mathsf{X}^{\mathcal{I}}$

Local GCI Rule: Bigger Problem

$\mathcal{T} \equiv \{C \sqsubseteq \mathsf{D}, \quad \mathsf{D} \sqsubseteq \mathsf{F}, \quad \neg C \sqcup \mathsf{F} \sqsubseteq \mathsf{X}\}$
$\mathcal{A} \equiv \{a : \mathsf{A}\}$

Question: are there any entailments for a?

$(\mathcal{T}, \mathcal{A}) \models a : \mathsf{X}$, can prove inconsistency of $(\mathcal{T}, \mathcal{A} \cup \{a : \neg\mathsf{X}\})$?

Not really, stuck with $L(a) = \{\mathsf{A}\}$

Reasoning by cases: in every $\mathcal{I}$, $a^{\mathcal{I}} \in C^{\mathcal{I}}$ or $a^{\mathcal{I}} \in (\neg C)^{\mathcal{I}}$

I: $a^{\mathcal{I}} \in C^{\mathcal{I}}$, thus $a^{\mathcal{I}} \in \mathsf{D}^{\mathcal{I}}$, thus $a^{\mathcal{I}} \in \mathsf{F}^{\mathcal{I}}$, thus $a^{\mathcal{I}} \in \mathsf{X}^{\mathcal{I}}$

II: $a^{\mathcal{I}} \in (\neg C)^{\mathcal{I}}$, thus $a^{\mathcal{I}} \in \mathsf{X}^{\mathcal{I}}$

Local GCI Rule: Bigger Problem

$\mathcal{T} \equiv \{C \sqsubseteq \mathsf{D}, \quad \mathsf{D} \sqsubseteq \mathsf{F}, \quad \neg C \sqcup \mathsf{F} \sqsubseteq \mathsf{X}\}$
$\mathcal{A} \equiv \{a : \mathsf{A}\}$

Question: are there any entailments for a?

$(\mathcal{T}, \mathcal{A}) \models a : \mathsf{X}$, can prove inconsistency of $(\mathcal{T}, \mathcal{A} \cup \{a : \neg\mathsf{X}\})$?

Not really, stuck with $L(a) = \{\mathsf{A}\}$

Reasoning by cases: in every $\mathcal{I}$, $a^{\mathcal{I}} \in C^{\mathcal{I}}$ or $a^{\mathcal{I}} \in (\neg C)^{\mathcal{I}}$

I: $a^{\mathcal{I}} \in C^{\mathcal{I}}$, thus $a^{\mathcal{I}} \in \mathsf{D}^{\mathcal{I}}$, thus $a^{\mathcal{I}} \in \mathsf{F}^{\mathcal{I}}$, thus $a^{\mathcal{I}} \in \mathsf{X}^{\mathcal{I}}$

II: $a^{\mathcal{I}} \in (\neg C)^{\mathcal{I}}$, thus $a^{\mathcal{I}} \in \mathsf{X}^{\mathcal{I}}$

Lesson: GCIs can be inherently global and apply to all individuals

Global GCI Rule

Instead we use the following global GCI rule:

GCI-rule if $C \sqsubseteq D \in \mathcal{T}$ and $\dot{\neg} C \sqcup D \notin L(a)$ for some a, then add $\dot{\neg} C \sqcup D$ to $L(a)$

Global GCI Rule

Instead we use the following global GCI rule:

GCI-rule if $C \sqsubseteq D \in \mathcal{T}$ and $\dot{\neg} C \sqcup D \notin L(a)$ for some a,
then add $\dot{\neg} C \sqcup D$ to $L(a)$

The GCI-rule adds a disjunction per individual and GCI. This is:

- Bad due to massive non-determinism
- Stupid for GCIs with a concept name on its left hand side

Global GCI Rule

Instead we use the following global GCI rule:

GCI-rule if $C \sqsubseteq D \in \mathcal{T}$ and $\dot{\neg} C \sqcup D \notin L(a)$ for some a,
then add $\dot{\neg} C \sqcup D$ to $L(a)$

The GCI-rule adds a disjunction per individual and GCI. This is:

- Bad due to massive non-determinism
- Stupid for GCIs with a concept name on its left hand side

We retain the local rule :

GCI_0-rule if $\mathsf{A} \sqsubseteq C \in \mathcal{T}$ and $\mathsf{A} \in L(a)$ and $C \notin L(a)$ for some a,
then add C to $L(a)$

Ontology Consistency: Example

$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$

$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$

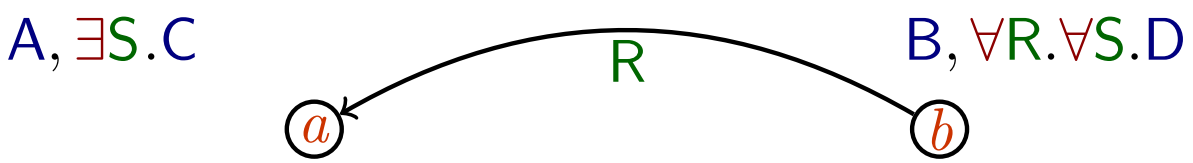
Ontology Consistency: Example

$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$

$\mathcal{G}_0$

Ontology Consistency: Example

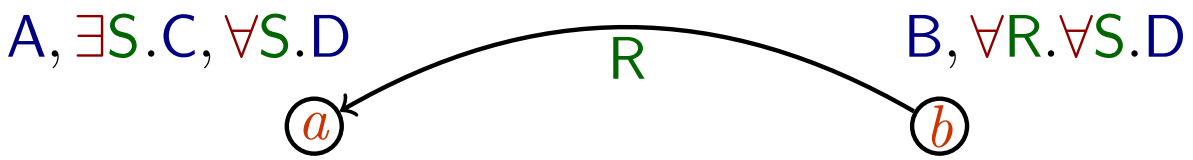
$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



$\mathcal{G}_0$

Ontology Consistency: Example

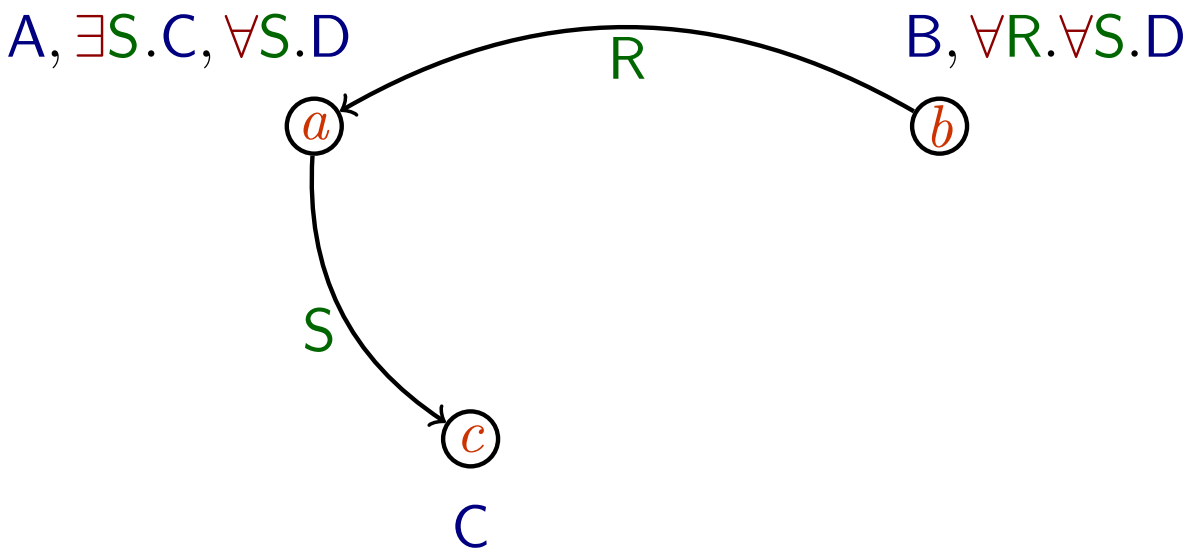
$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



$\mathcal{G}_0$

Ontology Consistency: Example

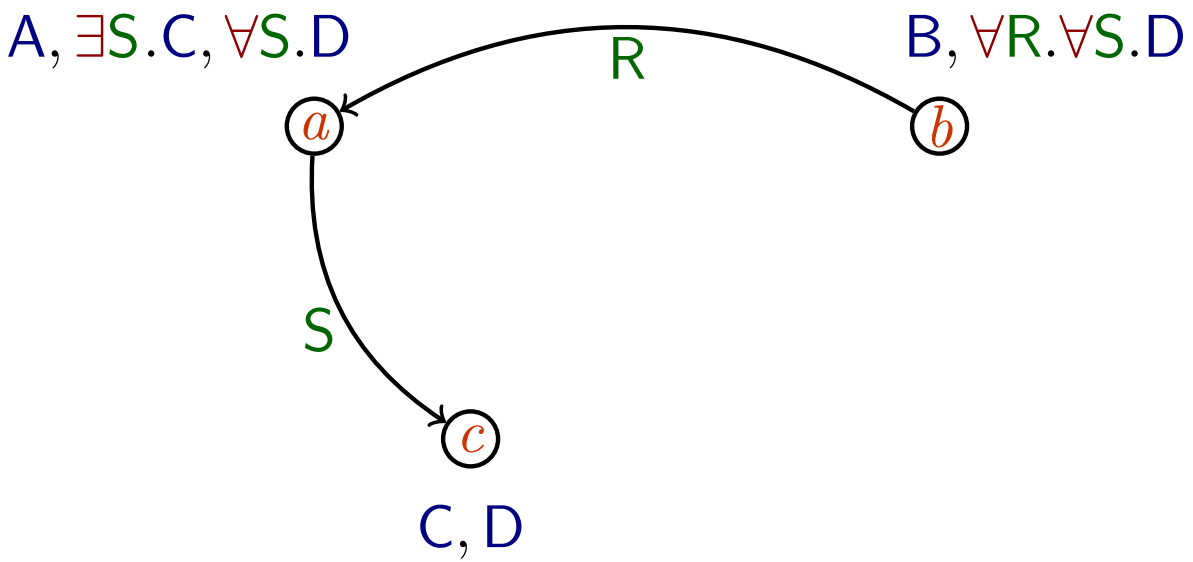
$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



