TECHNISCHE UNIVERSITÄT DRESDEN

Computational Logic ∴ Group

Foundations of Knowledge Representation

Lecture 3: Horn Logics and Datalog

Hannes Straß

based on slides of
Bernardo Cuenca Grau,
Ian Horrocks, and
Przemysław Wałęga

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Propositional Horn Fragment

PL Horn Fragment: only allows the following formulas ($n > 0$):

$$P_1 \wedge \ldots \wedge P_n \rightarrow Q \qquad \textit{rules}$$
$$P \qquad \textit{facts}$$

With P_i, Q being atoms, and where Q can be $\bot$.

Horn Clauses: Clauses with at most one positive literal.

$$\neg P_1 \vee \ldots \vee \neg P_n \vee Q$$

(Fact) entailment. Instance is set $\mathcal{H}$ of Horn formulas and atom P
Answer is true if every model of $\mathcal{H}$ is also a model of P
and false otherwise.

PL Horn entailment is solvable in polynomial time.

Lifting PL Horn to FOL Horn

First-Order Horn Clauses: Clauses with at most one positive literal

But now, atoms can contain variables, constants, and functions

Some examples of First-Order Horn clauses:

$$\neg JuvArthritis(x) \vee Arthritis(x)$$
$$\neg Arthritis(x) \vee \neg JuvDisease(x) \vee JuvArthritis(x)$$
$$\neg Child(x) \vee \neg Adult(x)$$
$$\neg Affects(x, y) \vee Person(y)$$
$$\neg JuvDisease(x) \vee Affects(x, f(x))$$
$$JuvDisease(JRA)$$

Horn Logics

Horn Formulas: FOL sentences that in CNF yield Horn clauses.

Horn Logics: Syntactic FOL fragments allowing only Horn Formulas.

Some examples of Horn formulas:

$$\forall x.(\mathit{Arthritis}(x) \land \mathit{JuvDisease}(x) \rightarrow \mathit{JuvArthritis}(x))$$
$$\forall x.(\mathit{Child}(x) \land \mathit{Adult}(x) \rightarrow \bot)$$
$$\forall x.(\forall y.(\mathit{Affects}(x, y) \rightarrow \mathit{Person}(y)))$$
$$\forall x.(\mathit{JuvDisease}(x) \rightarrow \exists y.(\mathit{Affects}(x, y) \land \mathit{Child}(y)))$$
$$\forall x.(\forall y.(\forall z.(\mathit{fatherOf}(x, y) \land \mathit{brotherOf}(x, z) \rightarrow \mathit{uncleOf}(z, y))))$$
$$\mathit{JuvDisease}(\mathit{JRA})$$

Expressivity

We cannot express "disjunctive formulas":

- Covering statements:

$$\forall x.(\mathit{Person}(x) \rightarrow \mathit{Adult}(x) \vee \mathit{Child}(x) \vee \mathit{Teenager}(x))$$

- Negation on the left of implication

$$\forall x.(\mathit{Person}(x) \wedge \neg \mathit{Woman}(x) \rightarrow \mathit{Man}(x))$$

As well as many others . . .

Note, however, that some formulas apparently "disjunctive" are Horn

$$\forall x.(\mathit{Adult}(x) \vee \mathit{Child}(x) \vee \mathit{Teenager}(x) \rightarrow \mathit{Person}(x))$$

Existential Rules

$$\forall\vec{x}.\forall\vec{z}.(\varphi(\vec{x},\vec{z}) \rightarrow \exists\vec{y}.\psi(\vec{x},\vec{y})) \qquad \textit{Existential Rule}$$
$$\forall\vec{x}.\forall\vec{z}.(\varphi(\vec{x},\vec{z}) \rightarrow \bot) \qquad \bot\textit{-Rule}$$
$$P(\vec{a}) \qquad \textit{Fact}$$

$\varphi(\vec{x},\vec{z})$: conjunction of function-free atoms with vars $\vec{x} \cup \vec{z}$.
$\psi(\vec{x},\vec{y})$: conjunction of function-free atoms with vars $\vec{x} \cup \vec{y}$.

