Chain Resolution

A Variant of the Resolution Calculus tailored towards Semantic Web Applications

Uwe Keller
uwe.keller@deri.org

Digital Enterprise Research Institute (DERI)
Department of Computer Science
University of Innsbruck, Austria

DERI Research Seminar – October 20th, 2005

based on a paper by Tanel Tammet at IJCAR 2004 [Tam04]

Outline

Outline

Knowledge-based Applications

- Use explicit and machine-processable representation of knowledge about some domain
- Explicit model of a domain allows for „knowledgable" Apps & „intelligent" Agents
- Domain model can be considered as a (huge) knowledgebase that is formalized in some logic $\mathcal{L}(\Sigma)$

What are Semantic Web Apps?

- Semantic Web Apps are data-intensive knowledge-based Apps
- Knowledge Representation based on Ontologies
  - Provide at least agreed-upon terminology (within some community) (Signature Σ)
  - Connected (via annotation) with a huge set of instances
  - (Semantic) Relations between terminology: concept & role hierarchies etc. (Schema) and complex interactions (Axioms)
  - Current technologies in SW such as RDF(S) & OWL can be understood as well-behaved subsets of FOL

What are Semantic Web Apps?

- Semantic Web Apps are data-intensive knowledge-based Apps
- Knowledge Representation based on Ontologies
  - Provide at least agreed-upon terminology (within some community) (Signature Σ)
  - Connected (via annotation) with a huge set of instances
  - (Semantic) Relations between terminology: concept & role hierarchies etc. (Schema) and complex interactions (Axioms)
  - Current technologies in SW such as RDF(S) & OWL can be understood as well-behaved subsets of FOL

What are Semantic Web Apps?

- Semantic Web Apps are data-intensive knowledge-based Apps
- Knowledge Representation based on Ontologies
  - Provide at least agreed-upon terminology (within some community) (Signature Σ)
  - Connected (via annotation) with a huge set of instances
  - (Semantic) Relations between terminology: concept & role hierarchies etc. (Schema) and complex interactions (Axioms)
  - Current technologies in SW such as RDF(S) & OWL can be understood as well-behaved subsets of FOL

What are Semantic Web Apps?

- Semantic Web Apps are data-intensive knowledge-based Apps
- Knowledge Representation based on Ontologies
  - Provide at least agreed-upon terminology (within some community) (Signature Σ)
  - Connected (via annotation) with a huge set of instances
  - (Semantic) Relations between terminology: concept & role hierarchies etc. (Schema) and complex interactions (Axioms)
  - Current technologies in SW such as RDF(S) & OWL can be understood as well-behaved subsets of FOL

Outline

The Basic Resolution Calculus

- Calculus for automated deduction in various logics
  - Propositional Logic, First-order Predicate Logics, some Modal Logics ...
  - Can not be applied to arbitrary logics in general.
- Works on formulae in Clause Normal Form only
- Checks whether a given set of formulae $\Omega^\star$ is unsatisfiable
- Can be used to do reasoning:

$$\Omega \models \phi \text{ iff. } \Omega \cup \{\neg\phi\} \text{ is unsatisfiable}$$

- Essential break-through and stimulus for the ATP field [Rob65].

The Basic Resolution Calculus

- Calculus for automated deduction in various logics
  - Propositional Logic, First-order Predicate Logics, some Modal Logics ...
  - Can not be applied to arbitrary logics in general.
- Works on formulae in Clause Normal Form only
- Checks whether a given set of formulae $\Omega^\star$ is unsatisfiable
- Can be used to do reasoning:

$$\Omega \models \phi \text{ iff. } \Omega \cup \{\neg\phi\} \text{ is unsatisfiable}$$

- Essential break-through and stimulus for the ATP field [Rob65].

The Basic Resolution Calculus

- Calculus for automated deduction in various logics
  - Propositional Logic, First-order Predicate Logics, some Modal Logics ...
  - Can not be applied to arbitrary logics in general.
- Works on formulae in Clause Normal Form only
- Checks whether a given set of formulae $\Omega^\star$ is unsatisfiable
- Can be used to do reasoning:

$$\Omega \models \phi \text{ iff. } \Omega \cup \{\neg\phi\} \text{ is unsatisfiable}$$

- Essential break-through and stimulus for the ATP field [Rob65].

The Basic Resolution Calculus

- Calculus for automated deduction in various logics
  - Propositional Logic, First-order Predicate Logics, some Modal Logics ...
  - Can not be applied to arbitrary logics in general.
- Works on formulae in Clause Normal Form only
- Checks whether a given set of formulae $\Omega^\star$ is unsatisfiable
- Can be used to do reasoning:

$$\Omega \models \phi \text{ iff. } \Omega \cup \{\neg\phi\} \text{ is unsatisfiable}$$

- Essential break-through and stimulus for the ATP field [Rob65].

The Basic Resolution Calculus

- Calculus for automated deduction in various logics
  - Propositional Logic, First-order Predicate Logics, some Modal Logics ...
  - Can not be applied to arbitrary logics in general.
- Works on formulae in Clause Normal Form only
- Checks whether a given set of formulae $\Omega^\star$ is unsatisfiable
- Can be used to do reasoning:

$$\Omega \models \phi \text{ iff. } \Omega \cup \{\neg\phi\} \text{ is unsatisfiable}$$

- Essential break-through and stimulus for the ATP field [Rob65].

