



Query Answering with Ontologies - Optimizing in EXPTIME and beyond: lessons learnt and challenges ahead

Volker Haarslev
Department of Computer Science and
Software Engineering
Concordia University, Montréal, Canada

The research on Racer is joint work with Ralf Möller,
Hamburg University of Technology

STS Talk, Hamburg, Germany, August 5, 2009

Overview

- Review of OWL
- Tbox/Abox Inference Services
- Overview of optimization techniques in Racer
- Instance retrieval
- Qualifying cardinality restrictions and nominals

OWL Language

- Three species of OWL
 - OWL Full is union of OWL syntax and RDF
 - OWL DL restricted to description logic (FOL) fragment
 - OWL Lite is “easier to implement” subset of OWL DL
- Semantic layering
 - OWL DL is part of OWL Full that is within DL fragment
 - DL semantics officially definitive
- OWL DL based on SHIQ Description Logic
 - In fact it is equivalent to SHOIN(Dn) DL
- OWL DL Benefits from many years of DL research
 - Well defined semantics
 - Formal properties well understood (complexity, decidability)
 - Known reasoning algorithms
 - Implemented systems (highly optimised)

OWL Class Constructors

Constructor	DL Syntax	Example
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	Human \sqcap Male
unionOf	$C_1 \sqcup \dots \sqcup C_n$	Doctor \sqcup Lawyer
complementOf	$\neg C$	\neg Male
oneOf	$\{x_1\} \sqcup \dots \sqcup \{x_n\}$	{john} \sqcup {mary}
allValuesFrom	$\forall P.C$	\forall hasChild.Doctor
someValuesFrom	$\exists P.C$	\exists hasChild.Lawyer
maxCardinality	$\leq nP$	≤ 1 hasChild
minCardinality	$\geq nP$	≥ 2 hasChild

- XMLS **datatypes** as well as classes in $\forall P.C$ and $\exists P.C$
 - E.g., \forall hasAge.nonNegativeInteger
- Arbitrarily complex **nesting** of constructors
 - E.g., Person $\sqcup \forall$ hasChild.(Doctor $\sqcap \exists$ hasChild.Doctor)

OWL Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human \sqsubseteq Animal \sqcap Biped
equivalentClass	$C_1 \equiv C_2$	Man \equiv Human \sqcap Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$ Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} \equiv {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg$ {peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter \sqsubseteq hasChild
equivalentProperty	$P_1 \equiv P_2$	cost \equiv price
inverseOf	$P_1 \equiv P_2^-$	hasChild \equiv hasParent ⁻
transitiveProperty	$P^+ \sqsubseteq P$	ancestor ⁺ \sqsubseteq ancestor
functionalProperty	$\top \sqsubseteq \leq 1P$	$\top \sqsubseteq \leq 1$ hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leq 1P^-$	$\top \sqsubseteq \leq 1$ hasSSN ⁻

- Axioms (mostly) reducible to inclusion (\sqsubseteq)
 - $C \equiv D$ iff both $C \sqsubseteq D$ and $D \sqsubseteq C$

Overview

- Review of OWL
- **Tbox/Abox Inference Services**
- Tableau rules for ALC
- Racer architecture
- Overview of optimization techniques in Racer
- Instance retrieval
- Qualifying cardinality restrictions and nominals

Characteristics of DL Semantics

- Interpretation domain can be chosen arbitrarily
- Distinguishing features of description logics
 - domain can be infinite
 - open world assumption
- A concept C is satisfiable iff there exists an interpretation \mathcal{I} such that $C^{\mathcal{I}} \neq \emptyset$
 - \mathcal{I} is called a model of C

Tbox Inference Services

- **Satisfiability of concepts** defined w.r.t. a Tbox \mathcal{T}
- Subsumption can be reduced to satisfiability
 - $\text{subsumes}(C,D) \Leftrightarrow \neg \text{sat}(\neg C \sqcap D)$
 - denoted as $C \sqsupseteq D$ or $D \sqsubseteq C$
- Other inference services
 - **Tbox coherence**: List all unsatisfiable concept names in \mathcal{T}
 - **compute subsumption hierarchy** (taxonomy) of concept names in \mathcal{T}
 - Why emphasize concept names?
 - ontological decisions of users
 - important concepts will be named

Concept Examples

$\top \sqsubseteq \forall \text{has_child.person}$

Ensure that the range of `has_child` is `person`

$\text{woman} \equiv \text{person} \sqcap \text{female}$

$\text{parent} \equiv \text{person} \sqcap \exists \text{has_child.person}$

$\text{mother} \equiv \text{parent} \sqcap \text{female}$

$\text{mother_having_only_female_kids} \equiv \text{mother} \sqcap$
 $\quad \quad \quad \forall \text{has_child.female}$

$\text{mother_having_only_daughters} \equiv \text{woman} \sqcap$

$\quad \quad \quad \text{parent} \sqcap$
 $\quad \quad \quad \forall \text{has_child.woman}$