$$\forall x.(\mathit{Arthritis}(x) \wedge \mathit{JuvDisease}(x) \rightarrow \mathit{JuvArthritis}(x)) \qquad \textit{Rule}$$
$$\forall x.(\mathit{Child}(x) \wedge \mathit{Adult}(x) \rightarrow \bot) \qquad \bot\textit{-Rule}$$
$$\forall x.(\mathit{JuvDisease}(x) \rightarrow \exists y.(\mathit{Affects}(x,y) \wedge \mathit{Child}(y)))) \qquad \textit{Rule}$$
$$\mathit{JuvDisease}(\mathit{JRA}) \qquad \text{Fact}$$

Examples of Horn formulas outside this logic:

$$\forall x.(\mathit{Adult}(x) \vee \mathit{Child}(x) \vee \mathit{Teenager}(x) \rightarrow \mathit{Person}(x))$$

Reasoning with Existential Rules

Fact Entailment: An instance is a pair $\langle \mathcal{R}, \mathcal{F} \rangle$ of rules and facts and a fact P
The answer is true iff $\langle \mathcal{R}, \mathcal{F} \cup \{\neg P\} \rangle$ is unsatisfiable.

Resolution can be optimised for Horn clauses.

General strategy: allow only certain kinds of resolution inferences:

- Need to show completeness

  Unsatisfiability must imply that the empty clause is derivable.

- No need to show soundness

  Still just resolution, which is sound.

Recall FOL Resolution Rule

$$\frac{\alpha \lor \phi \quad \neg\beta \lor \psi}{(\phi \lor \psi)MGU(\alpha,\beta)}$$

α, β are atoms
MGU(α, β) is Most General Unifier of α and β

Examples:

$$\frac{(\neg ArthritisPat(x) \lor Affects(f(x), x)) \quad ArthritisPat(g(a))}{Affects(f(g(a)), g(a)))} \qquad \{x \mapsto g(a)\}$$

$$\frac{Affects(x, John) \quad \neg Affects(JRA, y)}{\square} \qquad \{x \mapsto JRA, y \mapsto John\}$$

$$\frac{JuvDisease(h(g(f(x), a))) \quad \neg JuvDisease(h(g(y, y)))}{\textit{Rule not applicable}}$$

Recall FOL Factoring Rule

$$\frac{\gamma \vee \delta \vee \psi}{(\gamma \vee \psi) MGU(\gamma, \delta)} \quad \gamma, \delta \text{ literals, same sign}$$

Examples:

$$\frac{ArthritisPat(x) \vee Affects(f(x), x) \vee ArthritisPat(g(a))}{Affects(f(g(a)), g(a)) \vee ArthritisPat(g(a))} \quad \{x \mapsto g(a)\}$$

$$\frac{Affects(x, John) \vee Affects(JRA, y)}{Affects(JRA, John)} \quad \{x \mapsto JRA, y \mapsto John\}$$

$$\frac{\neg JuvDisease(h(g(f(x), a))) \vee \neg JuvDisease(h(g(y, z)))}{\neg JuvDisease(h(g(f(x), a)))} \quad \{y \mapsto f(x), z \mapsto a\}$$

Recall FOL Resolution Procedure

```
procedure SAT(𝒮)
    repeat
        for all clauses C1, C2 in 𝒮 do
            𝒮 := 𝒮 ∪ resolve(C1, C2)
        end for
    until No new clause can be added to 𝒮 or □ ∈ 𝒮
    If □ ∈ 𝒮 return false
    return true
end procedure
```

Function resolve(C_1, C_2) applies FO resolution in all possible ways, and then applies factoring in all possible ways.

Resolution with Free Selection

Resolution with free selection: a complete strategy

- Calculus parameterised by a Selection Function S
- S assigns to each Horn clause C a non-empty subset of its atoms:
  - $S(C)$ contains the single positive literal, OR
  - $S(C)$ contains a subset of negative literals
- Restrict resolution such that we only resolve on selected atoms

We are free to design the selection function ourselves!

If we satisfy the basic constraints, completeness is guaranteed

Resolution with Free Selection

A reasonable selection function:

- Select the set of all negative literals in each clause
- If there is no negative literal, select the (unique) positive literal

$$
\begin{aligned}
A(x) \to \exists y.R(x,y) \land B(y) &\quad\leadsto\quad \neg A(x) \lor R(x, f(x)) \\
&\quad\quad \neg A(x) \lor B(f(x)) \\
B(x) \to C(x) &\quad\leadsto\quad \neg B(x) \lor C(x) \\
R(x,y) \land C(y) \to D(y) &\quad\leadsto\quad \neg R(x,y) \lor \neg C(y) \lor D(x) \\
A(a) &\quad\leadsto\quad A(a)
\end{aligned}
$$

We want to see whether $D(a)$ follows

Resolution with Free Selection

$$\neg A(x) \vee R(x, f(x)) \qquad (1)$$
$$\neg A(x) \vee B(f(x)) \qquad (2)$$
$$\neg B(x) \vee C(x) \qquad (3)$$
$$\neg R(x, y) \vee \neg C(y) \vee D(x) \qquad (4)$$
$$A(a) \qquad (5)$$
$$\neg D(a) \qquad (6)$$

With this selection, we don't need to resolve (1) and (4)

Observation: This strategy amounts to Unit Resolution

One of the premises of resolution must be a unit clause !!

