The implication order ($\models$)

Recall subsumption $A \succeq_s B$ iff $\exists\, \theta$ s.t. $A\theta \subseteq B$

$A \models B$ iff $M_A \subseteq M_B$.

Q: Will implication benefit us more?

(1) It can handle relationships between recursive clauses

$$C = P(f(x)) \leftarrow P(x)$$
$$D = P(f^2(x)) \leftarrow P(x)$$

- C_1 may entail C_2 though not subsume C_2
- Two tautologies may NOT be subsume equivalent

(2) Subsumption (esp for LGG) can over-generalize

$D_1 = P(f^2(a)) \leftarrow P(a)$ $\quad D_2 = P(f(b)) \leftarrow P(b)$

$\{(a, \langle 1,1,1\rangle), (f(a), \langle 1,1\rangle), (f^2(a), \langle 1\rangle), (a, \langle 2\rangle)\}$

$(b, \langle 1,1\rangle), (f(b), \langle 1\rangle), (b, \langle 2\rangle)$

C_1: $P(f(a)) \leftarrow P(a)$

C_2: $P(f(b)) \leftarrow P(b)$

LGS: $P(f(x)) \leftarrow P(x)$

D_1: $P(f(a)) \leftarrow P(a)$

* D_2: $P(f^2(b)) \leftarrow P(b)$

LGS: $P(f(y)) \leftarrow P(x)$

More desirable: $P(f(x)) \leftarrow P(x)$

LGI

Claim that under $\succcurlyeq_S$, $\text{LGI} \preccurlyeq_S \text{LGS}$

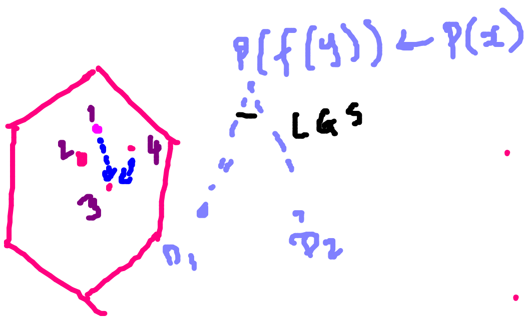
Implication is reflexive & transitive

$\Downarrow$

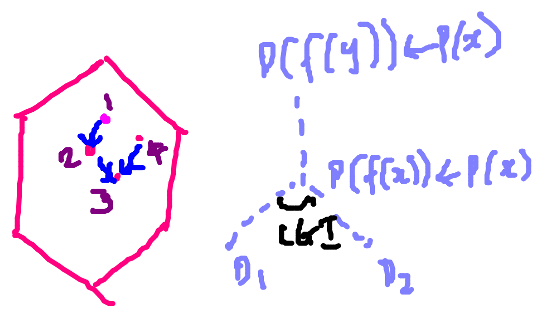
Quasi order

$\Downarrow$

Partial order over $\underbrace{\text{equivalent classes}}_{\text{w.r.t implication}}$



Subsumption
PO



Implication
PO

{ More fine-grained
Higher resolution }

Holds even for clauses without function symbols!

D_1 :- $P(x, y, z) \leftarrow P(y, z, x)$

D_2 :- $P(x, y, z) \leftarrow P(z, x, y)$

Check using resolution (cycling) that

$D_1 \models D_2$

& $D_2 \models D_1$

$\Rightarrow$ $D_1 \equiv D_2$

$LGI(D_1, D_2) = D_1 \equiv D_2$

$LGS(D_1, D_2) = ?$

Note: $D_1 \not\geq_s D_2$ & $D_2 \not\geq_s D_1$

$LGS(D_1, D_2) = \{ \underbrace{p(x,y,z) \leftarrow p(u,v,w)}_{\text{Overgeneralization}} \}$

Problem area :- Recursive clauses in subsumption order

$\underbrace{P}_{+}$ $\quad \neg \underbrace{P}_{-}$

③ Subsumption cannot help us compare between a theory Σ with another theory Σ' (or clause C). Implication CAN

Theory refinement

$\Sigma \rightarrow \Sigma'$

$\Sigma = \{ \underbrace{(P \leftarrow Q)}_{\text{CNF}}, (Q \leftarrow R) \}$ $\quad C = P \leftarrow R$

$\Sigma \models C$ ... but subsumption not helpful.

