The Foundations of Inductive Logic Programming

By Chongbing Liu

Outlines

☐ Resolution Based Proof Procedures
☐ ILP Problem Specification
☐ Generality Orders on Clauses
☐ Refinement Operators
☐ Conclusions
Resolution Based Proof Procedures

Proof Procedures

- Very often we need to prove that $\Sigma \vdash E$
- But this is in general undecidable
- When $\Sigma \vdash E$ is true, we could have some procedures to generate proofs
- Ideal properties: complete, sound, work mechanically, efficient and applicable to all $\Sigma$ and $E$
Resolution Based Proof Procedures

\[ \Sigma: \quad C, D, E, F, G, H \]

\[ R, U, V, C, D, E, F, G, H \]

\[ W, R, U, V, C, D, E, F, G, H \]

\[ \ldots \]

Unconstrained

C, D from \( \Sigma \) or all the intermediate resolvents

Incomplete

Inefficient
Resolution Based Proof Procedures

Resolution

C = f, x  D = ¬f, y

R = x ∨ y
Resolvent

Σ:
- h(x)←f(x), g(x).
- f(x).
- g(x).

resolution

Σ ⊢ h(x).

Σ ⊢ h(a).

C, D from Σ or all the intermediate resolvents

Unconstrained

Incomplete
Inefficient

10/19/2004
The Foundations of Inductive Logic Programming
Resolution Based Proof Procedures

**Deduction**

resolution rule

C = f, x  D = ¬f, y  

R = x ∨ y  
Resolvent

Σ:

h(x) ← f(x), g(x).

f(x).

g(x).

Σ ⊨ h(x).

C subsume D if exists

θ s.t. C θ ⊆ D

Σ ⊨ h(a).

Unconstrained

Deduction

C, D from Σ or all the
intermediate resolvents

Complete Inefficient

10/19/2004

The Foundations of Inductive Logic Programming
Resolution Based Proof Procedures

**Linear Resolution**

C, D, E, F, G, H

R

U

V

W

C is the resolvent of the last step, D from all

C, D from Σ or all the intermediate resolvents

Linear

Unconstrained

Deduction

Deduction

Complete Inefficient

Complete Inefficient

10/19/2004

The Foundations of Inductive Logic Programming
Resolution Based Proof Procedures

Resolution rule

\[ C = f, x \quad D = \neg f, y \]

\[ R = x \lor y \]

Resolvent

Input Resolution

\[ C, \quad D, \quad E, \quad F, \quad G, \quad H \]

\[ R \]

\[ U \]

\[ V \]

C is the resolvent of the last step, D from \( \Sigma \)

Input

Deduction

Subsumption

Incompleteness Efficient

C is the resolvent of the last step, D from all

Linear

Deduction

Subsumption

Complete Incompleteness

C, D from \( \Sigma \) or all the intermediate resolvents

Unconstrained

Deduction

Subsumption

Complete Incompleteness
Resolution Based Proof Procedures

Resolution rule

\[ C = f, x \quad D = \neg f, y \]

\[ R = x \lor y \]

Resolvent

\[ \Sigma \text{ consists of Horn clauses} \]

\[ C \text{ is the resolvent of the last step, } D \text{ from } \Sigma \]

\[ C, D \text{ from } \Sigma \text{ or all the intermediate resolvers} \]

SLD Resolution

Input

SLD

Deduction

Complete Efficient

Linear

Complete Inefficient

Unconstrained

Complete Inefficient

Deduction

Deduction

Deduction

Deduction

10/19/2004

The Foundations of Inductive Logic Programming
### Resolution Based Proof Procedures

Resolvent Rule:

\[ C = f, x \quad D = \neg f, y \]

\[ R = x \lor y \]

**SLDNF Resolution**

- **SLDNF**
  - Allow negative literals in the clause body;
  - Use Negation as Failure

- **SLD**
  - \( \Sigma \) consists of Horn clauses

- **Input**
  - \( C \) is the resolvent of the last step, \( D \) from \( \Sigma \)

- **Linear**
  - \( C \) is the resolvent of the last step, \( D \) from all

- **Unconstrained**
  - \( C, D \) from \( \Sigma \) or all the intermediate resolvents

**Deduction**

- Incomplete
- Efficient

- Complete
- Inefficient

**Subsumption**

10/19/2004

The Foundations of Inductive Logic Programming
ILP Problem Specification

Given:

A finite set of clauses \( B \) (background knowledge), and sets of clauses \( E^+ \) and \( E^- \)

Find:

A theory \( \Sigma \), such that \( \Sigma \cup B \) is correct with respect to \( E^+ \) and \( E^- \)
ILP Problem Specification

Correct theory

\[ \Sigma \cup B \text{ is correct with respect to } E^+ \text{ and } E^- \text{ if} \]

1. \[ \Sigma \cup B \models E^+ \] (completeness)

and

2. \[ \Sigma \cup B \cup \neg E^- \text{ is satisfiable} \] (consistency).

