A RESOLUTION DECISION PROCEDURE FOR SHOIQ

Yevgeny Kazakov and Boris Motik

The University of Manchester

August 20, 2006

SHOIQ IS A DESCRIPTION LOGIC!

Description Logics:

- a family language for knowledge representation:

$$\mathsf{HappyFather} \equiv \mathsf{Human} \sqcap (\geqslant 2\, \mathsf{hasChild}) \sqcap \\ \sqcap \forall \mathsf{hasChild}.(\mathsf{Famous} \sqcup \mathsf{Rich})$$

DESCRIPTION LOGICS:

- a family language for knowledge representation:

$$\mathsf{HappyFather} \equiv \mathsf{Human} \sqcap (\geqslant 2\, \mathsf{hasChild}) \sqcap \sqcap \forall \mathsf{hasChild}.(\mathsf{Famous} \sqcup \mathsf{Rich})$$

- Distinguished by:
  - Formal semantics (set-theoretic)
  - Decidability for key reasoning problems (satisfiability, subsumption, instance)

Description Logics:

- a family language for knowledge representation:

$$\mathsf{HappyFather} \equiv \mathsf{Human} \sqcap (\geqslant 2\,\mathsf{hasChild}) \sqcap \sqcap \forall \mathsf{hasChild}.(\mathsf{Famous} \sqcup \mathsf{Rich})$$

- Distinguished by:
  - Formal semantics (set-theoretic)
  - Decidability for key reasoning problems (satisfiability, subsumption, instance)
- Related to:
  - (Multi-) Modal Logics, Dynamic Logics
  - Fragments of First-Order Logic (guarded, two-variable)

Application of Description Logics

- Databases (Schema Integration)
- Ontologies (Knowledge Bases):
  - Rigorous description of terms in specific domains (Anatomy, Food, Cars)
  - Access information by performing queries:

$$?-\ \mathsf{Car} \sqcap \exists \mathsf{hasTransmittion}.\mathsf{Automatic} \sqcap \sqcap \exists \mathsf{hasPart}.(\mathsf{Engene} \sqcap (\geqslant 6\, \mathsf{hasPart}.\mathsf{Cylider}))$$

- Semantic Web:
  - Ontology Web Language $\mathcal{OWL}$ (W3C standard)
  - Annotation of enteries using “semantic” mark-up
  - Provide “the meaning” of entries

```
<owl:Class rdf:ID="http:
//www.ontology.com/US/states/WA">Washington</owl>
```

WHAT IS IT ABOUT SHOIQ?

What is it About SHOIQ?

- DL $\mathcal{SHOIQ}$ is a logical counterpart of $\mathcal{OWL\ DL}$
- Development of $\mathcal{OWL\ DL}$-ontologies requires reasoning:
  - computation of class trees (Heart $\sqsubseteq$ Organ)
  - evaluation of queries (?− Car $\sqcap$ . . .)
- reasoning ($\mathcal{OWL}$) = theorem proving ($\mathcal{SHOIQ}$)

What is it About SHOIQ?

- DL $\mathcal{SHOIQ}$ is a logical counterpart of $\mathcal{OWL\ DL}$
- Development of $\mathcal{OWL\ DL}$-ontologies requires reasoning:
  - computation of class trees (Heart $\sqsubseteq$ Organ)
  - evaluation of queries (?– Car $\sqcap$ . . .)
- reasoning ($\mathcal{OWL}$) = theorem proving ($\mathcal{SHOIQ}$)
- Reasoning in $\mathcal{SHOIQ}$ can be reduced to $\mathcal{C}^2$ (the two variable fragment with counting)
  - $\mathcal{C}^2$ is decidable [Grädel et al., 1997]
  - $\mathcal{C}^2$ is NExpTime-compete [Pacholski et al., 2000], [Pratt-Hartmann, 2005]

What is it About SHOIQ?

- DL $\mathcal{SHOIQ}$ is a logical counterpart of $\mathcal{OWL\ DL}$
- Development of $\mathcal{OWL\ DL}$-ontologies requires reasoning:
  - computation of class trees (Heart $\sqsubseteq$ Organ)
  - evaluation of queries (?– Car $\sqcap$ . . .)
- reasoning ($\mathcal{OWL}$) = theorem proving ($\mathcal{SHOIQ}$)
- Reasoning in $\mathcal{SHOIQ}$ can be reduced to $\mathcal{C}^2$ (the two variable fragment with counting)
  - $\mathcal{C}^2$ is decidable [Grädel et al., 1997]
  - $\mathcal{C}^2$ is NExpTime-compete [Pacholski et al., 2000], [Pratt-Hartmann, 2005]
  - but these procedures are not practical (“guess-and-check”)

What is it About SHOIQ?

- DL $\mathcal{SHOIQ}$ is a logical counterpart of $\mathcal{OWL\ DL}$
- Development of $\mathcal{OWL\ DL}$-ontologies requires reasoning:
  - computation of class trees (Heart $\sqsubseteq$ Organ)
  - evaluation of queries (?– Car $\sqcap$ . . .)
- reasoning ($\mathcal{OWL}$) = theorem proving ($\mathcal{SHOIQ}$)
- Reasoning in $\mathcal{SHOIQ}$ can be reduced to $\mathcal{C}^2$ (the two variable fragment with counting)
  - $\mathcal{C}^2$ is decidable [Grädel et al., 1997]
  - $\mathcal{C}^2$ is NExpTime-compete [Pacholski et al., 2000], [Pratt-Hartmann, 2005]
  - but these procedures are not practical (“guess-and-check”)
- [Horrocks & Sattler, 2005] – the first (and the only up until now) goal-directed procedure for $\mathcal{SHOIQ}$

What is it About SHOIQ?

- DL $\mathcal{SHOIQ}$ is a logical counterpart of $\mathcal{OWL\ DL}$
- Development of $\mathcal{OWL\ DL}$-ontologies requires reasoning:
  - computation of class trees (Heart $\sqsubseteq$ Organ)
  - evaluation of queries (?– Car $\sqcap$ . . .)
- reasoning ($\mathcal{OWL}$) = theorem proving ($\mathcal{SHOIQ}$)
- Reasoning in $\mathcal{SHOIQ}$ can be reduced to $\mathcal{C}^2$ (the two variable fragment with counting)
  - $\mathcal{C}^2$ is decidable [Grädel et al., 1997]
  - $\mathcal{C}^2$ is NExpTime-compete [Pacholski et al., 2000], [Pratt-Hartmann, 2005]
  - but these procedures are not practical (“guess-and-check”)
- [Horrocks & Sattler, 2005] – the first (and the only up until now) goal-directed procedure for $\mathcal{SHOIQ}$
- Now we can decide $\mathcal{SHOIQ}$ also by resolution!