The Basic Resolution Calculus (2)

The simplest possible calculus for Automated Deduction

- Consists of two simple deduction rules only:
  Resolution proof principle

(Binary) Resolution – BR

$$\frac{C \vee A_1 \qquad D \vee \neg A_2}{(C \vee D)\sigma}$$

(Positive) Factoring – FACT

$$\frac{C \vee A_1 \vee A_2}{(C \vee A_1)\sigma}$$

where σ is the most general unifier of atomic formulae A_1, A_2

- Simplicity due to use of Clause Normal Form
- Binary Resolution Rule $\approx$ Cut (on atomic formulae)
- There are only finitely many resolvents for a finite set of clauses (why?)

The Basic Resolution Calculus (2)

The simplest possible calculus for Automated Deduction

- Consists of two simple deduction rules only:
  Resolution proof principle

(Binary) Resolution – BR

$$\frac{C \vee A_1 \qquad D \vee \neg A_2}{(C \vee D)\sigma}$$

(Positive) Factoring – FACT

$$\frac{C \vee A_1 \vee A_2}{(C \vee A_1)\sigma}$$

where σ is the most general unifier of atomic formulae A_1, A_2

- Simplicity due to use of Clause Normal Form
- Binary Resolution Rule $\approx$ Cut (on atomic formulae)
- There are only finitely many resolvents for a finite set of clauses (why?)

The Basic Resolution Calculus (2)

The simplest possible calculus for Automated Deduction

- Consists of two simple deduction rules only:
  Resolution proof principle

(Binary) Resolution – BR

$$\frac{C \vee A_1 \qquad D \vee \neg A_2}{(C \vee D)\sigma}$$

(Positive) Factoring – FACT

$$\frac{C \vee A_1 \vee A_2}{(C \vee A_1)\sigma}$$

where σ is the most general unifier of atomic formulae A_1, A_2

- Simplicity due to use of Clause Normal Form
- Binary Resolution Rule $\approx$ Cut (on atomic formulae)
- There are only finitely many resolvents for a finite set of clauses (why?)

The Basic Resolution Calculus (2)

The simplest possible calculus for Automated Deduction

- Consists of two simple deduction rules only:
  Resolution proof principle

(Binary) Resolution – BR

$$\frac{C \vee A_1 \qquad D \vee \neg A_2}{(C \vee D)\sigma}$$

(Positive) Factoring – FACT

$$\frac{C \vee A_1 \vee A_2}{(C \vee A_1)\sigma}$$

where σ is the most general unifier of atomic formulae A_1, A_2

- Simplicity due to use of Clause Normal Form
- Binary Resolution Rule $\approx$ Cut (on atomic formulae)
- There are only finitely many resolvents for a finite set of clauses (why?)

The Basic Resolution Calculus (2)

The simplest possible calculus for Automated Deduction

- Consists of two simple deduction rules only:
  Resolution proof principle

(Binary) Resolution – BR

$$\frac{C \vee A_1 \qquad D \vee \neg A_2}{(C \vee D)\sigma}$$

(Positive) Factoring – FACT

$$\frac{C \vee A_1 \vee A_2}{(C \vee A_1)\sigma}$$

where σ is the most general unifier of atomic formulae A_1, A_2

- Simplicity due to use of Clause Normal Form
- Binary Resolution Rule $\approx$ Cut (on atomic formulae)
- There are only finitely many resolvents for a finite set of clauses (why?)

Basic Properties of the Calculus

- Basic Resolution Calculus is sound and (refutationally) complete:

$$\Omega \text{ unsatisfiable iff. } \Omega \vdash_{BR} \square$$

- Reformulation: Ω unsatisfiable iff. every (wrt. BR) saturated set R that contains Ω also contains the empty clause $\square$
- Simple proof procedure: Construct from Ω a clause set that is saturated (wrt. the inference system BR):
  - Repeatedly apply all inferences in BR that are possible for Ω, adding to Ω the conclusions of these inferences.
  - If the empty clause $\square$ is proved, terminate with success. If no inference rule is applicable, terminate with failure.

Basic Properties of the Calculus

- Basic Resolution Calculus is sound and (refutationally) complete:

$$\Omega \text{ unsatisfiable iff. } \Omega \vdash_{BR} \square$$

- Reformulation: Ω unsatisfiable iff. every (wrt. BR) saturated set R that contains Ω also contains the empty clause $\square$
- Simple proof procedure: Construct from Ω a clause set that is saturated (wrt. the inference system BR):
  - Repeatedly apply all inferences in BR that are possible for Ω, adding to Ω the conclusions of these inferences.
  - If the empty clause $\square$ is proved, terminate with success. If no inference rule is applicable, terminate with failure.

Basic Properties of the Calculus

- Basic Resolution Calculus is sound and (refutationally) complete:

$$\Omega \text{ unsatisfiable iff. } \Omega \vdash_{BR} \square$$

- Reformulation: Ω unsatisfiable iff. every (wrt. BR) saturated set R that contains Ω also contains the empty clause $\square$
- Simple proof procedure: Construct from Ω a clause set that is saturated (wrt. the inference system BR):
  - Repeatedly apply all inferences in BR that are possible for Ω, adding to Ω the conclusions of these inferences.
  - If the empty clause $\square$ is proved, terminate with success. If no inference rule is applicable, terminate with failure.

