Theorem Proving Strategies

Summary

- Theorem-proving and search
- Fair derivation strategies [Ambrosius-Johann 7.5]

Objective of Theorem Proving

Prove validity

- Given a set of assumption H
- Given a conjecture ψ
- Prove whether $H \models \psi$, i.e. prove $H \cup \{\neg\psi\}$ unsatisfiable

Automated Theorem Proving

Build computer programs that prove validity

- H and ψ written in a formal language, e.g. FOL
- Deduction of $\Box$ from $H \cup \{\neg\psi\}$
- A deduction is a sequence of statements in the formal language (e.g. FOL formulas) logically connected by inference rules

Inference Rule

Inference Rule

$$f : \frac{\psi_1, \cdots, \psi_n}{\psi}$$

- f inference rule
- $\psi_1, \cdots, \psi_n$ premises
- ψ consequence
- Inference system: collection of inference rules

Example (Binary Resolution)

Inference rule

$$\frac{L_1 \vee C \;\; L_2 \vee D}{(C \vee D)\sigma} L_1\sigma = \neg L_2\sigma \quad \sigma \text{ Most General Unifier}$$

Inference System I

Example (Inference System)

- Binary resolution
- Factoring
- Tautology elimination
- Subsumption elimination

General properties of Inference Systems

Correctness and Completeness

- Correcteness of inference rules
  - consequences are logical consequences of premises: $\psi_1, \cdots, \psi_n \models \psi$
- Completeness
  - if $H \models \psi$
  - there is a deduction of ψ from H
- Refutational completeness
  - there is a deduction of $\Box$ if $H \cup \{\psi\}$ is unsatisfiable

Theorem Proving as a Search Problem

States and Production Rules

Completeness means if there is a proove we will find it, we still do not know how

- We can see theorem proving as a search problem
- States: sets of possible formulas (e.g. sets of clauses)
- Transformation or production rules: inference rules
- Successful states: containing complete proofs (e.g., states containing $\Box$)

Search Plan

Search plan Σ

- Rule selecting function: Given history of states returns which inference rule to use
- Premises selection function: Given history of states returns which premises to use for the inference rule
- Termination detection: Given the current state return true iff state is successful

The sequence of states obtained by applying Σ to I is a derivation

Theorem-Proving Strategy

Search Plan

- applying rules from I results in a non-deterministic derivation
- Applying Σ to I we have a deterministic derivation
- $I + \Sigma =$ theorem proving strategy
- We want Σ to be fair
- Σ is fair: if there is a successful state Σ will find it

Classification of Strategies

Classification

- Ordering Based:
  - work on a set of objects
  - implicitly generate many proof attempts
  - Ordering and Contraction very important
  - Ordered resolution with Level Saturation
- Goal Based:
  - work on one object
  - explicitly generate one proof attempt
  - backtrack if the current proof attempt cannot be completed into a proof
  - Linear resolution with ordered clause and tree expansion policies

Characteristics of Strategies

main features

	Ordering Based	Goal Based
data	set of objects	one object
proof attempt	many implicit	one explicit
backtracking	No	Yes

Ordering Based Strategies

Basic concepts

- transform $H \cup \neg\psi$ into Clauses
- S is the clausal form of theorem proving problem
- $\neg\psi$ is an additional assumption
- Inference rules work on S

Inference Systems for Ordering Based Strategies

Inference Rules

- General form:

$$f : \frac{S}{S'}$$

Example

$$\frac{S \cup \{L_1 \vee D, L_2 \vee C\}}{S \cup \{L_1 \vee D, L_2 \vee C, (C \vee D)\sigma\}} L_1\sigma = \neg L_2\sigma$$

The Inference Systems $\mathcal{R}$

Inference Rules

- Expansion rules (Binary Resolution + Factoring)

$$f : \frac{S}{S'} S \subseteq S'$$

- Contraction rules (Tautology elimination + Subsumption)

$$f : \frac{S}{S'} S' \subseteq S$$

- Orderings on clauses (e.g. simplification orderings) are frequently used to:
  - restrict application of inference rule containing expansion of S
  - decide which clause can be deleted.(e.g., clause entailed by smaller clauses)

Contraction and Redundancy

Contraction rules

- Forward: reduces newly generated clauses
- Backward: use new clauses to reduce existing ones

Contraction key in ordering based methods

- Delete existing clauses
- Prevents generation of useless clauses
- Aim: delete redundant clauses

Fair Derivation

Basic idea

- Completeness of $\mathcal{R}$ depends upon derivation strategies used
- Want to generate all useful clauses
- Fair derivation strategy: every rule in $\mathcal{R}$ that can be applied to clauses in the derivation is applied eventually
- In a fair derivation, every rule in $\mathcal{R}$ eventually fails to add new clauses

Fair Derivation: definition

Definition (Fair Derivation)

An $\mathcal{R}$-derivation $S_0, S_1, S_2, \cdots$ is fair if

$$S^\infty = \bigcup_{k \geq 0} \bigcap_{j \geq k} S_j$$

is closed under $\mathcal{R}$.

- S^∞ is called the set of persistent clauses

Set of persistent clauses

Example

Given $S = \{\neg P \vee Q, P, \neg Q\}$, compute S^{∞} considering $S_0, S_1, S_2, \cdots$ as follow:

- $S_0 = \{\,\}$
- $S_1 = S$
- $\cdots$
- $S_{n+2} = S_{n+1} \cup \{\text{Resolvents of } C_1 \text{ and } C_2 | C_1 \in S_{n+1} \text{ and } C_2 \in S_{n+1} \setminus S_n\}$

Level Saturation is Fair

Fairness of Level Saturation

- Assume no contraction
- At each stage generate all possible resolvents
- Maintains all resolvents in the set
- Eventually all possible resolvents will be generated

A redundancy criterion

redundant clause

A clause C is redundant with respect to S_i if

- C is a Tautology
- C is subsumed by a clause $D \in S_i$

Level Saturation with solution

Level Saturation with solution

- $S_0 = \{\,\}$
- $S_1 = S$
- $\ldots$
- One saturation step
  - $S_{2n+1} = S_{2n} \cup \{\text{Resolvents of } C_1 \text{ and } C_2 | C_1 \in S_{2n} \text{ and } C_2 \in S_{2n} \setminus S_{2n-2}\}$
- One reduction step
  - $S_{2n+2} = S_{2n+1} \setminus \{C \in S_{2n+1} | C \text{ is redundant with respect to } S_{2n}\}$
- Solution: reduction step

Level Saturation with solution is fair

Fairness

- Every clause C not in S^{∞} is redundant with respect to S^{∞}
- Suppose C and D are in S^{∞}
- Then C and D are not redundant otherwise eliminated before
- Let R be a resolvent of C and D.
- Then R was necessarily generated in some S_{2n} by construction
- If R was removed then R is subsumed by some other clause in some S_j $j > n$.
- Therefore R is redundant with respect to S^{∞}