Logic: TD as search, Datalog (variables)

Computer Science cpsc322, Lecture 23

(Textbook Chpt 5.2 & some basic concepts from Chpt 12)

Nov, 1, 2013
Lecture Overview

• Recap Top Down
• TopDown Proofs as search
• Datalog
Top-down Ground Proof Procedure

Key Idea: search backward from a query $G$ to determine if it can be derived from $KB$.

Bottom Up

$KB \xrightarrow{Bu} \{C\}$

$G$ is proved if $G \subseteq C$

Looks at the query only at the end

Top Down

Query $G$

$KB$

Answer
Notation: An answer clause is of the form:

\[ \text{yes} \leftarrow a_1 \land a_2 \land \ldots \land a_m \]

Express query as an answer clause
(e.g., query \[ a_1 \land a_2 \land \ldots \land a_m \])

\[ \text{yes} \leftarrow \text{query} \land b_1 \land b_2 \land \ldots \land b_p \land a_{i+1} \land \ldots \land a_m \]

Rule of inference (called SLD Resolution)
Given an answer clause of the form:

\[ \text{yes} \leftarrow a_1 \land a_2 \land \ldots \land a_m \]

and the clause:

\[ a_i \leftarrow b_1 \land b_2 \land \ldots \land b_p \]

You can generate the answer clause

\[ \text{yes} \leftarrow a_1 \land \ldots \land a_{i-1} \land b_1 \land b_2 \land \ldots \land b_p \land a_{i+1} \land \ldots \land a_m \]
• **Successful Derivation:** When by applying the inference rule you obtain the answer clause $\text{yes} \leftarrow$.

Query: $a$ (two ways)

\[
\begin{align*}
\text{yes} & \leftarrow a. \\
& \quad \downarrow \\
& \quad \text{Fail} \\
& \quad \text{Fail} \\
\end{align*}
\]
Lecture Overview

• Recap Top Down
• TopDown Proofs as search
• Datalog
Systematic Search in different R&R systems

Constraint Satisfaction (Problems):
- **State**: assignments of values to a subset of the variables
- **Successor function**: assign values to a “free” variable
- **Goal test**: set of constraints
- **Solution**: possible world that satisfies the constraints
- **Heuristic function**: none (all solutions at the same distance from start)

Planning (forward):
- **State**: possible world
- **Successor function**: states resulting from valid actions
- **Goal test**: assignment to subset of vars
- **Solution**: sequence of actions
- **Heuristic function**: empty-delete-list (solve simplified problem)

Logical Inference (top Down):
- **State**: answer clause
- **Successor function**: states resulting from substituting one atom with all the clauses of which it is the head
- **Goal test**: empty answer clause
- **Solution**: start state
- **Heuristic function**: see next slide
Search Graph

KB

\[
\begin{align*}
    a &\leftarrow b \land c. \\
    a &\leftarrow h. \\
    b &\leftarrow k. \\
    d &\leftarrow p. \\
    f &\leftarrow p. \\
    g &\leftarrow f. \\
    h &\leftarrow m. \\
    b &\leftarrow j. \\
    d &\leftarrow m. \\
    f &\leftarrow m. \\
    g &\leftarrow m. \\
    k &\leftarrow m. \\
    p &. 
\end{align*}
\]

Prove: \( \text{?} \leftarrow a \land d. \)

Heuristics?
Prove: ?← a ∧ d.

Possible Heuristic?
Number of atoms in the answer clause
Admissible?

A. Yes  B. No  C. It Depends
Search Graph

Prove: \( a \land d \leftarrow \) ?

KB

\[
\begin{align*}
a & \leftarrow b \land c. \\
a & \leftarrow g. \\
a & \leftarrow h. \\
b & \leftarrow j. \\
b & \leftarrow k. \\
d & \leftarrow m. \\
d & \leftarrow p. \\
f & \leftarrow m. \\
g & \leftarrow f. \\
g & \leftarrow m. \\
h & \leftarrow m. \\
p & \leftarrow \end{align*}
\]

Heuristics?

Admissible

\# of atoms in answer clause

because you need at least that number of resolution steps to obtain \( yes \)

i.e. the goal state

Heuristics?

& 4 atoms in answer clause

because you need at least that number of resolution steps to obtain \( yes \)

i.e. the goal state
If the body of an answer clause contains a symbol that does not match the head of any clause in the KB what should the most informative heuristic value for that answer clause be?