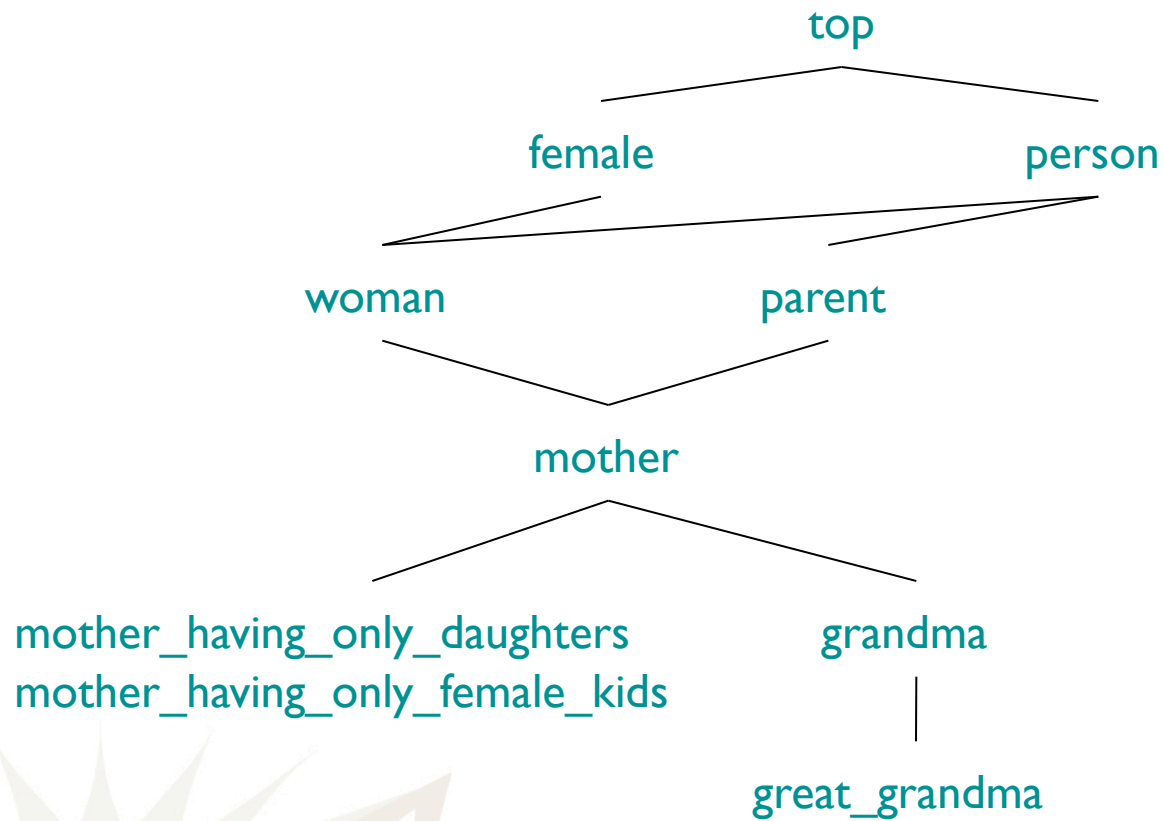
} equivalent

$\text{grandma} \equiv \text{woman} \sqcap \exists \text{has_child.parent}$

$\text{great_grandma} \equiv \text{woman} \sqcap$

$\quad \quad \quad \exists \text{has_child}.\exists \text{has_child.parent}$

Example Taxonomy



World Description or Abox

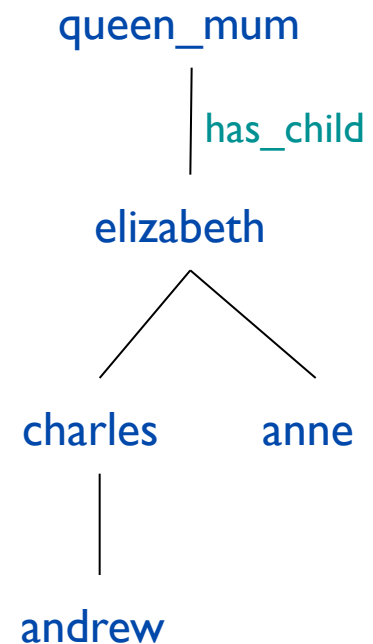
- How can we assert knowledge about individuals?
- Assertional axioms
 - concept assertion for an individual **a**
 - $a:C$ satisfied if $a^I \in C^I$
 - example: `elizabeth:mother`
 - role assertion for two individuals **a** and **b**
 - $(a,b):R$ satisfied if $(a^I, b^I) \in R^I$
 - example: `(elizabeth,charles):has_child`
- Unique name assumption possible
 - Different names denote different individuals
 - $a^I \neq b^I$

Abox Inference Services (1)

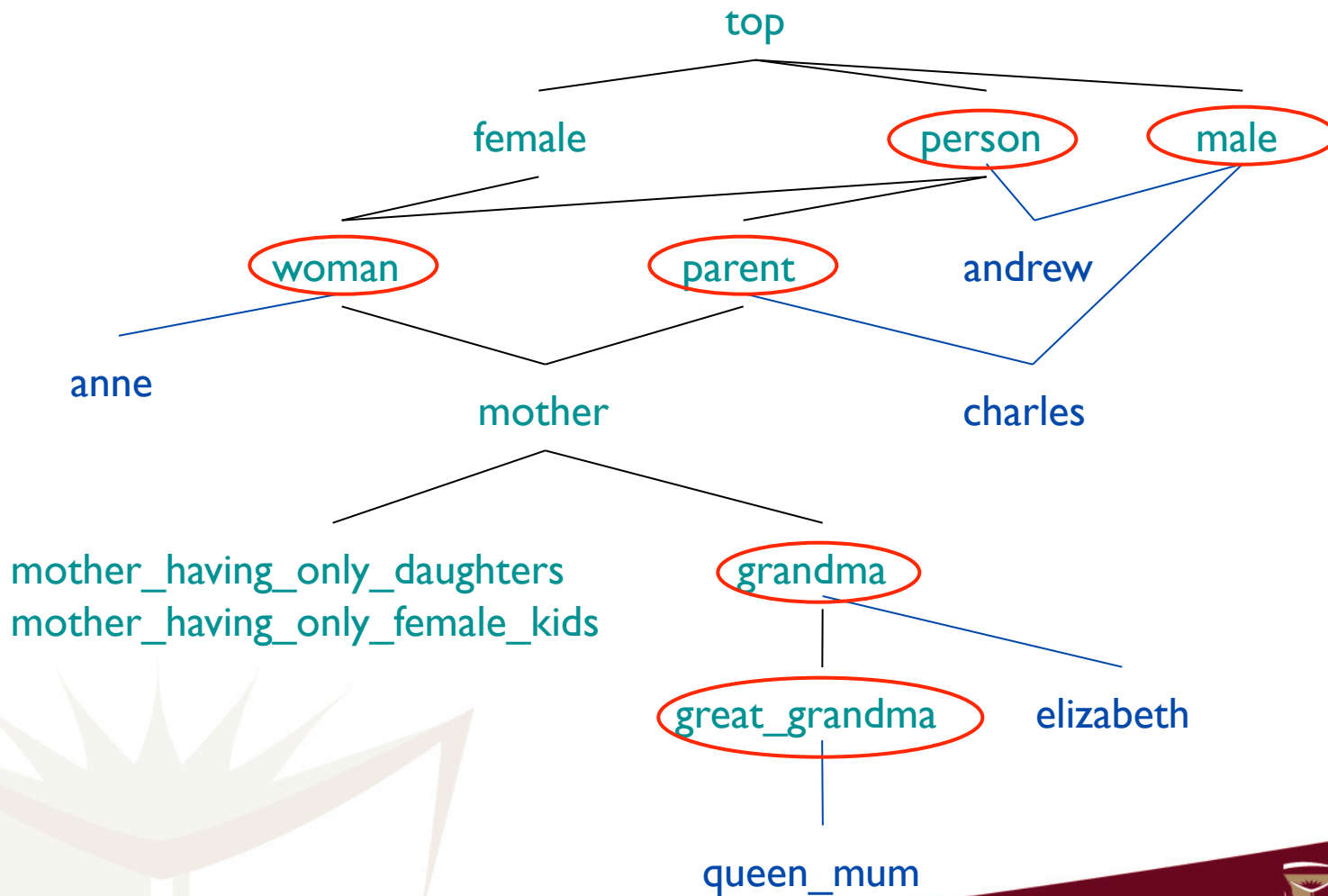
- Satisfiability of assertions defined w.r.t.
 - Abox \mathcal{A}
 - Tbox \mathcal{T}
- Inference services
 - **Abox satisfiability**: Is the collection \mathcal{A} of assertions satisfiable?
 - **Instance checking**: $\text{instance?}(a, C, \mathcal{A})$
Is a an instance of concept C or subsumes C the individual a ?
 - **Abox realization**: compute for all individuals in \mathcal{A} their most-specific concept names w.r.t. Tbox \mathcal{T}
 - **Instance retrieval**: find all instances of a given concept
 $\text{instance_retrieval}(C, \mathcal{A})$