Resolution with Free Selection

$$\neg A(x) \vee R(x, f(x)) \quad (1)$$

$$\neg A(x) \vee B(f(x)) \quad (2)$$

$$\neg B(x) \vee C(x) \quad (3)$$

$$\neg R(x, y) \vee \neg C(y) \vee D(x) \quad (4)$$

$$A(a) \quad (5)$$

$$\neg D(a) \quad (6)$$

$$R(a, f(a)) \quad (7)$$

$$B(f(a)) \quad (8)$$

$$C(f(a)) \quad (9)$$

$$\neg C(f(a)) \vee D(a) \quad (10)$$

$$D(a) \quad (11)$$

$$\square \quad (12)$$

Resolution with Free Selection

We still have termination problems . . .

$$\begin{aligned} A(x) \to \exists y.R(x,y) \land A(y) \quad &\rightsquigarrow \quad \neg A(x) \lor R(x, f(x)) \\ &\qquad\quad \neg A(x) \lor A(f(x)) \\ A(a) \quad &\rightsquigarrow \quad A(a) \end{aligned}$$

Resolution with Free Selection

$$\neg A(x) \lor R(x, f(x))$$
$$\neg A(x) \lor A(f(x))$$
$$A(a)$$
$$R(a, f(a))$$
$$A(f(a))$$
$$R(a, f(f(a)))$$
$$A(f(f(a)))$$
$$\dots$$

Theorem

Unsatisfiability and fact entailment over existential rules are undecidable (semi-decidable).

That is, as difficult as checking unsatisfiability in FOL.

Datalog

To achieve decidability we need to sacrifice expressivity.

Datalog: the quintessential rule-based KR language

$$\forall \vec{x}.\forall \vec{z}.(\varphi(\vec{x},\vec{z}) \rightarrow \psi(\vec{x})) \qquad \textit{Rule}$$

$$\forall \vec{x}.\forall \vec{z}.(\varphi(\vec{x},\vec{z}) \rightarrow \bot) \qquad \bot\text{–}\textit{Rule}$$

$$P(\vec{a}) \qquad \textit{Fact}$$

$\varphi(\vec{x},\vec{z})$ and $\psi(\vec{x})$: conjunctions of function-free atoms

We can still express

$$\forall x.(\forall y.(\forall z.(\mathit{fatherOf}(x,y) \wedge \mathit{brotherOf}(x,z) \rightarrow \mathit{uncleOf}(z,y))))$$

$$\forall x.(\forall y.(\mathit{Affects}(x,y) \rightarrow \mathit{Person}(y)))$$

But, we can no longer express

$$\forall x.(\mathit{JuvDisease}(x) \rightarrow \exists y.(\mathit{Affects}(x,y) \wedge \mathit{Child}(y))))$$

Decidability of Entailment

Theorem

Fact entailment in Datalog is decidable.

Decidability follows directly from Herbrand's theorem

- Our problem reduces to unsatisfiability of $\mathcal{S} = \mathcal{R} \cup \mathcal{F} \cup \{\neg P\}$
- $\mathcal{R} \cup \mathcal{F} \cup \{\neg P\}$ is a set of clauses without function symbols

  so Herbrand universe finite
- Gilmore's FOL unsatisfiability algorithm terminates.