ILP Search all the clauses for correct \( \Sigma \)
ILP Problem Specification

Correct theory

\[ \Sigma \cup B \text{ is correct with respect to } E^+ \text{ and } E^- \text{ if} \]

1. \[ \Sigma \cup B \models E^+ \] (completeness)

and

2. \[ \Sigma \cup B \cup \neg E^- \text{ is satisfiable} \] (consistency).

\[ (\Sigma \cup B) \text{ implies no } e \in E^- \] (easier, proof procedures)
ILP Problem Specification

Consistency Condition

$$\Sigma \cup \mathcal{B}$$ is correct with respect to $$E^+$$ and $$E^-$$ if

1. $$\Sigma \cup \mathcal{B} \models E^+$$ \hspace{1cm} (completeness)

and

2. $$\Sigma \cup \mathcal{B} \cup \neg E^-$$ is satisfiable \hspace{1cm} (consistency).

Example: (let $$\mathcal{B} = \emptyset$$)

$$\Sigma = \{ P(a) \lor P(b) \}$$

$$E^- = \{ P(a), P(b) \}$$

($$\Sigma \cup \mathcal{B}$$) implies no $$e \in E^-$$

(easier, proof procedures)

10/19/2004 The Foundations of Inductive Logic Programming
ILP Problem Specification

Admissibility

\( \Sigma \cup B \) is correct with respect to \( E^+ \) and \( E^- \) if

1. \( \Sigma \cup B \models E^+ \) (completeness)

and

2. \( \Sigma \cup B \cup \neg E^- \) is satisfiable (consistency).

\( (\Sigma \cup B) \) implies no \( e \in E^- \) (easier, proof procedures)

If \( \langle E, \Sigma \rangle \) are admissible:

- \( \langle \text{ground atoms, Horn clauses} \rangle \)
- \( \langle \text{ground literals, clauses} \rangle \)
ILP Problem Specification

Correct theory

Σ ∪ B is correct with respect to E⁺ and E⁻ if

1. Σ ∪ B |= E⁺ (completeness)

and

2. (Σ ∪ B) implies no e ∈ E⁻ (consistency).

(easier, proof procedures)

If ⟨E, Σ⟩ are admissible:
⟨ground atoms, Horn clauses⟩
⟨ground literals, clauses⟩
ILP Problem Specification

**Correct theory**

\[ \sum \cup B \text{ is correct with respect to } E^+ \text{ and } E^- \text{ if} \]

1. \[ \sum \cup B \models E^+ \] (completeness)

and

\[ \text{Reduced Search Space! (bias)} \]

2. \[ (\sum \cup B) \text{ implies no } e \in E^- \] (consistency).
   (easier, proof procedures)

If \( \langle E, \sum \rangle \) are admissible:

\( \langle \text{ground atoms, Horn clauses} \rangle \)

\( \langle \text{ground literals, clauses} \rangle \)
ILP Problem Specification

ILP as a search problem (search space)

Theory Space

Clause space (Language bias)
ILP Problem Specification

ILP as a search problem (generality orders)

Ordered Clause space
ILP Problem Specification

ILP as a search problem (generality orders)

Ordered Theory Space

Ordered Clause space
ILP Problem Specification

ILP as a search problem (A General Scheme)

Start with some initial theory
Repeat

If $\Sigma$ is too strong, specialize it

If $\Sigma$ is too weak, generalize it

until $\Sigma \cup B$ is correct with respect to $E^+$ and $E^-$
ILP Problem Specification

Operations

Start with some initial theory
Repeat
If \( \sum \) is too strong, specialize it
If \( \sum \) is too weak, generalize it
} \[ \text{Refinement operators} \]
until \( \sum \cup B \) is correct with respect to \( E^+ \) and \( E^- \)
Generality Orders on Clauses

Basic Concepts

- Quasi-order $\succeq$ on set S: Reflexive and transitive
- Least generalization(S): Least Upper Bound (lub)
- Greatest specialization(S): Greatest Lower Bound (glb)
- Lattice: Exist lub and glb for any S
- Downward Cover(C): $\{ D \mid C \succeq D, \text{ and no } E \text{ s.t. } C > E > D \}$
- Upward Cover(C): $\{ D \mid D \succeq C, \text{ and no } E \text{ s.t. } D > E > C \}$
Generality Orders on Clauses

(no background knowledge)

• Subsumption order on atoms
• Subsumption order on clauses
• Implication order on clauses
Generality Orders on Clauses

(no background knowledge)

