Tableau Algorithms for Description Logics

Franz Baader
Theoretical Computer Science
RWTH Aachen
Germany

- Short introduction to Description Logics (terminological KR languages, concept languages, KL-ONE-like KR languages, ...).
- A tableau algorithm for $\mathcal{ALC}$ (i.e., multi-modal K).
- Extensions that can handle number restrictions, terminological axioms, and role constructors.

Description logics

class of knowledge representation formalisms

- Descended from structured inheritance networks [Brachman 78].
- Tried to overcome ambiguities in semantic networks and frames
  that were due to their lack of a formal semantics.
- Restriction to a small set of "epistemologically adequate" operators
  for defining concepts (classes).
- Importance of well-defined basic inference procedures:
  subsumption and instance problem.
- First realization: system KL-ONE [Brachman&Schmolze 85],
  many successor systems (Classic, Crack, FaCT, Flex, Kris, Loom, Race...).
- First application: natural language processing;
  now also other domains (configuration of technical systems, databases,
  chemical process engineering, medical terminology, ...)

Description logic systems

structure

description language

- constructors for building complex concepts and roles out of atomic concepts and roles
- formal, logic-based semantics

TBox

defines terminology of the application domain

ABox

states facts about a specific "world"

knowledge base

reasoning component

- derive implicitly represented knowledge (e.g., subsumption)
- "practical" algorithms

Rheinisch-Westfälische Technische Hochschule Aachen
RWTH

Description language

examples of typical constructors:

$C \sqcap D$, $\neg C$, $\forall r . C$, $\exists r . C$, $(\geq n\ r)$

A man	Human $\sqcap$ $\neg$ Female $\sqcap$
that is married to a doctor, and	$\exists$ married-to . Doctor $\sqcap$
has at least 5 children,	($\geq$ 5 has-child) $\sqcap$
all of whom are professors.	$\forall$ has-child . Professor

TBox

definition of concepts

Happy-man = Human $\sqcap$ ...

statement of constraints

$\exists$ married-to . Doctor $\sqsubseteq$ Doctor

ABox

properties of individuals

Happy-Man(Franz)

has-child(Franz,Luisa)

has-child(Franz,Julian)

Formal semantics

based on interpretations as in predicate logic

An interpretation I associates

- concepts C with sets C^I and
- roles r with binary relations r^I.

The semantics of the constructors is defined through identities:

- $(C \sqcap D)^I = C^I \cap D^I$
- $(\geq n\ r)^I = \{ d \mid \#\{e \mid (d,e) \in r^I\} \geq n \}$
- $(\forall r . C)^I = \{ d \mid \forall e: (d,e) \in r^I \Rightarrow e \in C^I \}$
- …

$I \models A = C$ iff $A^I = C^I$

$I \models C \sqsubseteq D$ iff $C^I \subseteq D^I$

$I \models C(a)$ iff $a^I \in C^I$

$I \models r(a,b)$ iff $(a^I,b^I) \in r^I$

Reasoning

makes implicitly represented knowledge explicit,
is provided as service by the DL system, e.g.:

polynomial reductions

Subsumption: Is C a subconcept of D?

$C \sqsubseteq D$ iff $C^I \subseteq D^I$ for all interpretations I.

Satisfiability: Is the concept description C non-contradictory?

C is satisfiable iff there is an I such that $C^I \neq \emptyset$.

Consistency: Is the ABox $\mathcal{A}$ non-contradictory?

$\mathcal{A}$ is consistent iff it has a model.

Instantiation: Is e an instance of C w.r.t. the given ABox $\mathcal{A}$?

$\mathcal{A} \models C(e)$ iff $e^I \in C^I$ for all models I of $\mathcal{A}$.

in presence of negation

Focus of DL research

- decidability/complexity of reasoning
- requires restricted description language
- systems and complexity results available for various combinations of constructors

- application relevant concepts must be definable
- some application domains require very expressive DLs
- efficient algorithms in practice for very expressive DLs?

Reasoning feasible ⟷ versus ⟷ Expressivity sufficient

Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH

DL research

historical overview

Phase 1

mostly system development (KL-ONE, LOOM, ...)

early eighties

- expressive description languages, but no disjunction, negation, exist. quant.
- use of so-called structural subsumption algorithms (polynomial)
- no formal investigation of reasoning problems and properties of algorithms

Phase 2

first formal investigations

mid-eighties

- formal, logic-based semantics
- first undecidability and complexity results
- incompleteness of structural subsumption algorithms
  - incompleteness as feature (Loom, Back)
  - restrict expressive power (Classic)

Phase 3

tableau algorithms for DLs and thorough complexity analysis

end eighties to mid-nineties

- Schmidt-Schauß and Smolka describe the first complete (tableau-based) subsumption algorithm for a non-trivial DL;

  $\mathcal{ALC}$: propositionally closed (negation, disjunction, existential restrictions);

  complexity result: subsumption in $\mathcal{ALC}$ is PSPACE-complete.