A. Zero
B. Infinity
C. Twice the number of clauses in the KB
D. None of the above
Lecture Overview

• Recap Top Down
• TopDown Proofs as search
• Datalog
Representation and Reasoning in Complex domains

- In complex domains expressing knowledge with propositions can be quite limiting.

There is no notion that the system can reason about:

- $up(s_2)$
- $up(s_3)$
- $ok(cb_1)$
- $ok(cb_2)$
- $live(w_1)$
- $connected(w_1, w_2)$

- It is often natural to consider individuals and their properties.

$up(s_2)$ $up(s_3)$ $ok(cb_1)$ $ok(cb_2)$ $live(w_1)$ $connected(w_1, w_2)$
What do we gain....

By breaking propositions into relations applied to individuals?

• Express **knowledge** that **holds for set of individuals** (by introducing **variables**)

\[
\text{live}(W) \leftarrow \text{connected_to}(W, W1) \land \text{live}(W1) \land \\
\text{wire}(W) \land \text{wire}(W1).
\]

• We can **ask generic queries** (i.e., containing **variables**)

\[
? \text{connected_to}(W, w_1)
\]
Datalog vs PDCL (better with colors)

First Order Logic

\[ \forall X \exists Y p(X, Y) \iff \neg q(Y) \]

- \( p(a_1, a_2) \)
- \( \neg q(a_3) \)

Propositional Logic

\[ 7(p \lor q) \rightarrow (r \lor s \lor t) \]

- \( p \)
- \( r \)

Datalog

\[ p(X) \leftarrow q(X) \land r(X, Y) \]

- \( r(X, Y) \leftarrow s(Y) \)
- \( s(a_1), q(a_2) \)

PDCL

\[ p \leftarrow s \lor t \]

- \( r \leftarrow s \lor q \lor p \)
- \( r \leftarrow p \)
Datalog: a relational rule language

Datalog expands the syntax of PDCL….

<table>
<thead>
<tr>
<th><strong>Variable</strong></th>
<th>is a symbol starting with an upper case letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples:</td>
<td>X, Y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Constant</strong></th>
<th>is a symbol starting with lower-case letter or a sequence of digits.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples:</td>
<td>alan, w1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Term</strong></th>
<th>is either a variable or a constant.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples:</td>
<td>X, Y, alan, w1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Predicate symbol</strong></th>
<th>is a symbol starting with a lower-case letter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples:</td>
<td>live, connected, part-of, in</td>
</tr>
</tbody>
</table>
Datalog Syntax (cont’d)

An **atom** is a symbol of the form $p$ or $p(t_1 \ldots t_n)$ where $p$ is a predicate symbol and $t_i$ are terms

Examples: sunny, in(alan,X)

A **definite clause** is either an atom (a fact) or of the form:

$$h \leftarrow b_1 \land \ldots \land b_m$$

where $h$ and the $b_i$ are atoms (Read this as `h if b`.)

Example: in(X,Z) $\leftarrow$ in(X,Y) $\land$ part-of(Y,Z)

A **knowledge base** is a set of definite clauses
Datalog: Top Down Proof Procedure

• Extension of Top-Down procedure for PDCL. How do we deal with variables?
  • Idea:
    - Find a clause with head that matches the query
    - Substitute variables in the clause with their matching constants
  • Example:

  **Query:** yes ← in(alan, cs_building).

  in(alan, r123).
  part_of(r123,cs_building).
  in(X,Y) ← part_of(Z,Y) ∧ in(X,Z).

  with Y = cs_building
  X = alan
Example proof of a Datalog query

```
in(alan, r123).
part_of(r123,cs_building).
in(X,Y) ← part_of(Z,Y) ∧ in(X,Z).
```

**Query:** yes ← in(alan, cs_building).

```
yes ← part_of(Z,cs_building) ∧ in(alan, Z).
```

A. yes ← part_of(Z, r123) ∧ in(alan, Z).
B. yes ← in(alan, r123).
C. yes ←.
D. None of the above
Example proof of a Datalog query

\[
in(\text{alan}, \text{r123}). \\
\text{part_of}(\text{r123}, \text{cs\_building}). \\
in(X,Y) \leftarrow \text{part_of}(Z,Y) \land in(X,Z).
\]

Query: \( \text{yes} \leftarrow in(\text{alan}, \text{cs\_building}). \)

\[
\text{yes} \leftarrow \text{part_of}(Z, \text{cs\_building}) \land in(\text{alan}, Z).
\]

