Rule Induction

Learning Sets of Rules

Rules are very easy to understand; popular in data mining.

- **Variable Size**. Any boolean function can be represented.
- **Deterministic**.
- **Discrete and Continuous Parameters**.

Learning algorithms for rule sets can be described as

- **Constructive Search**. The rule set is built by adding rules; each rule is constructed by adding conditions.
- **Eager**.
- **Batch**.

Rule Set Hypothesis Space

- **Each rule is a conjunction of tests**. Each test has the form $x_j = v$, $x_j \leq v$, or $x_j \geq v$, where v is a value for x_j that appears in the training data.

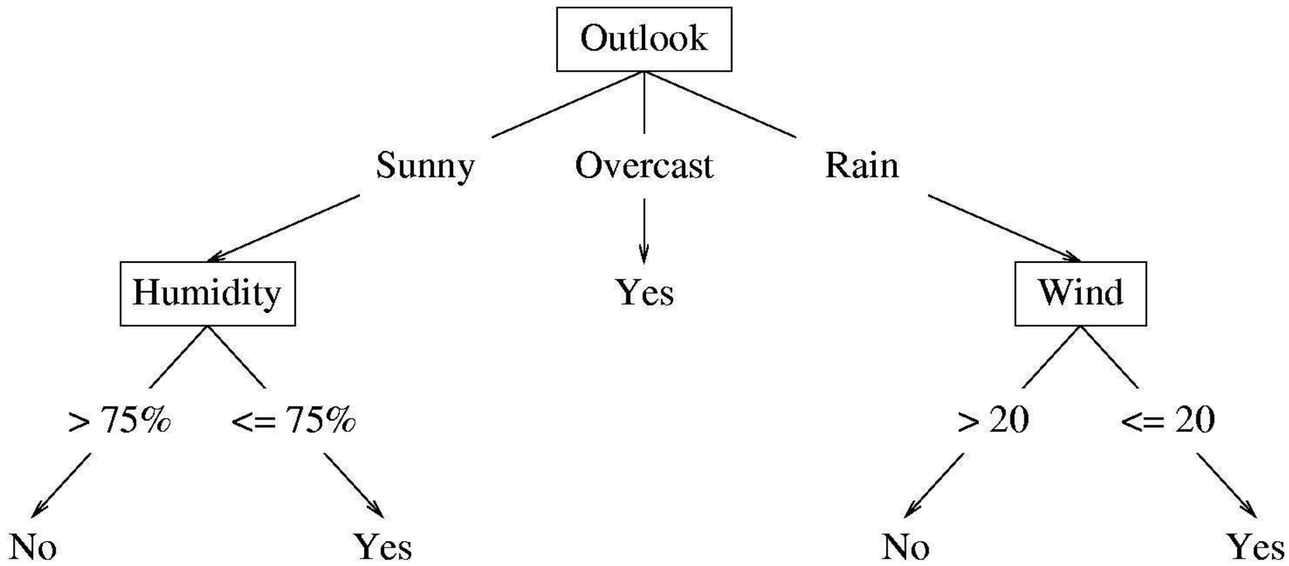
$$x_1 = \mathit{Sunny} \wedge x_2 \leq 75\% \Rightarrow y = 1$$

- **A rule set is a disjunction of rules**. Typically all of the rules are for one class (e.g., $y = 1$). An example is classified into $y = 1$ if *any* rule is satisfied.

$$\begin{aligned} x_1 = \mathit{Sunny} \wedge x_2 \leq 75\% &\Rightarrow y = 1 \\ x_1 = \mathit{Overcast} &\Rightarrow y = 1 \\ x_1 = \mathit{Rain} \wedge x_3 \leq 20 &\Rightarrow y = 1 \end{aligned}$$

