Logic: Domain Modeling /Proofs +
Top-Down Proofs

Computer Science cpsc322, Lecture 22

(Textbook Chpt 5.2)

Oct, 30, 2013
Department of Computer Science
Undergraduate Events

More details @ https://www.cs.ubc.ca/students/undergrad/life/upcoming-events

Mastering LinkedIn Workshop
Date:  Mon., Oct 28  
Time:  5:00 pm  
Location:  Wesbrook 100

Resume Drop-in Editing
Date:  Tues., Oct 29  
Time:  12:30 – 3:30 pm  
Location:  ICCS 253

Graduate Recruitment Panel
Date:  Wed., Oct 30  
Time:  12:30 – 1:30 pm  
Location:  X836, ICICS/CS

CSSS Meet the Profs Luncheon
Date:  Thurs., Oct 31  
Time:  12:30 – 2 pm  
Location:  X836, ICICS/CS

E-Portfolio Info Session & Talk by Eric Diep, Co-Founder, A Thinking Ape  
Date:  Tues., Nov 5  
Time:  5:15 – 6:45 pm  
Location:  DMP 110

Programming Interview Resources Drop-in Clinic
Date:  Wed., Nov 6  
Time:  12:30 – 2 pm  
Location:  ICCS 202

Speed Mentoring Dinner
Date:  Wed., Nov 6  
Time:  5:45 – 7:15 pm  
Location:  ICCS X860
Lecture Overview

• Recap
• Using Logic to Model a Domain (Electrical System)
• Reasoning/Proofs (in the Electrical Domain)
• Top-Down Proof Procedure
Soundness & completeness of proof procedures

- A proof procedure $X$ is sound ...

$$\frac{KB \vdash_x G}{KB \models G}$$

- A proof procedure $X$ is complete ....

$$\frac{KB \models G}{KB \vdash_x G}$$

- BottomUp for PDCL is sound & complete

- We proved this in general even for domains represented by thousands of propositions and corresponding KB with millions of definite clauses!
Lecture Overview

- Recap
- Using PDCL Logic to Model a Domain (Electrical System)
- Reasoning/Proofs (in the Electrical Domain)
- Top-Down Proof Procedure
Electrical Environment

outside power

- circuit breaker
- on/off switch
- two-way switch
- light
- power outlet
- wires

CPSC 322, Lecture 21
Let’s define relevant propositions

• For each wire $w$
• For each circuit breaker $cb$
• For each switch $s$
• For each light $l$
• For each outlet $p$

How many interpretations?

- $7$
- $2$
- $3 \times 2$
- $2$

$19$

$\sim 5 \times 10^5$
Let’s now tell system knowledge about how the domain works

\[
\begin{align*}
\text{live}_L_L_1 & \leftarrow \text{live}_w_w_0 \\
\text{live}_w_w_0 & \leftarrow \text{up}_s_s_2 \land \text{live}_w_w_1 \\
\text{live}_w_w_0 & \leftarrow \text{down}_s_s_2 \land \text{live}_w_w_2 \\
\text{live}_w_w_1 & \leftarrow \text{up}_s_s_1 \land \text{live}_w_w_3
\end{align*}
\]
More on how the domain works….

\[
\begin{align*}
\text{live}_w_2 & \leftarrow \text{live}_w_3 \land \text{down}_s_1. \\
\text{live}_l_2 & \leftarrow \text{live}_w_4. \\
\text{live}_w_4 & \leftarrow \text{live}_w_3 \land \text{up}_s_3. \\
\text{live}_p_1 & \leftarrow \text{live}_w_3.
\end{align*}
\]
More on how the domain works….

\[
\begin{align*}
\text{live}_w_3 & \leftarrow \text{live}_w_5 \land \text{ok}_cb_1. \\
\text{live}_p_2 & \leftarrow \text{live}_w_6. \\
\text{live}_w_6 & \leftarrow \text{live}_w_5 \land \text{ok}_cb_2. \\
\text{live}_w_5 & \leftarrow \text{live}_\text{outside}.
\end{align*}
\]
What else we may know about this domain?

- That some simple propositions are true

\[ \text{live\_outside}. \]
What else we may know about this domain?

- That some additional simple propositions are true

\[ \text{down}_s_1. \quad \text{up}_s_2. \quad \text{up}_s_3. \quad \text{ok}_cb_1. \quad \text{ok}_cb_2. \quad \text{live}_outside. \]
All our knowledge…..

\[
\begin{align*}
\text{down\_s}_1. \\
\text{up\_s}_2. \\
\text{up\_s}_3. \\
\text{ok\_cb}_1. \\
\text{ok\_cb}_2. \\
\text{live\_outside} & \quad \leftarrow \text{live\_l}_1 \leftarrow \text{live\_w}_0 \\
\text{live\_w}_0 & \leftarrow \text{live\_w}_1 \land \text{up\_s}_2. \\
\text{live\_w}_0 & \leftarrow \text{live\_w}_2 \land \text{down\_s}_2. \\
\text{live\_w}_1 & \leftarrow \text{live\_w}_3 \land \text{up\_s}_1. \\
\text{live\_w}_2 & \leftarrow \text{live\_w}_3 \land \text{down\_s}_1. \\
\text{live\_l}_2 & \leftarrow \text{live\_w}_4. \\
\text{live\_w}_4 & \leftarrow \text{live\_w}_3 \land \text{up\_s}_3. \\
\text{live\_p}_1 & \leftarrow \text{live\_w}_3. \\
\text{live\_w}_3 & \leftarrow \text{live\_w}_5 \land \text{ok\_cb}_1. \\
\text{live\_p}_2 & \leftarrow \text{live\_w}_6. \\
\text{live\_w}_6 & \leftarrow \text{live\_w}_5 \land \text{ok\_cb}_2. \\
\text{live\_w}_5 & \leftarrow \text{live\_outside}. 
\end{align*}
\]
Lecture Overview