$\mathcal{G}_0$

Ontology Consistency: Example

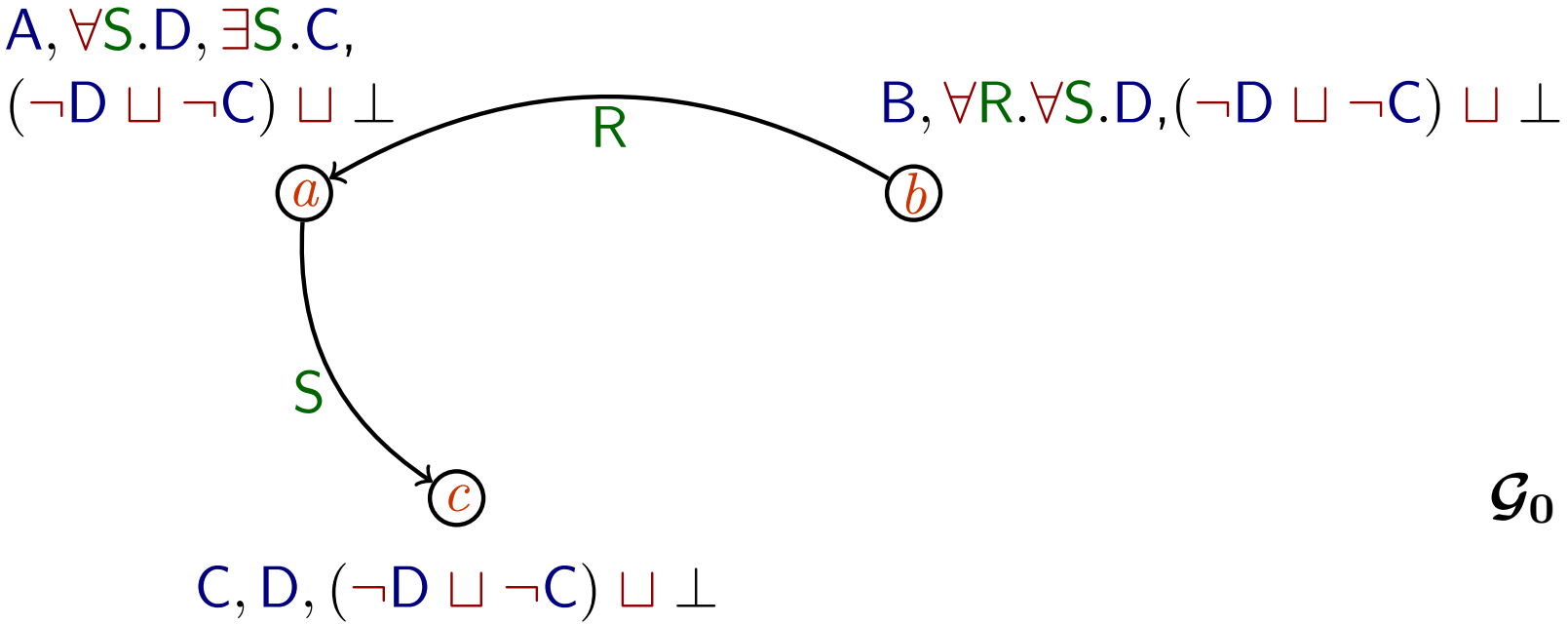
$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



$\mathcal{G}_0$

Ontology Consistency: Example

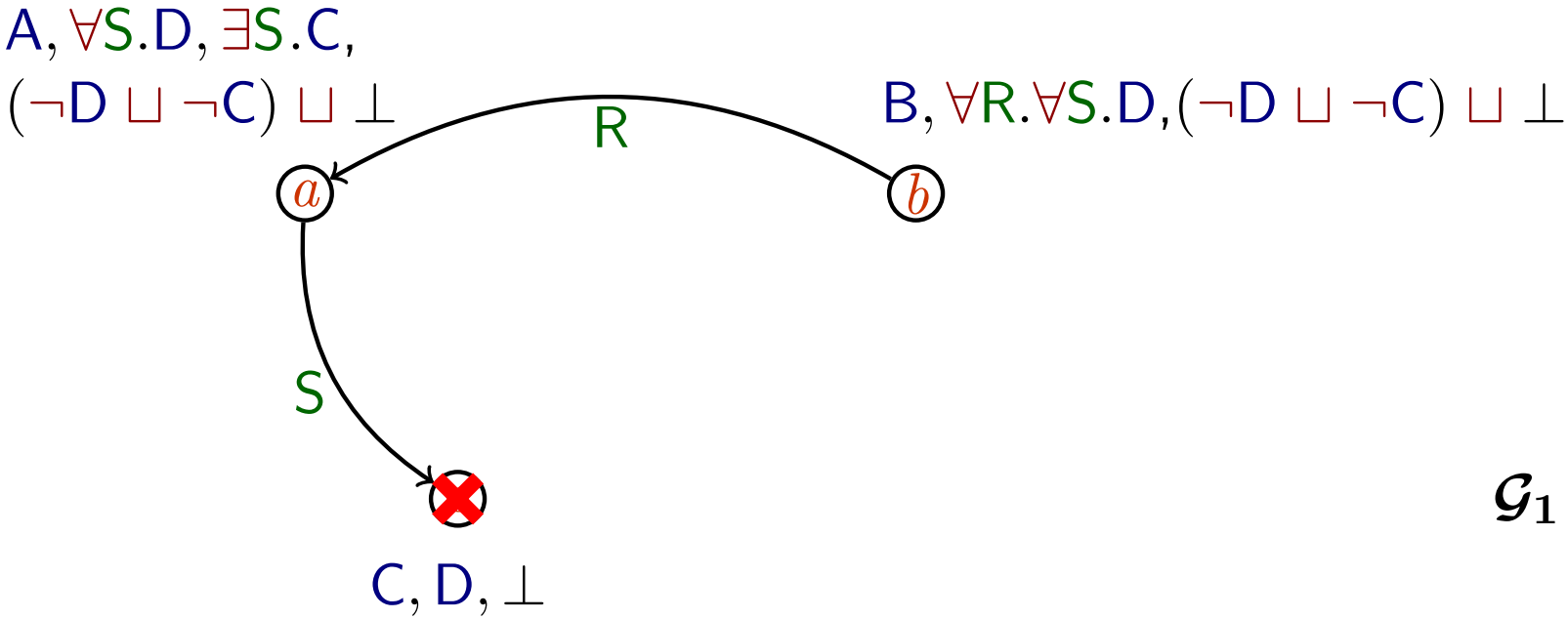
$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



$\mathcal{G}_0$

Ontology Consistency: Example

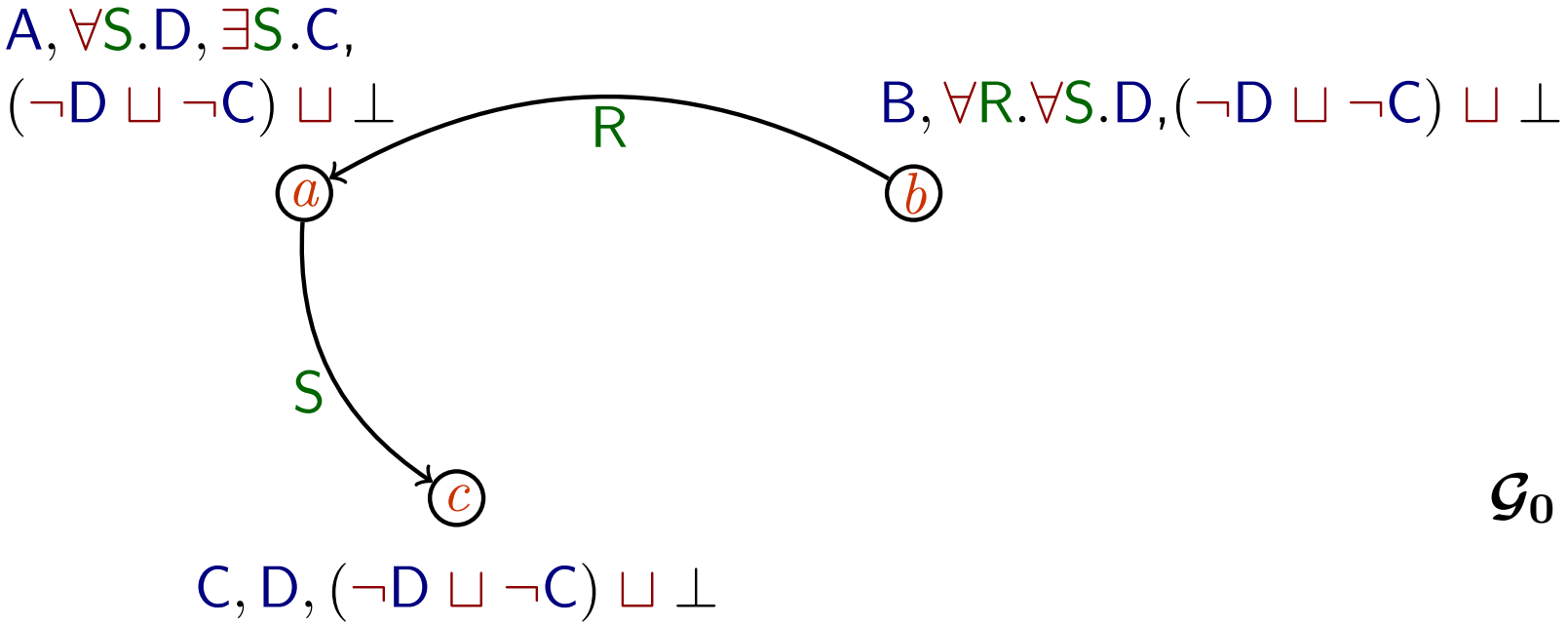
$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



