Constraint Solving
Outline

• Datalog
• Boolean Satisfiability
• Network Policy Generation (Adam)
Datalog

• A query language for (deductive) databases
  ‣ Given a DB and Datalog rules, can infer other facts

• Datalog query evaluation is based on first-order logic
  ‣ Thus is sound and complete

• Is a restricted form of Prolog
  ‣ Disallows complex terms in predicates (no functions of arity > 0)
  ‣ Limits assignments that are possible under recursion and negation (stratification)
  ‣ Only allows range-restricted variables (variables in consequent must appear in antecedent, non-negated)

• Result: Datalog terminates (all possible proofs are finite), unlike Prolog
Datalog Programs

- \text{In\_role}(alice, \text{accountant})
- \text{Is\_senior}(\text{accountant}, \text{clerk})
- \text{Is\_senior}(\text{clerk}, \text{employee})
- \text{In\_role}(X, R1) \iff \text{In\_role}(X, R2), \text{Is\_senior}(R2, R1)
Datalog Programs

• \text{In\_role}(alice, accountant)
• \text{Is\_senior}(accountant, clerk)
• \text{Is\_senior}(clerk, employee)
• \text{In\_role}(X, R1) \leftarrow \text{In\_role}(X, R2), \text{Is\_senior}(R2, R1)

• FOL Concepts:
  ‣ Alphabet of \textit{variables}, \textit{function symbols}, and \textit{predicate symbols}
  ‣ Functions and predicates have \textit{arity} (0 or more args)
  ‣ A function symbol of arity 0 is a \textit{constant}
• `in_role(alice, accountant)`
• `is_senior(accountant, clerk)`
• `is_senior(clerk, employee)`
• `in_role(X, R1) ← in_role(X, R2), is_senior(R2, R1)`

• Predicate symbols: `in_role`, `is_senior`
• Constant symbols: `alice`, `accountant`, `clerk`, `employee`
• Variables: `??`
Datalog Programs

- `ln_role(alice, accountant)`
- `is_senior(accountant, clerk)`
- `is_senior(clerk, employee)`
- `ln_role(X, R1) ← ln_role(X, R2), is_senior(R2, R1)`

- **FOL Concepts:**
  - *Atomic formula* (atom) is `p(t_1, ..., t_n)`, where `p` is a predicate and `t_i` is a term (constant, variable, or function in general)
  - *Formulae* are formed using atoms, conjunction, disjunction, negation, implication, and logical equivalence, including quantifiers
Datalog Programs

- \text{in\_role}(alice, accountant)
- \text{Is\_senior}(accountant, clerk)
- \text{Is\_senior}(clerk, employee)
- \text{in\_role}(X, R1) \leftarrow \text{in\_role}(X, R2), \text{Is\_senior}(R2, R1)

- FOL Concepts:
  - \textit{Literal} is an atom or the negation of an atom
  - A \textit{clause} is a disjunction of literals
Horn Clauses

- Datalog uses *Horn clauses*
  - A clause with at most one positive literal
    - Write one out
  - What is the equivalent formulation using implication?
- The result is a Prolog rule
  - Although remember that Datalog limits the possible rules
  - A Horn clause is a Datalog clause if it does not have function symbols with arity > 0
Datalog Analysis for Security

- Encode security state as facts (literals)
- Logical implications relationships in the security state as rules (Horn clauses)
- Queries may be issued to determine whether certain properties hold
  - E.g., Is Alice capable of performing actions authorized to clerks and employees?
  - Why might you care whether this is true?
Least Herbrand Model

- Property of Datalog for processing queries
- If query is a negation of a goal clause, query evaluation can be performed efficiently

- Definitions
  - The set \( U_A \) of all ground terms constructed over alphabet \( A \) is a *Herbrand universe*
  - The set of all ground atomic formulae is a *Herbrand base*
  - A *Herbrand interpretation* \( I \) of program \( P \) is a subset of the Herbrand base of \( P \)
Least Herbrand Model

- Property of Datalog for processing queries
- If query is a negation of a goal clause, query evaluation can be performed efficiently

Definitions

- A ground rule is satisfied by a Herbrand interpretation \( I \) if either \( a_0 \) in \( I \) or at least one of \( a_1, \ldots, a_n \) is not in \( I \)
  - That is, either \( a_0 \) is true and all \( a_i \) are true, or some \( a_i \) is not true and \( a_0 \) is not true

- An \( I \) is a **Herbrand model** of program \( P \) if each clause in \( P \) is satisfied by \( I \)
Unique Least Herbrand Model

• Each program $P$ must have at least one model describing what is true in that model

• Each program $P$ must have a unique least Herbrand model

• Problem: compute the least Herbrand model for a program
  ▸ Why?
Computing in Datalog

• Immediate Consequence Operator

\[ T_P(I) = \{ A | (A \leftarrow B_1, \ldots, B_n) \in Gnd(P) \land B_i \in I \} \]

• Since Herbrand universe and Herbrand base are finite
  ‣ Can compute as a fixed point where termination is guaranteed

\[ T_P(\emptyset) \subseteq T_P(T_P(\emptyset)) \subseteq T_P(T_P(T_P(\emptyset))) \subseteq \ldots \]

• In linear time in size of program P
Computing in Datalog

- Query: Is atom \( a \) true in \( P \)?
- Compute least Herbrand model of \( P \) and see if \( a \) is there
  - I believe this is called hyperresolution
  - Not goal-directed
- Instead: Query negation of a goal clause
  - Query: there exists \( X \), s.t. \((\text{In}_\text{role}(X, \text{accountant}) \land \text{In}_\text{role}(X, \text{clerk}))\)?
  - Verify using the negative of the query
    - Find if \((P \cup \text{not } Q)\) does not have a model
- SLD resolution – may not terminate
- SLG resolution is guaranteed to terminate
Boolean Satisfiability

- Malik and Zhang paper
Summary

• Datalog
  ‣ Efficient method for reasoning about the state of a system

• Boolean Satisfaction
  ‣ Practical methods exist for solving these problems
Questions