Why a $\boxed{\text{Resolution-Based}}$ Decision Procedure for SHOIQ?

Why a Resolution-Based Decision Procedure for SHOIQ?

- different from the tableau-based approach
  - search for proofs vs. search for models

Why a Resolution-Based Decision Procedure for SHOIQ?

- different from the tableau-based approach
  - search for proofs vs. search for models
- likely to behave differently for different types of problems:
  - Tableau is good for reasoning with large schema (terminologies)
  - Resolution is useful for reasoning with large data (assertions) [Hustadt, Motik & Sattler, 2004]

Description Logics: Syntax

Axioms

$$\text{Researcher} \equiv \text{Human} \sqcap \forall \text{produce}.\text{Paper}$$

$$\text{Researcher}(\text{Rob})$$

DESCRIPTION LOGICS: SYNTAX

AXIOMS

$\text{Researcher} \equiv \text{Human} \sqcap \forall \text{produce}.\text{Paper}$ ◀ Terminology

$\text{Researcher}(\text{Rob})$ ◀ Assertions

Description Logics: Syntax

Axioms

Researcher $\equiv$ Human $\sqcap$ $\forall$produce.Paper ◄ Terminology

Researcher (Rob) ◄ Assertions

- Basic building blocks of DLs:
  - Concept names
  - Role names
  - Individuals
  - Operators

DESCRIPTION LOGICS: SYNTAX

AXIOMS

$\text{Researcher} \equiv \text{Human} \sqcap \forall \text{produce}.\text{Paper}$	Terminology
$\text{Researcher}(\text{Rob})$	Assertions

- Basic building blocks of DLs:
  - Concept names – sets: Researcher, Human, Paper
  - Role names
  - Individuals
  - Operators

Description Logics: Syntax

Axioms

$\text{Researcher} \equiv \text{Human} \sqcap \forall \text{produce}.\text{Paper}$ ◀ Terminology

$\text{Researcher}(\text{Rob})$ ◀ Assertions

- Basic building blocks of DLs:
  - Concept names – sets: Researcher, Human, Paper
  - Role names – binary relations: produces
  - Individuals
  - Operators

Description Logics: Syntax

Axioms

$\text{Researcher} \equiv \text{Human} \sqcap \forall \text{produce}.\text{Paper}$ ◀ Terminology

$\text{Researcher}(\text{Rob})$ ◀ Assertions

- Basic building blocks of DLs:
  - Concept names – sets: Researcher, Human, Paper
  - Role names – binary relations: produces
  - Individuals – constants: Rob
  - Operators

Description Logics: Syntax

Axioms

$\text{Researcher} \equiv \text{Human} \sqcap \forall \text{produce}.\text{Paper}$ ◀ Terminology

$\text{Researcher}(\text{Rob})$ ◀ Assertions

- Basic building blocks of DLs:
  - Concept names – sets: Researcher, Human, Paper
  - Role names – binary relations: produces
  - Individuals – constants: Rob
  - Operators – logical constructors: $C_1 \sqcap C_2$, $\forall r.C$, $A \equiv C$

Description Logics: Semantics

Axioms

$\text{Researcher} \equiv \text{Human} \sqcap \forall \text{produces}.\text{Paper}$

$\text{Researcher}(\text{Rob})$

- Basic building blocks of DLs:
  - Concept names: Researcher, Human, Paper
  - Role names: produces
  - Individuals: Rob
  - Operators: $C_1 \sqcap C_2$, $\forall r.C$, $A \equiv C$

Description Logics: Semantics

Axioms

Researcher $\equiv$ Human $\sqcap$ $\forall$produces.Paper

Researcher(Rob)

- Basic building blocks of DLs:
  - Concept names — Researcher, Human, Paper
    ⇝ unary atoms: — Researcher(x), Human(x), Paper(x)
  - Role names — produces
  - Individuals — Rob
  - Operators — $C_1 \sqcap C_2$, $\forall r.C$, $A \equiv C$

Description Logics: Semantics

Axioms

$\text{Researcher} \equiv \text{Human} \sqcap \forall \text{produces}.\text{Paper}$

$\text{Researcher}(\text{Rob})$

- Basic building blocks of DLs:
  - Concept names
    ⇝ unary atoms: Researcher(x), Human(x), Paper(x)
  - Role names produces
    ⇝ binary atoms: produces(x,y)
  - Individuals Rob
  - Operators $C_1 \sqcap C_2$, $\forall r.C$, $A \equiv C$

Description Logics: Semantics

Axioms

$\text{Researcher} \equiv \text{Human} \sqcap \forall \text{produces}.\text{Paper}$

$\text{Researcher}(\text{Rob})$

- Basic building blocks of DLs:
  - Concept names
    $\rightsquigarrow$ unary atoms: Researcher(x), Human(x), Paper(x)
  - Role names
    $\rightsquigarrow$ binary atoms: produces(x,y)
  - Individuals Rob
    $\rightsquigarrow$ constants: Rob
  - Operators $C_1 \sqcap C_2$, $\forall r.C$, $A \equiv C$

Description Logics: Semantics

Axioms

$\text{Researcher} \equiv \text{Human} \sqcap \forall \text{produces}.\text{Paper}$

$\text{Researcher}(\text{Rob})$

- Basic building blocks of DLs:
  - Concept names
    $\rightsquigarrow$ unary atoms: Researcher(x), Human(x), Paper(x)
  - Role names
    $\rightsquigarrow$ binary atoms: produces(x,y)
  - Individuals
    $\rightsquigarrow$ constants: Rob
  - Operators
    $\rightsquigarrow$ constructors:

$C_1 \sqcap C_2$,	$\forall r.C$,	$A \equiv C$
$C_1(x) \wedge C_2(x)$,	$\forall y.[r(x,y) \rightarrow C(y)]$,	$\forall x.[A(x) \equiv C(x)]$

Description Logics: Semantics

Axioms

$$\mathsf{Researcher}(x) \equiv \mathsf{Human}(x) \sqcap \forall y.[\mathsf{produces}(x,y) \rightarrow \mathsf{Paper}(y)]$$

$$\mathsf{Researcher}(\mathrm{Rob})$$

- Basic building blocks of DLs:
  - Concept names
    $\rightsquigarrow$ unary atoms: Researcher(x), Human(x), Paper(x)
  - Role names
    $\rightsquigarrow$ binary atoms: produces(x,y)
  - Individuals
    $\rightsquigarrow$ constants: Rob
  - Operators
    $\rightsquigarrow$ constructors: $C_1(x) \wedge C_2(x)$, $\forall y.[r(x,y) \rightarrow C(y)]$, $\forall x.[A(x) \equiv C(x)]$