$\mathcal{G}_1$

Ontology Consistency: Example

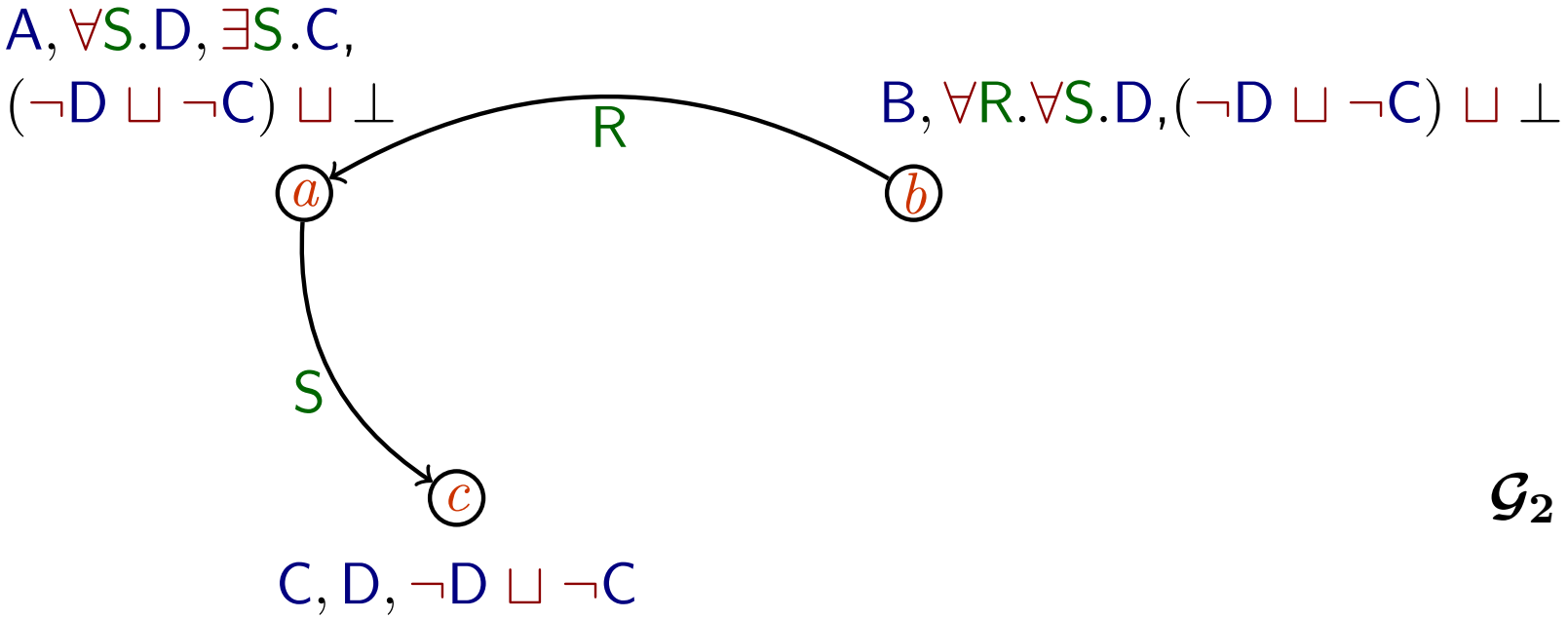
$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



$\mathcal{G}_0$

Ontology Consistency: Example

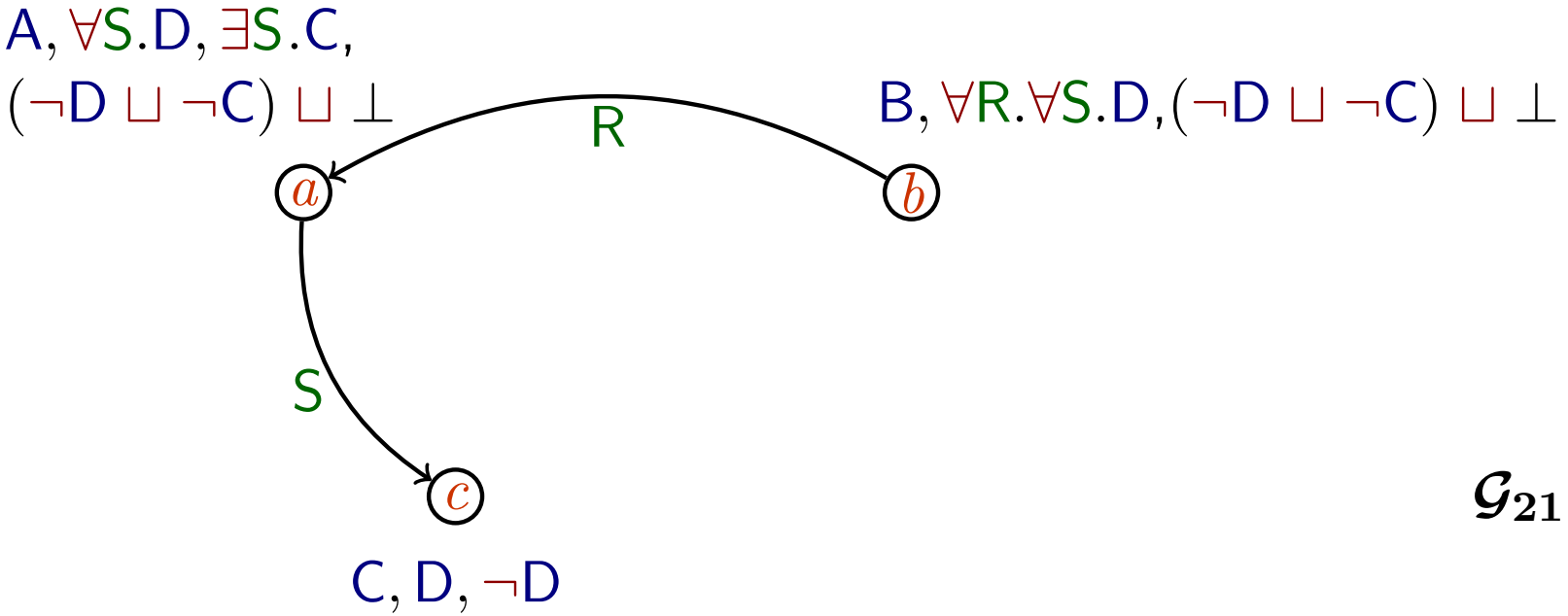
$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



$\mathcal{G}_2$

Ontology Consistency: Example

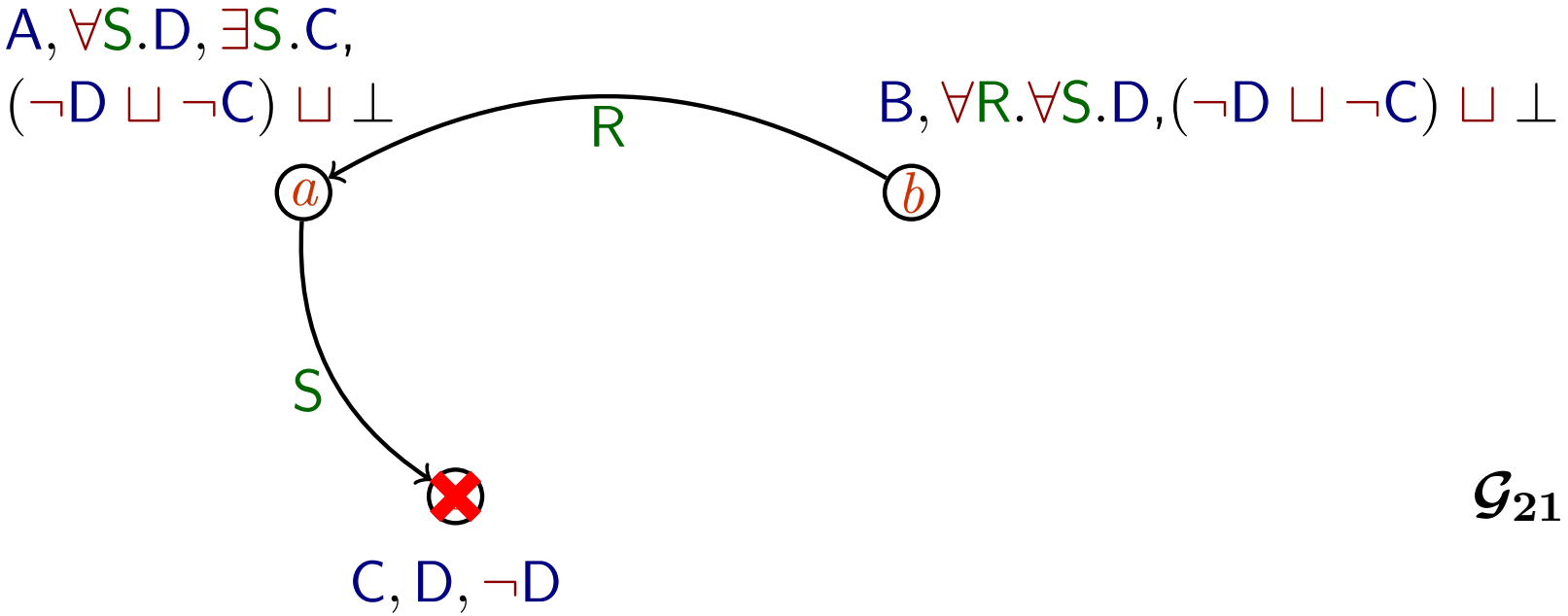
$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



$\mathcal{G}_{21}$

Ontology Consistency: Example

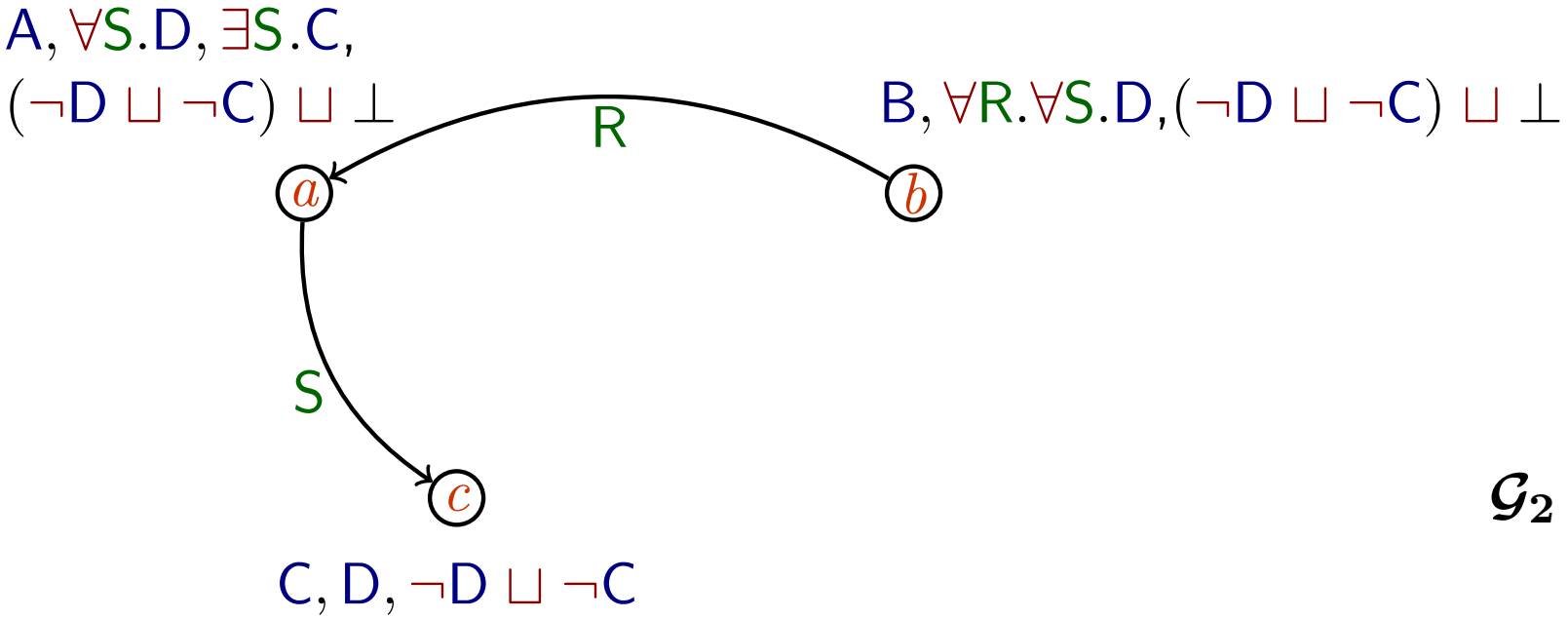
$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