Relationship to Decision Trees

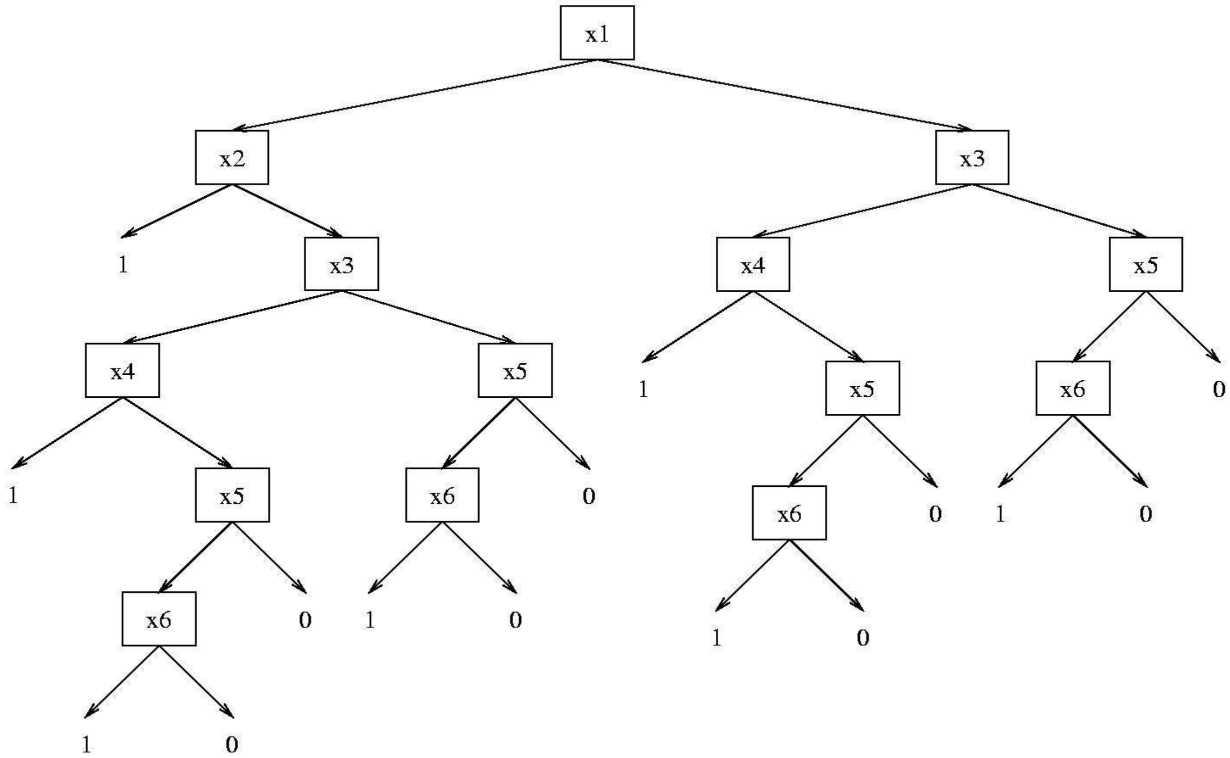
Any decision tree can be converted into a set of rules. The previous set of rules corresponds to this tree:



Relationship to Decision Trees

A small set of rules can correspond to a big decision tree, because of the *Replication Problem*.

$x_1 \wedge x_2 \Rightarrow y = 1$ $\qquad$ $x_3 \wedge x_4 \Rightarrow y = 1$ $\qquad$ $x_5 \wedge x_6 \Rightarrow y = 1$



Learning a Single Rule

We grow a rule by starting with an empty rule and adding tests one at a time until the rule "covers" only positive examples.

GROWRULE(S)

$R = \{\ \}$

repeat

choose best test $x_j \Theta v$ to add to R, where $\Theta \in \{=, \neq, \leq, \geq\}$

$S := S -$ all examples that do not satisfy $R \cup \{x_j \Theta v\}$.

until S contains only positive examples.

Choosing the Best Test

- Current rule R covers m_0 negative examples and m_1 positive examples.
  Let $p = \frac{m_1}{m_0+m_1}$.
- Proposed rule $R \cup \{x_j \Theta v\}$ covers m'_0 and m'_1 examples.
  Let $p' = \frac{m'_1}{m'_0+m'_1}$.
- $Gain = m'_1 \left[(-p \lg p) - (-p' \lg p')\right]$

We want to reduce our surprise (to the point where we are *certain*), but we also want the rule to cover many examples. This formula tries to implement this tradeoff.

Learning a Set of Rules
(Separate-and-Conquer)

GrowRuleSet(S)

$A = \{\ \}$

repeat

  $R :=$ GrowRule(S)

  Add R to A

  $S := S-$ all positive examples that satisfy R.

until S is empty.

return A

More Thorough Search Procedures

All of our algorithms so far have used greedy algorithms. Finding the smallest set of rules is NP-Hard. But there are some more thorough search procedures that can produce better rule sets.