- Exact worst-case complexity of satisfiability and subsumption for various DLs (DFKI, University of Rome I).

- Development of tableau-based algorithms for a great variety of DLs (DFKI, University of Rome I, RWTH Aachen, ...).

- First DL systems with tableau algorithms: Kris (DFKI), Crack (IRST Trento); first optimization techniques for DL systems with tableau algorithms.

- Schild notices a close connection between DLs and modal logics.

$\mathcal{ALC}$ is a syntactic variant of multi-modal K

[Schild 91]

concept name A	propositional variable A
role name r	modal parameter r
$C \sqcap D$	$t(C) \land t(D)$
$C \sqcup D$	$t(C) \lor t(D)$
$\neg C$	$\neg t(C)$
$\exists r . C$	$\langle r \rangle t(C)$
$\forall r . C$	$[r]t(C)$

translation $\xrightarrow{t}$

interpretation I	Kripke structure $\mathcal{K} = (\mathcal{W}, \mathcal{R})$
set of individuals dom(I)	set of worlds $\mathcal{W}$
interpretation of role names r^I	accessibility relation R_r
interpretation of concept names A^I	worlds in which A is true

Phase 4

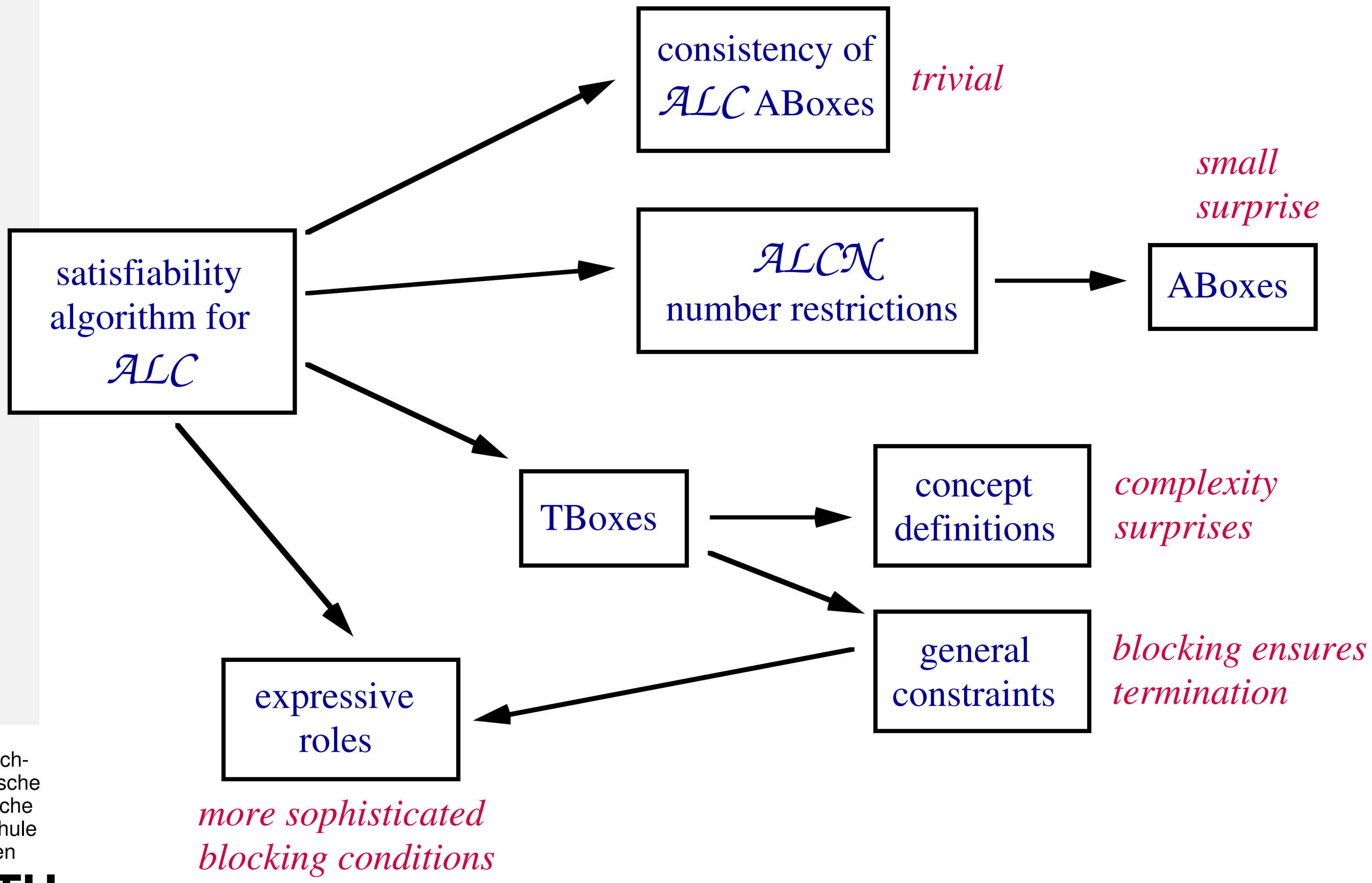
algorithms and systems for very expressive DLs (e.g., without finite model property)

late nineties

- Decidability results for very expressive DLs by translation into PDL (propositional dynamic logic) (Uni Roma I), strong complexity results; motivated by database applications.
- Intensive optimization of tableau algorithms (Uni Manchester, IRST Trento, Bell Labs): very efficient systems for expressive DLs.
- Design of practical tableau algorithms for very expressive DLs (Uni Manchester, RWTH Aachen).