Efficiency of Saturation Strategies

Definition (Resolution Operator)

Let Ω be a set of clauses and $Res(\Omega)$ denote the set of all resolvents derivable from Ω. Then we define the operator R of unrestricted resolution and its deductive closure $R^\star$ by

$$R(\Omega) = \Omega \cup Res(\Omega),$$
$$R^0(\Omega) = \Omega, R^{i+1}(\Omega) = R(R^i(\Omega)), R^\star(\Omega) = \bigcup_{i \geq 0} R^i(\Omega)$$

- Nondeterministic calculus BR implemented as breadth-first search.
- At time $i + 1$, the clause set $R^i(\Omega)$ represent the current search space.
- Crucial: Find „intelligent” saturation algorithms A that construct $\square$ quickly – Fairness of A needed.

Efficiency of Saturation Strategies

Definition (Resolution Operator)

Let Ω be a set of clauses and $Res(\Omega)$ denote the set of all resolvents derivable from Ω. Then we define the operator R of unrestricted resolution and its deductive closure $R^\star$ by

$$R(\Omega) = \Omega \cup Res(\Omega),$$
$$R^0(\Omega) = \Omega, R^{i+1}(\Omega) = R(R^i(\Omega)), R^\star(\Omega) = \bigcup_{i \geq 0} R^i(\Omega)$$

- Nondeterministic calculus BR implemented as breadth-first search.
- At time $i + 1$, the clause set $R^i(\Omega)$ represent the current search space.
- Crucial: Find „intelligent” saturation algorithms A that construct $\square$ quickly – Fairness of A needed.

Efficiency of Saturation Strategies

Definition (Resolution Operator)

Let Ω be a set of clauses and $Res(\Omega)$ denote the set of all resolvents derivable from Ω. Then we define the operator R of unrestricted resolution and its deductive closure $R^\star$ by

$$R(\Omega) = \Omega \cup Res(\Omega),$$
$$R^0(\Omega) = \Omega, R^{i+1}(\Omega) = R(R^i(\Omega)), R^\star(\Omega) = \bigcup_{i \geq 0} R^i(\Omega)$$

- Nondeterministic calculus BR implemented as breadth-first search.
- At time $i + 1$, the clause set $R^i(\Omega)$ represent the current search space.
- Crucial: Find „intelligent” saturation algorithms A that construct $\square$ quickly – Fairness of A needed.

Refinements of the Calculus

Restricting the Resolution Principle

- Observation & idea:
  - At time $i+1$, $R^i(\Omega)$ represents the search space $\approx$ i-th application of the unrestricted Resolution operator R
  - $|Res(\Omega)| \approx O(|\Omega|^2)$ is too big, thus $R^i(\Omega)$ grows rapidly
  - Not all resolvents in $Res(\Omega)$ are needed to construct $\square$

  $\Rightarrow$ Get rid of (obviously) unnecessary resolvents in $Res(\Omega)$

Definition (Resolution Refinement Operator)

A resolution refinement opertor R_x is a mapping on sets of clauses defined by an (one-step) operator ρ_x s.t.
$R_x(\Omega) = \Omega \cup \rho_x(\Omega)$, ρ_x is recursive,
and $\rho_x(\Omega)$ is a finite subset of $R^*(\Omega)$

Refinements of the Calculus

Restricting the Resolution Principle

- Observation & idea:
  - At time $i+1$, $R^i(\Omega)$ represents the search space $\approx$ i-th application of the unrestricted Resolution operator R
  - $|Res(\Omega)| \approx O(|\Omega|^2)$ is too big, thus $R^i(\Omega)$ grows rapidly
  - Not all resolvents in $Res(\Omega)$ are needed to construct $\square$

  $\Rightarrow$ Get rid of (obviously) unnecessary resolvents in $Res(\Omega)$

Definition (Resolution Refinement Operator)

A resolution refinement opertor R_x is a mapping on sets of clauses defined by an (one-step) operator ρ_x s.t.
$R_x(\Omega) = \Omega \cup \rho_x(\Omega)$, ρ_x is recursive,
and $\rho_x(\Omega)$ is a finite subset of $R^*(\Omega)$

Refinements of the Calculus

Restricting the Resolution Principle

- Observation & idea:
  - At time $i+1$, $R^i(\Omega)$ represents the search space $\approx$ i-th application of the unrestricted Resolution operator R
  - $|Res(\Omega)| \approx O(|\Omega|^2)$ is too big, thus $R^i(\Omega)$ grows rapidly
  - Not all resolvents in $Res(\Omega)$ are needed to construct $\square$

  $\Rightarrow$ Get rid of (obviously) unnecessary resolvents in $Res(\Omega)$

Definition (Resolution Refinement Operator)

A resolution refinement opertor R_x is a mapping on sets of clauses defined by an (one-step) operator ρ_x s.t.
$R_x(\Omega) = \Omega \cup \rho_x(\Omega)$, ρ_x is recursive,
and $\rho_x(\Omega)$ is a finite subset of $R^\star(\Omega)$

Outline

ATP Application Scenarios:
Mathematics vs. Semantic Web Applications

- Class 1: ATP on Mathematical Problems
  - Small, compact axiom sets, difficult problems **(traditional)**
- Class 2: ATP on Semantic Web Applications
  - Data-centric, Ontology-based, large data-sets, simple formulae **(only recently)**
- Current ATP systems are tailored towards & measured against their performance on problem in **Class 1**. Not surprisingly, there performance on problems of **Class 2** is not as good as it could be / needs to be.
- **Needed**: specific techniques for Class 2 problems.
- **Chain Resolution** is one step in this direction for resolution techniques!