Decidability of Entailment

Our algorithm is an adaptation of Gilmore's when Herbrand Universe is finite

```
procedure DATALOG-GIL(⟨R, F⟩, P)
    Compute Herbrand Universe U
    R′ := ground(R, U)
    return Horn-Prop(⟨R′, F⟩, P)
end procedure
```

Subroutine Horn-Prop solves entailment problem for Horn PL

Complexity Considerations

$$\forall x.(\forall y.(\forall z.(\mathit{fatherOf}(x,y) \wedge \mathit{brotherOf}(x,z) \rightarrow \mathit{uncleOf}(z,y))))$$
$$\mathit{fatherOf}(\mathit{John},\mathit{Mary}), \mathit{brotherOf}(\mathit{John},\mathit{Peter})$$

Herbrand Univ: constants in $\langle \mathcal{R}, \mathcal{F} \rangle$

$$U = \{\mathit{John}, \mathit{Mary}, \mathit{Peter}\}$$

Grounding leads to exponential size set of propositional clauses

$$\mathit{fatherOf}(\mathit{John},\mathit{John}) \wedge \mathit{brotherOf}(\mathit{John},\mathit{John}) \rightarrow \mathit{uncleOf}(\mathit{John},\mathit{John})$$
$$\mathit{fatherOf}(\mathit{John},\mathit{Mary}) \wedge \mathit{brotherOf}(\mathit{John},\mathit{Mary}) \rightarrow \mathit{uncleOf}(\mathit{Mary},\mathit{Mary})$$
$$\mathit{fatherOf}(\mathit{John},\mathit{Peter}) \wedge \mathit{brotherOf}(\mathit{John},\mathit{Peter}) \rightarrow \mathit{uncleOf}(\mathit{Peter},\mathit{Peter})$$

and so on

Size of the grounding grows as $\mathcal{O}(c^v)$, where

- c is the max. number of constants in facts.
- v is the max. number of variables in rules.

Complexity Considerations

Propositional entailment in PL can be decided in polynomial time

Overall process takes exponential time (because of grounding)

Theorem

Fact entailment in Datalog is decidable in ExpTime

In fact, the problem is also ExpTime-hard (beyond this course)

Naive grounding algorithm is worst-case optimal
($\Rightarrow$) The problem is ExpTime-Hard !

Practical Considerations

From a practical point of view, we can do much better:

- Avoid computing the grounding upfront
- Instantiate variables to constants "on the fly"

We develop two resolution-based strategies:

1. Forward-chaining:

   Start from facts and instantiate rules to derive new facts whenever possible until goal is derived

2. Backwards-chaining:

   Start from goal and proceed "backwards" to derive the empty clause

Both strategies can be seen as Resolution with Free Selection.

Forward Chaining (Example)

Start from facts and instantiate rules to derive new facts whenever possible until goal (or $\Box$) is derived

$$\forall x.(JuvArthritis(x) \rightarrow JuvDisease(x)) \tag{13}$$

$$\forall x.(\forall y.(JuvDisease(x) \land Affects(x, y) \rightarrow Child(y))) \tag{14}$$

$$JuvArthritis(JRA) \tag{15}$$

$$Affects(JRA, John) \tag{16}$$

Match existing facts to rule bodies to derive new facts.

From Fact (15) and Rule (13) we obtain the following by unit resolution

$$JuvDisease(JRA) \tag{17}$$

From Facts (17) and (16) and Rule (14), derive goal and stop.

$$Child(John)$$

Forward Chaining and Resolution

$\mathcal{S}_{fw}$: select all negative literals in clauses, and the (unique) positive literal if the clause doesn't have negative literals.

$$\neg \mathit{JuvArthritis}(x) \vee \mathit{JuvDisease}(x)$$
$$\mathit{JuvArthritis}(\mathit{JRA})$$

We obtain the following by resolution:

$$\mathit{JuvDisease}(\mathit{JRA})$$

Forward Chaining and Resolution

$\mathcal{S}_{fw}$: select all negative literals in clauses, and the (unique) positive literal if the clause doesn't have negative literals.

Deriving a new fact by matching other facts to a rule may require several resolution steps (Hyperresolution).

$$\neg JuvDisease(x) \lor \neg Affects(x, y) \lor Child(y)$$
$$Affects(JRA, John)$$
$$JuvDisease(JRA)$$

We obtain the following by resolution:

$$\neg JuvDisease(JRA) \lor Child(John)$$
$$Child(John)$$

In forward chaining, we do both steps in one

Forward Chaining

procedure FORWARD($\langle \mathcal{R}, \mathcal{F} \rangle$, P)
  $\mathcal{F}' := \mathcal{F}$
  repeat
    for each rule $R = \neg B_1 \vee \neg B_2 \vee \ldots, \vee \neg B_n \vee H \in \mathcal{R}$ **do**
      if $\{D_1, \ldots, D_n\} \subseteq \mathcal{F}'$ such that B_i unifies with D_i **then**
        $\theta :=$ Unify($\{B_1 = D_1, \ldots, B_n = D_n\}$)
        $\mathcal{F}' := \mathcal{F}' \cup \{H\theta\}$
      end if
    end for
  until No new atom can be added to $\mathcal{F}'$ or $P \in \mathcal{F}'$ or $\Box \in \mathcal{F}'$
  if $P \in \mathcal{F}'$ or $\Box \in \mathcal{F}'$ **then**
    return true
  else
    return false
  end if
end procedure