- **Round-Robin Replacement**. After growing a complete rule set, we can delete the first rule, compute the set S of training examples not covered by any rule, and one or more new rules, to cover S. This can be repeated with each of the original rules. This process allows a later rule to "capture" the positive examples of a rule that was learned earlier.
- **Backfitting**. After each new rule is added to the rule set, we perform a few iterations of Round-Robin Replacement (it typically converges quickly). We repeat this process of growing a new rule and then performing Round-Robin Replacement until all positive examples are covered.
- **Beam Search**. Instead of growing one new rule, we grow B new rules. We consider adding each possible test to each rule and keep the best B resulting rules. When no more tests can be added, we choose the best of the B rules and add it to the rule set.

Probability Estimates From Small Numbers

When m_0 and m_1 are very small, we can end up with

$$p = \frac{m_1}{m_0 + m_1}$$

being very unreliable (or even zero).

Two possible fixes

- **Laplace Estimate**. Add 1/2 to the numerator and 1 to the denominator:

  $$p = \frac{m_1 + 0.5}{m_0 + m_1 + 1}$$

  This is essentially saying that in the absence of any evidence, we expect $p = 1/2$, but our belief is very weak (equivalent to 1/2 of an example).

- **General Prior Estimate**. If you have a prior belief that $p = 0.25$, you can add any number k to the numerator and $4k$ to the denominator.

  $$p = \frac{m_1 + k}{m_0 + m_1 + 4k}$$

  The larger k is, the stronger our prior belief becomes.

Many authors have added 1 to both the numerator and denominator in rule learning cases (weak prior belief that $p = 1$).

Learning Rules for Multiple Classes

What if rules for more than one class?

Two possibilities:

- Order rules (decision list)
- Weighted vote (e.g., weight = accuracy $\times$ coverage)

Learning First-Order Rules

Why do that?

- Can learn sets of rules such as

  $$Ancestor(x, y) \leftarrow Parent(x, y)$$
  $$Ancestor(x, y) \leftarrow Parent(x, z) \wedge Ancestor(z, y)$$

- The PROLOG programming language:
  programs are sets of such rules

First-Order Rule for Classifying Web Pages

[Slattery, 1997]

course(A) $\leftarrow$
  has-word(A, instructor),
  $\neg$ has-word(A, good),
  link-from(A, B),
  has-word(B, assign),
  $\neg$ link-from(B, C)

Train: 31/31, Test: 31/34

FOIL (First-Order Inductive Learner)

Same as propositional separate-and-conquer, except:

- Different candidate specializations (literals)
- Different evaluation function

Specializing Rules in FOIL

Learning rule: $P(x_1, x_2, \ldots, x_k) \leftarrow L_1 \ldots L_n$

Candidate specializations add new literal of form:

- $Q(v_1, \ldots, v_r)$, where at least one of the v_i in the created literal must already exist as a variable in the rule.
- $Equal(x_j, x_k)$, where x_j and x_k are variables already presentin the rule
- The negation of either of the above forms of literals