ABox Example

- $(\text{male} \sqsubseteq \neg \text{female})$ additional axiom ensuring disjointness
- $\text{queen_mum} : \text{woman}$
- $(\text{queen_mum}, \text{elizabeth}) : \text{has_child}$
- $\text{elizabeth} : \text{woman}$
- $(\text{elizabeth}, \text{charles}) : \text{has_child}$
- $(\text{elizabeth}, \text{anne}) : \text{has_child}$
- $\text{charles} : \text{parent} \sqcap \text{male}$
- $\text{anne} : \text{woman}$
- $(\text{charles}, \text{andrew}) : \text{has_child}$
- $\text{andrew} : \text{person} \sqcap \text{male}$



TBox Taxonomy plus Individuals



Abox Inference Services (2)

- New basic inference service: Abox satisfiability
 - $\text{asat}(\mathcal{A})$
- All other inference services can be reduced to asat
 - instance checking:
 $\text{instance?}(a, C, \mathcal{A}) \equiv \neg \text{asat}(\mathcal{A} \cup \{a: \neg C\})$
 - concept satisfiability:
 $\text{sat}(C) \equiv \text{asat}(\{a: C\})$
 - concept subsumption:
 $\text{subsumes}(C, D) \equiv \neg \text{sat}(\neg C \sqcap D) \equiv \neg \text{asat}(\{a: \neg C \sqcap D\})$

Standard Inference Services

- Consistency of class/concept description
 - catch design errors
 - example: vegetarian eats meat
- Subsumption between classes
 - example: a **mother** is always a **parent**
- Taxonomy of class names (classification)
 - ordered by subsumption relationship
 - from very general to very specific
- Consistency of individual descriptions
 - Is the knowledge specified for an individual **joe** consistent with other known individuals and classes
 - **joe** (vegetarian) makes a reservation for a restaurant that offers only meals containing meat
- Find classes that match known instances
 - if **susy** is **female** and has a **child**, she is an instance of **mother**

Two Reasoning views

- Traditional view from knowledge engineering
 - defined concept express domain knowledge
 - primitive concepts express only necessary conditions
 - axioms ensure global consistency criteria
 - importing inferences services
 - taxonomy
 - unsatisfiable concepts
- Theorem prover
 - domain knowledge is expressed as a set of arbitrary axioms
 - inference services
 - taxonomy gives no interesting information
 - unsatisfiable concepts
 - is a hypothesis implied by the set of axioms?

Basic Reasoning Assumptions

- Reasoning is monotonic
 - Inferred knowledge cannot contradict known knowledge
- Open world assumption
 - In contrast to databases
 - The absence of knowledge cannot be used for inferences
 - No premature inferences
 - Ideal for web context
- Incomplete knowledge
 - Default assumption
 - Granularity and depth of knowledge may vary

Overview

- Review of OWL-DL
- Tbox/Abox Inference Services
- **Overview of optimization techniques in Racer**
- Instance retrieval
- Qualifying cardinality restrictions and nominals

There's more to Racer ...

- ... than a collection of optimized (inference) engines
- For many inference problems:
- Motto: Avoid using the optimized tableaux prover (and the other provers) at all!
- So, exploit given information
 - $A \sqsubseteq B \sqcap C \sqcap \exists R.D$
- ... and do not compute anything twice
- Nevertheless: the tableau prover is important

Preprocessing (1)

- Normalization
 - canonical order of concept descriptions
 - $B \sqcap C$ instead of $C \sqcap B$
- Simplification
 - remove duplicates: $B \sqcap C \sqcap B \rightarrow B \sqcap C$
 - remove redundant descriptions: $(A \sqcup \neg A) \rightarrow \top$, $(A \sqcap \neg A) \rightarrow \perp$
- Extract for named concepts
 - told subsumer: $A \sqsubseteq B \sqcap C \rightarrow \{B, C\}$
 - told disjoints: $A \sqsubseteq \neg D \sqcap \neg E \rightarrow \{D, E\}$
 - A refers-to: $A \sqsubseteq B \sqcap \exists R.C \rightarrow \{B\}, \{C\}$
 - C referenced-by: $A \sqsubseteq B \sqcap \exists R.C \rightarrow \{A\}$
- Terminological cycles: $A \sqsubseteq B \sqcap \exists R.A$

Preprocessing (2)

- Determine logic of concept expressions
 - static analysis while parsing input
 - used to dynamically switch optimizations on/off
 - per satisfiability, subsumption, ... test
 - triggers different clash detection
 - number restrictions present
 - no clash possible
 - triggers different blocking techniques
 - subset, equality, double blocking
 - dynamic analysis for inference tests with new input
 - emphasis on fragments such as ELH, L, CD (completely defined)

Optimization Techniques: SAT (1)

- Trace technique searches depth-first
 - give priority to \neg , \sqcup , \forall operators
 - afterwards process at-least and at-most restrictions
 - space economic but not always optimal for inverse roles
- Dependency-directed backtracking
 - skip choice points during backtracking that did not contribute to encountered clash
 - potential runtime improvement:
number of choices: from exponential to linear

- $(C_1 \sqcup D_1) \sqcap \dots \sqcap (C_n \sqcup D_n)$

Optimization Techniques: SAT (2)

- Selection of disjunctions
 - previously failed (closed) disjuncts disabled
 - weighted measure of
 - number of open disjuncts
 - total number of disjuncts
 - depth of or-dependencies in backtracking tree
- Selection of open disjuncts
 - frequency count of (negated) disjuncts in (all) disjunctions
- Semantic branching
 - $(A \sqcup B)$: test A and $\neg A$ instead of A and B
 - constraint propagator: mark
 - satisfied disjunctions or
 - closed disjuncts

Optimization Techniques: SAT (3)

- Selection of existential restrictions
 - depth of or-dependencies in or-tree
 - give priority if restricted by at-most restrictions
- Caching of satisfiability status of named concepts
 - using so-called pseudo models derived from a SAT test
 - reused for
 - subsumption testing
 - modal successor testing if no cache entry found using model merging
- Modal successor caching
 - (un)satisfiability of $\exists R.C \sqcap \forall R.D$ reused for $\exists S.D \sqcap \forall S.C$
 - depend on logic
- Both caching techniques require a dependency management in order to remain sound

Optimization Techniques: SAT (4)

- Process qualified number restrictions with integer linear programming
 - (un)satisfiability of conjunction of qualified number restrictions is equivalent to (un)satisfiability of a corresponding set of linear inequations
 - using Simplex and branch and bound/cut to achieve non-negative integer solutions
 - recently extended to deal with nominals
 - calculus for SHOQ

Optimization Techniques: Abox (1)

- Graph transformations
 - rolling-up of chains into concept descriptions
 - $(a,b):R, (b,c):S, b:C, c:D$ transformed into $a:\exists R.(C \sqcap \exists S.D)$
- Fast non-instance test
 - more than 90% of all instance checking tests are unsuccessful
 - individual pseudo model merging
 - extracted from Abox consistency test
 - a is not an instance of C if the pseudo models of $\neg C$ and a are mergable

Optimization Techniques: Abox (2)

- Exploiting completion information
 - **completion**: based on Abox consistency test
 - sound but incomplete
 - individual a is not an instance of C if $a:\neg C$ added to the completion does not cause a clash
 - **precompletion**: “deterministic” part of completion
 - sound and complete
 - individual a is not an instance of C if $a:\neg C$ added to the completion does not cause a clash
 - otherwise a is an instance of C

Optimization Techniques: Abox (3)

- Partitioning into independent parts
 - currently only connectedness of graphs
- Linear instance retrieval
 - iterate over all individuals to find instances of a given concept **C**
- Binary instance retrieval
 - find all instances of a given concept **C**
 - binary split of individual candidate sets
 - one “expensive” test eliminates a large set of candidates
- Dependency-based instance retrieval
 - split based on clash dependencies

Overview

- Review of OWL-DL
- Tbox/Abox Inference Services
- Overview of optimization techniques in Racer
- Instance retrieval
- **Qualifying cardinality restrictions and nominals**

Why the focus on qualified number restrictions?