Backward Chaining (Example)

Check whether following rules and facts imply $Child(John)$

$$\forall x.(JuvArthritis(x) \rightarrow JuvDisease(x)) \tag{18}$$

$$\forall x.(\forall y.(JuvDisease(x) \land Affects(x, y) \rightarrow Child(y))) \tag{19}$$

$$JuvArthritis(JRA) \tag{20}$$

$$Affects(JRA, John) \tag{21}$$

Match "goal" $Child(John)$ to rule heads and facts to derive new goals

To prove $Child(John)$, by Rule (19) it is sufficient to show

$$JuvDisease(x) \quad \text{and} \quad Affects(x, John)$$

Then, by Fact (21), it would be sufficient to show

$$JuvDisease(JRA)$$

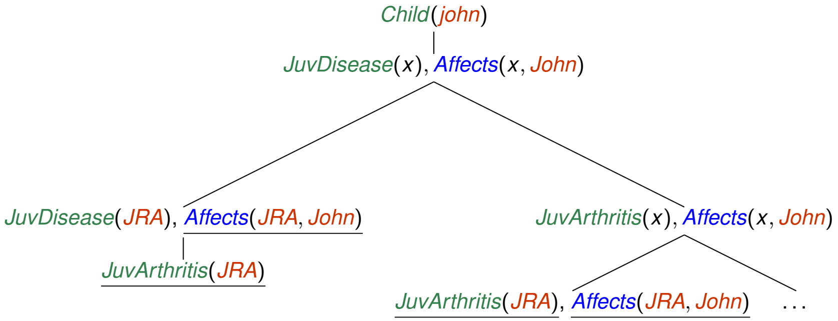
Another possibility: use Rule (18) and get the following sub-goals

$$JuvArthritis(x) \quad \text{and} \quad Affects(x, John)$$

And so on. . . ,

Backward Chaining (Example)

We can represent this kind of backwards reasoning in an AND-OR tree



$$\forall x.(JuvArthritis(x) \rightarrow JuvDisease(x))$$

$$\forall x.(\forall y.(JuvDisease(x) \land Affects(x, y) \rightarrow Child(y)))$$

$$JuvArthritis(JRA)$$

$$Affects(JRA, John)$$

Backwards Chaining and Resolution

$\mathcal{S}_{bw}$: select the unique positive literal in clauses, and
all negative literals if the clause doesn't have positive literals.

Matching the goal to a rule head or a fact corresponds to one resolution step.

$$\frac{\neg JuvDisease(x) \vee \neg Affects(x, y) \vee Child(y) \quad \neg Child(John)}{\neg JuvDisease(x) \vee \neg Affects(x, John)}$$

Termination Issues

Resolution with free selection may not terminate with $\mathcal{S}_{bw}$

Example: show that john is a Scientist.

$$\neg worksWith(x, y) \lor \neg Scientist(y) \lor Scientist(x) \tag{22}$$

$$worksWith(john, mary) \tag{23}$$

$$\neg Scientist(john) \tag{24}$$

We start resolving on selected atoms:

$$\neg worksWith(john, y) \lor \neg Scientist(y)$$

$$\neg worksWith(john, y_1) \lor \neg worksWith(y_1, y_2) \lor \neg Scientist(y_2)$$

$$\ldots$$

Keep on generating clauses with chains of *worksWith* atoms of increasing length (variable proliferation).

Thus, the backwards-chaining tree can have infinite branches.

Other Considerations

Implementing Forward and Backwards chaining efficiently is non-trivial:

- Forward-chaining: set of deduced facts might get huge
- Backwards-chaining: recursion may be too deep or search tree too wide.

There are many ways to optimise these algorithms

Semi-naive evaluation, Magic sets, ...

But, this is beyond the scope of this course.

Many optimised systems that implement forward/backwards chaining

The KR languages we have described are related to:

- Databases: Datalog query language, and deductive databases
- Logic programming: Prolog