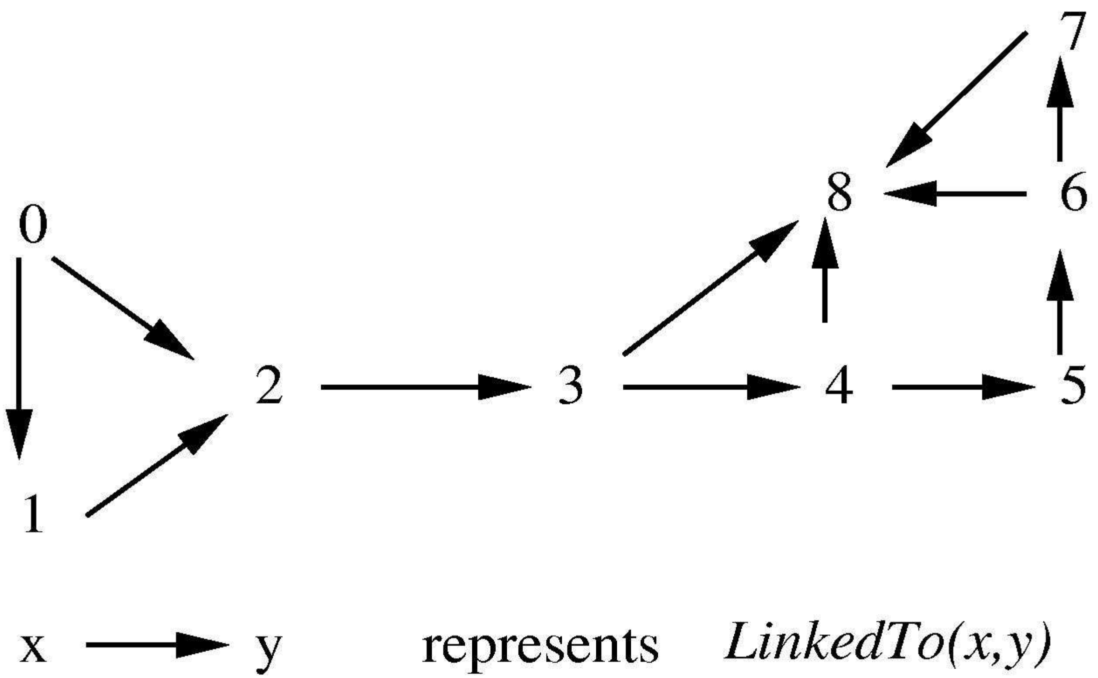
Information Gain in FOIL

$$Foil\_Gain(L, R) \equiv t \left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right)$$

Where

- L is the candidate literal to add to rule R
- p_0 = number of positive bindings of R
- n_0 = number of negative bindings of R
- p_1 = number of positive bindings of $R + L$
- n_1 = number of negative bindings of $R + L$
- t = no. of positive bindings of R also covered by $R + L$

FOIL Example



Target function:

- *CanReach(x,y)* true iff directed path from x to y

Instances:

- Pairs of nodes, e.g $\langle 1, 5 \rangle$, with graph described by literals *LinkedTo(0,1)*, $\neg$ *LinkedTo(0,8)* etc.

Hypothesis space:

- Each $h \in H$ is a set of Horn clauses using predicates $LinkedTo$ (and $CanReach$)

Induction as Inverted Deduction

Induction is finding h such that

$$(\forall \langle x_i, f(x_i) \rangle \in D) \ B \wedge h \wedge x_i \vdash f(x_i)$$

where

- x_i is ith training instance
- $f(x_i)$ is the target function value for x_i
- B is other background knowledge

So let's design inductive algorithm by inverting operators for automated deduction.

Induction as Inverted Deduction

"Pairs of people $\langle u, v \rangle$ such that child of u is v"

$$\begin{aligned} f(x_i) &: \quad Child(Bob, Sharon) \\ x_i &: \quad Male(Bob), Female(Sharon), Father(Sharon, Bob) \\ B &: \quad Parent(u,v) \leftarrow Father(u,v) \end{aligned}$$

What satisfies $(\forall \langle x_i, f(x_i) \rangle \in D) \; B \wedge h \wedge x_i \vdash f(x_i)$?

$$\begin{aligned} h_1 &: \quad Child(u,v) \leftarrow Father(v,u) \\ h_2 &: \quad Child(u,v) \leftarrow Parent(v,u) \end{aligned}$$

Induction as Inverted Deduction

We have mechanical *deductive* operators $F(A, B) = C$, where $A \wedge B \vdash C$

Need *inductive* operators

$$O(B, D) = h \text{ where } (\forall \langle x_i, f(x_i) \rangle \in D)\ (B \wedge h \wedge x_i) \vdash f(x_i)$$

Induction as Inverted Deduction

Positives:

- Subsumes earlier idea of finding h that "fits" training data
- Domain theory B helps define meaning of "fit" the data

$$B \wedge h \wedge x_i \vdash f(x_i)$$

- Suggests algorithms that search H guided by B

Induction as Inverted Deduction

Negatives:

- Doesn't allow for noisy data. Consider

$$(\forall \langle x_i, f(x_i) \rangle \in D)\ (B \wedge h \wedge x_i) \vdash f(x_i)$$