- Reasoning with qualified number (or cardinality) restrictions is known to scale not very well
- OWL 2 supports qualified number restrictions
 - One can expect more ontologies using these constructs
 - Performance bottlenecks will discourage potential users
- Useful in many domains
 - engineering (≥ 8 has_part cylinders)
 - medicine (≥ 37 has_part bone)
 - ...

Qualified number restrictions in our everyday life

Listening to elevator conversations (incomplete knowledge)

- Day 1: *I bought a mini-van (8 seats) that fits our family* \Rightarrow
 ≤ 6 has_child
- Day 2: *My youngest daughter won a prize* \Rightarrow
 \exists has_child.(female \sqcap won_price) \sqcap ≥ 2 has_child.female
- Day 3: *Two of my sons are hockey fans* \Rightarrow
 ≥ 2 has_child.(male \sqcap hockey_fan)
- Day 4: *One of my daughters is a computer whiz* \Rightarrow
 \exists has_child.(female \sqcap computer_whiz)
- Day 5: *I have three sons* \Rightarrow
 ≤ 3 has_child.male \sqcap ≥ 3 has_child.male
- Day 6: *All my daughters study at Concordia* \Rightarrow
 ≤ 3 has_child.(female \sqcap concordia_student)

Qualified number restrictions: Do we need numbers greater than 7?

$Student \sqsubseteq \forall hasCredit.(Science \sqcup Engineering \sqcup Business) \sqcap$
 $\geq 120 hasCredit.(Science \sqcup Engineering) \sqcap$
 $\geq 140 hasCredit \sqcap$
 $\leq 32 hasCredit.(Science \sqcup Business) \sqcap$
 $\leq 91 hasCredit.Engineering$

- Typical inference service requested:
Is the concept *Student* satisfiable?
- In many domains numbers greater than 7 are essential in qualified number restrictions

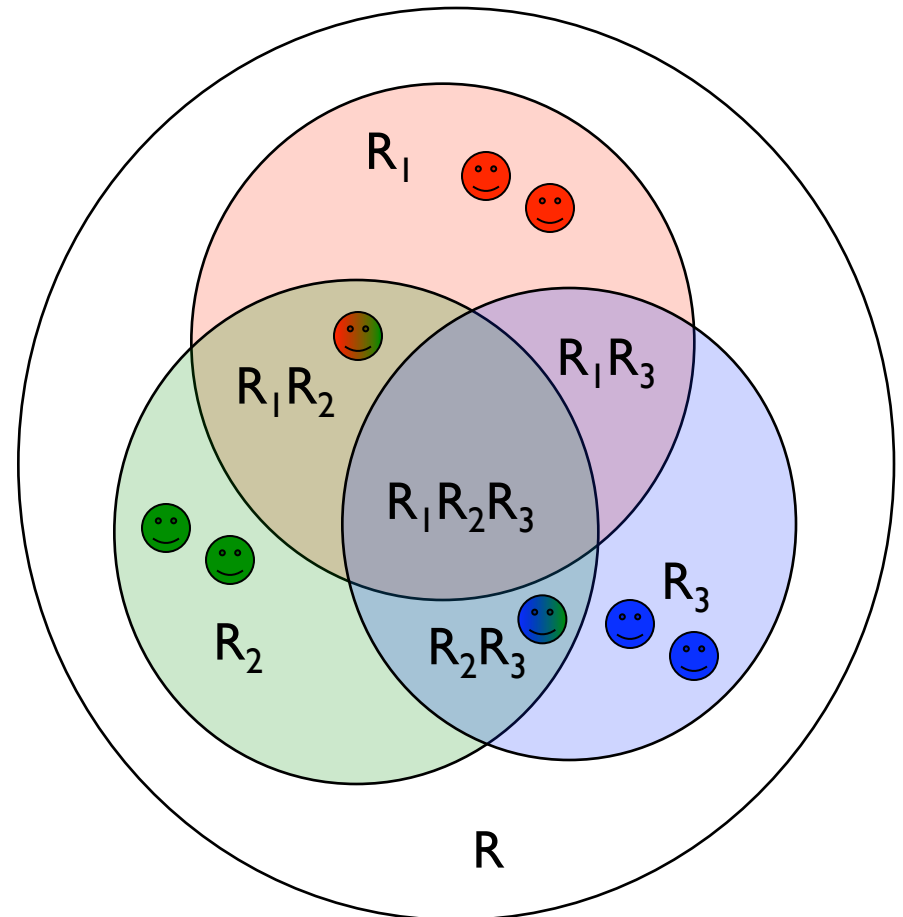
Standard Tableau Rules for Q

- At-least rule: $\geq n R.C$
 - if necessary, create n distinct R -successors which become members of C
- At-most rule: $\leq n R.C$
 - if there exist more than n R -successors which are members of C , then merge two (non-distinct) of these R -successors
- Choose rule:
 - if there exists an R -successor whose membership for C or $\sim C$ is unknown, determine whether this successor is a member of C or $\sim C$
- Treatment of numbers is highly inefficient
- Individual-wise computation of set cardinalities
- No representation of sets
- Blind search tries to fulfill at-least and at-most restrictions

Standard Tableau Reasoning Example

≥ 2 has_child.rich \sqcap
 ≥ 2 has_child.prof \sqcap
 ≥ 2 has_child.lawyer \sqcap
 ≤ 3 has_child
prof $\sqsubseteq \neg$ rich

R_1
 R_2
 R_3
 R



- Standard tableau calculi do not represent set cardinalities
- Solve implicit arithmetic problems by trying all possible combinations
- Merge pairs of individuals