ATP Application Scenarios:
Mathematics vs. Semantic Web Applications

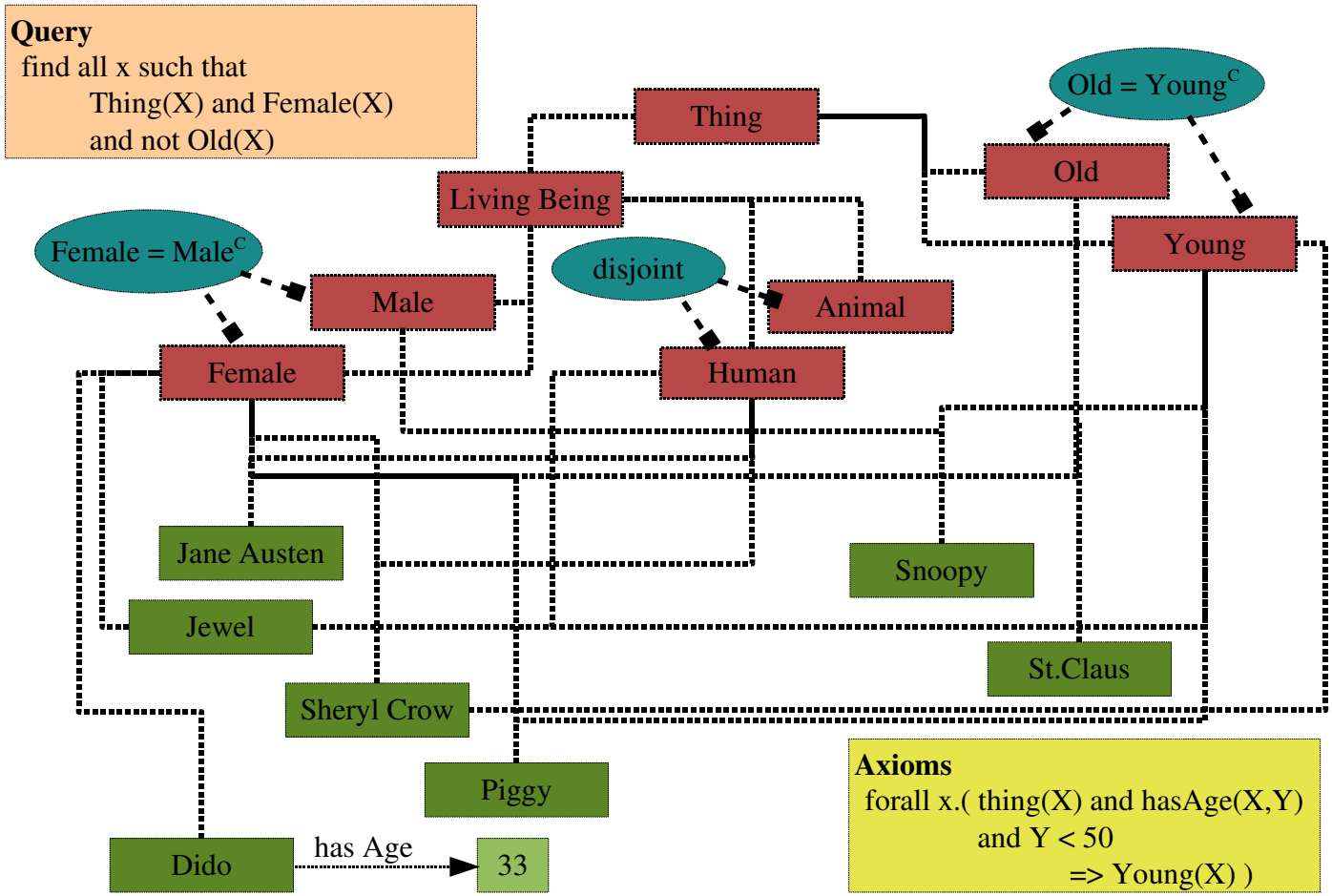
- Class 1: ATP on Mathematical Problems
  - Small, compact axiom sets, difficult problems **(traditional)**
- Class 2: ATP on Semantic Web Applications
  - Data-centric, Ontology-based, large data-sets, simple formulae **(only recently)**
- Current ATP systems are tailored towards & measured against their performance on problem in **Class 1**. Not surprisingly, there performance on problems of **Class 2** is not as good as it could be / needs to be.
- **Needed**: specific techniques for Class 2 problems.
- **Chain Resolution** is one step in this direction for resolution techniques!