Hierarchy of DLs

$$\left.\begin{array}{ll} \text{Basic Description Logic } \mathcal{ALC}\text{:} & \sqcap, \sqcup, \neg, \forall r.C, \exists r.C, \sqsubseteq \\ \text{Transitive Roles:} & \text{Transitive}(r) \end{array}\right\} \mathcal{S}$$

The University of Manchester

Hierarchy of DLs

- Basic Description Logic $\mathcal{ALC}$: $\sqcap, \sqcup, \neg, \forall r.C, \exists r.C, \sqsubseteq$ } $\mathcal{S}$
- Transitive Roles: Transitive(r) } $\mathcal{S}$
- Role Hierarchies: $r_1 \sqsubseteq r_2$ $\mathcal{H}$
- Inverse Roles: r_2^{-} $\mathcal{I}$
- Qualified Number Restrictions: $(\geqslant n\, r.C)$, $(\leqslant n\, r.C)$ $\mathcal{Q}$

$= \mathcal{SHIQ}$

Hierarchy of DLs

- Basic Description Logic $\mathcal{ALC}$: $\sqcap, \sqcup, \neg, \forall r.C, \exists r.C, \sqsubseteq$
- Transitive Roles: Transitive(r) $\Big\} \mathcal{S}$
- Role Hierarchies: $r_1 \sqsubseteq r_2$ $\mathcal{H}$
- Inverse Roles: r_2^- $\mathcal{I}$
- Qualified Number Restrictions: $(\geqslant n\, r.C)$, $(\leqslant n\, r.C)$ $\mathcal{Q}$

$= \mathcal{SHIQ}$

- Nominals: $\{i\}$ $\mathcal{O}$

$= \mathcal{SHOIQ}$

Expressive Power of $\mathcal{SHOIQ}$

- Cardinality restrictions: $|C| \leq n$
  - $C \sqsubseteq \{i_1\} \sqcup \{i_2\} \sqcup \cdots \sqcup \{i_n\}$ $\qquad |C| \leq n$

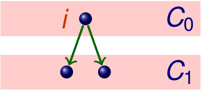
Expressive Power of $\mathcal{SHOIQ}$

- Cardinality restrictions: $|C| \leq n$, $|C| \geq n$
  - $C \sqsubseteq \{i_1\} \sqcup \{i_2\} \sqcup \cdots \sqcup \{i_n\}$ $\qquad |C| \leq n$
  - $C \sqsupseteq \{i_1\} \sqcup \{i_2\} \sqcup \cdots \sqcup \{i_n\}$ $\qquad |C| \geq n$

    $\{i_p\} \sqcap \{i_q\} \sqsubseteq \bot, \quad p < q$

Expressive Power of $\mathcal{SHOIQ}$

- Cardinality restrictions: $|C| \leq n$, $|C| \geq n$
- Large cardinality restrictions:
  - $C_0 \sqsupseteq \{i\}$
    $C_0 \sqsubseteq (\geqslant 2\, r.C_1)$

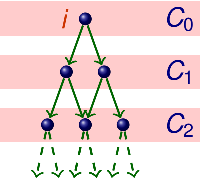


$|C_0| \geq 1$
$|C_1| \geq 2$

The University of Manchester

Expressive Power of $\mathcal{SHOIQ}$

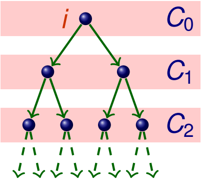
- Cardinality restrictions: $|C| \leq n$, $|C| \geq n$
- Large cardinality restrictions:
  - $C_0 \sqsupseteq \{i\}$
    $C_0 \sqsubseteq (\geqslant 2\, r.C_1)$
    $C_1 \sqsubseteq (\geqslant 2\, r.C_2)$
    $\ldots$
    $C_{n-1} \sqsubseteq (\geqslant 2\, r.C_n)$



$|C_0| \geq 1$
$|C_1| \geq 2$

Expressive Power of $\mathcal{SHOIQ}$

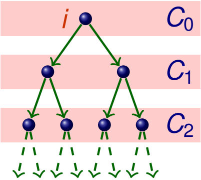
- Cardinality restrictions: $|C| \leq n$, $|C| \geq n$
- Large cardinality restrictions:
  - $C_0 \sqsupseteq \{i\}$
    $C_0 \sqsubseteq (\geqslant 2\, r.C_1)$
    $C_1 \sqsubseteq (\geqslant 2\, r.C_2)$
    $\dots$
    $C_{n-1} \sqsubseteq (\geqslant 2\, r.C_n)$
    $\top \sqsubseteq (\leqslant 1\, r^-.\top)$



$|C_0| \geq 1$
$|C_1| \geq 2$

Expressive Power of $\mathcal{SHOIQ}$

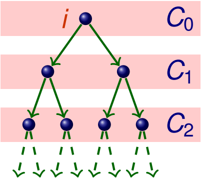
- Cardinality restrictions: $|C| \leq n$, $|C| \geq n$
- Large cardinality restrictions: $|C| \geq 2^n$
  - $C_0 \sqsupseteq \{i\}$
    $C_0 \sqsubseteq (\geqslant 2\, r.C_1)$
    $C_1 \sqsubseteq (\geqslant 2\, r.C_2)$
    $\dots$
    $C_{n-1} \sqsubseteq (\geqslant 2\, r.C_n)$
    $\top \sqsubseteq (\leqslant 1\, r^-.\top)$



$|C_0| \geq 1$
$|C_1| \geq 2$
$\dots$
$|C_n| \geq 2^n$

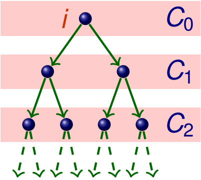
Expressive Power of $\mathcal{SHOIQ}$

- Cardinality restrictions: $|C| \leq n$, $|C| \geq n$
- Large cardinality restrictions: $|C| \geq 2^n$, $|C| \leq 2^n$
  - $C_0 \sqsupseteq \{i\}$
    $C_0 \sqsubseteq (\geqslant 2\, r.C_1)$
    $C_1 \sqsubseteq (\geqslant 2\, r.C_2)$
    $\cdots$
    $C_{n-1} \sqsubseteq (\geqslant 2\, r.C_n)$
    $\top \sqsubseteq (\leqslant 1\, r^-.\top)$