Subsumption order \((\preceq)\) on the set of atoms

- **Definition**: \(A \preceq B\) if \(A \theta \subseteq B\) for some \(\theta\)
- **Existence Of Least Generalization**: Yes
- **Existence Of Greatest Specialization**: Yes
- **Upward covers**: finite
- **Downward cover**: finite
Generality Orders on Clauses

(no background knowledge)

Subsumption order \( \preceq \) on the set of \textbf{clauses}

- Definition: \( A \preceq B \) if \( A \theta \subseteq B \) for some \( \theta \)
- Existence Of Least Generalization : Yes
- Existence Of Greatest Specialization: Yes
- On Horn clauses : Lattice
- Upward covers : not always exist or finite
- Downward cover : not always exist or finite
Generality Orders on Clauses

(no background knowledge)

Implication order ( |- ) on the set of clauses

☐ Definition: logical consequence
☐ Existence Of Least Generalization: Yes
☐ Existence Of Greatest Specialization: No
☐ On Horn clauses: No
☐ Upward covers: not always exist or finite
☐ Downward cover: not always exist or finite

Only when S contains at least One function-free clause
Generality Orders on Clauses

(with background knowledge)

- Relative Subsumption order
- Relative Implication order
- Generalized Subsumption order
Generality Orders on Clauses

(with background knowledge)

**Relative Subsumption order \( (\preceq_B) \)**

- **Definition:** \( C \preceq_B D \) if \( B \vdash \forall (C \theta \subseteq D) \) for some \( \theta \)
- **Existence Of Least Generalization:** Yes, when \( B \) is a set of ground literals
- **On Horn clauses:** Yes, when \( B \) is ground atoms
- **Deduction:** Exist a deduction of \( D \) from \( \{C\} \cup B \) where \( C \) occurs at most once
Generality Orders on Clauses

(with background knowledge)

Relative Implication order (\(\vdash_B\))

- Definition: \(C \vdash_B D\) if \((B \cup \{C\}) \vdash D\)
- Existence Of Least Generalization: Yes, when \(B\) is a set of function-free ground literals and \(S\) contains at least one on function-free clause
- On Horn clauses: NO
- Deduction: Exist a deduction of \(D\) from \(\{C\} \cup B\)
Generality Orders on Clauses

(with background knowledge)

Generalized Subsumption order ($\geq_B$)

- Definition: $C \geq_B D$ if with $B$, $C$ can be used to prove at least as many results as $D$
- Existence Of Least Generalization: Yes, but if $S$ is a set of atoms, or $S$ and $B$ are all function-free or $B$ is ground
- On Horn clauses: Yes, e.g., if $B$ is ground definite program and $S$ is a set of definite program clause with same heads
- Deduction: Exist a SLD-deduction of $D$, where $C$ is the top clause and members in $B$ are input clauses
Generality Orders on Clauses

(with background knowledge)

- $C \succeq_B D$ if there exists a SLD-deduction of $D$, with $C$ as top clause and members of $B$ as input clauses.
- $C \prec_B D$ if there exists a deduction of $D$ from $\{C\} \cup B$ where $C$ occurs at most once as a leaf.
- $C \vdash_B D$ if there exists a deduction of $D$ from $\{C\} \cup B$. 
Generality Orders on Clauses

summary

Generalized subsumption \(\text{Weaker than}\) Relative subsumption \(\text{Weaker than}\) Relative implication

Subsumption \(\text{Weaker than}\) Relative subsumption \(\text{Weaker than}\) Implication

10/19/2004  The Foundations of Inductive Logic Programming
Refinement Operators

- functions from a single clause to a set of clauses:
  \[ \rho(C) : \text{downward refinement operators} \]
  \[ \delta(C) : \text{upward refinement operators} \]

- **Ideal (downward) operators:**
  - Locally finite : \( \rho(C) \) is finite
  - Complete : \( \forall C > D, \exists E \in \rho^*(c) \) s.t. \( D \equiv E \)
  - Proper : \( \rho(C) \subseteq \{ D \mid C \triangleright D \} \)
Refinement Operators

- Ideal $\rho(C)$ exists $\iff$ every $C$ has a finite set of downward cover set
- Ideal $\delta(C)$ exists $\iff$ every $C$ has a finite set of upward cover set
- Only subsumption order on set of atoms has finite downward and upward cover sets. Others don’t.
- So ideal operators do not exist for clauses structured by most practical orders.
Refinement Operators

- In practice we drop the properness, and use
- locally finite and complete operators.

- Such operators exist for clauses structured by
- subsumption order.
Conclusions

- Resolution based proof procedures are useful in ILP.
- ILP is a search problem.
- Different orders may be defined on the search space.
- The search could be achieve by applying refinement operators.
Thank you.