Using clause: \( \text{in}(X,Y) \leftarrow \text{part_of}(Z,Y) \land in(X,Z) \), with \( Y = \text{cs\_building} \) and \( X = \text{alan} \)

\[
\text{yes} \leftarrow in(\text{alan}, \text{r123}).
\]

Using clause: \( \text{part_of}(\text{r123}, \text{cs\_building}) \), with \( Z = \text{r123} \)

\[
\text{yes} \leftarrow \text{part_of}(Z, \text{r123}), in(\text{alan}, Z).
\]

Using clause: \( \text{in}(X,Y) \leftarrow \text{part_of}(Z,Y) \land in(X,Z) \), with \( X = \text{alan} \) and \( Y = \text{r123} \)

\[\text{fail}\]

No clause with matching head: \( \text{part_of}(Z, \text{r123}) \).
Tracing Datalog proofs in Alspace

- You can trace the example from the last slide in the Alspace Deduction Applet at http://aispace.org/deduction/ using file ex-Datalog available in course schedule

- Question 4 of assignment 3 asks you to use this applet
Datalog: queries with variables

in(alan, r123).
part_of(r123, cs_building).
in(X,Y) ← part_of(Z,Y) & in(X,Z).

Query:
in(alan, X1).
   yes(X1) ← in(alan, X1).

What would the answer(s) be?
Datalog: queries with variables

in(alan, r123).
part_of(r123, cs_building).
in(X, Y) ← part_of(Z, Y) & in(X, Z).

Query: in(alan, X1).
yes(X1) ← in(alan, X1).

What would the answer(s) be?

yes(r123).
yes(cs_building).

Again, you can trace the SLD derivation for this query in the AIspace Deduction Applet.
Logics in AI: Similar slide to the one for planning

Propositional Definite Clause Logics

Propositional Logics

First-Order Logics

Semantics and Proof Theory

Satisfiability Testing (SAT)

Description Logics

Ontologies

Production Systems

Cognitive Architectures

Hardware Verification

Semantic Web

Video Games

Summarization

Tutoring Systems

Information Extraction

you will know a little

Some Applications

CPSC 322, Lecture 8
Big Picture: R&R systems

Environment

Deterministic
- Arc Consistency
- Search

Stochastic
- SLS

Static

Constraint Satisfaction
- Vars + Constraints
- Search
- SLS

Query

Logics
- Search
- CSP

Sequential

STRIPS
- CSP
- For planning

Representation

Reasoning Technique

Decision Nets
- Var. Elimination
- Value Iteration

Belief Nets
- Var. Elimination
- For inference

Markov Processes

CPSC 322, Lecture 2

Slide 25
Midterm review

Average 77 😊
Best 103!
32 students > 90%
6 students <50%

How to learn more from midterm
• Carefully examine your mistakes (and our feedback)
• If you still do not see the correct answer/solution go back to your notes, the slides and the textbook
• If you are still confused come to office hours with specific questions
Full Propositional Logics (not for 322)

DEFs.

Literal: an atom or a negation of an atom

Clause: is a disjunction of literals

Conjunctive Normal Form (CNF): a conjunction of clauses

INFERENECE:

• Convert all formulas in KB and \( \neg \alpha \) in CNF

• Apply **Resolution Procedure** (at each step combine two clauses containing complementary literals into a new one)

• Termination
  • No new clause can be added
  • Two clause resolve into an empty clause
Propositional Logics: Satisfiability (SAT problem)

Does a set of formulas have a model? Is there an interpretation in which all the formulas are true?

(Stochastic) Local Search Algorithms can be used for this task!

Evaluation Function: number of unsatisfied clauses

WalkSat: One of the simplest and most effective algorithms:
Start from a randomly generated interpretation

- Pick an unsatisfied clause
- Pick an proposition to flip (randomly 1 or 2)
  1. To minimize # of unsatisfied clauses
  2. Randomly
Full First-Order Logics (FOLs)

We have **constant symbols, predicate symbols and function symbols**

So interpretations are much more complex (but the same basic idea – one possible configuration of the world)

- constant symbols => individuals, entities
- predicate symbols => relations
- function symbols => functions

**INFERENCExE:**

- **Semidecidable:** algorithms exists that says yes for every entailed formulas, but no algorithm exists that also says no for every non-entailed sentence
- **Resolution Procedure** can be generalized to FOL