$|C_0| \geq 1$
$|C_1| \geq 2$
$\cdots$
$|C_n| \geq 2^n$

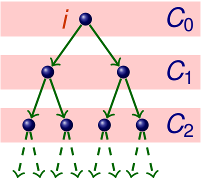
  - $C_0 \sqsubseteq \{i\}$
    $C_1 \sqsubseteq (\geqslant 1\, r^-.C_0)$
    $C_2 \sqsubseteq (\geqslant 1\, r^-.C_1)$
    $\cdots$
    $C_n \sqsubseteq (\geqslant 1\, r^-.C_{n-1})$
    $\top \sqsubseteq (\leqslant 2\, r.\top)$



$|C_0| \leq 1$
$|C_1| \leq 2$
$\cdots$
$|C_n| \leq 2^n$

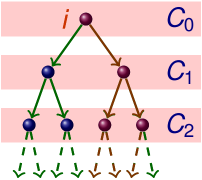
Expressive Power of $\mathcal{SHOIQ}$

- Cardinality restrictions: $|C| \leq n$, $|C| \geq n$
- Large cardinality restrictions: $|C| \geq 2^n$, $|C| \leq 2^n$
  - $C_0 \sqsupseteq \{i\}$
    $C_0 \sqsubseteq (\geqslant 2\, r.C_1)$
    $C_1 \sqsubseteq (\geqslant 2\, r.C_2)$
    $\cdots$
    $C_{n-1} \sqsubseteq (\geqslant 2\, r.C_n)$
    $\top \sqsubseteq (\leqslant 1\, r^-.\top)$



$|C_0| \geq 1$
$|C_1| \geq 2$
$\cdots$
$|C_n| \geq 2^n$

  - $C_0 \sqsubseteq \{i\}$
    $C_1 \sqsubseteq (\geqslant 1\, r^-.C_0)$
    $C_2 \sqsubseteq (\geqslant 1\, r^-.C_1)$
    $\cdots$
    $C_n \sqsubseteq (\geqslant 1\, r^-.C_{n-1})$
    $\top \sqsubseteq (\leqslant 2\, r.\top)$



$|C_0| \leq 1$
$|C_1| \leq 2$
$\cdots$
$|C_n| \leq 2^n$

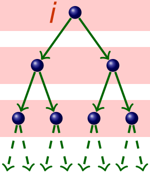
Expressive Power of $\mathcal{SHOIQ}$

- Cardinality restrictions: $|C| \leq n$, $|C| \geq n$
- Large cardinality restrictions: $|C| \geq 2^n$, $|C| \leq 2^n$
- Huge cardinality restrictions: $|C| \geq 2^{2^n}$, $|C| \leq 2^{2^n}$

Expressive Power of $\mathcal{SHOIQ}$

- Cardinality restrictions: $|C| \leq n$, $|C| \geq n$
- Large cardinality restrictions: $|C| \geq 2^n$, $|C| \leq 2^n$
- Huge cardinality restrictions: $|C| \geq 2^{2^n}$, $|C| \leq 2^{2^n}$

$$\mathsf{B}_n \sqcap \cdots \sqcap \mathsf{B}_0 \sqsubseteq \{i\}$$
$$\top \sqsubseteq (\geqslant 1\, r^{-}.\top)$$
$$\top \sqsubseteq (\leqslant 2\, r\ .\top)$$



$\mathsf{B}_n \sqcap \cdots \sqcap \mathsf{B}_1 \sqcap \mathsf{B}_0 \quad = 0$

$\mathsf{B}_n \sqcap \cdots \sqcap \mathsf{B}_1 \sqcap \neg\mathsf{B}_0 \quad = 1$

$\mathsf{B}_n \sqcap \cdots \sqcap \neg\mathsf{B}_1 \sqcap \mathsf{B}_0 \quad = 2$

$\cdots$

$= 2^n$

$$\mathsf{B}_0 \sqsubseteq \forall r.\neg\mathsf{B}_0$$
$$\neg\mathsf{B}_0 \sqsubseteq \forall r.\ \mathsf{B}_0$$
$$\mathsf{B}_{i+1} \sqcap \mathsf{B}_i \sqsubseteq \forall r.\ \mathsf{B}_{i+1} \qquad \text{– bits "count" over } r$$
$$\neg\mathsf{B}_{i+1} \sqcap \mathsf{B}_i \sqsubseteq \forall r.\neg\mathsf{B}_{i+1}$$
$$\mathsf{B}_{i+1} \sqcap \neg\mathsf{B}_i \sqsubseteq \forall r.[(\neg\mathsf{B}_{i+1} \sqcap \mathsf{B}_i) \sqcup (\mathsf{B}_{i+1} \sqcap \neg\mathsf{B}_i)]$$
$$\neg\mathsf{B}_{i+1} \sqcap \neg\mathsf{B}_i \sqsubseteq \forall r.[(\mathsf{B}_{i+1} \sqcap \mathsf{B}_i) \sqcup (\neg\mathsf{B}_{i+1} \sqcap \neg\mathsf{B}_i)]$$

RESOLUTION-BASED PROCEDURES: THE BASIC PRINCIPLES

- Invented by Joyner Jr. (1976)
- Allows one to use existing automated theorem provers (SPASS, VAMPIRE) as decision procedures
- The general idea is as follows:
  1. Define a clause class for the target fragment
  2. Show that this class is closed under inferences
  3. Show the class is finite for a fixed signature
- Many decision procedures are based on this principle:
  - clause classes $\mathcal{E}$, $\mathcal{S}^+$, $\mathcal{E}^+$, etc. [Fermüller et al., 1993]
  - modal logics [Schmidt, 1997], [Hustadt, 1999],
  - fragments of first-order logic [Bachmair et al., 1993], [Ganzinger & de Nivelle, 1999].

How to Turn Resolution Into a Decision Procedure?

- Tweak the parameters of a prover (ordering and selection function) so that the size of clauses does not grow

How to Turn Resolution Into a Decision Procedure?

- Tweak the parameters of a prover (ordering and selection function) so that the size of clauses does not grow
- Problematic situations:

Example

$A(c)$ $\quad\rightsquigarrow\quad$ $\underline{A(c)}$ $\qquad \neg\underline{A(x)} \vee A(f(x))$

$A \sqsubseteq \exists r.A$ $\qquad\qquad\qquad$ $\underline{A(f(c))}$

$\underline{A(f(f(c)))}$

$\ldots$

- Problem: the depth grows

How to Turn Resolution Into a Decision Procedure?