ATP Application Scenarios:
Mathematics vs. Semantic Web Applications

- Class 1: ATP on Mathematical Problems
  - Small, compact axiom sets, difficult problems **(traditional)**
- Class 2: ATP on Semantic Web Applications
  - Data-centric, Ontology-based, large data-sets, simple formulae **(only recently)**
- Current ATP systems are tailored towards & measured against their performance on problem in **Class 1**. Not surprisingly, there performance on problems of **Class 2** is not as good as it could be / needs to be.
- **Needed**: specific techniques for Class 2 problems.
- **Chain Resolution** is one step in this direction for resolution techniques!

Example

Applying Resolution on Ontologies



Chain Resolution: The Problem

- Fairly common situation in practical Apps of Resolution: Proof Tasks consists of
  - an „ordinary” FOL part well-suited for Resolution
  - a large special theory not well-suited for Resolution
- Class and role hierarchies are such a „theory”
  - Make up significant parts of ontologies [TRBH04]
  - Are often & heavily used in SW Apps (using Ontologies)
  - Lead to unnecessary blow up of the search space
  - Especially problematic for instance retrieval tasks
- Solution Approach: Extend Resolution by a specific Black Box system, that can deal efficiently with the special theory

Chain Resolution: The Problem

- Fairly common situation in practical Apps of Resolution: Proof Tasks consists of
  - an „ordinary” FOL part well-suited for Resolution
  - a large special theory not well-suited for Resolution
- Class and role hierarchies are such a „theory”
  - Make up significant parts of ontologies [TRBH04]
  - Are often & heavily used in SW Apps (using Ontologies)
  - Lead to unnecessary blow up of the search space
  - Especially problematic for instance retrieval tasks
- Solution Approach: Extend Resolution by a specific Black Box system, that can deal efficiently with the special theory

Chain Resolution: The Problem

- Fairly common situation in practical Apps of Resolution: Proof Tasks consists of
  - an „ordinary” FOL part well-suited for Resolution
  - a large special theory not well-suited for Resolution
- Class and role hierarchies are such a „theory”
  - Make up significant parts of ontologies [TRBH04]
  - Are often & heavily used in SW Apps (using Ontologies)
  - Lead to unnecessary blow up of the search space
  - Especially problematic for instance retrieval tasks
- Solution Approach: Extend Resolution by a specific Black Box system, that can deal efficiently with the special theory

Outline

Chain Resolution: Idea

- Chain Resolution is
  - Resolution search strategy for large ontologies
  - Such a simple extension by a black box, called Chain Box
- Targets only at specific clauses in Ω: (Chain Clauses)

$$A(x_1, \ldots, x_n) \vee B(x_1, \ldots, x_n)$$

where A and B are *signed* predicates, all x_j are different

  - Consider chain clauses as simple implications.
  - At each stage during proof search, move chain clauses from Ω in Chain Box
  - Use Chain Box as an efficient (deductive) oracle in essential steps during proof search: BR, FAC, SUBSUMPT rules
  - Everything else in Ω is treated as before.

Chain Resolution: Idea

- Chain Resolution is
  - Resolution search strategy for large ontologies
  - Such a simple extension by a black box, called Chain Box
- Targets only at specific clauses in Ω: (Chain Clauses)

$$A(x_1, \ldots, x_n) \vee B(x_1, \ldots, x_n)$$

where A and B are *signed* predicates, all x_j are different

  - Consider chain clauses as simple implications.
  - At each stage during proof search, move chain clauses from Ω in Chain Box
  - Use Chain Box as an efficient (deductive) oracle in essential steps during proof search: BR, FAC, SUBSUMPT rules
  - Everything else in Ω is treated as before.

Chain Resolution: Idea

- Chain Resolution is
  - Resolution search strategy for large ontologies
  - Such a simple extension by a black box, called Chain Box
- Targets only at specific clauses in Ω: (Chain Clauses)

$$A(x_1, \ldots, x_n) \vee B(x_1, \ldots, x_n)$$

where A and B are *signed* predicates, all x_j are different

  - Consider chain clauses as simple implications.
  - At each stage during proof search, move chain clauses from Ω in Chain Box
  - Use Chain Box as an efficient (deductive) oracle in essential steps during proof search: BR, FAC, SUBSUMPT rules
  - Everything else in Ω is treated as before.

Chain Resolution: Idea

- Chain Resolution is
  - Resolution search strategy for large ontologies
  - Such a simple extension by a black box, called Chain Box
- Targets only at specific clauses in Ω: (Chain Clauses)

$$A(x_1, \ldots, x_n) \vee B(x_1, \ldots, x_n)$$

where A and B are *signed* predicates, all x_j are different

  - Consider chain clauses as simple implications.
  - At each stage during proof search, move chain clauses from Ω in Chain Box
  - Use Chain Box as an efficient (deductive) oracle in essential steps during proof search: BR, FAC, SUBSUMPT rules
  - Everything else in Ω is treated as before.

Chain Resolution: Idea

- Chain Resolution is
  - Resolution search strategy for large ontologies
  - Such a simple extension by a black box, called Chain Box
- Targets only at specific clauses in Ω: (Chain Clauses)

$$A(x_1, \ldots, x_n) \vee B(x_1, \ldots, x_n)$$

where A and B are *signed* predicates, all x_j are different

  - Consider chain clauses as simple implications.
  - At each stage during proof search, move chain clauses from Ω in Chain Box
  - Use Chain Box as an efficient (deductive) oracle in essential steps during proof search: BR, FAC, SUBSUMPT rules
  - Everything else in Ω is treated as before.