- First order logic gives a *huge* hypothesis space H

  $\rightarrow$ Overfitting

  $\rightarrow$ Intractability of calculating all acceptable h's

Deduction: Resolution Rule

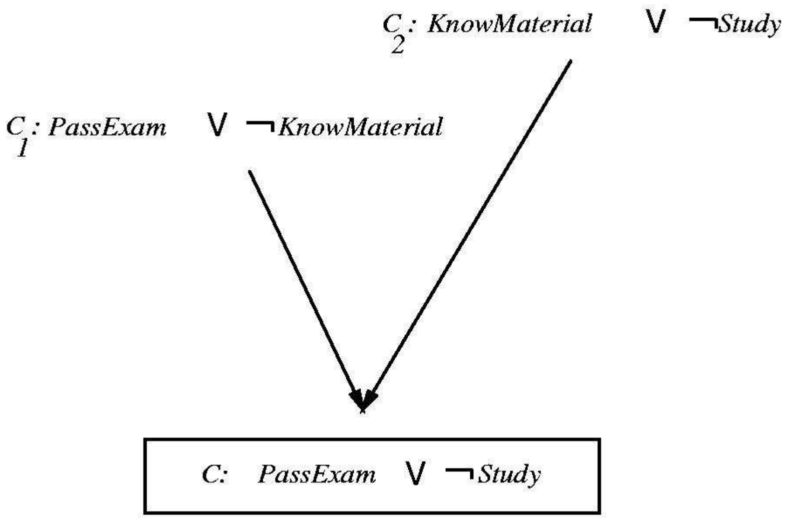
$$\frac{\begin{array}{ccc} P & \vee & L \\ \neg L & \vee & R \end{array}}{\begin{array}{ccc} P & \vee & R \end{array}}$$

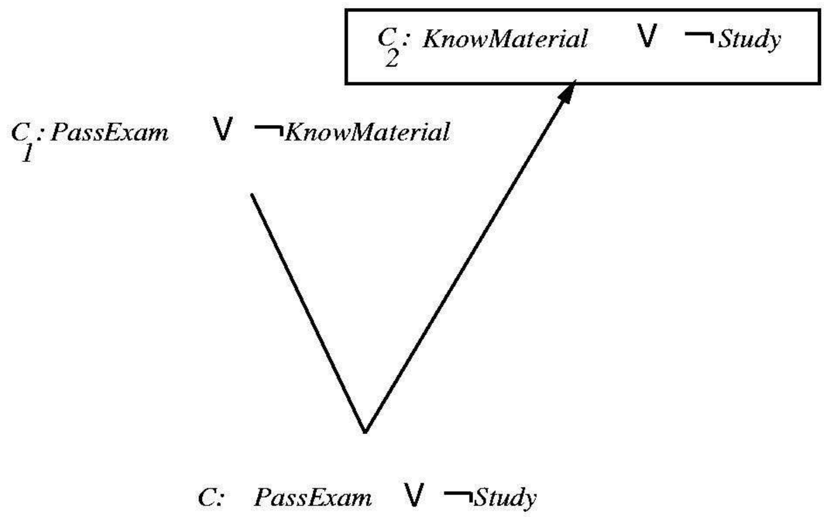
1. Given initial clauses C_1 and C_2, find a literal L from clause C_1 such that $\neg L$ occurs in clause C_2
2. Form the resolvent C by including all literals from C_1 and C_2, except for L and $\neg L$. More precisely, the set of literals occurring in the conclusion C is

$$C = (C_1 - \{L\}) \cup (C_2 - \{\neg L\})$$

where $\cup$ denotes set union, and "$-$" is set difference

Inverting Resolution





Inverted Resolution (Propositional)

1. Given initial clauses C_1 and C, find a literal L that occurs in clause C_1, but not in clause C.

2. Form the second clause C_2 by including the following literals

$$C_2 = (C - (C_1 - \{L\})) \cup \{\neg L\}$$

First-Order Resolution

1. Find a literal L_1 from clause C_1, literal L_2 from clause C_2, and substitution θ such that $L_1\theta = \neg L_2\theta$