Tableau algorithms for DLs

overview of the rest of the talk



Satisfiability algorithm

Idea generate a finite interpretation I such that $C_0^I \neq \emptyset$

Data structure for describing (partial) interpretations: ABoxes

(w.l.o.g. all concept descriptions in negation normal form)

Approach ABox assertions are viewed as constraints;

propagate constraints.

- Starting with $\mathcal{A}_0 := \{C_0(x_0)\}$, the algorithm applies transformation rules until all constraints are satisfied or an obvious contradiction is detected.
- Every rule corresponds to one constructor.
- Disjunction requires non-deterministic rule: two alternatives.

Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH

The $\rightarrow_{\sqcap}$-rule

Condition: $\mathcal{A}$ contains $(C_1 \sqcap C_2)(x)$, but not both $C_1(x)$ and $C_2(x)$.

Action: $\mathcal{A}' := \mathcal{A} \cup \{C_1(x), C_2(x)\}$.

The $\rightarrow_{\sqcup}$-rule

Condition: $\mathcal{A}$ contains $(C_1 \sqcup C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$.

Action: $\mathcal{A}' := \mathcal{A} \cup \{C_1(x)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{C_2(x)\}$.

The $\rightarrow_{\exists}$-rule

Condition: $\mathcal{A}$ contains $(\exists r.C)(x)$, but there is no individual name z such that $C(z)$ and $r(x, z)$ are in $\mathcal{A}$.

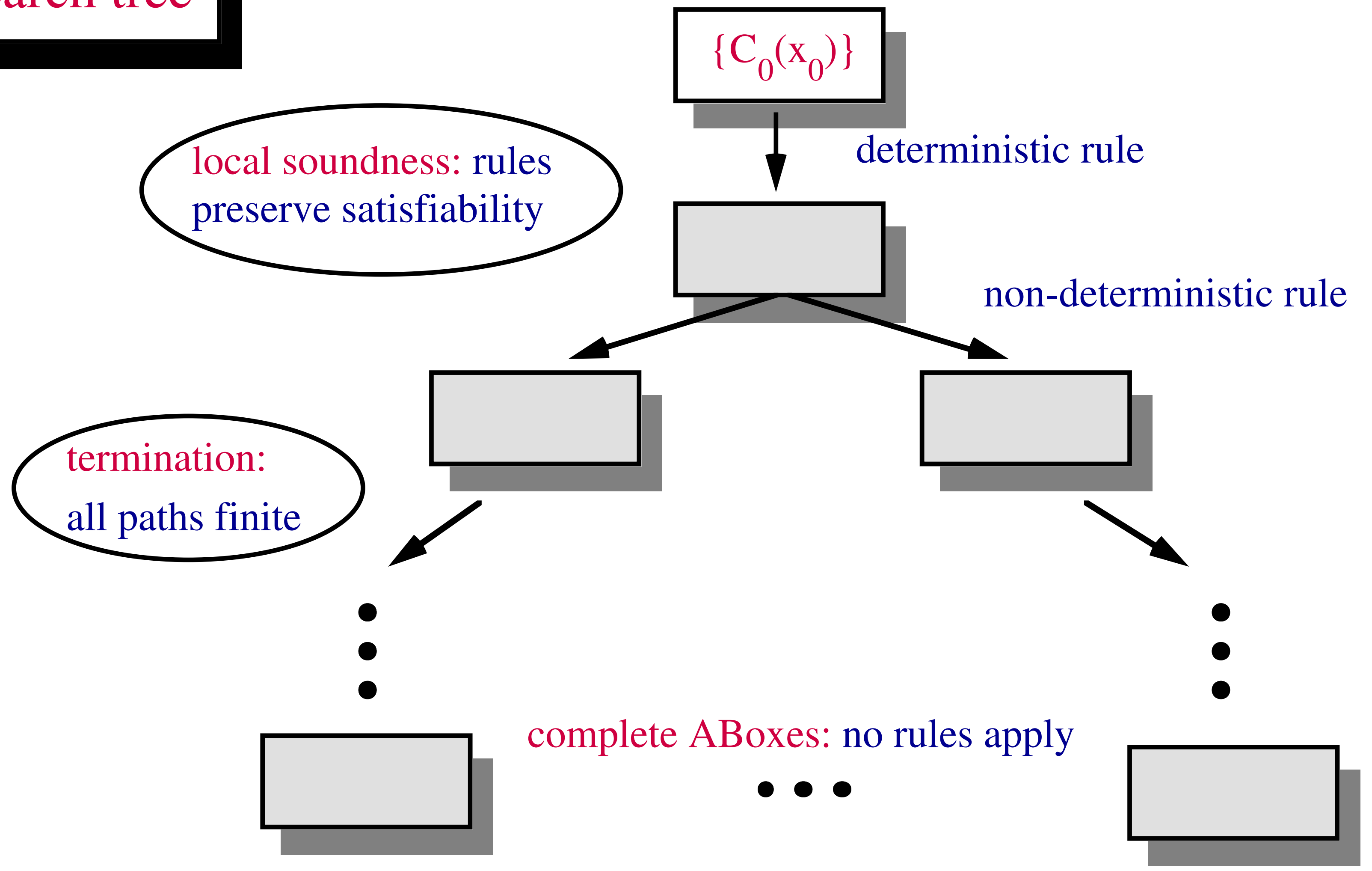
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(y), r(x, y)\}$ where y is an individual name not occurring in $\mathcal{A}$.