- Tweak the parameters of a prover (ordering and selection function) so that the size of clauses does not grow
- Problematic situations:

Example

$A(c)$ $\quad\leadsto\quad$ $\underline{A(c)}$ $\quad \underline{\neg A(x)} \vee A(f(x))$

$A \sqsubseteq \exists r.A$ $\quad$ $\underline{A(f(c))}$

$\underline{A(f(f(c)))}$

$\ldots$

- Problem: the depth grows
- The reason: the selected literal is not the deepest one

How to Turn Resolution Into a Decision Procedure?

- Tweak the parameters of a prover (ordering and selection function) so that the size of clauses does not grow
- Problematic situations:

Example

$$A(c) \qquad\qquad \rightsquigarrow \qquad\qquad \underline{A(c)} \qquad \neg\underline{A(x)} \vee \boxed{A(f(x))}$$

$$A \sqsubseteq \exists r.A \qquad\qquad \underline{A(f(c))}$$

$$\underline{A(f(f(c)))}$$

$$\ldots$$

- Problem: the depth grows
- The reason: the selected literal is not the deepest one
- Solution: resolve on the depest literal

How to Turn Resolution Into a Decision Procedure?

- Tweak the parameters of a prover (ordering and selection function) so that the size of clauses does not grow
- Problematic situations:

Example

$$A(c) \qquad\qquad \rightsquigarrow \qquad \underline{A(c)} \qquad \neg\underline{A(x)} \vee \neg R(x,y) \vee A(y)$$
$$A \sqsubseteq \forall R.A \qquad\qquad\qquad \neg R(c, y_1) \vee \underline{A(y_1)}$$
$$\neg R(c, y_1) \vee \neg R(y_1, y_2) \vee \underline{A(y_2)}$$
$$\ldots$$

- Problem: variables got duplicated

How to Turn Resolution Into a Decision Procedure?

- Tweak the parameters of a prover (ordering and selection function) so that the size of clauses does not grow
- Problematic situations:

Example

$A(c)$ $\quad\rightsquigarrow\quad$ $\underline{A(c)} \qquad \underline{\neg A(x)} \vee \neg R(x,y) \vee A(y)$

$A \sqsubseteq \forall R.A$ $\qquad\qquad\qquad \neg R(c, y_1) \vee \underline{A(y_1)}$

$$\neg R(c, y_1) \vee \neg R(y_1, y_2) \vee \underline{A(y_2)}$$

$$\ldots$$

- Problem: variables got duplicated
- The reason: the unified expression does not contain all variables of the clause

How to Turn Resolution Into a Decision Procedure?

- Tweak the parameters of a prover (ordering and selection function) so that the size of clauses does not grow
- Problematic situations:

Example

$$
\begin{array}{lcll}
A(c) & \rightsquigarrow & \underline{A(c)} & \neg \underline{A(x)} \vee \boxed{\neg R(x,y)} \vee A(y) \\
A \sqsubseteq \forall R.A & & \neg R(c, y_1) \vee \underline{A(y_1)} & \\
 & & \neg R(c, y_1) \vee \neg R(y_1, y_2) \vee \underline{A(y_2)} & \\
 & & \ldots &
\end{array}
$$

- Problem: variables got duplicated
- The reason: the unified expression does not contain all variables of the clause
- Solution: resolve on the expression with all variables

How to Turn Resolution Into a Decision Procedure?

- Tweak the parameters of a prover (ordering and selection function) so that the size of clauses does not grow
- Problematic situations: depth or no. of variables grows
- Decidability is typically a consequence that all expressions in clauses are covering:

How to Turn Resolution Into a Decision Procedure?

- Tweak the parameters of a prover (ordering and selection function) so that the size of clauses does not grow
- Problematic situations: depth or no. of variables grows
- Decidability is typically a consequence that all expressions in clauses are covering:
- every functional term of an expression contains all variables

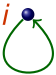
Example

$\neg A(x) \vee \mathrm{r}(x, \mathrm{f}(x, y))$	term $\mathrm{f}(x, y)$ is covering
$\neg A(x) \vee x \simeq c$	term c is not covering

DIFFICULTIES WITH $\mathcal{SHOIQ}$ IN RESOLUTION

EXAMPLE

$O \sqsubseteq \{i\}$	$\leadsto$	1. $\neg O(x) \vee x \simeq i$
$O \sqsubseteq \exists r.O$	$\leadsto$	2. $\neg O(x) \vee r(x, f(x))$
	$\leadsto$	3. $\neg O(x) \vee O(f(x))$
$\top \sqsubseteq \leqslant 1\, r^{-}.\top$	$\leadsto$	4. $\neg r(x, y) \vee x \simeq g(y)$



DIFFICULTIES WITH $\mathcal{SHOIQ}$ IN RESOLUTION

EXAMPLE

$$
\begin{array}{lcll}
\mathsf{O} \sqsubseteq \{i\} & \rightsquigarrow & 1.\ \neg\underline{\mathsf{O}(x)} \vee x \simeq i & \text{– not covering} \\
\mathsf{O} \sqsubseteq \exists \mathsf{r}.\mathsf{O} & \rightsquigarrow & 2.\ \neg\underline{\mathsf{O}(x)} \vee \underline{\mathsf{r}(x, f(x))} & \\
 & \rightsquigarrow & 3.\ \neg\mathsf{O}(x) \vee \underline{\mathsf{O}(f(x))} & \\
\top \sqsubseteq \leqslant 1\, \mathsf{r}^{-}.\top & \rightsquigarrow & 4.\ \neg\underline{\mathsf{r}(x, y)} \vee x \simeq g(y) & \text{– not covering}
\end{array}
$$

DIFFICULTIES WITH $\mathcal{SHOIQ}$ IN RESOLUTION

EXAMPLE

$$
\begin{array}{llll}
\mathsf{O} \sqsubseteq \{i\} & \rightsquigarrow & 1. & \neg \underline{\mathsf{O}(x)} \vee x \simeq i \\
\mathsf{O} \sqsubseteq \exists \mathsf{r}.\mathsf{O} & \rightsquigarrow & 2. & \neg \mathsf{O}(x) \vee \underline{\mathsf{r}(x, f(x))} \\
 & \rightsquigarrow & 3. & \neg \mathsf{O}(x) \vee \underline{\mathsf{O}(f(x))} \\
\top \sqsubseteq \leqslant 1\, \mathsf{r}^{-}.\top & \rightsquigarrow & 4. & \neg \underline{\mathsf{r}(x, y)} \vee x \simeq g(y)
\end{array}
$$

$$
\begin{array}{l}
\mathsf{OR}[1;3] : 5.\ \neg \mathsf{O}(x) \vee \underline{f(x)} \simeq i \\
\mathsf{OR}[2;4] : 6.\ \neg \mathsf{O}(x) \vee x \simeq g(\underline{f(x)}) \\
\mathsf{OP}[5;6] : 7.\ \neg \mathsf{O}(x) \vee x \simeq g(i)
\end{array}
$$