$\mathcal{G}_{21}$

Ontology Consistency: Example

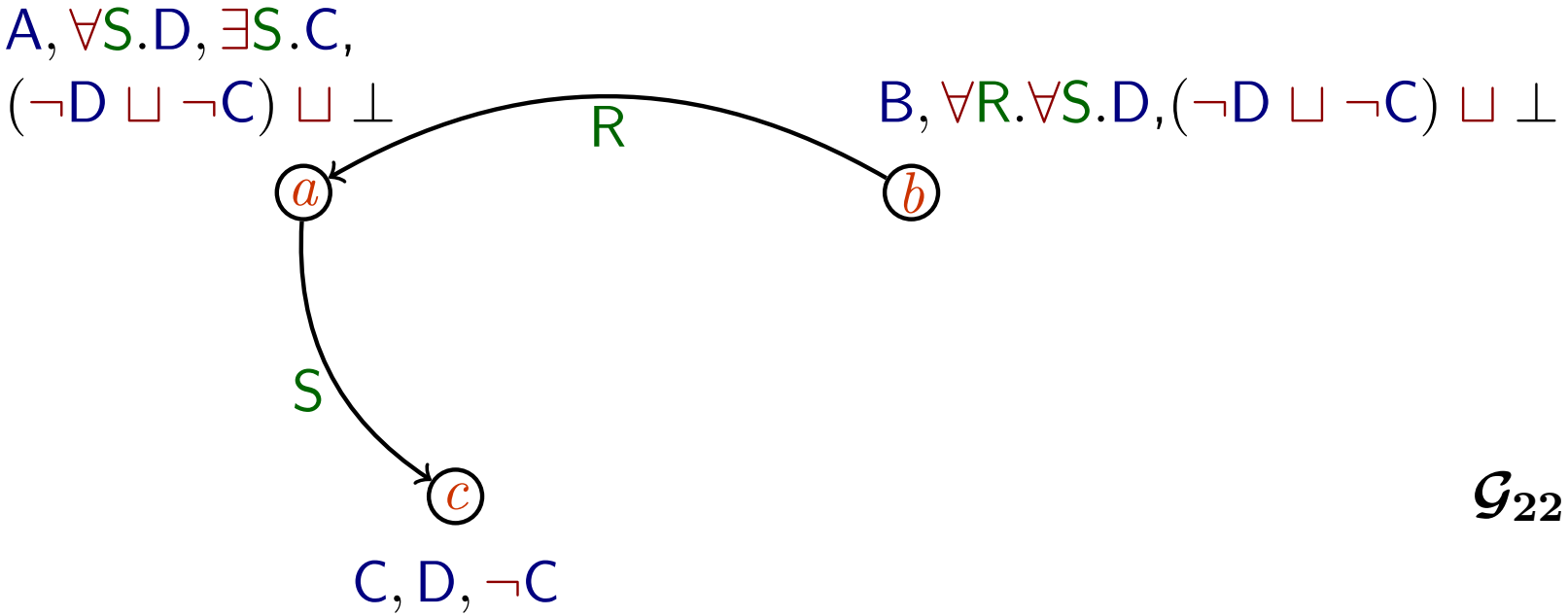
$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



$\mathcal{G}_2$

Ontology Consistency: Example

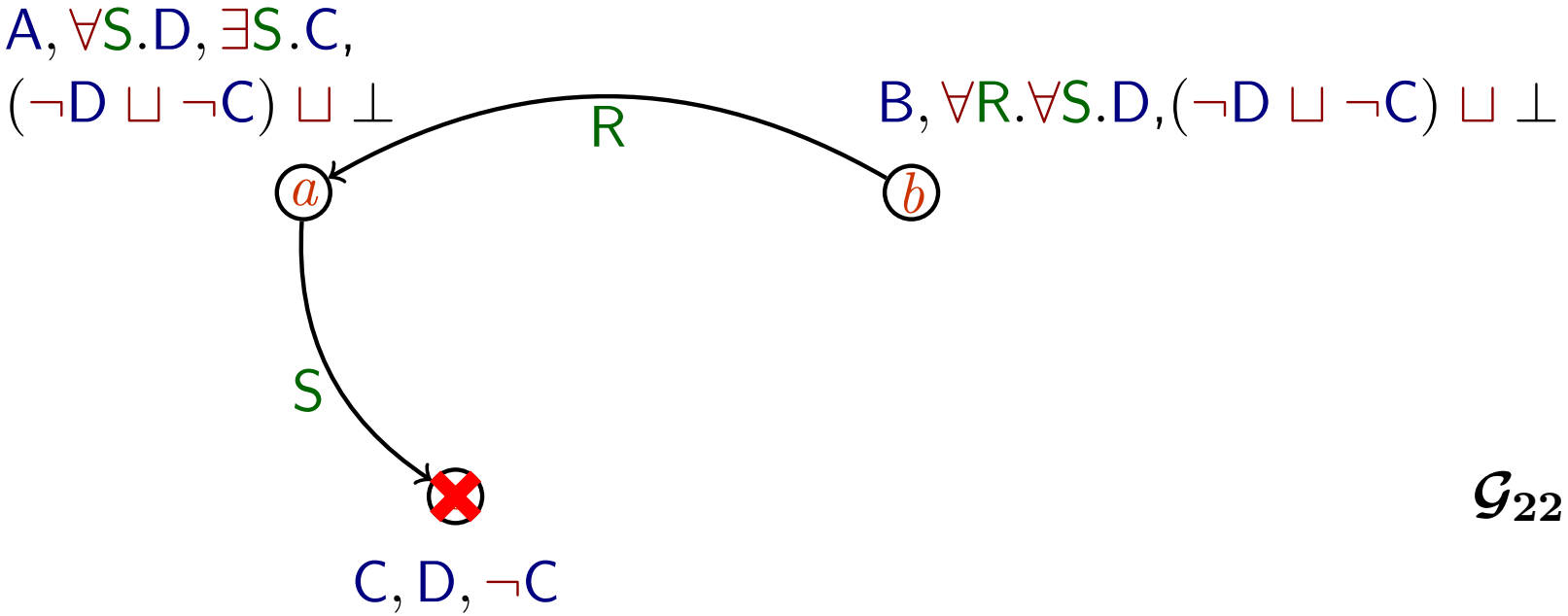
$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



$\mathcal{G}_{22}$

Ontology Consistency: Example

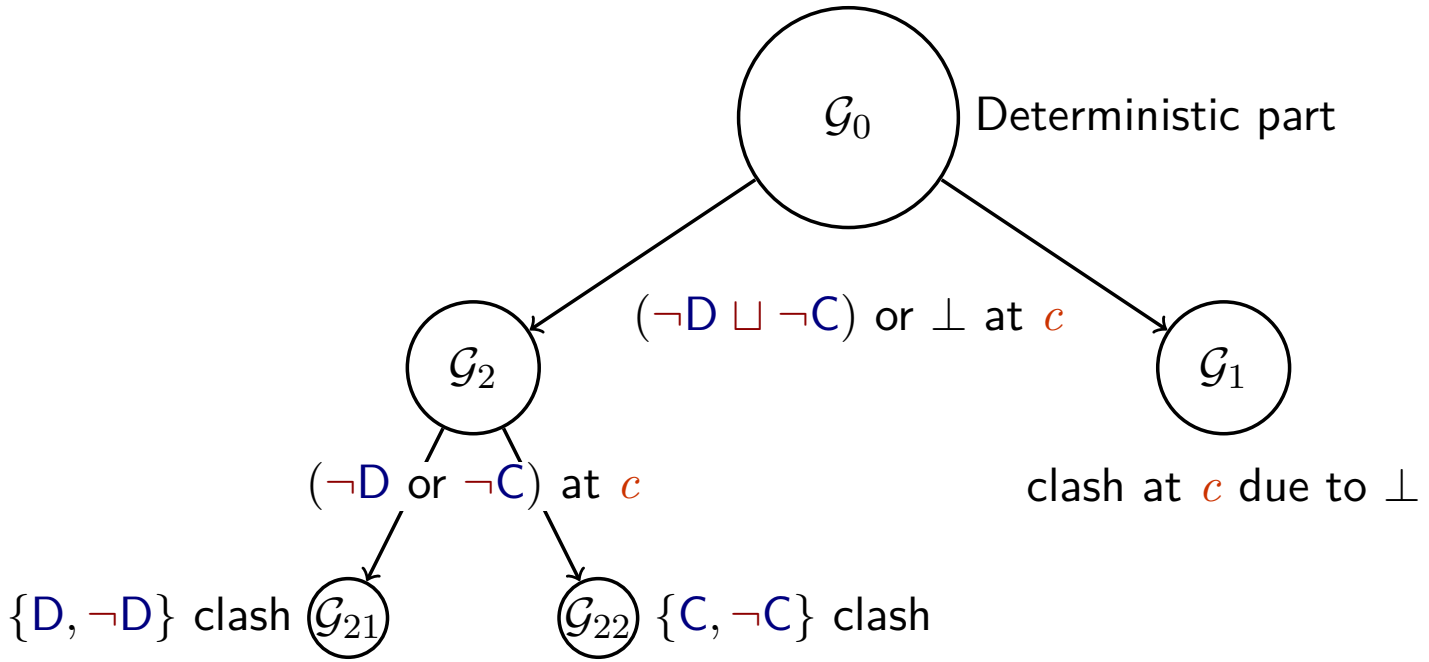
$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