LGI & GSI

→ For general formulae α_1 & α_2.

? for clauses

[sometimes used] $LGI(\alpha_1, \alpha_2) = \alpha_1 \wedge \alpha_2 \vDash \alpha_1,\ \vDash \alpha_2$

[hardly used] $GSI(\alpha_1, \alpha_2) = \alpha_1 \vee \alpha_2 \Dashv \alpha_2,\ \Dashv \alpha_1$

holds even if α_1 & α_2 are clauses

$$
\begin{tikzcd}
 & \phi_1 \wedge \phi_2 \arrow[dl] \arrow[dr] & \\
\phi_1 \arrow[dr] & & \phi_2 \arrow[dl] \\
 & \phi_1 \vee \phi_2 &
\end{tikzcd}
$$

If ϕ_1 & ϕ_2 are clauses is $\phi_1 \vee \phi_2$ a clause?

Please standardize apart ϕ_1 & ϕ_2

LGI for clauses

↳ Exists ($\mathcal{A}$ is computable) for every finite set of clauses Σ containing at least one (1) function free, (2) non-tautologous clause

INPUT: A finite set $\mathcal{S}$ of clauses, containing at least one non-tautologous function-free clause;
OUTPUT: An LGI of $\mathcal{S}$ in $\mathcal{C}$;
Remove all tautologies from $\mathcal{S}$, call the remaining set $\mathcal{S}'$;
Let m be the number of distinct terms (including subterms) in $\mathcal{S}'$, let $V = \{x_1, \ldots x_m\}$;

f(c) c

Let $\mathcal{G}$ be the (finite) set of all clauses which can be constructed from predicate symbols and constants in $\mathcal{S}'$ and variables in V; → Do not make extra fn applications
Let $\{U_1, \ldots, U_n\}$ be the set of all subsets of $\mathcal{G}$;
Let H_i be an LGS (computed using algorithm in Figure 2.4) of U_i, for every $1 \leq i \leq n$;
Remove from $\{H_1, \ldots, H_n\}$ all clauses which do not imply $\mathcal{S}'$ (since each H_i is function-free, this implication is decidable), and standardize the remaining clauses $\{H_1, \ldots, H_q\}$ apart. ;

Provable

return $H = H_1 \cup \ldots \cup H_q$;

Note :- S' is constructed from S

If Σ = finite set of ground clauses & C is ground cl. s.t C = $L_1 \vee \ldots \vee L_n$, A is finite set of ground atoms from Σ & C, then . .

$\Sigma \models C$ iff $\Sigma \cup \{\neg L_1, \ldots, \neg L_n\}$ is satisfiable

by deduction thm

iff $\Sigma \cup \{\neg L_1 \ldots \neg L_n\}$ has no herbrand model

finite check ⇒ decidable iff no subset of A is a H-model of $\Sigma \cup \{\neg L_1 \ldots \neg L_n\}$.

U_H (H-univ)

Negative Results

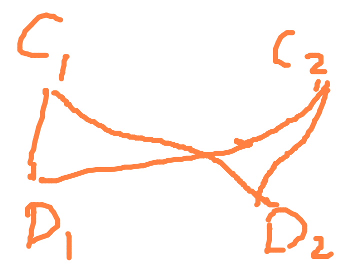
① $\exists$ pairs of horn clauses that have no LGI in $\mathcal{H}$.

$$D_1 = P(f^2(x)) \leftarrow P(x)$$

$$D_2 = P(f^3(x)) \leftarrow P(x)$$

$$C_1 = P(f(x)) \leftarrow P(x)$$

$$C_2 = P(f^2(y)) \leftarrow P(x)$$



But $LGI(D_1, D_2)$ exists in C

$$= P(f(x)) \vee P(f^2(y)) \vee \neg P(x)$$

② ∃ pairs of horn clauses with no GSI in $\mathcal{H}$.

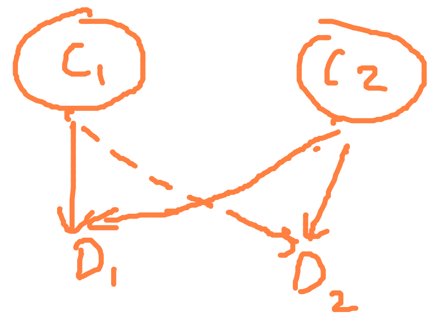
$D_1 = P(f^2(x)) \leftarrow P(x)$

$D_2 = P(f^3(x)) \leftarrow P(x)$

$C_1 = P(f(x)) \leftarrow P(x)$

$C_2 = P(f^2(y)) \leftarrow P(x)$

∃ no $GSI(C_1, C_2)$



③ for general clause all having fn symbols,

LGI is open question

$\underbrace{\text{LGI}}$ → existence, computation

④ Covers --- -ve results carry over. . .