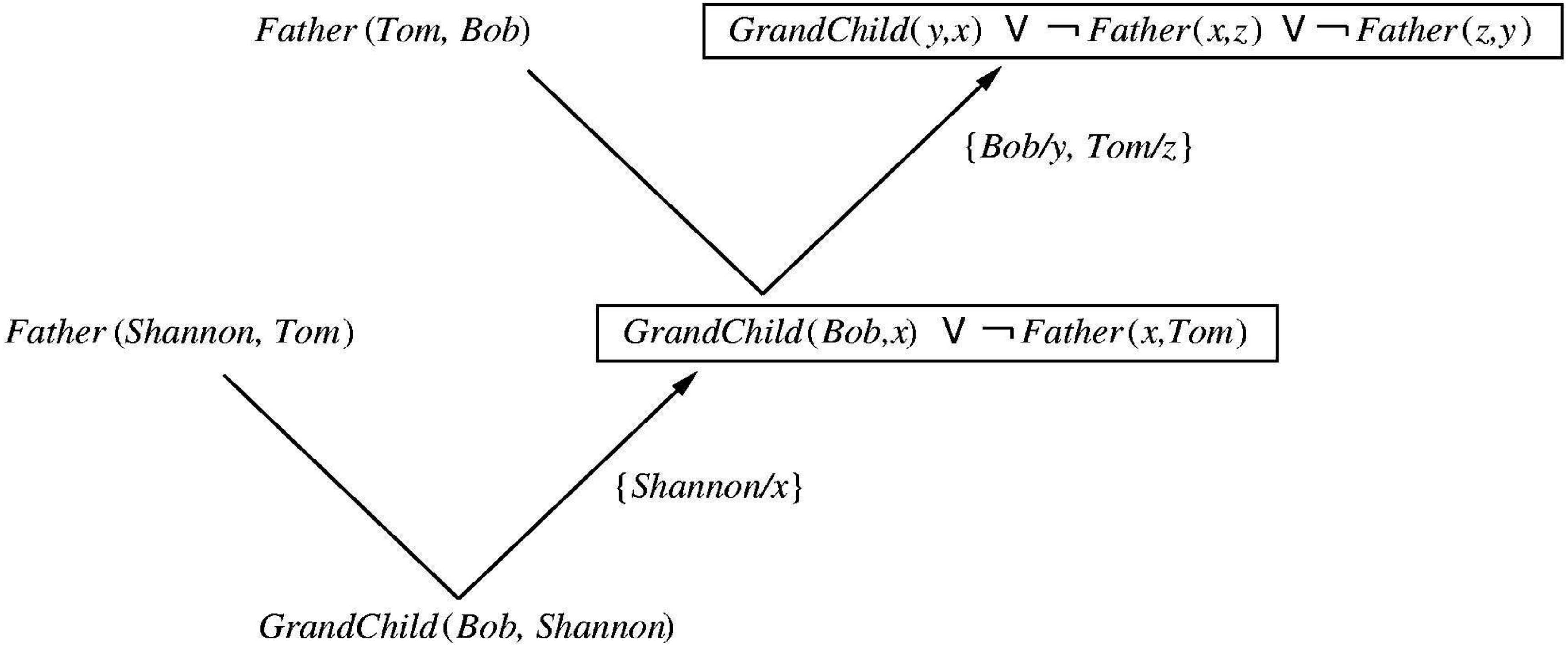
2. Form the resolvent C by including all literals from $C_1\theta$ and $C_2\theta$, except for $L_1\theta$ and $\neg L_2\theta$. More precisely, the set of literals occurring in the conclusion C is

$$C = (C_1 - \{L_1\})\theta \cup (C_2 - \{L_2\})\theta$$

Inverting First-Order Resolution

$$C_2 = (C - (C_1 - \{L_1\})\theta_1)\theta_2^{-1} \cup \{\neg L_1 \theta_1 \theta_2^{-1}\}$$

Cigol



Father (Tom, Bob)
GrandChild(y,x) ∨ ¬ Father(x,z) ∨ ¬ Father(z,y)
{Bob/y, Tom/z}
Father (Shannon, Tom)
GrandChild(Bob,x) ∨ ¬ Father(x,Tom)
{Shannon/x}
GrandChild(Bob, Shannon)

Progol

PROGOL: Reduce comb explosion by generating the most specific acceptable h

1. User specifies H by stating predicates, functions, and forms of arguments allowed for each
2. PROGOL uses sequential covering algorithm.
   For each $\langle x_i, f(x_i)\rangle$
   - Find most specific hypothesis h_i s.t.
     $B \wedge h_i \wedge x_i \vdash f(x_i)$
     - actually, considers only k-step entailment
3. Conduct general-to-specific search bounded by specific hypothesis h_i, choosing hypothesis with minimum description length

Rule Induction: Summary

- Rule grown by adding one antecedent at a time
- Rule set grown by adding one rule at a time
- Propositional or first-order
- Alternative: inverse resolution