Some definitions ...

Definition (Propositional Variant)

A clause C' is a propositional variant of clause C iff C' is derivable by a sequence of binary resolution steps from C and a set of chain clauses.

Definition (Search State)

A search state of a resolution prover is a set of clauses kept by the prover at some point in time during the proof search

Outline

How does it work in detail?

- Chain Box essentially represents a mapping of signed predicate symbols to sets of signed predicate symbols that captures the deductive closure of the set of chain clauses

Definition (Chain Box Key, Chain, Row)

A signed predicate P is called a key of a chain box in case P has a set of signed predicate symbols associated with it in the chain box. The associated set is called chain, also denoted as $chain(P)$. The pair of a key P and the chain set of P is called a row of the chain box.

- A chain box does not contain literals, but predicates
- A row $(P, \{\ldots P' \ldots\}) \approx P(\overline{x}) \rightarrow P'(\overline{x})$

How does it work in detail?

- Chain Box essentially represents a mapping of signed predicate symbols to sets of signed predicate symbols that captures the deductive closure of the set of chain clauses

Definition (Chain Box Key, Chain, Row)

A signed predicate P is called a key of a chain box in case P has a set of signed predicate symbols associated with it in the chain box. The associated set is called chain, also denoted as $chain(P)$. The pair of a key P and the chain set of P is called a row of the chain box.

- A chain box does not contain literals, but predicates
- A row $(P, \{\ldots P' \ldots\}) \approx P(\overline{x}) \rightarrow P'(\overline{x})$

How does it work in detail?

- Chain Box essentially represents a mapping of signed predicate symbols to sets of signed predicate symbols that captures the deductive closure of the set of chain clauses

Definition (Chain Box Key, Chain, Row)

A signed predicate P is called a key of a chain box in case P has a set of signed predicate symbols associated with it in the chain box. The associated set is called chain, also denoted as $chain(P)$. The pair of a key P and the chain set of P is called a row of the chain box.

- A chain box does not contain literals, but predicates
- A row $(P, \{\ldots P' \ldots\}) \approx P(\overline{x}) \rightarrow P'(\overline{x})$

Moving Chain Clauses to the Chain Box

- Chain Box Initialization
  - For each predicate P in Ω create two rows $(P, \{P\})$ and $(\neg P, \{\neg P\})$
  - Before the proof search starts, move all chain clauses in Ω to the chain box and remove them from Ω. In case that such a move produces a new unit clause, add this clause to Ω.
- During Proof Search
  - Move all chain clauses produced during proof search in Ω to the chain box and remove them from the search state. All unit clauses that result from the movements, are addeded to the passive list part of the search state.

Moving Chain Clauses to the Chain Box (2)

- How to represent the full semantics of a chain clause?
  - $P(\overline{x}) \vee P'(\overline{x}) \equiv \neg P(\overline{x}) \rightarrow P'(\overline{x}) \wedge \neg P'(\overline{x}) \rightarrow P(\overline{x})$
  - Hence, given $P(\overline{x}) \vee P'(\overline{x})$ insert entries in two rows.
  - Additionally, we need to capture the transitivity of $\rightarrow$ wrt. the entries already present in the chain box
  - Finally, care about potentially derivable unit clause: Rows with key p and $chain(p) = \{\ldots, r, \ldots \neg r, \ldots\}$ by inserting them to the passive list of the search state

Moving Chain Clauses to the Chain Box (3)

- Algorithm

addchain(p,p') := addchain-trans($\neg p$, p'); addchain-trans(p,$\neg p'$)
addchain-trans(k, p) :=
    if $p \notin chain(k)$ then do
        chain(k) := chain(k) $\cup \{p\}$
        forall $l \in chain(p)$ do addchain(k,l)
        forall keys r where $k \in chain(r)$ do
            forall $m \in chain(k)$ do addchain(r,m)

Moving Chain Clauses to the Chain Box (4)

- Chain Box represents all the derivations that can be made among chain clauses using the BR rule and (because of the initializations of rows with $(p, \{p\})$) and the FACT rule.
- Every derivable conclusions in BR among chain clauses not representable in the chain box (unit clauses) are put back in the search state (to not lose possible derivations)
- This ensures completeness of Chain Resolution later on, since every proof in BR can be simulated in the Chain Resolution Calculus

Adapting the Rules of the Calculus
Binary Resolution

- Modify existing rules s.t. they work modulo the chain box
- Easiest solution: Modify Literal Unification & Literal Subsumption

- Chain Resolution Algorithm
  - Two literals $A(t_1, \ldots, t_n)$ and $B(t'_1, \ldots, t'_n)$ can be resolved upon iff $A(t_1, \ldots, t_n)$ and $A(t'_1, \ldots, t'_n)$ are unifiable (in the standard sense) and $B = \neg A$ or $B \in \mathit{chain}(\neg A)$
  - Substitution to be applied does not change.
- Chain Resolution Rule now becomes a macro inference rule wrt. the BR rule.