$\mathcal{G}_{22}$

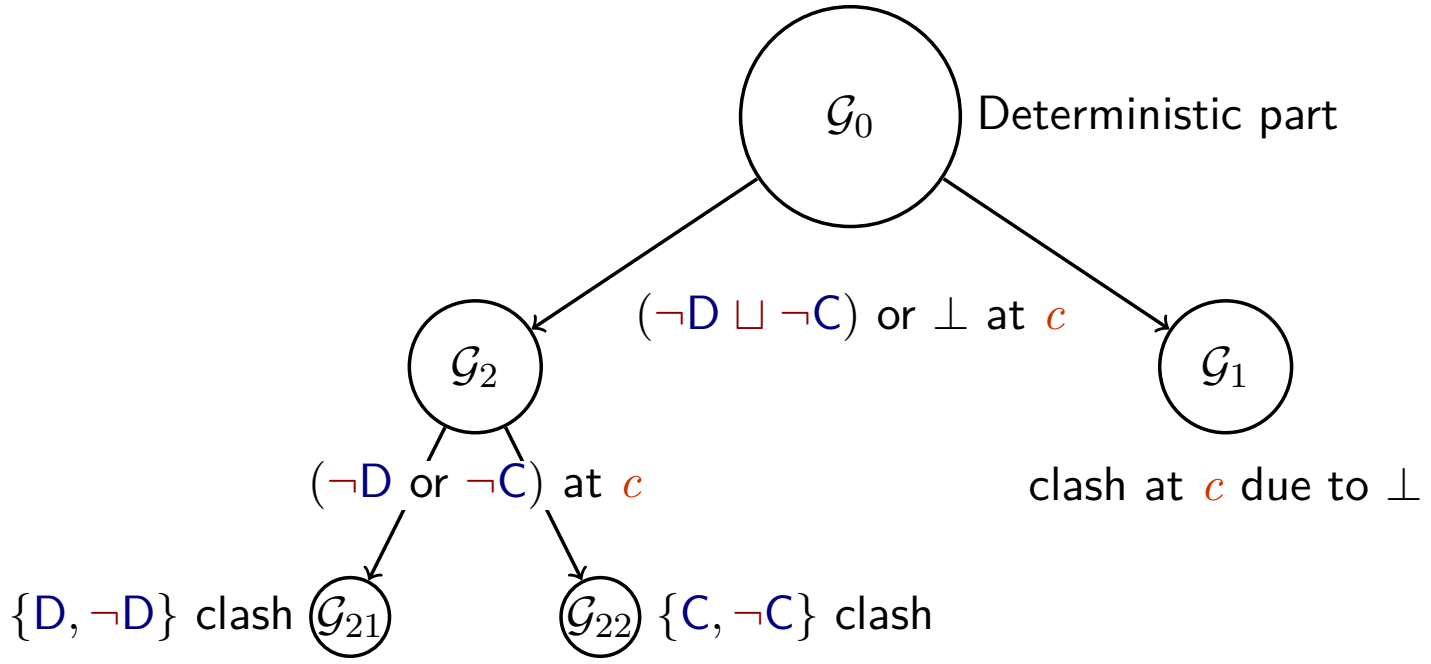
It's All About Choices...

$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



It's All About Choices...

$\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{S}.\mathsf{C}, \quad \mathsf{B} \sqsubseteq \forall \mathsf{R}.\forall \mathsf{S}.\mathsf{D}, \quad \mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot\}$
$\mathcal{A} \equiv \{a : \mathsf{A}, \quad b : \mathsf{B}, \quad (b, a) : \mathsf{R}\}$



The space of choices explodes huge and very-very fast (lots of backtracking)

Didn't We Lose Anything?..

Question: did we retain soundness, completeness, and termination?

Didn't We Lose Anything?..

Question: did we retain soundness, completeness, and termination?

Cyclic schema: $\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{R}.\mathsf{A}\}, \quad \mathcal{A} \equiv \{a\colon \mathsf{A}, b\colon \mathsf{B}\}$

Didn't We Lose Anything?..

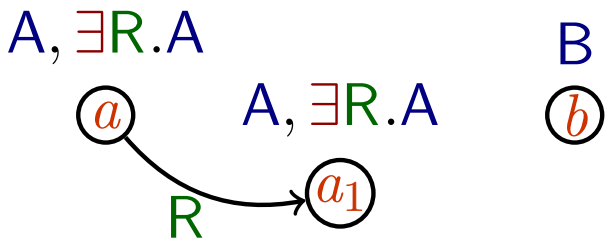
Question: did we retain soundness, completeness, and termination?

Cyclic schema: $\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{R}.\mathsf{A}\}, \quad \mathcal{A} \equiv \{a\colon \mathsf{A}, b\colon \mathsf{B}\}$

Didn't We Lose Anything?..

Question: did we retain soundness, completeness, and termination?

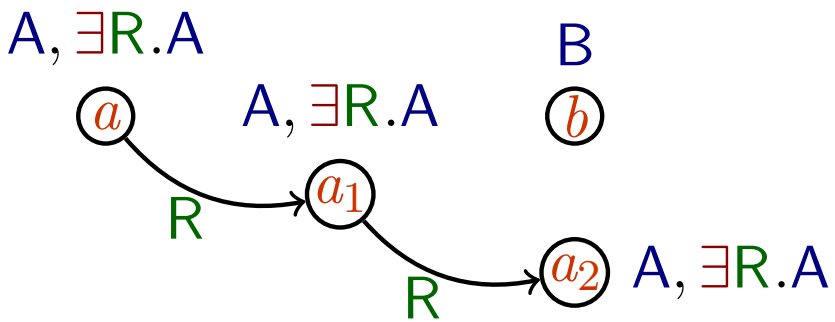
Cyclic schema: $\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{R}.\mathsf{A}\}, \quad \mathcal{A} \equiv \{a\colon \mathsf{A}, b\colon \mathsf{B}\}$



Didn't We Lose Anything?..

Question: did we retain soundness, completeness, and termination?

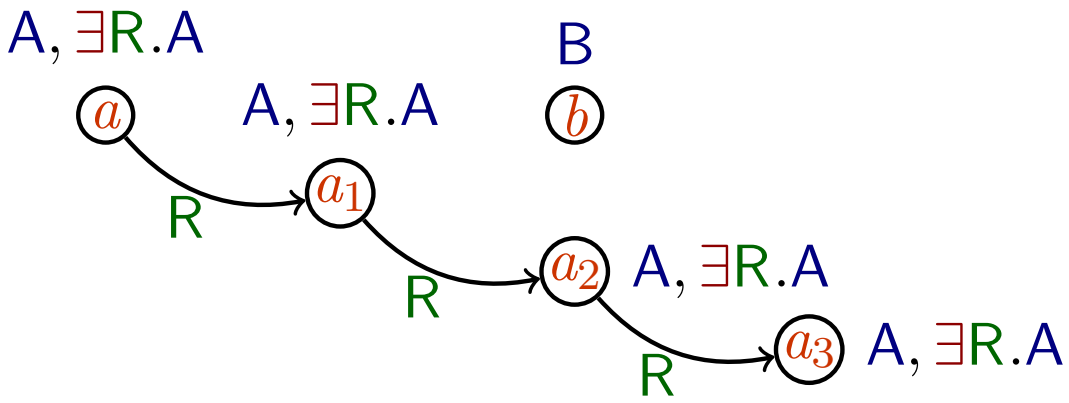
Cyclic schema: $\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{R}.\mathsf{A}\}, \quad \mathcal{A} \equiv \{a\colon \mathsf{A}, b\colon \mathsf{B}\}$



Didn't We Lose Anything?..

Question: did we retain soundness, completeness, and termination?

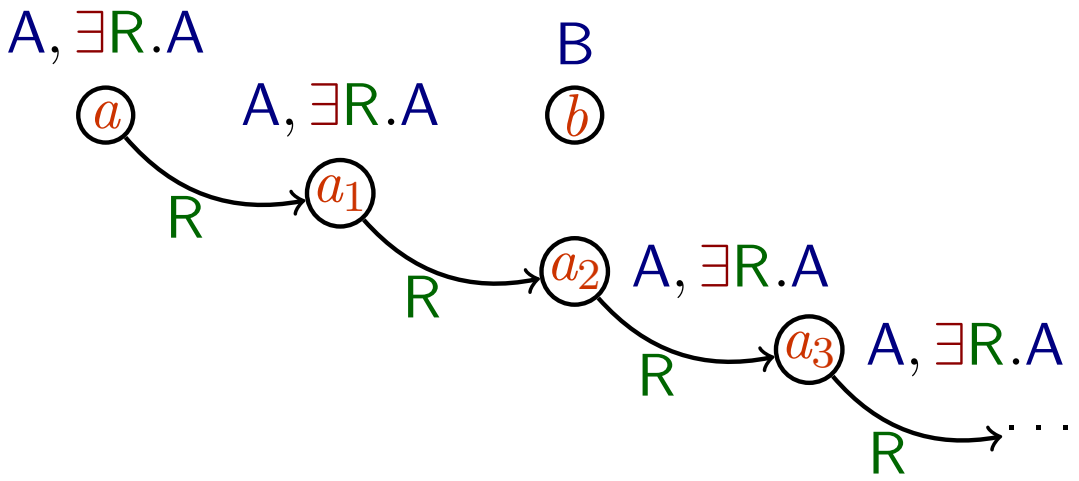
Cyclic schema: $\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{R}.\mathsf{A}\}, \quad \mathcal{A} \equiv \{a\colon \mathsf{A}, b\colon \mathsf{B}\}$



Didn't We Lose Anything?..

Question: did we retain soundness, completeness, and termination?

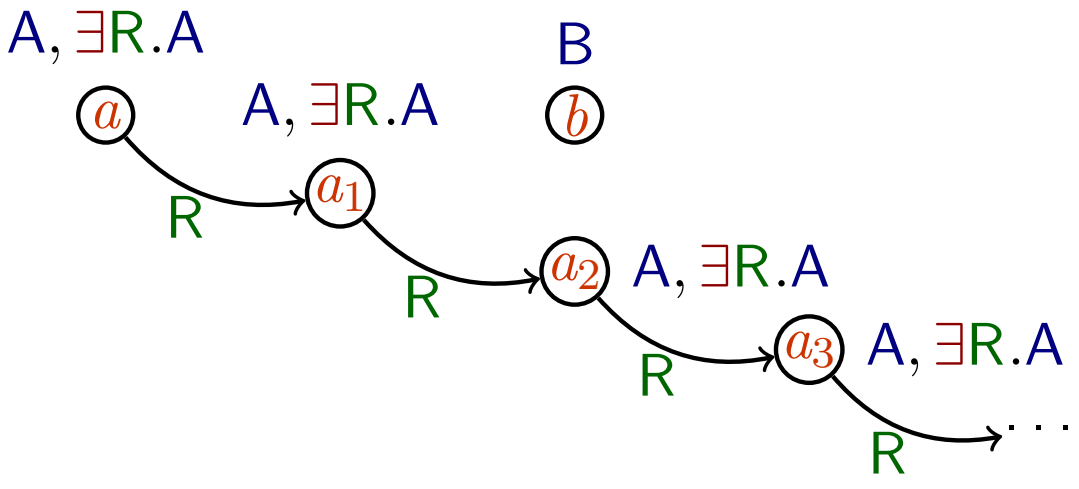
Cyclic schema: $\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{R}.\mathsf{A}\}, \quad \mathcal{A} \equiv \{a\colon \mathsf{A}, b\colon \mathsf{B}\}$



Didn't We Lose Anything?..

Question: did we retain soundness, completeness, and termination?