$\{P(x_1, x_2)\}$ has no upward cover.

$\{P(x_1, x_2), P(x_2, x_1)\}$ has no finite complete set of downward covers.

Positive results

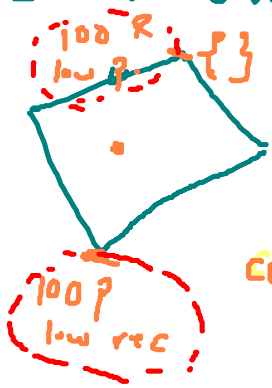
① If C is a set of function-free clauses, then $\langle C, \models \rangle$ is a lattice

ILP & Structured search space : Depends on language of H, & on notion of cover $\Downarrow$ Prof Proc.

Recall that ILP is for developing H given B s.t $P(\text{covers}(H, B, E^+))$ is high & $P(\text{covers}(H, B, E^-))$ is low

covers so far ignored B

① Lattice structure is a generality order over the hypothesis $\Rightarrow$ gives you a search space



cover few in E^+ & few in E^-

cover many in E^+ & possibly many in E^-. Eg:- $\{\} \equiv \top \models$ everything

② To prune large parts of search space

Might require use of monotonic $\boxed{\text{function}}$

1. When generalizing C to C', $C' \succ C$, all the examples covered by C will also be covered by C' (since if $\mathcal{B} \cup \{C\} \models e$ (e being an example) holds then also $\mathcal{B} \cup \{C'\} \models e$ holds). This property is used to prune the search of more general clauses when e **is a negative example: if e is inconsistent (covers a negative example) then all its generalizations will also be inconsistent**. Hence, the generalizations of C do not need to be considered.

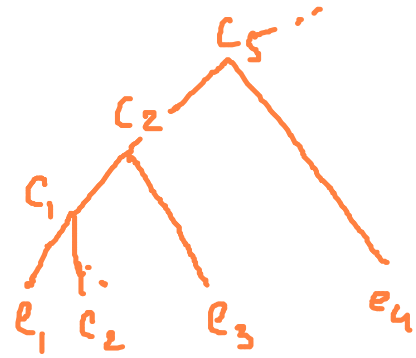
   $f_{e^-}(c) \uparrow$

2. When specializing C to C', $C \succ C'$, an example not covered by C will not be covered by any of its specializations either (since if $\mathcal{B} \cup \{C\} \not\models e$ holds then also $\mathcal{B} \cup \{C'\} \not\models e$ holds). This property is used to prune the search of more specific clauses when e **is an uncovered positive example: if C does not cover a positive example none of its specializations will**. Hence, the specializations of C do not need to be considered.

   $-f_{e^+}(c) \downarrow$

③ Basis for 2 ILP techniques.

① Bottom up building of LGGs from training examples, relative to background knowledge



May want some pre/post processing

② Top-down search using refinement operators (along "covers" edge)

Comparing generality ordering

$$\text{if } C \succeq_\theta D \text{ then } C \succeq_\models D$$

but

not vice – versa

For example, the above holds for:

$C:\ natural(s(X)) \leftarrow natural(X)$

$D:\ natural(s(s(X))) \leftarrow natural(X)$

Subsumption theorem explains the asymmetry

If Σ is a set of clauses and D is a clause, then $\Sigma \models D$ iff D is a tautology, or there exists a clause $D' \succeq_\theta D$ which can be derived from Σ using some form of resolution.

Σ

T_*

D' $\quad T$

$\theta\checkmark$

D

Asymmetry is because $C \in \Sigma$ may be self-recursive OR D may be a tautology.