Adapting the Rules of the Calculus (2)
Factorisation

- Chain Factorization Algorithm
  - A bit more complicated since factorized literal depends on implication encoded in the chain box
  - $A(t_1, \ldots, t_n)$ and $B(t'_1, \ldots, t'_n)$ can be factorized iff $A(t_1, \ldots, t_n)$ and $A(t'_1, \ldots, t'_n)$ are unifiable (in the standard sense) by mgu σ and $B = A$ or $A \in \mathit{chain}(B)$ or $B \in \mathit{chain}(A)$
  - Use the less specific formlua as the factorization:
    - if $B = A$ then resulting lit. is $A(t_1, \ldots, t_n)\sigma$
    - if $A \in \mathit{chain}(B)$ then resulting lit. is $A(t_1, \ldots, t_n)\sigma$
    - if $B \in \mathit{chain}(A)$ then resulting lit. is $B(t_1, \ldots, t_n)\sigma$
- Chain Factorization Rule now becomes a macro inference rule wrt. the FACT rule.

Adapting the Rules of the Calculus (3)
Subsumption

- Chain Subsumption Algorithm of Literals
  - Literal $A(t_1, \ldots, t_n)$ subsumes $B(t'_1, \ldots, t'_n)$ iff. $A(t_1, \ldots, t_n)$ subsumes $B(t'_1, \ldots, t'_n)$ (in the standard sense) and either $B = A$ or $B \in \mathit{chain}(A)$
  - Important: In comparison to standard literal subsumption, there are now potentially a lot more subsumed clauses possible.
  - This is a key factor for efficiency when using resolution techniques!
  - Standard clause subsumption algorithm stays the same!

Why is it an efficient Strategy?

- Context:
  Experiments in the context of resolution-based proof search for query answering in large ontologies [Tam04]

- Tammet realized in these experiments that
  - New chain clauses are produced during proof search
  - Some chain clauses are typically present in the proofs, making a proof larger and hence harder to find
  - Chain clauses produce a large number of propositional variations of non-chain clauses, making the search space larger

Why is it an efficient Strategy?

- Context:
  Experiments in the context of resolution-based proof search for query answering in large ontologies [Tam04]

- Tammet realized in these experiments that
  - New chain clauses are produced during proof search
  - Some chain clauses are typically present in the proofs, making a proof larger and hence harder to find
  - Chain clauses produce a large number of propositional variations of non-chain clauses, making the search space larger

Why is it an efficient Strategy? (2)

- The effect of chain clauses in the search space
  - Assume that there is a clause C in the search space s.t.

$$C = A_1(t_1) \vee \ldots \vee A_n(t_n) \vee \Gamma$$

  - Assume the search space contains m_i clauses of the form

$$\neg A_1(x) \vee A'_1 \ldots, \ldots, \neg A_i(x) \vee A'_{m_i} ldots$$

    for each $1 \leq i \leq n$.
- Then the resolution method will derive $m_1 \times m_2 \times \ldots \times m_n$ new clauses from C using chain clauses alone
- All these clauses are propositional variations of C

Why is it an efficient Strategy? (2)

- The effect of chain clauses in the search space
  - Assume that there is a clause C in the search space s.t.

$$C = A_1(t_1) \vee \ldots \vee A_n(t_n) \vee \Gamma$$

  - Assume the search space contains m_i clauses of the form

$$\neg A_1(x) \vee A'_1 \ldots, \ldots, \neg A_i(x) \vee A'_{m_i} ldots$$

    for each $1 \leq i \leq n$.
- Then the resolution method will derive $m_1 \times m_2 \times \ldots \times m_n$ new clauses from C using chain clauses alone
- All these clauses are propositional variations of C

Why is it an efficient Strategy? (3)

- Chain Resolution now achieves . . .
  - No chain clauses are kept in the search space (smaller search space)
  - Chain clauses are „compiled” into an efficient data structure: Chain Box reasoning is far more efficient than the usual proof rules.
  - Never use chain clauses in ordinary resolution steps
  - Macro inferences along class and role hierarchies is achieved (shorter proofs)
  - Strong redundancy criterion can be used: Propositional variations of clauses don’t need to be kept

Outline

Chain Resolution is an elegant solution

- Chain Resolution is sound and (refutationally) complete
- Chain Resolution can be implemented and integrated in existing systems very easily
- Expected to fulfill principle of graceful degradation
  - Non-chain clauses: Performance degradation should be minimal
  - Chain clauses: Performance enhancement should be significant
- Chain Resolution can be combined with other important strategies without loosing completeness
  - A-ordered Resolution
  - Set-of-Support Resolution

Chain Resolution is an elegant solution

- Chain Resolution is sound and (refutationally) complete
- Chain Resolution can be implemented and integrated in existing systems very easily
- Expected to fulfill principle of graceful degradation
  - Non-chain clauses: Performance degradation should be minimal
  - Chain clauses: Performance enhancement should be significant
- Chain Resolution can be combined with other important strategies without loosing completeness
  - A-ordered Resolution
  - Set-of-Support Resolution

Chain Resolution is an elegant solution

- Chain Resolution is sound and (refutationally) complete
- Chain Resolution can be implemented and integrated in existing systems very easily
- Expected to fulfill principle of graceful degradation
  - Non-chain clauses: Performance degradation should be minimal
  - Chain clauses: Performance enhancement should be significant
- Chain Resolution can be combined with other important strategies without loosing completeness
  - A-ordered Resolution
  - Set-of-Support Resolution