The $\rightarrow_{\forall}$-rule

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

Action: $\mathcal{A}' := \mathcal{A} \cup \{C(y)\}$.

search tree



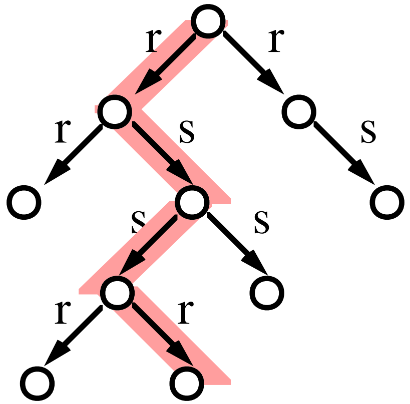
satisfiable iff one of the complete ABoxes is open, i.e., does not contain an obvious contradiction (clash) → $A(x), \neg A(x)$

Rheinisch-Westfälische Technische Hochschule Aachen
RWTH

Complexity

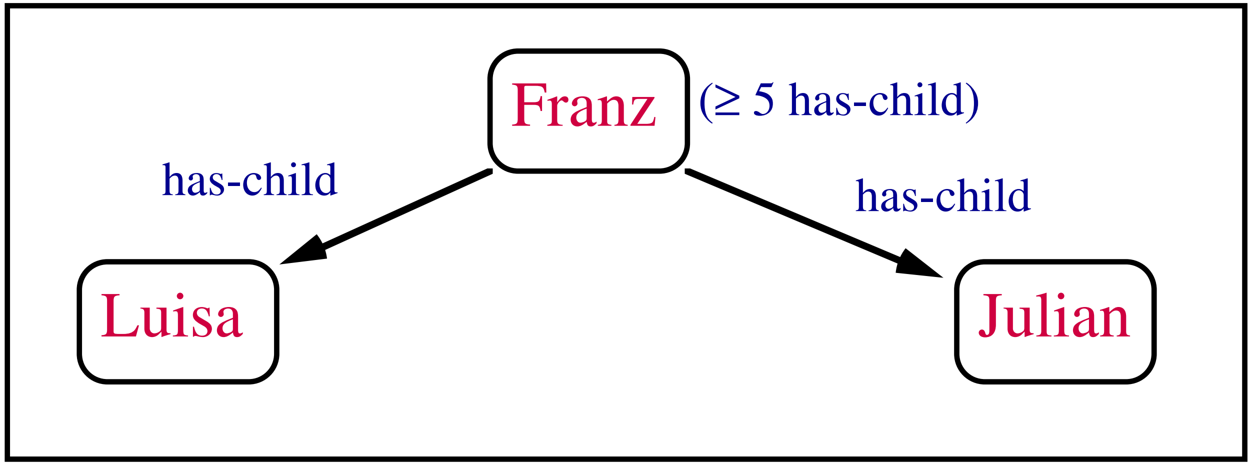
satisfiability in $\mathcal{ALC}$ is PSPACE-complete

- PSPACE-hard: reduction of QBF (Quantified Boolean Formulae)
- In PSPACE:
  - PSPACE = NPSPACE, i.e., forget about non-determinism
  - interpretations generated by the algorithm may be exponential, but:
  - they are trees of linear depth, whose branches can be generated separately