Difficulties with $\mathcal{SHOIQ}$ in Resolution

Example

$\mathsf{O} \sqsubseteq \{i\}$	$\leadsto$	1. $\neg\underline{\mathsf{O}(x)} \vee x \simeq i$
$\mathsf{O} \sqsubseteq \exists \mathsf{r}.\mathsf{O}$	$\leadsto$	2. $\neg\underline{\mathsf{O}(x)} \vee \underline{\mathsf{r}(x, f(x))}$
	$\leadsto$	3. $\neg\mathsf{O}(x) \vee \underline{\mathsf{O}(f(x))}$
$\top \sqsubseteq \leqslant 1\, \mathsf{r}^-.\top$	$\leadsto$	4. $\neg\underline{\mathsf{r}(x, y)} \vee x \simeq g(y)$

OR[1; 3] : 5. $\neg\mathsf{O}(x) \vee \underline{f(x)} \simeq i$

OR[2; 4] : 6. $\neg\mathsf{O}(x) \vee x \simeq g(\underline{f(x)})$

OP[5; 6] : 7. $\neg\underline{\mathsf{O}(x)} \vee x \simeq g(i)$ of the same form

Difficulties with $\mathcal{SHOIQ}$ in resolution

Example

$O \sqsubseteq \{i\}$	$\rightsquigarrow$	1. $\neg \underline{O(x)} \vee x \simeq i$ ◀
$O \sqsubseteq \exists r.O$	$\rightsquigarrow$	2. $\neg O(x) \vee \underline{r(x, f(x))}$
	$\rightsquigarrow$	3. $\neg O(x) \vee \underline{O(f(x))}$
$\top \sqsubseteq \leqslant 1\, r^{-}.\top$	$\rightsquigarrow$	4. $\neg \underline{r(x, y)} \vee x \simeq g(y)$

OR[1; 3] : 5. $\neg O(x) \vee \underline{f(x)} \simeq i$

OR[2; 4] : 6. $\neg O(x) \vee x \simeq g(\underline{f(x)})$

OP[5; 6] : 7. $\neg \underline{O(x)} \vee x \simeq g(i)$ ◀ of the same form

... 8. $\neg \underline{O(x)} \vee x \simeq g(g(i))$ ◀ produces deeper

... 9. $\neg \underline{O(x)} \vee x \simeq g(g(g(i)))$ ◀ clauses

Difficulties with $\mathcal{SHOIQ}$ in resolution

Example

$O \sqsubseteq \{i\}$	$\rightsquigarrow$	1. $\neg \underline{O(x)} \vee x \simeq i$
$O \sqsubseteq \exists r.O$	$\rightsquigarrow$	2. $\neg \underline{O(x)} \vee \underline{r(x, f(x))}$
	$\rightsquigarrow$	3. $\neg O(x) \vee \underline{O(f(x))}$
$\top \sqsubseteq \leqslant 1\, r^{-}.\top$	$\rightsquigarrow$	4. $\neg \underline{r(x, y)} \vee x \simeq g(y)$

OR[1; 3] : 5. $\neg O(x) \vee \underline{f(x)} \simeq i$

OR[2; 4] : 6. $\neg O(x) \vee x \simeq g(\underline{f(x)})$

OP[5; 6] : 7. $\neg \underline{O(x)} \vee x \simeq g(i)$

add new: 8. $\neg \underline{O(x)} \vee i \simeq g(i)$ consequence of 1 and 7

Difficulties with $\mathcal{SHOIQ}$ in Resolution

Example

$O \sqsubseteq \{i\}$ $\rightsquigarrow$ 1. $\neg \underline{O(x)} \vee x \simeq i$

$O \sqsubseteq \exists r.O$ $\rightsquigarrow$ 2. $\neg \underline{O(x)} \vee r(x, f(x))$

$\rightsquigarrow$

$\top \sqsubseteq \leqslant 1\, r^{-}.\top$ $\rightsquigarrow$

Redundancy for Clauses

A clause is redundant if it follows from smaller clauses

$OR[1;3]$: 5. $\neg O(x) \vee \underline{f(x)} \simeq i$

$OR[2;4]$: 6. $\neg O(x) \vee x \simeq g(\underline{f(x)})$

$OP[5;6]$: 7. $\boxed{\neg \underline{O(x)} \vee x \simeq g(i)}$ follows from 1 and 8

8. $\neg \underline{O(x)} \vee i \simeq g(i)$ consequence of 1 and 7

Difficulties with $\mathcal{SHOIQ}$ in Resolution

Example

$O \sqsubseteq \{i\}$ $\rightsquigarrow$ 1. $\neg O(x) \vee x \simeq i$

$O \sqsubseteq \exists r.O$ $\rightsquigarrow$ 2. $\neg O(x) \vee r(x, f(x))$

$\rightsquigarrow$

$\top \sqsubseteq \leqslant 1\, r^{-}.\top$ $\rightsquigarrow$

Redundancy for Clauses

A clause is redundant if it follows from smaller clauses

OR[1; 3] : 5. $\neg O(x) \vee f(x) \simeq i$

OR[2; 4] : 6. $\neg O(x) \vee x \simeq g(f(x))$

OP[5; 6] : 7. $\boxed{\neg O(x) \vee x \simeq g(i)}$ follows from 1 and 8

larger than 1,

8. $\neg O(x) \vee i \simeq g(i)$

Difficulties with $\mathcal{SHOIQ}$ in Resolution

Example

$O \sqsubseteq \{i\} \rightsquigarrow$ 1. $\neg \underline{O(x)} \vee x \simeq i$

$O \sqsubseteq \exists r.O \rightsquigarrow$

$\rightsquigarrow$

$\top \sqsubseteq \leqslant 1\, r^{-}.\top \rightsquigarrow$

Redundancy for Clauses

A clause is redundant if it follows from smaller clauses

OR[1; 3] : 5. $\neg O(x) \vee \underline{f(x)} \simeq i$

OR[2; 4] : 6. $\neg O(x) \vee x \simeq g(\underline{f(x)})$

OP[5; 6] : 7. $\boxed{\neg \underline{O(x)} \vee x \simeq g(i)}$

follows from 1 and 8

larger than 1,

but not larger than 8!