Chain Resolution is an elegant solution

- Chain Resolution is sound and (refutationally) complete
- Chain Resolution can be implemented and integrated in existing systems very easily
- Expected to fulfill principle of graceful degradation
  - Non-chain clauses: Performance degradation should be minimal
  - Chain clauses: Performance enhancement should be significant
- Chain Resolution can be combined with other important strategies without loosing completeness
  - A-ordered Resolution
  - Set-of-Support Resolution

Summary

- Chain Resolution is a optimization technique for Resolution frameworks that addresses chain clauses
- Chain Clauses are expected to be of significant importance for Semantic Web Apps
- Chain Resolution is a very elegant solution
- Should be implemented on top of the GANDALF system. No evaluation results published so far.
- Chain Resolution is a nice example for an engineering approach to problem solving
- Open questions . . .
  - Are there further techniques that can be applied to typical SW Applications?
  - Can Chain Boxes be beneficially be applied in the context of other calculi as well, e.g. Tableaux calculi? And if so how?

Summary

- Chain Resolution is a optimization technique for Resolution frameworks that addresses chain clauses
- Chain Clauses are expected to be of significant importance for Semantic Web Apps
- Chain Resolution is a very elegant solution
- Should be implemented on top of the GANDALF system. No evaluation results published so far.
- Chain Resolution is a nice example for an engineering approach to problem solving
- Open questions . . .
  - Are there further techniques that can be applied to typical SW Applications?
  - Can Chain Boxes be beneficially be applied in the context of other calculi as well, e.g. Tableaux calculi? And if so how?

Summary

- Chain Resolution is a optimization technique for Resolution frameworks that addresses chain clauses
- Chain Clauses are expected to be of significant importance for Semantic Web Apps
- Chain Resolution is a very elegant solution
- Should be implemented on top of the GANDALF system. No evaluation results published so far.
- Chain Resolution is a nice example for an engineering approach to problem solving
- Open questions . . .
  - Are there further techniques that can be applied to typical SW Applications?
  - Can Chain Boxes be beneficially be applied in the context of other calculi as well, e.g. Tableaux calculi? And if so how?

Summary

- Chain Resolution is a optimization technique for Resolution frameworks that addresses chain clauses
- Chain Clauses are expected to be of significant importance for Semantic Web Apps
- Chain Resolution is a very elegant solution
- Should be implemented on top of the GANDALF system. No evaluation results published so far.
- Chain Resolution is a nice example for an engineering approach to problem solving
- Open questions . . .
  - Are there further techniques that can be applied to typical SW Applications?
  - Can Chain Boxes be beneficially be applied in the context of other calculi as well, e.g. Tableaux calculi? And if so how?

Summary

- Chain Resolution is a optimization technique for Resolution frameworks that addresses chain clauses
- Chain Clauses are expected to be of significant importance for Semantic Web Apps
- Chain Resolution is a very elegant solution
- Should be implemented on top of the GANDALF system. No evaluation results published so far.
- Chain Resolution is a nice example for an engineering approach to problem solving
- Open questions . . .
  - Are there further techniques that can be applied to typical SW Applications?
  - Can Chain Boxes be beneficially be applied in the context of other calculi as well, e.g. Tableaux calculi? And if so how?

Summary

- Chain Resolution is a optimization technique for Resolution frameworks that addresses chain clauses
- Chain Clauses are expected to be of significant importance for Semantic Web Apps
- Chain Resolution is a very elegant solution
- Should be implemented on top of the GANDALF system. No evaluation results published so far.
- Chain Resolution is a nice example for an engineering approach to problem solving
- Open questions . . .
  - Are there further techniques that can be applied to typical SW Applications?
  - Can Chain Boxes be beneficially be applied in the context of other calculi as well, e.g. Tableaux calculi? And if so how?

Summary

- Chain Resolution is a optimization technique for Resolution frameworks that addresses chain clauses
- Chain Clauses are expected to be of significant importance for Semantic Web Apps
- Chain Resolution is a very elegant solution
- Should be implemented on top of the GANDALF system. No evaluation results published so far.
- Chain Resolution is a nice example for an engineering approach to problem solving
- Open questions . . .
  - Are there further techniques that can be applied to typical SW Applications?
  - Can Chain Boxes be beneficially be applied in the context of other calculi as well, e.g. Tableaux calculi? And if so how?

References

J. A. Robinson.

A Machine-Oriented Logic Based on the Resolution Principle.

Journal of the ACM, 12(1):23 – 41, January 1965.

Tanel Tammet.

Chain Resolution for the Semantic Web.

In *Automated Reasoning – 2nd International Joint Conference on Automated Reasoning (IJCAR), Cork, Ireland*, volume 3097 of *LNAI*, pages 307 – 320, 2004.

Dmitry Tsarkov, Alexandre Riazanov, Sean Bechhofer, and Ian Horrocks.

Using Vampire to reason with OWL.

In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *Proc. of the 2004 International Semantic Web Conference (ISWC 2004)*, number 3298 in Lecture Notes in Computer Science, pages 471–485. Springer, 2004.