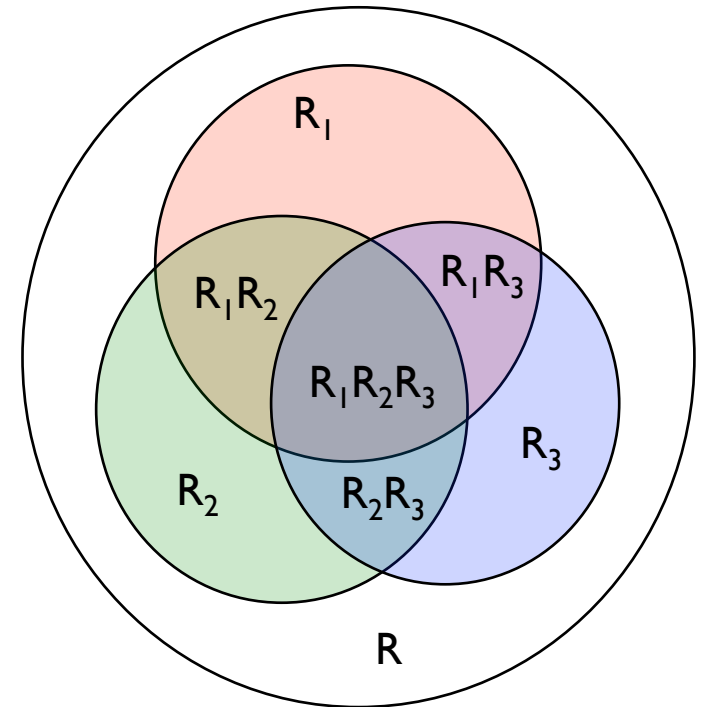
Hybrid Architecture

- Idea: Translate concept satisfiability w.r.t. qualifying number restrictions to solvability of a set of inequations
- A concept expression consisting of a conjunction of qualified number restrictions is satisfiable iff the corresponding set of inequations has a solution
- Our approach: Two interacting components
 - tableau reasoner for the logical part
 - reasoner for linear integer inequations

Joint work with **Nasim Farsiniamarj**

Atomic Decomposition

- Idea of a so-called atomic decomposition of role fillers
- Semantic partitioning into mutually disjoint sets of role fillers
- Sum of set cardinalities form part of linear inequations

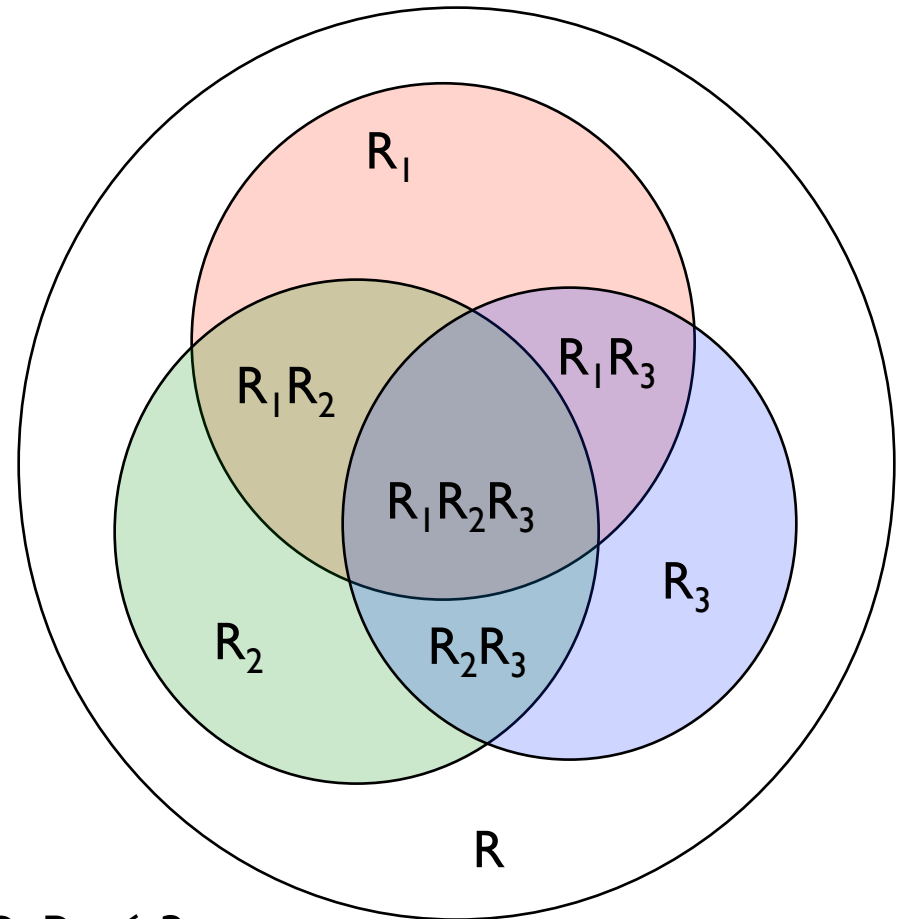


Inspired by: Ohlbach, H.J., Köhler, J., *Modal Logics, Description Logics and Arithmetic Reasoning*, *Journal of Artificial Intelligence* 1-2(1999), 1-31

Algebraic Reasoning Example

≥ 2 has_child.rich \sqcap
 ≥ 2 has_child.prof \sqcap
 ≥ 2 has_child.lawyer \sqcap
 ≤ 3 has_child
 prof \sqsubseteq \neg rich

R_1
 R_2
 R_3
 R



$R_i \geq 0, R_iR_j \geq 0, R_iR_jR_k \geq 0, i,j,k=1..3$

Goal: minimize sum of all variables

$$R_1 + R_1R_2 + R_1R_3 + R_1R_2R_3 \geq 2$$

$$R_2 + R_1R_2 + R_2R_3 + R_1R_2R_3 \geq 2$$

$$R_3 + R_1R_3 + R_2R_3 + R_1R_2R_3 \geq 2$$

$$R_1 + R_2 + R_3 + R_1R_2 + R_1R_3 + R_2R_3 + R_1R_2R_3 \leq 3$$

$$R_1R_2 \leq 0$$

$$R_1R_2R_3 \leq 0$$

Possibly concluded from the logical part

Evaluation

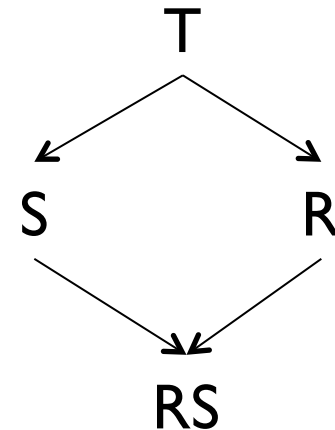
- Prototype reasoner only optimized for
 - Qualified number restrictions
 - Dependency-directed backtracking
- Synthetic benchmarks adequate for evaluation
- Comparison with Pellet (2.0) as representative tableau DL reasoner
- Effect of enabling/disabling implemented optimization techniques

Parameters affecting performance

- Size of numbers in number restrictions
- Type of backtracking
- Satisfiability vs. unsatisfiability of concepts
- Number of number restrictions
- (Ratio of number of at-least restrictions to number of at-most restrictions)

Increasing values of numbers (1)

- Two test concept patterns
- The value i is incremented
 - linear growth (2 – 10, 1 - 100)
 - exponential growth (10^1 – 10^6)