8. $\neg \underline{O(x)} \vee i \simeq g(i)$

Difficulties with $\mathcal{SHOIQ}$ in resolution

Example

$\mathsf{O} \sqsubseteq \{i\}$ $\rightsquigarrow$ 1. $\neg \underline{\mathsf{O}(x)} \vee x \simeq i$

$\mathsf{O} \sqsubseteq \exists \mathsf{r}.\mathsf{O}$ $\rightsquigarrow$ 2. $\underline{\mathsf{O}(x)} \vee \mathsf{r}(x, f(x))$

$\rightsquigarrow$

$\top \sqsubseteq\ \leqslant 1\, \mathsf{r}^{-}.\top$ $\rightsquigarrow$

OR[1; 3] : 5. $\neg \mathsf{O}(x) \vee \underline{f(x)} \simeq i$

OR[2; 4] : 6. $\neg \mathsf{O}(x) \vee x \simeq g(\underline{f(x)})$

OP[5; 6] : 7. $\neg \underline{\mathsf{O}(x)} \vee x \simeq g(i)$

Redundancy for Clauses

A clause is redundant if it follows from **smaller** clauses

DIFFICULTIES WITH $\mathcal{SHOIQ}$ IN RESOLUTION

EXAMPLE

$\mathsf{O} \sqsubseteq \{i\}$ $\rightsquigarrow$ 1. $\neg\underline{\mathsf{O}(x)} \vee x \simeq i$

$\mathsf{O} \sqsubseteq \exists \mathsf{r}.\mathsf{O}$ $\rightsquigarrow$ 2. $\neg\underline{\mathsf{O}(x)} \vee r(x, f(x))$

$\rightsquigarrow$

$\top \sqsubseteq \leqslant 1\, \mathsf{r}^{-}.\top$ $\rightsquigarrow$

REDUNDANCY FOR CLAUSES

A clause is redundant if it follows from smaller clauses

OR[1; 3] : 5. $\neg\mathsf{O}(x) \vee \underline{f(x)} \simeq i$

OR[2; 4] : 6. $\neg\mathsf{O}(x) \vee x \simeq g(\underline{f(x)})$

OP[5; 6] : 7. $\neg\underline{\mathsf{O}(x)} \vee x \simeq g(i)$ wait a bit...

OR[7; 3] : 8. $\neg\mathsf{O}(x) \vee \underline{f(x)} \simeq g(i)$

Difficulties with $\mathcal{SHOIQ}$ in Resolution

Example

$O \sqsubseteq \{i\}$ $\rightsquigarrow$ 1. $\neg \underline{O(x)} \vee x \simeq i$

$O \sqsubseteq \exists r.O$ $\rightsquigarrow$ 2. $\neg \underline{O(x)} \vee r(x, f(x))$

$\rightsquigarrow$

$\top \sqsubseteq \leqslant 1\, r^{-}.\top$ $\rightsquigarrow$

Redundancy for Clauses

A clause is redundant if it follows from smaller clauses

$OR[1;3]$: 5. $\neg O(x) \vee \underline{f(x)} \simeq i$

$OR[2;4]$: 6. $\neg O(x) \vee x \simeq g(\underline{f(x)})$

$OP[5;6]$: 7. $\neg \underline{O(x)} \vee x \simeq g(i)$ wait a bit...

$OR[7;3]$: 8. $\neg O(x) \vee \underline{f(x)} \simeq g(i)$

add: 9. $\neg \underline{O(x)} \vee i \simeq g(i)$ ◄ consequence of 5 and 8

Difficulties with $\mathcal{SHOIQ}$ in resolution

Example

$O \sqsubseteq \{i\}$ $\rightsquigarrow$ 1. $\neg O(x) \vee x \simeq i$

$O \sqsubseteq \exists r.O$ $\rightsquigarrow$ 2. $\neg O(x) \vee r(x, f(x))$

$\rightsquigarrow$

$\top \sqsubseteq \leqslant 1\, r^-.\top$ $\rightsquigarrow$

Redundancy for Clauses

A clause is redundant if it follows from smaller clauses

$OR[1;3]$: 5. $\neg O(x) \vee f(x) \simeq i$

$OR[2;4]$: 6. $\neg O(x) \vee x \simeq g(f(x))$

$OP[5;6]$: 7. $\neg O(x) \vee x \simeq g(i)$ wait a bit...

$OR[7;3]$: 8. $\boxed{\neg O(x) \vee f(x) \simeq g(i)}$ follows from 5 and 9 larger than 5, and larger than 9!

9. $\neg O(x) \vee i \simeq g(i)$ consequence of 5 and 8

Difficulties with $\mathcal{SHOIQ}$ in Resolution

Example

$O \sqsubseteq \{i\}$ $\rightsquigarrow$ 1. $\neg \underline{O(x)} \vee x \simeq i$

$O \sqsubseteq \exists r.O$ $\rightsquigarrow$ 2. $\neg O(x) \vee r(x, f(x))$

$\rightsquigarrow$

$\top \sqsubseteq \leqslant 1\, r^{-}.\top$ $\rightsquigarrow$

Redundancy for Clauses

A clause is redundant if it follows from smaller clauses

OR[1; 3] : 5. $\neg O(x) \vee \underline{f(x)} \simeq i$

OR[2; 4] : 6. $\neg O(x) \vee x \simeq g(\underline{f(x)})$

OP[5; 6] : 7. $\neg \underline{O(x)} \vee x \simeq g(i)$ wait a bit...

OR[7; 3] : $\boxed{\cancel{8. \neg O(x) \vee \underline{f(x)} \simeq g(i)}}$ remove!

9. $\neg \underline{O(x)} \vee i \simeq g(i)$ consequence of 5 and 8

- The saturation procedure terminates!

Nominal Generation

- The idea is developed into a new simplification rule that introduces constants

Nominal Generation

$$\frac{\alpha(x) \vee \bigvee_{i=1}^{n} f(x) \simeq t_i}{\begin{array}{r} \alpha(x) \vee \bigvee_{i=1}^{n} f(x) \simeq c_i \\ \alpha(x) \vee \bigvee_{j=1}^{n} c_i \simeq t_j \\ 1 \leq i \leq n \end{array}}$$

where (i) c_i are fresh constants for t_i and α

Nominal Generation

- The idea is developed into a new simplification rule that introduces constants
- the constants are reused when the rule has been applied to $\alpha(x)$ and $f(x)$ before.