Principled approach to generality ordering

Given a set of clauses S, clauses $C, D \in S$ and quasi-orders $\succeq_1$ and $\succeq_2$ on S, then $\succeq_1$ is *stronger* than $\succeq_2$ if $C \succeq_2 D$ implies $C \succeq_1 D$. If also for some $C, D \in S$ $C \not\succeq_2 D$ and $C \succeq_1 D$ then $\succeq_1$ is *strictly stronger* than $\succeq_2$

$\succeq_{\models}$ is strictly stronger than $\succeq_\theta$

SOME GENERALITY ORDERINGS

decreasing generality

– $C \succeq_{\models} D$ iff $C \models D$

– $C \succeq_\theta$ iff there is a substitution θ s.t. $C \subseteq D$

– $C \succeq_{\theta'} D$ iff every literal in D is *compatible* to a literal in C and $C \succeq_\theta D$

– $C \succeq_{\theta''} D$ iff $|C| \geq |D|$ and $C \succeq_{\theta'} D$

To avoid silly cases such as $\{P(x,y)\} \succeq_\theta \{P(x,y), P(y,x)\}$

To avoid silly cases of $\succeq_\theta$ such as $\underbrace{\{P(x,y)\}}_{C} \succeq \underbrace{\{P(x',y'), Q(y')\}}_{D}$

Which generality to choose?

↳ Strongest ordering (so that you dont overgeneralize or overspecialize)

↳ But practical. (LGI is too expensive)

↳ $\models$ is undecidable

↳ $\subseteq$ is decidable but NP-complete even for $\mathcal{H}$.

↳ Restrictions for making it practical

- Determinate Horn clauses: There exists an ordering of literals in C and exactly one substitution θ s.t. $C\theta \subseteq D$. *as before dict.*

- $k - local$ Horn clauses. Partition a Horn clause into k "disjoint" sub-parts and perform k independent subsumption tests.

In summary, the subsumption order on clausal languages is used most often in ILP as the generality order (in contrast to the implication order), owing to its following properties:

1. Further, subsumption is more tractable and efficiently implementable. Subsumption between clauses is a decidable relation, whereas implication is not. The flip side is that subsumption is a weaker relation.

2. Equivalence classes under subsumption can be represented by a single reduced clause. Reduction can be undone by inverse reduction (*c.f.* Section 2.3).

3. Every finite set of clauses (function free or not)has a least generalization (LGS) and greatest specialization (GSS) under subsumption in $\mathcal{C}$. Hence $< \mathcal{C}, \succeq >$ is a lattice. The same is not true for the implication quasi-ordering $\succeq_{\models}$ (for restricted languages *lubs* for $\succeq_{\models}$ may well exist).

Order	*lub*	*glb*
$\succeq_\theta$	$\surd$	$\surd$
$\succeq_{\models}$	$\times$	$\surd$

4. Every finite set of Horn clauses has a least generalization (LGS) and greatest specialization (GSS) under subsumption in $\mathcal{H}$. Hence $< \mathcal{H}, \succeq >$ is a lattice. The same does not hold for the implication order.

   Implication Story for $\mathcal{H}$ is sad

5. The negative results for covers hold for subsumption as well as implication.

Incorporating Background knowledge :

Why B ?

$H_1 = CP(x) \leftarrow F(x)$

LGS

$D_1 = CP(x) \leftarrow S(x), F(x), D(x),$

$D_2 = CP(x) \leftarrow F(x), C(x)$

Say D_1 & D_2 are +ve examples

Say we also have B.

$B_1 = P(x) \leftarrow C(x)$ .. D_2

$B_2 = P(x) \leftarrow D(x)$ ... D_1

$B_3 = S(x) \leftarrow C(x)$ .. D_2

H_1

more satisfactory
(TB is not a EP)

$H_B = CP(x) \leftarrow S(x), F(x), P(x)$

D_1 D_2

Note:- $H_B \cup B \vDash D_1$ & D_2 but $H_B \nvDash D_1 \wedge D_2$

Three generality orderings with $\mathcal{B}$

1. Plotkin's relative subsumption ($\succsim_{\mathcal{B}}$)

C & $\mathcal{B}$ arbit

if $\mathcal{B} = \phi$, $\succsim_{\mathcal{B}} \equiv \succsim_{\theta}$

2. Relative implications ($\models_{\mathcal{B}}$)

if $\mathcal{B} = \phi$, $\models_{\mathcal{B}} \equiv \succsim_{\models}$

-ve results for covers extend.

C & $\mathcal{B}$ should be definite ← 3. Generalized subsumption ($\geq_{\mathcal{B}}$)