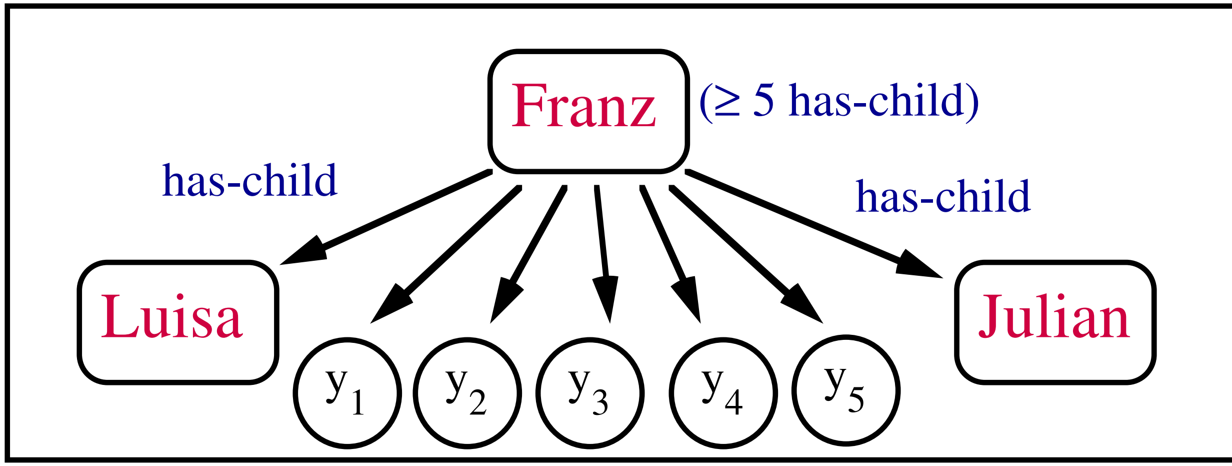


At-least restrictions

generate new distinct role successors, unless they are already present



$\longrightarrow_{\geq}$-rule



inequality assertions:

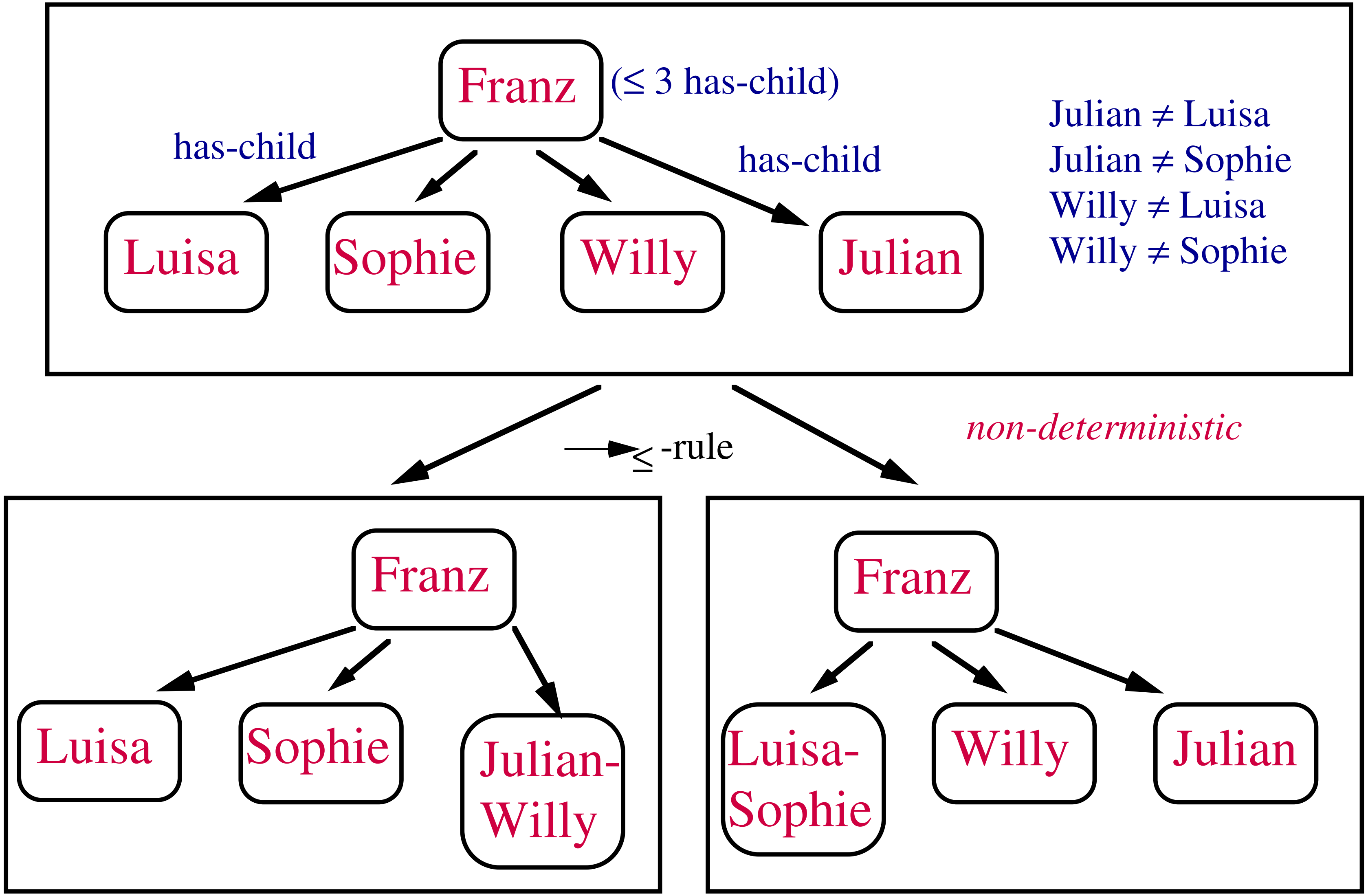
$y_1 \neq y_2$, $y_1 \neq y_3$,

...