Nominal Generation

$$\frac{\alpha(x) \vee \bigvee_{i=1}^{n} f(x) \simeq t_i}{\begin{array}{r}\alpha(x) \vee \bigvee_{i=1}^{k} f(x) \simeq c_i \\ \alpha(x) \vee \bigvee_{j=1}^{n} c_i \simeq t_j \\ 1 \leq i \leq k\end{array}}$$

where (i) c_i are fresh constants for t_i and α , (ii) k=n for the first application of rule for $\alpha(x)$ and $f(x)$, otherwise k and c_i are reused

Nominal Generation

- The idea is developed into a new simplification rule that introduces constants
- the constants are reused when the rule has been applied to $\alpha(x)$ and $f(x)$ before.
- there is a second variant of this rule for a different type of clauses

Nominal Generation 1

$$\frac{\alpha(x) \vee \bigvee_{i=1}^{n} f(x) \simeq t_i}{\alpha(x) \vee \bigvee_{i=1}^{k} f(x) \simeq c_i \qquad \begin{array}{r} \alpha(x) \vee \bigvee_{j=1}^{n} c_i \simeq t_j \\ 1 \leq i \leq k \end{array}}$$

where (i) c_i are fresh constants for t_i and α , (ii) k=n for the first application of rule for $\alpha(x)$ and $f(x)$, otherwise k and c_i are reused

Nominal Generation 2

$$\frac{\alpha(x) \vee \bigvee_{i=1}^{n} f(x) \simeq t_i \vee \bigvee_{i=1}^{n} x \simeq c_i}{\ldots\ldots\ldots\ldots}$$

Termination and Complexity Analysis

- Every application of the rule can increase the number of constants by at most a polynomial factor

Nominal Generation

$$\frac{\alpha(x) \vee \bigvee_{i=1}^{n} f(x) \simeq t_i}{\begin{array}{r} \alpha(x) \vee \bigvee_{i=1}^{k} f(x) \simeq c_i \\ \alpha(x) \vee \bigvee_{j=1}^{n} c_i \simeq t_j \\ 1 \leq i \leq k \end{array}}$$

where (i) c_i are fresh constants for t_i and α , (ii) k=n for the first application of rule for $\alpha(x)$ and $f(x)$, otherwise k and c_i are reused

Termination and Complexity Analysis

- Every application of the rule can increase the number of constants by at most a polynomial factor
- There are at most exponentially many applications possible (exponentially many pairs $\alpha(x)$ and $f(x)$)

Nominal Generation

$$\frac{\alpha(x) \vee \bigvee_{i=1}^{n} f(x) \simeq t_i}{\begin{array}{r}\alpha(x) \vee \bigvee_{i=1}^{k} f(x) \simeq c_i \\ \alpha(x) \vee \bigvee_{j=1}^{n} c_i \simeq t_j \\ 1 \leq i \leq k\end{array}}$$

where (i) c_i are fresh constants for t_i and α , (ii) k=n for the first application of rule for $\alpha(x)$ and $f(x)$, otherwise k and c_i are reused

Termination and Complexity Analysis

- Every application of the rule can increase the number of constants by at most a polynomial factor
- There are at most exponentially many applications possible (exponentially many pairs $\alpha(x)$ and $f(x)$)
- Hence the procedure terminates, with the upper bound: 3EXPTIME

Nominal Generation

$$\frac{\alpha(x) \vee \bigvee_{i=1}^{n} f(x) \simeq t_i}{\begin{array}{r}\alpha(x) \vee \bigvee_{i=1}^{k} f(x) \simeq c_i \\ \alpha(x) \vee \bigvee_{j=1}^{n} c_i \simeq t_j \\ 1 \leq i \leq k\end{array}}$$

where (i) c_i are fresh constants for t_i and α , (ii) k=n for the first application of rule for $\alpha(x)$ and $f(x)$, otherwise k and c_i are reused

Why Is It So Hard?

- In $\mathcal{SHOIQ}$ it is possible to express very large cardinality restrictions like $|C| \leq 2^{2^n}$, $|D| \geq 2^{2^m}$.

Why Is It So Hard?

- In $\mathcal{SHOIQ}$ it is possible to express very large cardinality restrictions like $|C| \leq 2^{2^n}$, $|D| \geq 2^{2^m}$.
- Hence, it is possible to encode combinatorial constraints involving very big numbers:

Example

$|A \sqcup B| \leq 2^{2^n}$, $|A \sqcup C| \geq 2^{2^m+k}$, $|B \sqcup C| \geq 2^{2^k}$, $|C| \leq 2^n$

Why Is It So Hard?

- In $\mathcal{SHOIQ}$ it is possible to express very large cardinality restrictions like $|C| \leq 2^{2^n}$, $|D| \geq 2^{2^m}$.
- Hence, it is possible to encode combinatorial constraints involving very big numbers
- Such problems (in particular, the pigeon hole principle) are known to be hard for resolution since it is not really capable to deal with numbers

CONCLUSIONS

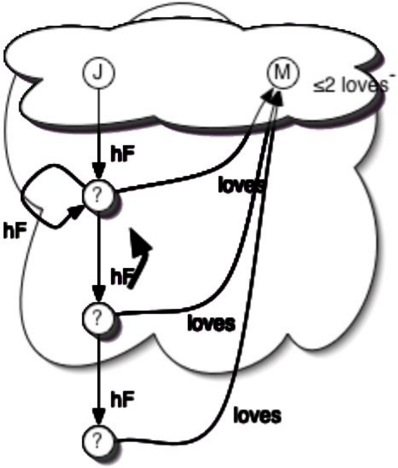
- We have found a decision procedure for $\mathcal{SHOIQ}$ based on basic superposition calculus which runs in 3ExpTime
- High complexity is due to combination of:
  nominals + number restrictions + inverse roles
- The restriction of the procedure to simpler languages ($\mathcal{SHOIQ}$, $\mathcal{ALC}$) behaves like procedures known before
- hence it exhibits "pay as you go" behaviour
- The restricted version for $\mathcal{SHIQ}$ has proved itself in practice in system KAON2 [1]
- No additional degree of non-determinism is introduced by NOMINAL GENERATION rules
- Future developments: Integration of algebraic reasoning into resolution?

[1] http://www.kaon2.semanticweb.org

Thank You!

Comparison with the Tableau Procedure

- Constants introduced by Nominal Generation correspond (in some way) to “nominal nodes”.
- The exact number of different constants is not guessed, but equality constraints are generated
- “Blocking” is native in resolution by subsumption deletion
- No “yo-yo” effect in resolution, since deletion of clauses is permanent



(A picture from the presentation by Horrocks & Sattler on “A Tableau Decision Procedure for SHOIQ” [2005])