Cyclic schema: $\mathcal{T} \equiv \{\mathsf{A} \sqsubseteq \exists \mathsf{R}.\mathsf{A}\}, \quad \mathcal{A} \equiv \{a\colon \mathsf{A}, b\colon \mathsf{B}\}$



The GCI propagates $\exists \mathsf{R}.\mathsf{A}$ along the infinite chain

This expansion does not terminate on cyclic TBoxes

Blocking

Need to block infinite duplication of individuals

Blocking

Need to block infinite duplication of individuals

Blocking: each rule is applicable to a if there is no node b s.t. $L(a) \subseteq L(b)$

Blocking

Need to block infinite duplication of individuals

Blocking: each rule is applicable to a if there is no node b s.t. $L(a) \subseteq L(b)$

In case we have

- a freshly created node a,
- another node b s.t.:
  - $L(a) \subseteq L(b)$
  - b is older than a

$\Rightarrow$ node b blocks node a

Blocking

Need to block infinite duplication of individuals

Blocking: each rule is applicable to a if there is no node b s.t. $L(a) \subseteq L(b)$

In case we have

- a freshly created node a,
- another node b s.t.:
  - $L(a) \subseteq L(b)$
  - b is older than a

$\Rightarrow$ node b blocks node a

$\sqcap$-rule: if $\exists R.C \in L(a)$ for some non-blocked a and there is no b s.t. $C \in L(b)$ and an R-edge from a to b is in $\mathcal{G}$

then create a new node b, add an R-edge from a to b and add C to $L(b)$

Theoretical Properties

The tableau algorithm with the global GCI rule and blocking is:

- Sound
- Complete
- Terminating

for standard reasoning problems in $\mathcal{ALC}$ (w.r.t. general TBoxes)

Theoretical Properties

The tableau algorithm with the global GCI rule and blocking is:

- Sound
- Complete
- Terminating

for standard reasoning problems in $\mathcal{ALC}$ (w.r.t. general TBoxes)

Complexity is trickier:

- No longer in PSPACE (exponentially long chains may be generated before blocking)
- It's actually in EXPSPACE while the problem is in EXPTIME

Theoretical Properties

The tableau algorithm with the global GCI rule and blocking is:

- Sound
- Complete
- Terminating

for standard reasoning problems in $\mathcal{ALC}$ (w.r.t. general TBoxes)

Complexity is trickier:

- No longer in PSPACE (exponentially long chains may be generated before blocking)
- It's actually in EXPSPACE while the problem is in EXPTIME

Implemented in FaCT++, RacerPro, Pellet, Konclude,...

Reminder: Reasoning Problems in DLs

$\mathcal{ALC}$ Tableau with TBoxes

Two Essential Optimizations

Non-Determinism

Disjunctive non-determinism is the major source of intractability

- Disjunction
- GCIs

Non-Determinism

Disjunctive non-determinism is the major source of intractability

- Disjunction
- GCIs

Non-determinism is an inherent feature of the logic!

$\mathcal{O} \models a\colon (D_1 \sqcup D_2)$ does not mean $\mathcal{O} \models a\colon D_1$ or $\mathcal{O} \models a\colon D_2$

($\mathcal{ALC}$ is non-convex or non-Horn)

Non-Determinism

Disjunctive non-determinism is the major source of intractability

- Disjunction
- GCIs

Non-determinism is an inherent feature of the logic!

$\mathcal{O} \models a : (D_1 \sqcup D_2)$ does not mean $\mathcal{O} \models a : D_1$ or $\mathcal{O} \models a : D_2$

($\mathcal{ALC}$ is non-convex or non-Horn)

Two essential optimizations:

- Backjumping: being smarter about non-determinism
- Absorption: reducing non-determinism

Absorption of General Concept Inclusions

Remember: for $\mathcal{T} = \{C_i \sqsubseteq D_i \mid 1 \leq i \leq n\}$,
where no C_i is a concept name,

each individual x will have n disjunctions $x\colon (\dot{\neg} C_i \sqcup D_i)$

Absorption of General Concept Inclusions

Remember: for $\mathcal{T} = \{C_i \sqsubseteq D_i \mid 1 \leq i \leq n\}$,
where no C_i is a concept name,
each individual x will have n disjunctions $x\colon (\dot{\neg} C_i \sqcup D_i)$

GCI$_0$-rule: if $\mathsf{A} \in L(a)$ for some non-blocked a
and $\mathsf{A} \sqsubseteq D \in \mathcal{T}$ and $D \notin L(a)$

then add D to $L(a)$

GCI-rule if $C \sqsubseteq D \in \mathcal{T}$
and $\dot{\neg} C \sqcup D \notin L(a)$ for some non-blocked a,

then add $\dot{\neg} C \sqcup D$ to $L(a)$

Absorption of General Concept Inclusions

Remember: for $\mathcal{T} = \{C_i \sqsubseteq D_i \mid 1 \leq i \leq n\}$,
where no C_i is a concept name,
each individual x will have n disjunctions $x\colon (\dot{\neg} C_i \sqcup D_i)$

GCI$_0$-rule: if $\mathsf{A} \in L(a)$ for some non-blocked a
and $\mathsf{A} \sqsubseteq D \in \mathcal{T}$ and $D \notin L(a)$

then add D to $L(a)$

GCI-rule if $C \sqsubseteq D \in \mathcal{T}$
and $\dot{\neg} C \sqcup D \notin L(a)$ for some non-blocked a,

then add $\dot{\neg} C \sqcup D$ to $L(a)$

Observation: many GCIs are of the form $\mathsf{A} \sqcap X_1 \sqcap X_2 \sqcap \ldots \sqsubseteq C$
for concept name A and arbitrary concepts X_i
e.g., $\mathsf{Human} \sqcap \ldots \sqsubseteq C$ or $\mathsf{Device} \sqcap \ldots \sqsubseteq C$

Absorption of General Concept Inclusions

Localise GCIs to concept names by rewriting $\mathsf{A} \sqcap X \sqsubseteq C$ into equivalent $\mathsf{A} \sqsubseteq \neg X \sqcup C$

Absorption of General Concept Inclusions

Localise GCIs to concept names by rewriting $\mathsf{A} \sqcap X \sqsubseteq C$ into equivalent $\mathsf{A} \sqsubseteq \neg X \sqcup C$

$\mathsf{Human} \sqcap \exists \mathsf{owns}.\mathsf{Dog} \sqsubseteq \mathsf{PetOwner}$

becomes

$\mathsf{Human} \sqsubseteq \neg\exists \mathsf{owns}.\mathsf{Dog} \sqcup \mathsf{PetOwner}$

Absorption of General Concept Inclusions

Localise GCIs to concept names by rewriting $A \sqcap X \sqsubseteq C$ into equivalent $A \sqsubseteq \neg X \sqcup C$

$\mathsf{Human} \sqcap \exists \mathsf{owns}.\mathsf{Dog} \sqsubseteq \mathsf{PetOwner}$

becomes

$\mathsf{Human} \sqsubseteq \neg\exists \mathsf{owns}.\mathsf{Dog} \sqcup \mathsf{PetOwner}$

Due to absorption the local GCI rule applies more often and the global, non-deterministic one less often

Absorption of General Concept Inclusions

Localise GCIs to concept names by rewriting $A \sqcap X \sqsubseteq C$ into equivalent $A \sqsubseteq \neg X \sqcup C$

$\mathsf{Human} \sqcap \exists \mathsf{owns}.\mathsf{Dog} \sqsubseteq \mathsf{PetOwner}$

becomes

$\mathsf{Human} \sqsubseteq \neg\exists \mathsf{owns}.\mathsf{Dog} \sqcup \mathsf{PetOwner}$

Due to absorption the local GCI rule applies more often and the global, non-deterministic one less often

In the best case nearly all GCIs are absorbed

In the worst case nothing changes

Absorption: Example

$\mathsf{Human} \sqcap \exists \mathsf{owns}.\mathsf{Dog} \sqsubseteq \mathsf{PetOwner}$ becomes

$\mathsf{Human} \sqsubseteq \neg\exists \mathsf{owns}.\mathsf{Dog} \sqcup \mathsf{PetOwner}$

Consider some node x in some $\mathcal{G}$:

Absorption: Example

$\mathsf{Human} \sqcap \exists \mathsf{owns}.\mathsf{Dog} \sqsubseteq \mathsf{PetOwner}$ becomes

$\mathsf{Human} \sqsubseteq \neg\exists \mathsf{owns}.\mathsf{Dog} \sqcup \mathsf{PetOwner}$

Consider some node x in some $\mathcal{G}$:

- $\mathsf{Human} \notin L(x)$ (most of the nodes) $\Rightarrow$ no GCI rule applies
  vs. adding $\neg\mathsf{Human} \sqcup \neg\exists \mathsf{owns}.\mathsf{Dog} \sqcup \mathsf{PetOwner}$ – 3 choices

Absorption: Example

$\mathsf{Human} \sqcap \exists \mathsf{owns}.\mathsf{Dog} \sqsubseteq \mathsf{PetOwner}$ becomes

$\mathsf{Human} \sqsubseteq \neg\exists \mathsf{owns}.\mathsf{Dog} \sqcup \mathsf{PetOwner}$

Consider some node x in some $\mathcal{G}$:

- $\mathsf{Human} \notin L(x)$ (most of the nodes) $\Rightarrow$ no GCI rule applies
  vs. adding $\neg\mathsf{Human} \sqcup \neg\exists \mathsf{owns}.\mathsf{Dog} \sqcup \mathsf{PetOwner}$ – 3 choices
- $\mathsf{Human} \in L(x) \Rightarrow$ adding $\neg\exists \mathsf{owns}.\mathsf{Dog} \sqcup \mathsf{PetOwner}$ – 2 choices
  vs. adding $\neg\mathsf{Human} \sqcup \neg\exists \mathsf{owns}.\mathsf{Dog} \sqcup \mathsf{PetOwner}$ – 3 choices

Absorption: Example

Human $\sqcap \exists$owns.Dog $\sqsubseteq$ PetOwner becomes

Human $\sqsubseteq \neg\exists$owns.Dog $\sqcup$ PetOwner

Consider some node x in some $\mathcal{G}$:

- Human $\notin L(x)$ (most of the nodes) $\Rightarrow$ no GCI rule applies
  vs. adding $\neg$Human $\sqcup \neg\exists$owns.Dog $\sqcup$ PetOwner – 3 choices
- Human $\in L(x) \Rightarrow$ adding $\neg\exists$owns.Dog $\sqcup$ PetOwner – 2 choices
  vs. adding $\neg$Human $\sqcup \neg\exists$owns.Dog $\sqcup$ PetOwner – 3 choices

Savings are exponential in the number of absorbed GCIs

Absorption: Example

Human $\sqcap \exists$owns.Dog $\sqsubseteq$ PetOwner becomes

Human $\sqsubseteq \neg\exists$owns.Dog $\sqcup$ PetOwner

Consider some node x in some $\mathcal{G}$:

- Human $\notin L(x)$ (most of the nodes) $\Rightarrow$ no GCI rule applies
  vs. adding $\neg$Human $\sqcup \neg\exists$owns.Dog $\sqcup$ PetOwner – 3 choices
- Human $\in L(x) \Rightarrow$ adding $\neg\exists$owns.Dog $\sqcup$ PetOwner – 2 choices
  vs. adding $\neg$Human $\sqcup \neg\exists$owns.Dog $\sqcup$ PetOwner – 3 choices

Savings are exponential in the number of absorbed GCIs

No real ontology with complex concepts on the left can be classified without absorption

Backjumping

Recall: when a clash is encountered, the non-deterministic algorithm backtracks i.e., returns to the last non-deterministic choice and tries the other possibility

Backjumping

Recall: when a clash is encountered, the non-deterministic algorithm backtracks i.e., returns to the last non-deterministic choice and tries the other possibility

It may re-discover the same clash again

Backjumping

Recall: when a clash is encountered, the non-deterministic algorithm backtracks i.e., returns to the last non-deterministic choice and tries the other possibility

It may re-discover the same clash again

$$L(x) = \{\exists R.(A \sqcap B) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A\}$$

Backjumping

Recall: when a clash is encountered, the non-deterministic algorithm backtracks i.e., returns to the last non-deterministic choice and tries the other possibility

It may re-discover the same clash again

$$L(x) = \{\exists R.(A \sqcap B) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A\}$$

The clash $\{A, \neg A\}$ is caused by the interaction between $\exists R.(A \sqcap B)$ and $\forall R.\neg A$

Backjumping

Recall: when a clash is encountered, the non-deterministic algorithm backtracks i.e., returns to the last non-deterministic choice and tries the other possibility

It may re-discover the same clash again

$$L(x) = \{\exists R.(A \sqcap B) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A\}$$

The clash $\{A, \neg A\}$ is caused by the interaction between $\exists R.(A \sqcap B)$ and $\forall R.\neg A$

This has nothing to do with massive branching on $C_i \sqcup D_i$

Backjumping

Recall: when a clash is encountered, the non-deterministic algorithm backtracks i.e., returns to the last non-deterministic choice and tries the other possibility

It may re-discover the same clash again

$$L(x) = \{\exists R.(A \sqcap B) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A\}$$

The clash $\{A, \neg A\}$ is caused by the interaction between $\exists R.(A \sqcap B)$ and $\forall R.\neg A$

This has nothing to do with massive branching on $C_i \sqcup D_i$

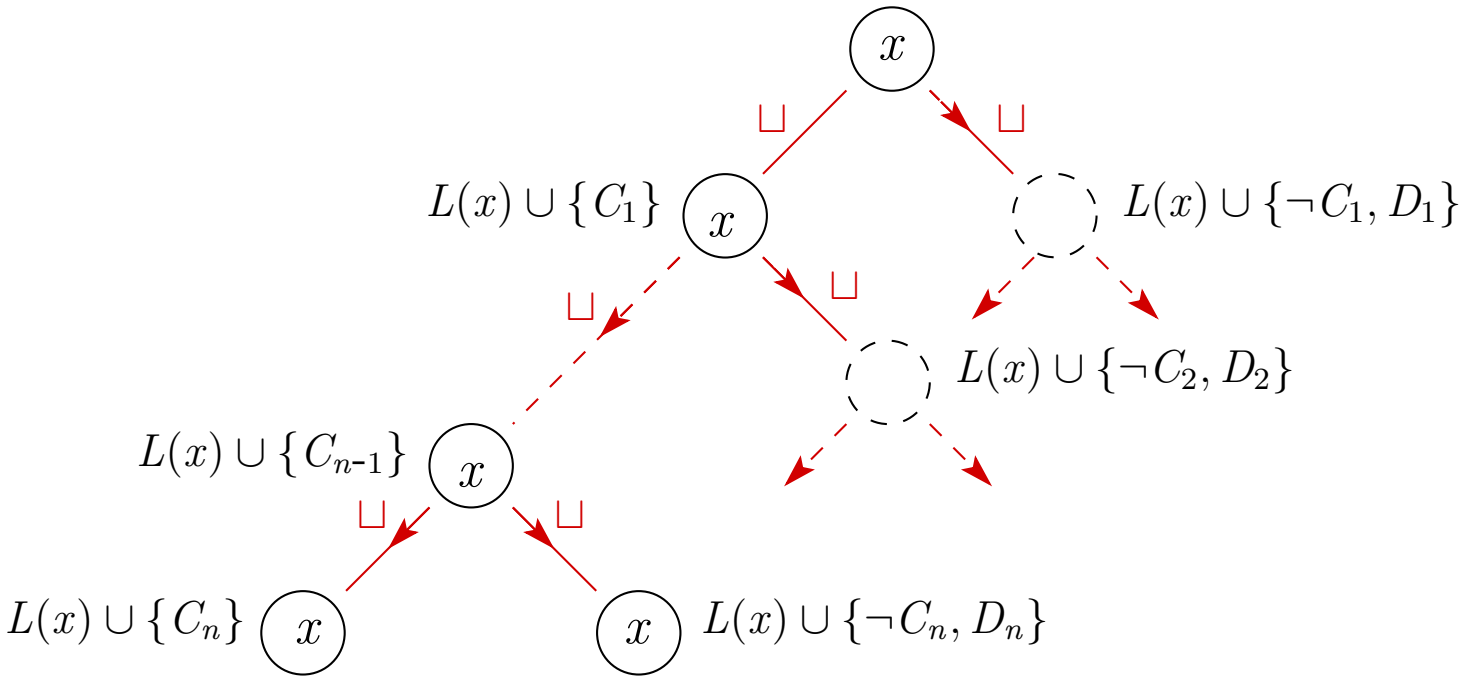
Backjumping attempts to recognize such interactions

Backjumping

$$L(x) = \{\exists R.(A \sqcap B) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A\}$$

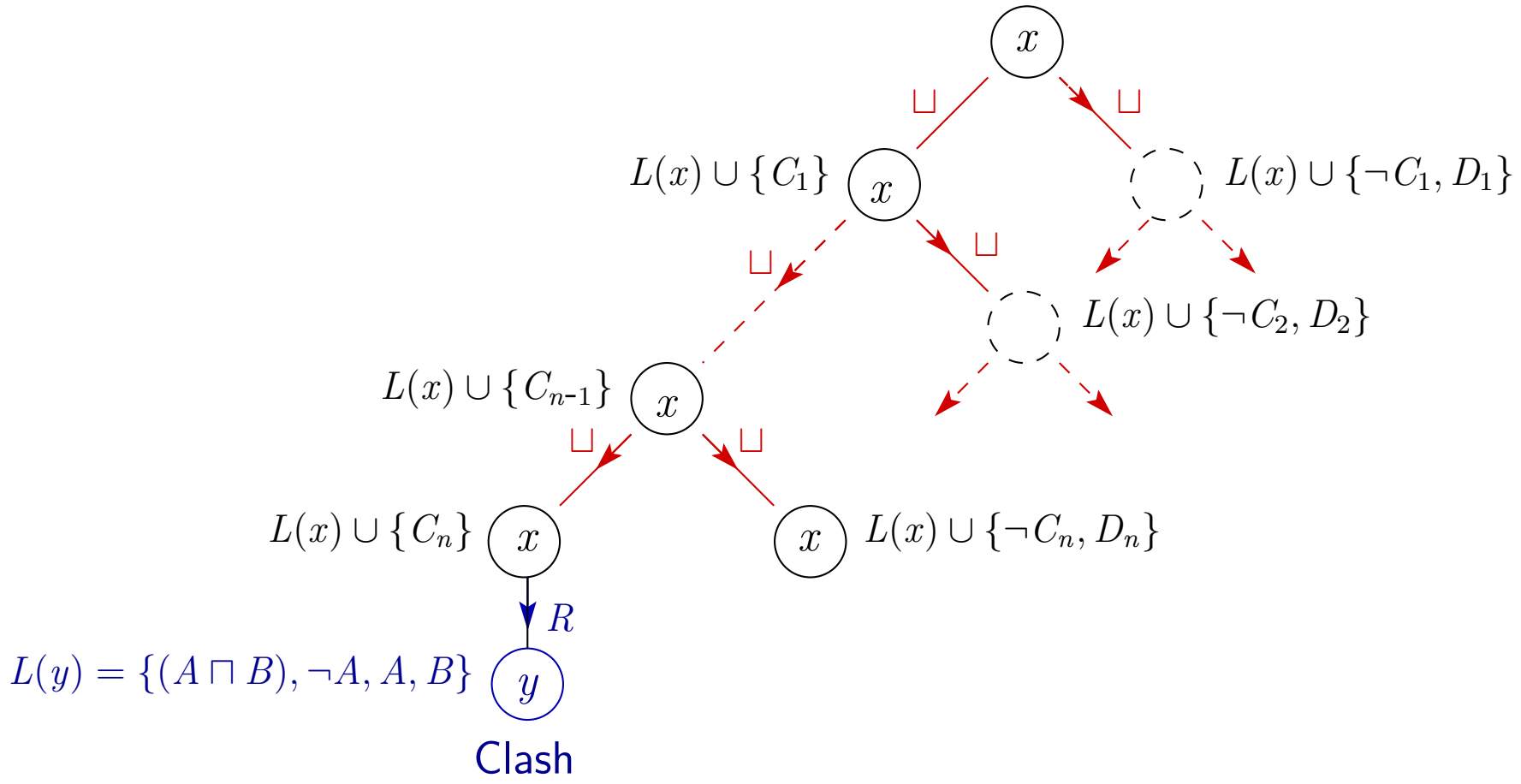
Backjumping

$$L(x) = \{\exists R.(A \sqcap B) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A\}$$