$y_3 \neq y_5$, $y_4 \neq y_5$

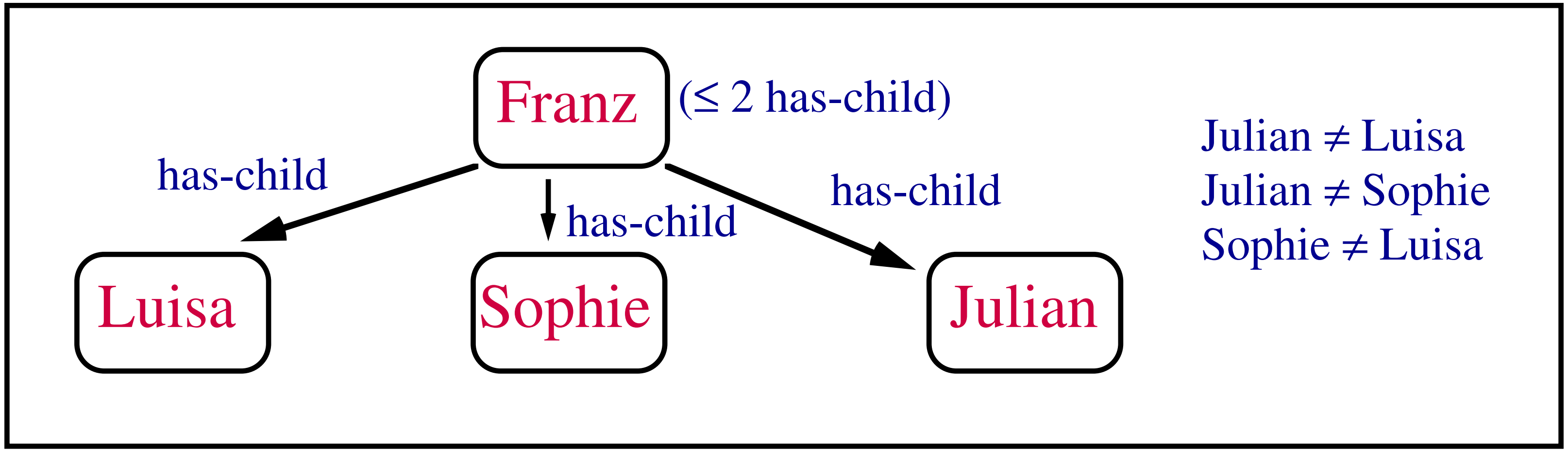
At-most restrictions

identify role successors if there are too many, unless they are asserted to be distinct



New type of clashes

if there are too many successors that are asserted to be distinct



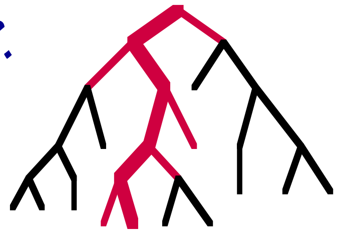
Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH

Complexity

satisfiability in $\mathcal{ALCN}$ is PSPACE-complete

- Unary coding of numbers: similar to the case of $\mathcal{ALC}$.

  Only one branch together with the direct successors of the nodes on the branch must be stored.



- Decimal coding of numbers: number n of direct successors exponential in the size of the decimal representation of n. However:

  - It is sufficient to generate only one representative for each at-least restriction, if
  - another type of clashes is used:

$(\leq n\ r), (\geq m\ r)$ for $n < m$

Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH

Qualified number restrictions

restrict number of successors belonging to a certain concept

$(\geq 3\ \text{has-child} . \text{Human}) \sqcap (\leq 1\ \text{has-child} . \text{Female}) \sqcap (\leq 1\ \text{has-child} . \neg\text{Female})$