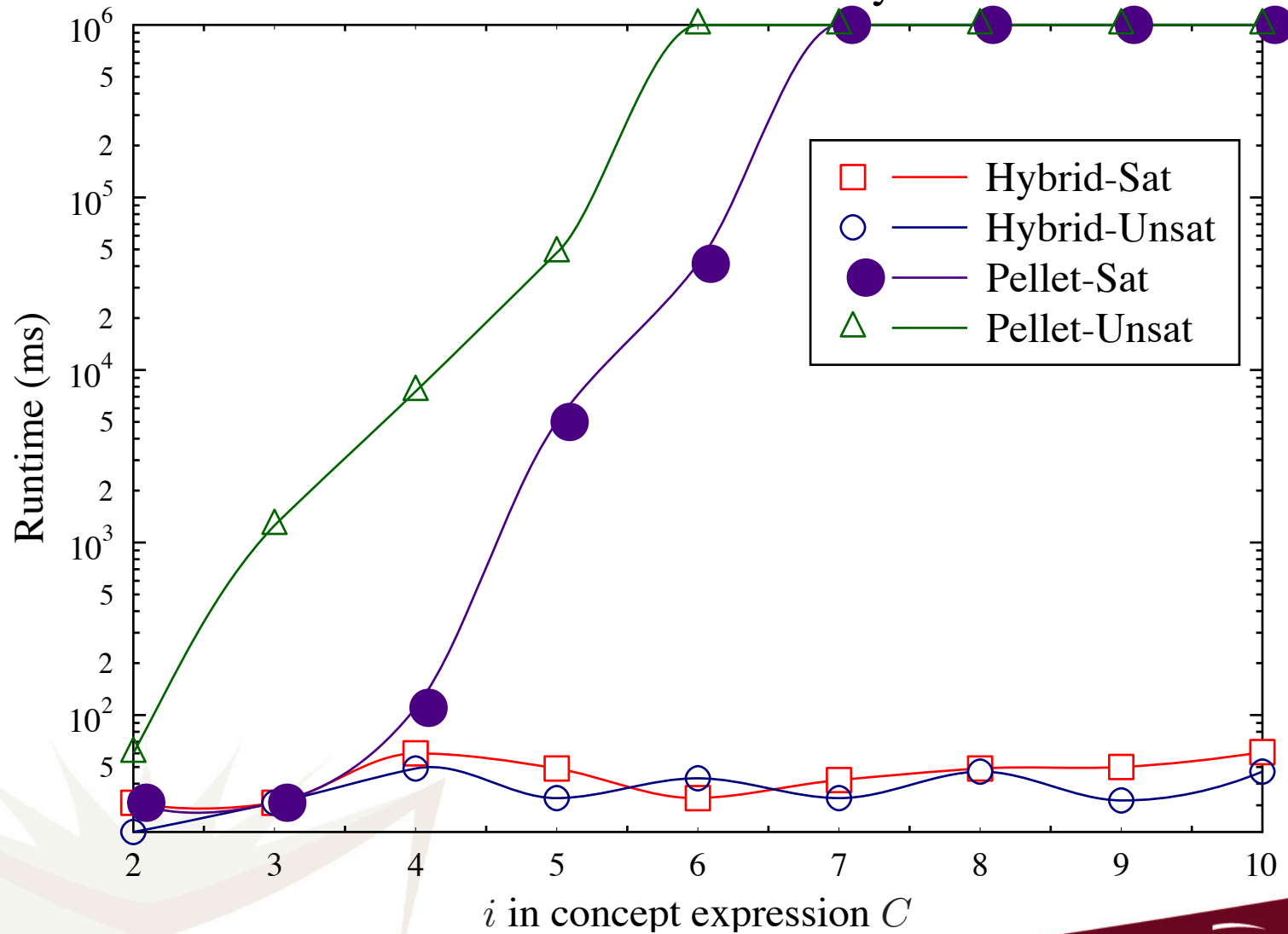
$$(\geq 2i RS.(A \sqcup B)) \sqcap (\leq i S.A) \sqcap (\leq i R.B) \sqcap (\leq (i-1) T.(\neg A)) \sqcup (\leq i T.(\neg B))$$

C_{SAT}

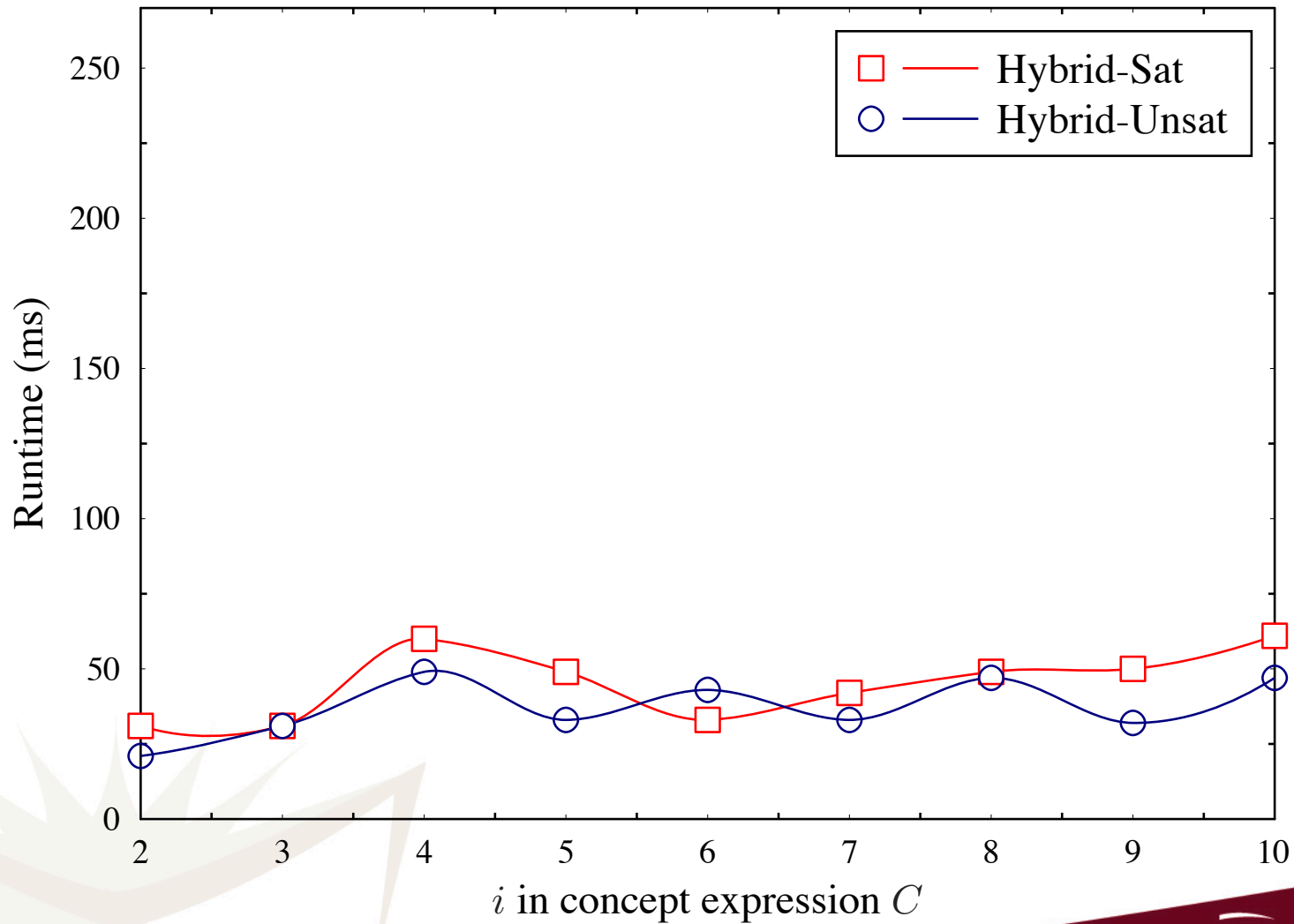
$$(\geq 2i RS.(A \sqcup B)) \sqcap (\leq i S.A) \sqcap (\leq i R.B) \sqcap (\leq (i-1) T.(\neg A)) \sqcup (\leq (i-1) T.(\neg B))$$