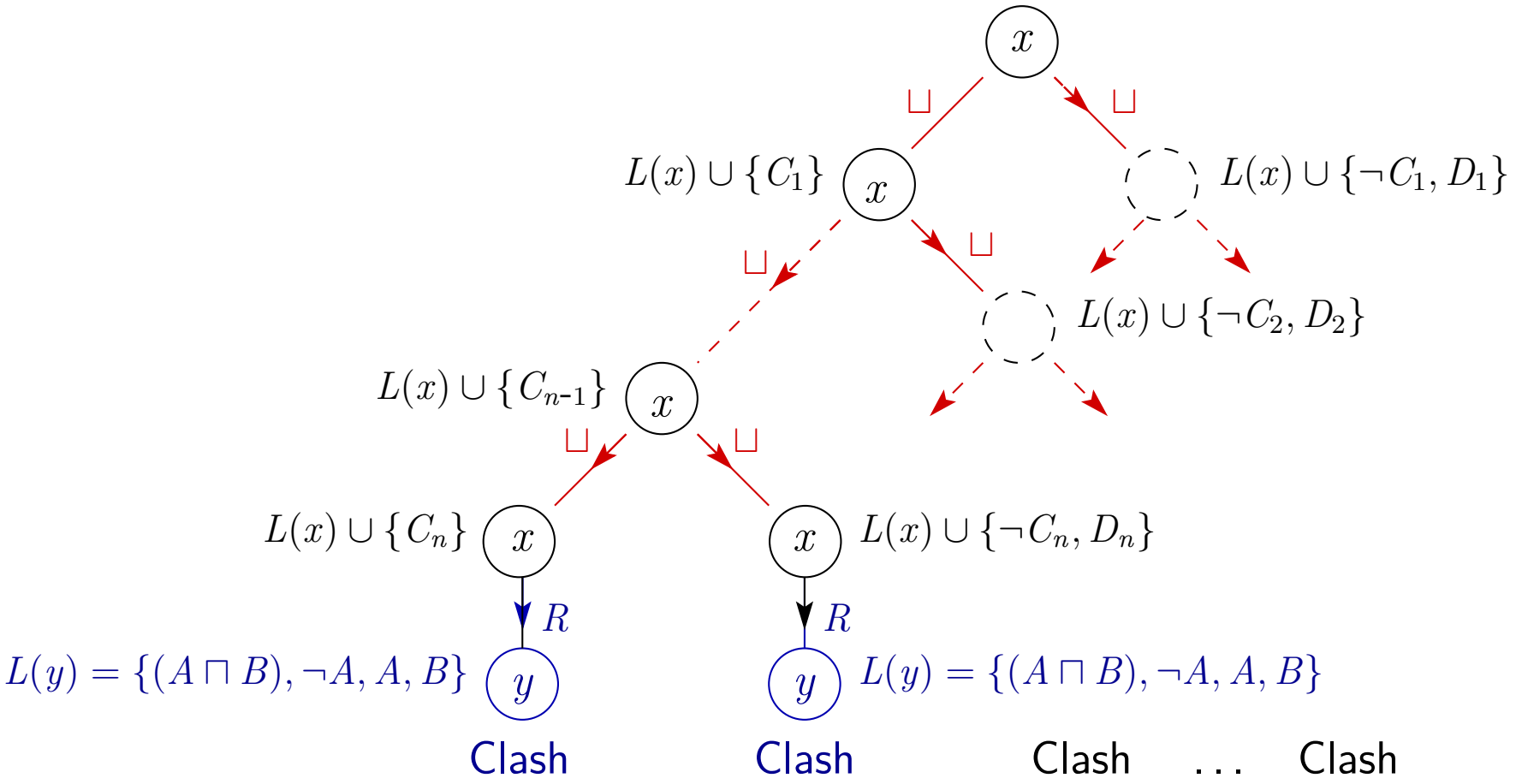
Backjumping

$$L(x) = \{\exists R.(A \sqcap B) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A\}$$



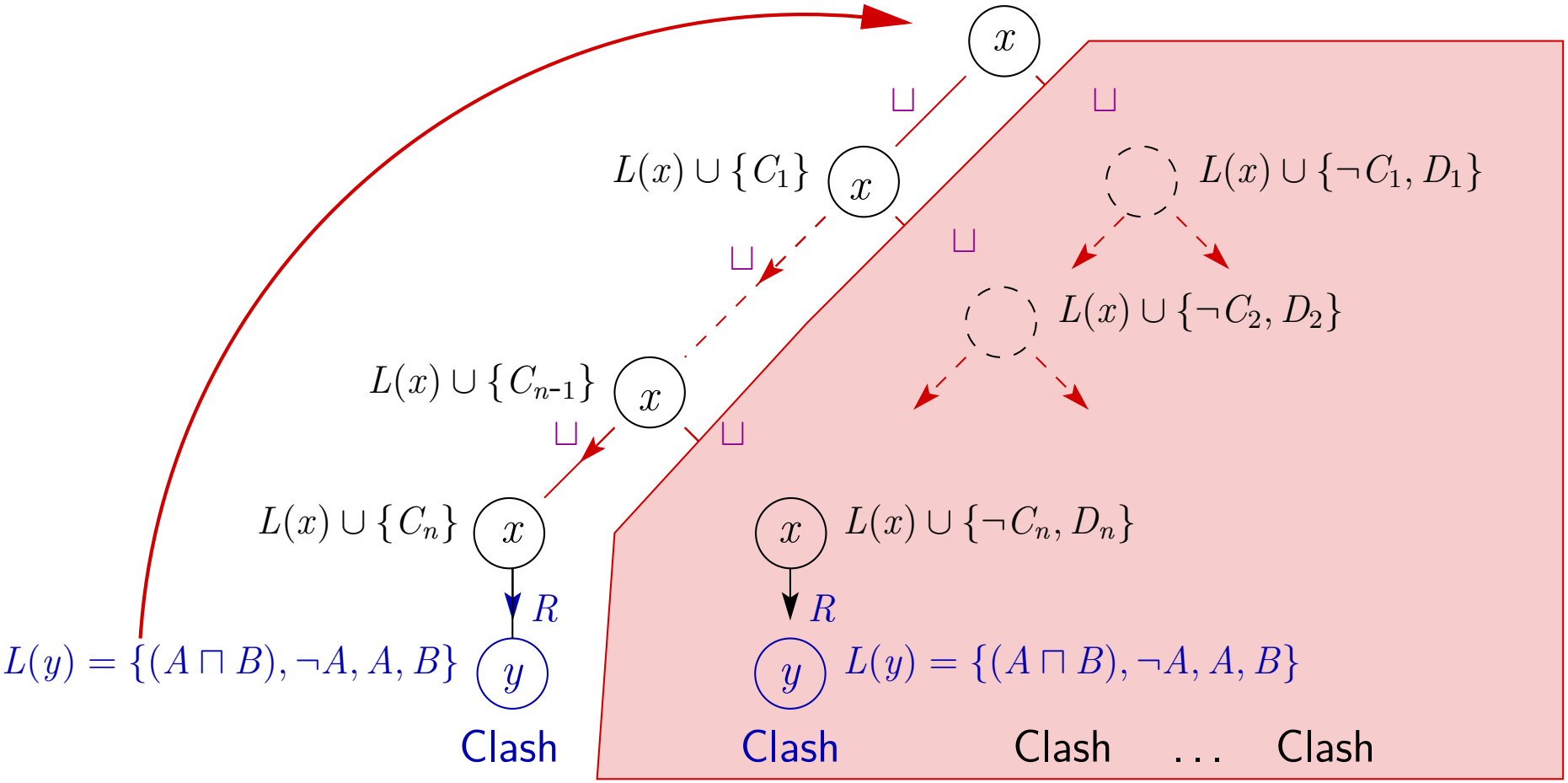
Backjumping

$$L(x) = \{\exists R.(A \sqcap B) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A\}$$



Backjumping

$$L(x) = \{\exists R.(A \sqcap B) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A\}$$



Is Everything OK?

$$L(x) = \{\exists \mathsf{R}.(\mathsf{A} \sqcap \mathsf{B}) \sqcap ((\mathsf{C}_1 \sqcup \mathsf{D}_1) \sqcap \ldots \sqcap (\mathsf{C}_n \sqcup \mathsf{D}_n)) \sqcap \forall \mathsf{R}.\neg \mathsf{A}\}$$

Is Everything OK?

$$L(x) = \{\exists R.(A \sqcap B) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A\}$$

Remember the rule ordering

- Cheap rules first
- Costly rules after

Is Everything OK?

$$L(x) = \{\exists R.(A \sqcap B) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A\}$$

Remember the rule ordering

- Cheap rules first
- Costly rules after

$\sqcap$-rule: $\exists R.(A \sqcap B), (C_1 \sqcup D_1), \ldots, (C_n \sqcup D_n), \forall R.\neg A$

$\exists$-rule: $A \sqcap B, \neg A \subseteq L(v)$, which is a clash

Another Backjumping Example

$$(\exists R.(A \sqcap B) \sqcup C_0) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A$$

Another Backjumping Example

$$(\exists R.(A \sqcap B) \sqcup C_0) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A$$

No inconsistency any more

Another Backjumping Example

$(\exists R.(A \sqcap B) \sqcup C_0) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A$

No inconsistency any more

But the first choice of $\exists R.(A \sqcap B)$ is extremely unfortunate:

- $\exists R.(A \sqcap B) \sqcap C_i \sqcap \ldots \sqcap \forall R.\neg A$
  is unsatisfiable for every $i > 0$
- Naive tableau will systematically try each C_i or D_i

Another Backjumping Example

$(\exists R.(A \sqcap B) \sqcup C_0) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A$

No inconsistency any more

But the first choice of $\exists R.(A \sqcap B)$ is extremely unfortunate:

- $\exists R.(A \sqcap B) \sqcap C_i \sqcap \ldots \sqcap \forall R.\neg A$
  is unsatisfiable for every $i > 0$
- Naive tableau will systematically try each C_i or D_i

Backjumping realizes that instead it should jump back and try C_0 instead of $\exists R.(A \sqcap B)$

Another Backjumping Example

$(\exists R.(A \sqcap B) \sqcup C_0) \sqcap ((C_1 \sqcup D_1) \sqcap \ldots \sqcap (C_n \sqcup D_n)) \sqcap \forall R.\neg A$

No inconsistency any more

But the first choice of $\exists R.(A \sqcap B)$ is extremely unfortunate:

- $\exists R.(A \sqcap B) \sqcap C_i \sqcap \ldots \sqcap \forall R.\neg A$ is unsatisfiable for every $i > 0$
- Naive tableau will systematically try each C_i or D_i

Backjumping realizes that instead it should jump back and try C_0 instead of $\exists R.(A \sqcap B)$

Clause/choice ordering and other SAT heuristics help further

Stuff Seen Today

- standard reasoning problems for $\mathcal{ALC}$ ontologies
- tableau algorithm for $\mathcal{ALC}$ ontologies that
  - requires blocking for termination
  - is a decision procedure for the standard reasoning problems
  - works on a set of labeled graphs in a non-deterministic way with backtracking
  - is implemented in state-of-the-art reasoners
- some essential optimizations

Tomorrow: Classification algorithm and performance experiments

Stuff Seen Today

- standard reasoning problems for $\mathcal{ALC}$ ontologies
- tableau algorithm for $\mathcal{ALC}$ ontologies that
  - requires blocking for termination
  - is a decision procedure for the standard reasoning problems
  - works on a set of labeled graphs in a non-deterministic way with backtracking
  - is implemented in state-of-the-art reasoners
- some essential optimizations

Tomorrow: Classification algorithm and performance experiments

Ask your questions!