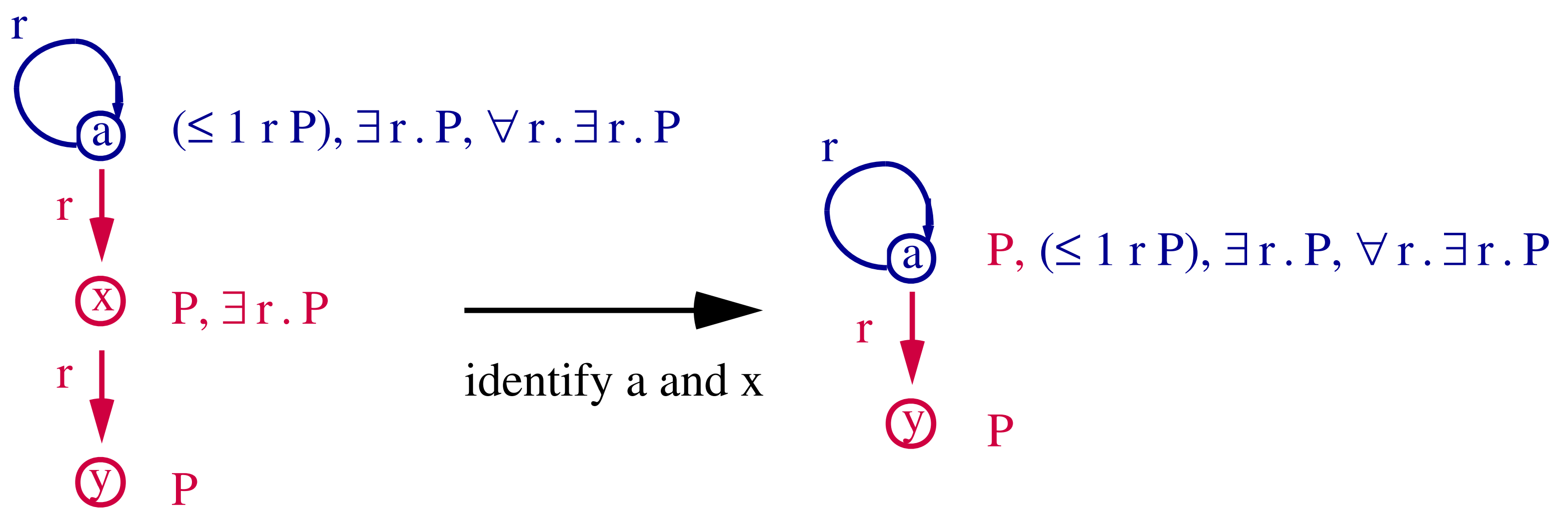
- Naive extension of the algorithm for $\mathcal{ALCN}$ [van der Hoek&de Rijke, 95] does not work.
- One needs an additional non-deterministic rule [Hollunder&Baader, 91]:

  If $(\leq n\ r . C)(a)$ is present, then choose $C(b)$ or $\neg C(b)$ for each r-successor b of a.

- Complexity: PSPACE-complete
  - Unary coding: same as for $\mathcal{ALCN}$
  - Decimal coding: introducing one representative is not sufficient! [Tobies, 99] uses counters and new types of clashes.

Consistency of $\mathcal{ALCN}$-Aboxes

- Naive extension of the satisfiability algorithm for $\mathcal{ALCN}$ [Hollunder, 90] does not terminate:



- Solution: use a strategy that applies generating rules with lower priority.

- Complexity: PSPACE-complete [Hollunder, 96].
  Pre-completion: first, apply rules only to "old" individuals; then forget about the role assertions.

Reasoning modulo concept definitions

acyclic,
w/o multiple definitions

- Defined names (lhs of defs) are just abbreviations (macros).
- Unfolding of concept descriptions: replace defined names by their definitions until no defined name occurs.
- Unfolding reduces reasoning modulo definitions to reasoning w/o definitions.
- Most papers consider only reasoning w/o definitions.

- However, unfolding may be exponential:

$$A_1 = \forall r . A_0 \sqcap \forall s . A_0, \quad \ldots, \quad A_n = \forall r . A_{n-1} \sqcap \forall s . A_{n-1}$$

- Complexity result for small language $(\forall, \sqcap)$ [Nebel, 90]:
  - subsumption w/o definitions is polynomial,
  - subsumption modulo concept definitions is coNP-complete.

- Folk theorem: this difference does not occur for $\mathcal{ALC}$.

Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH

Complexity results

for reasoning modulo concept definitions in more expressive DLs [Lutz, 99]

$\mathcal{ALC}$

complexity does not change when adding definitions

- subsumption/satisfiability PSPACE-complete
- subsumption/satisfiability modulo concept definitions PSPACE-complete
  - unfolding "on the fly"

$\mathcal{ALCF}$

complexity changes dramatically

- subsumption/satisfiability PSPACE-complete
  - connected feature subgraphs in model cannot be investigated "branche-wise"
  - their size is, however, polynomial
- subsumption/satisfiability modulo concept definitions NEXPTIME-complete
  - concept definitions can compactly represent large connected feature subgraphs

$\mathcal{ALCF}$

extends $\mathcal{ALC}$ by features and agreements on feature chains

Features

functional roles, interpreted as partial functions

has-father is functional whereas has-child is not

Agreements

on feature chains assert that successors exist and are identical

Human ⊓ (has-mother∘has-eye-colour = has-eye-colour)

Humans having the same eye colour as their mother

General constraints

$C \sqsubseteq D$ for arbitrary concept descriptions C, D

- Considering one constraint of the form $Top \sqsubseteq D$ is sufficient:

  $C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n \longrightarrow Top \sqsubseteq (\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap \neg C_n \sqcup D_n)$

- General constraints make reasoning considerably harder:

  - satisfiability/subsumption in $\mathcal{ALC}$ with general constraints

    EXPTIME-hard (proof very similar to Exptime-hardness of PDL)

  - satisfiability/subsumption in $\mathcal{ALCF}$ with general constraints

    undecidable (reduction from word problem for groups)

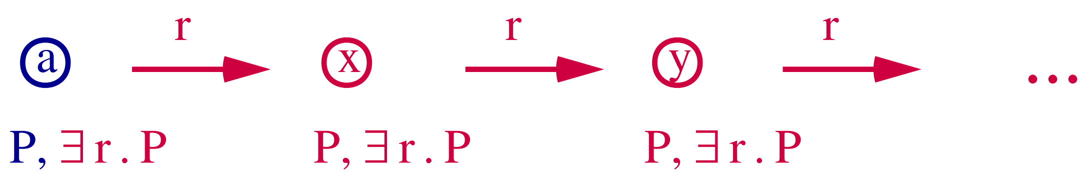
Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH

Satisfiability algorithm

for $\mathcal{ALC}$ with general constraints

- New rule: to take the constraint Top ⊑ D into account, assert D(b) for each individual b.

- This may obviously cause non-termination:
  test satisfiability of P under the constraint Top ⊑ ∃ r . P



- Blocking yields termination:
  - y is blocked by x in $\mathcal{A}$ iff {D | D(y) in $\mathcal{A}$} ⊆ {D | D(x) in $\mathcal{A}$}
  - generating rules not applied to blocked individuals
  - successors of "blocking individual" can be re-used for "blocked individual"

Complexity of $\mathcal{ALC}$ with general constraints

- Satisfiability/subsumption in $\mathcal{ALC}$ with general constraints is EXPTIME-complete:
  - in EXPTIME: translation into PDL or direct automata construction
- The tableau algorithm (as presented) yields only a NEXPTIME upper bound
  - optimized implementation shows very good behaviour in practice [Horrocks, 98]
  - designing an EXPTIME tableau algorithm for $\mathcal{ALC}$ with general constraints is rather hard [Donini&Massacci, 99].

Expressive roles

Role constructors

inverse of roles, composition of roles, transitive closure, Boolean operators, ...

has-parent as inverse of has-child

Restricted interpretation of roles

transitive roles, functional roles, role hierarchy, ...

has-offspring is transitive

has-son is sub-role of has-child

Transitive roles and role hierarchies

- Can simulate general constraints:

satisfiability of C under the constraint Top $\sqsubseteq$ D	$\rightarrow$	satisfiability of $C \sqcap D \sqcap \forall u . D$ where u is a transitive super-role of all roles in C, D

- Complexity: satisfiability/subsumption in $\mathcal{ALC}$ with transitive roles and role hierarchies is EXPTIME-complete (translation into PDL).
- The tableau algorithm can handle role hierarchies by replacing the condition "r(x,y) in $\mathcal{A}$" by "s(x,y) in $\mathcal{A}$ for a sub-role s of r".
- The tableau algorithm can handle transitive roles by an additional rule: if $\forall r . D(x)$ and r(x,y) is in $\mathcal{A}$ and r is transitive, then add $\forall r . D(y)$.
- Both ideas must be combined; blocking required for termination.

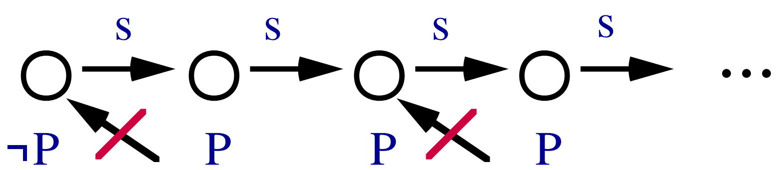
Rheinisch-
Westfälische
Technische
Hochschule
Aachen
RWTH

Transitive roles, role hierarchies, inverse roles in $\mathcal{ALCN}$

- No finite model property:

$$\neg P \sqcap \exists s.P \sqcap \forall r.(\exists s.P \sqcap (\leq 1\ s^{-1}))$$

where r is a transitive super-role of s



- The tableau algorithm in [Horrocks&Sattler, 99] tries to generate a finite pre-model that can be "unravelled" to a model.
- Requires more sophisticated blocking conditions.

Rheinisch-Westfälische Technische Hochschule Aachen
RWTH

Conclusion

tableau algorithms for DLs

- Main focus of research not on theoretical complexity results:
  - tableau approach yields worst-case optimal algorithms for PSPACE DLs
  - most tableau algorithms for EXPTIME DLs are not worst-case optimal
- Focus on practical algorithms: remarkable evolution in the last 15 years
  - eighties: polynomial structural algorithms
  - mid-nineties: optimized PSPACE tableau algorithms
  - end nineties: optimized tableau algorithms for EXPTIME DLs