C_{UNSAT}

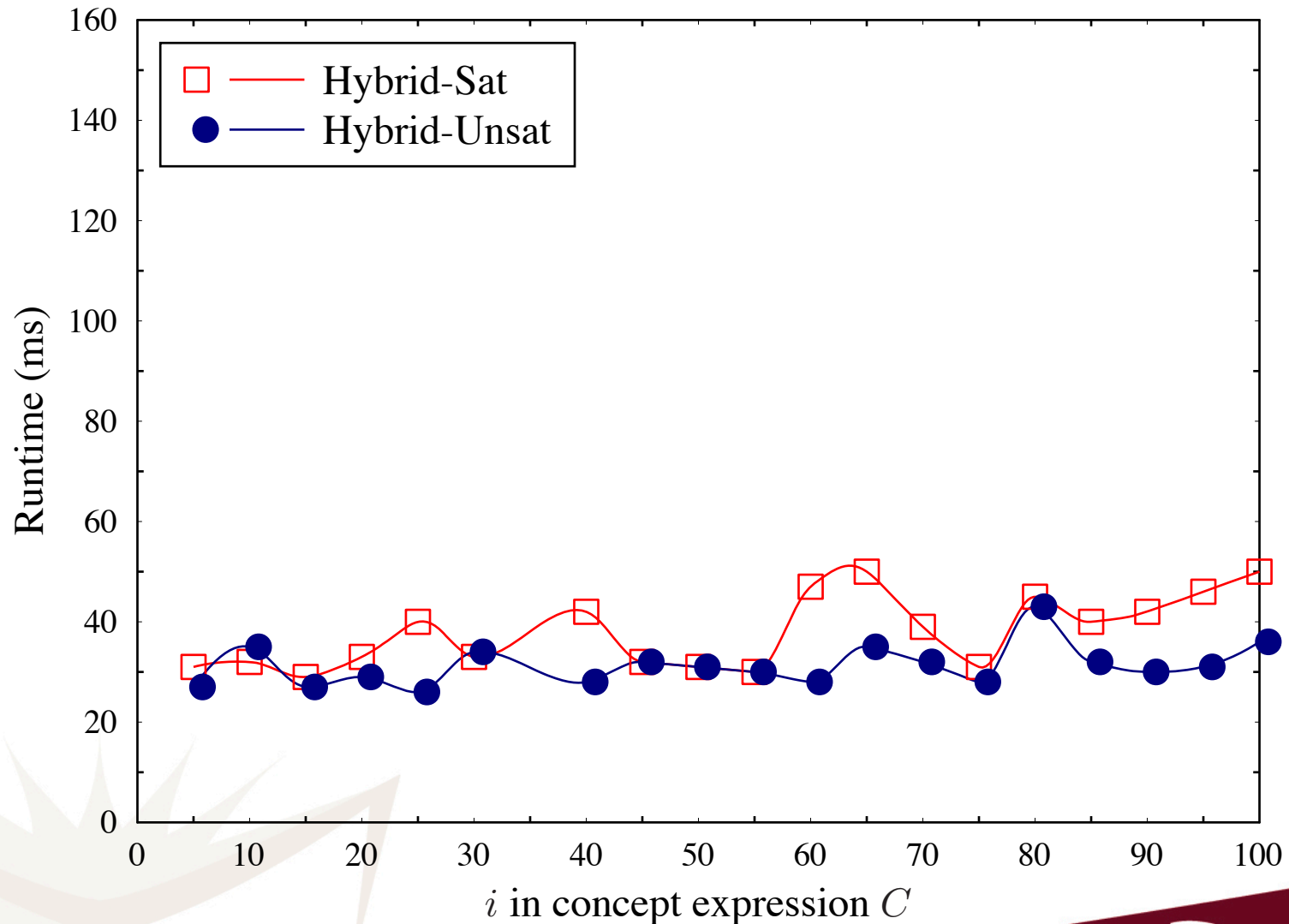
Increasing values of numbers (2)



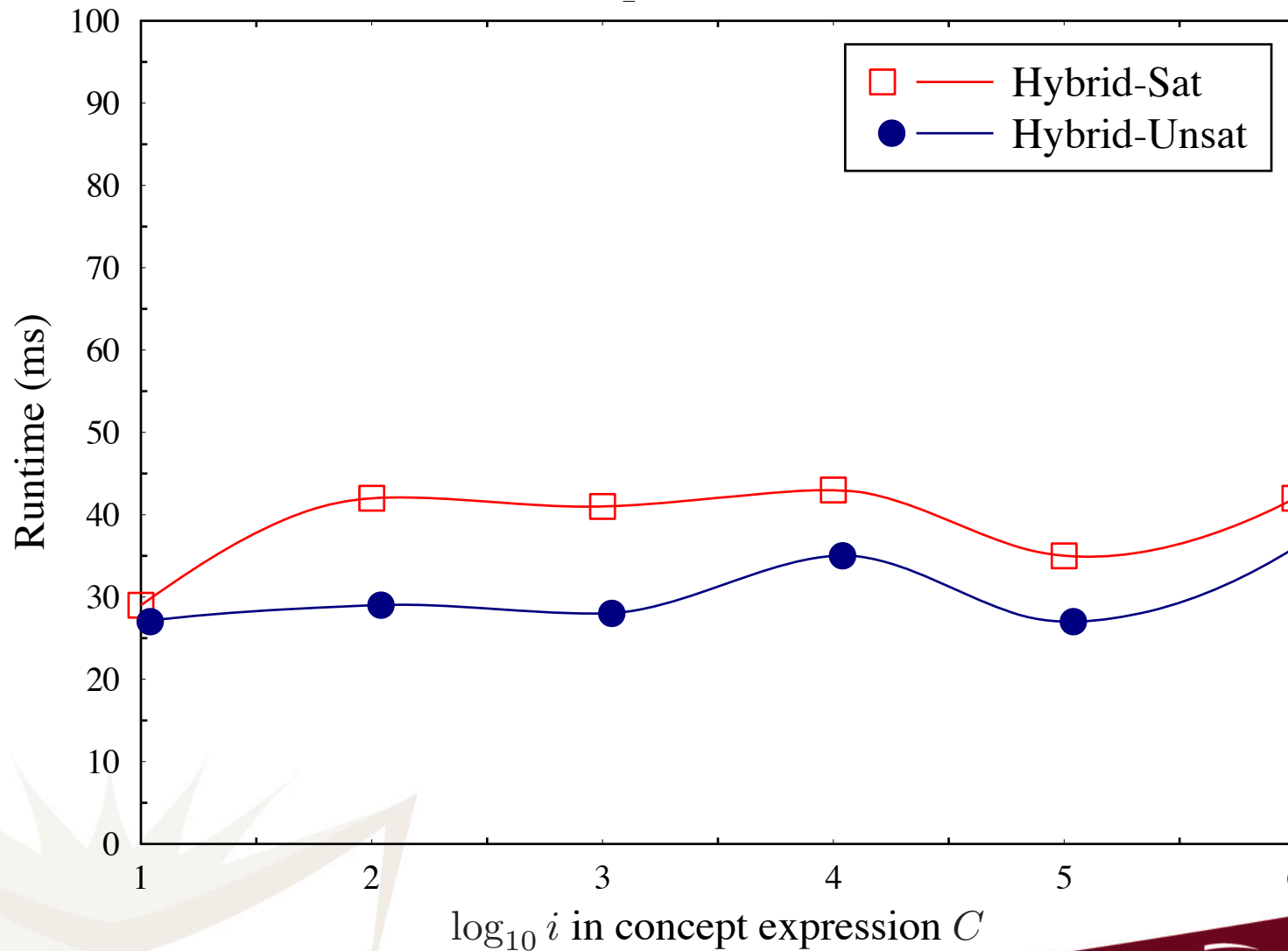
Increasing values of numbers (3)



Increasing values of numbers (4)



Increasing values of numbers (5)



Number of Qualified Number Restrictions (1)

- Satisfiable concept
- Hybrid algorithm encounters no clashes
- In each step increase the number of at-least and at-most restrictions ($2i + 1$ restrictions)
- Linear growth of i (3 – 17)

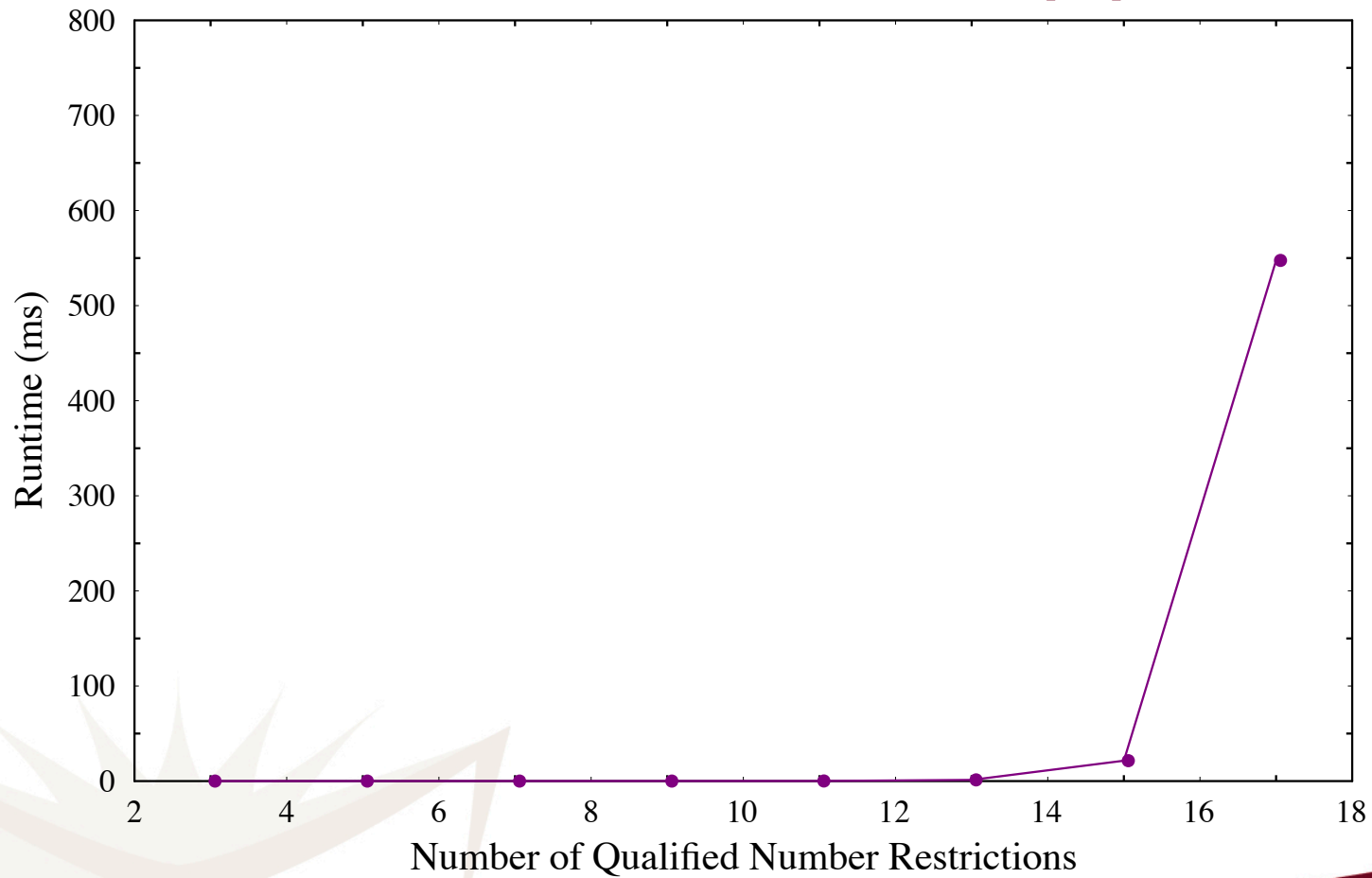
$$RS \sqsubseteq R$$

$$\geq 20 RS \sqcap \geq 10 R.C_1 \sqcap \dots \sqcap \geq 10 R.C_i \sqcap$$

$$\geq 5 R.(\neg C_2 \sqcup \neg C_3) \sqcap \dots \sqcap \geq 5 R.(\neg C_i \sqcup \neg C_{i+1}) \sqcap$$

$$\leq 5 R.(\neg C_1 \sqcup \neg C_2)$$

Number of Qualified Number Restrictions (2)



Conclusion for Hybrid Calculus

- Calculus better informed about implicit cardinality restrictions
- Hybrid calculus essential for
 - Qualified number restrictions with big numbers
 - Conjunctions of interdependent qualified number restrictions
- Could be used to design a practical algorithm for SHQ that is worst-case optimal
- Work by Yu Ding showed a worst-case optimal calculus for SHIQ using global caching and linear integer programming

Future/Related Work for Hybrid Calculus

- Improve prototype by avoiding to process all variables
- Optimize arithmetic reasoner
 - Apply column generation technique for large number of variables in inequations
- Extend to more complex DLs
 - Adding nominals is solved now (hybrid calculus for SHOQ), PhD thesis by Jocelyne Faddoul
 - Adding inverse roles to SHOQ is under investigation
 - Conjecture: extended calculus remains double exponential in the worst case

Current/Future Work

- DL Optimizations essential for practical use
 - New approach: map reasoning with qualified number restrictions and nominals to integer linear programming
 - Subtableau caching for DLs with inverse roles
 - Recognize (tractable) fragments of OWL
- Maintenance and revision of ontologies
 - Tracability/trust/justification of changes required
- Scalability of OWL reasoners will be very important
 - How to deal with huge Abox data (billions of RDF triples), e.g., only 1 msec per individual applied to 1,000,000 individuals
 - Connection to RDF triple stores
 - Inclusion of legacy data stored in traditional DBs
 - Parallelization of reasoning