• Recap

• Using Logic to Model a Domain (Electrical System)

• Reasoning/Proofs (in the Electrical Domain)

• Top-Down Proof Procedure
What Semantics is telling us

• Our KB (all we know about this domain) is going to be true only in a subset of all possible \(2^{19}\) interpretations.

• What is logically entailed by our KB are all the propositions that are true in all those models.

• This is what we should be able to derive given a sound and complete proof procedure.
If we apply the bottom-up (BU) proof procedure down_s_1, up_s_2, up_s_3, ok_cb_1, ok_cb_2, live_outside

BU generates C all the atoms added to care in green

down_s_1 = live_l_1 \lor live_l_2
up_s_2 = live_w_0
up_s_3 = live_w_0 \land up_s_2
ok_cb_1 = live_w_0 \land down_s_2
ok_cb_2 = live_w_0 \land down_s_1
live_l_2 = live_w_4
live_w_4 = live_w_3 \land up_s_3
live_p_1 = live_w_3
live_w_3 = live_w_5 \land ok_cb_1
live_p_2 = live_w_6
live_w_6 = live_w_5 \land ok_cb_2
live_w_5 = live_outside

live_l_1 \not< C \Rightarrow KB \vdash_{BU} live_l_1 \not< \Rightarrow KB \vdash live_l_2

which is not the case for live_l_1
Lecture Overview

• Recap
• Using Logic to Model a Domain (Electrical System)
• Reasoning/Proofs (in the Electrical Domain)
• Top-Down Proof Procedure
Bottom-up vs. Top-down

Bottom-up

KB → C

G is proved if $G \subseteq C$

When does BU look at the query $G$?

A. In every loop iteration  B. Never
C. Only at the end  D. Only at the beginning
Bottom-up vs. Top-down

- **Key Idea of top-down**: search backward from a query $G$ to determine if it can be derived from $KB$.

**Bottom-up**

- $KB$ → $C$
- $G$ is proved if $G \subseteq C$

**Top-down**

- Query $G$
- $KB$ → Answer

When does BU look at the query $G$?
- At the end

TD performs a backward search starting at $G$
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 $G \subseteq C$

look at the query only at the end

Top Down

query $G$

answer

KB

CPSC 322, Lecture 22
Top-down Proof Procedure: Basic elements

**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 \( \text{query } a_1 \land a_2 \land \ldots \land a_m \))

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

**Rule of inference** (called SLD Resolution)

Given an **answer clause** of the form:

\[ \varnothing \]

and the clause:

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

You can generate the answer clause

\[ \text{yes} \leftarrow a_1 \land a_2 \land \ldots \land a_{i-1} \land b_1 \land b_2 \land \ldots \land b_p \land a_{i+1} \land \ldots \land a_m \]
Rule of inference: Examples

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 KB clause:

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

You can generate the answer clause

\[ \text{yes} \leftarrow a_1 \land \ldots \land a_{i-1} \land b_1 \land b_2 \land \ldots \land b_p \land a_{i+1} \land \ldots \land a_m \]

\[ \text{yes} \leftarrow b \land c. \]

\[ b \leftarrow k \land f. \]

\[ \Rightarrow \text{yes} \leftarrow k \land f \land c \]

\[ \text{yes} \leftarrow e \land f. \]

\[ e. \leftarrow \Rightarrow \text{yes} \leftarrow f \]
(successful) Derivations

• An **answer** is an answer clause with $m = 0$. That is, it is the answer clause $yes \leftarrow$.

• A (successful) **derivation** of query $\text{"?} q_1 \land \ldots \land q_k \text{"}$ from $KB$ is a sequence of answer clauses $\gamma_0, \gamma_1, \ldots, \gamma_n$ such that
  • $\gamma_0$ is the answer clause $yes \leftarrow q_1 \land \ldots \land q_k$
  • $\gamma_i$ is obtained by **resolving** $\gamma_{i-1}$ with a clause in $KB$, and
  • $\gamma_n$ is an answer. $yes \leftarrow$.

• An **unsuccessful derivation**.....

$yes \leftarrow a \land b$
Example: derivations

- $a \leftarrow e \land f$
- $c \leftarrow e$
- $f \leftarrow j \land e$
- $a \leftarrow b \land c$
- $d \leftarrow k$
- $f \leftarrow c$
- $b \leftarrow k \land f$
- $e$
- $j \leftarrow c$

Query: $a$ (two ways)

1. yes $\leftarrow a$
2. yes $\leftarrow a$

- $u \leftarrow b \land c$
- $u \leftarrow k \land f \land c$

$K$ cannot be eliminated so will Fail.

- $u \leftarrow e \land f$
- $u \leftarrow f$
- $u \leftarrow c$
- $u \leftarrow e$
- yes $\leftarrow ...$
Example: derivations

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

Query: \( b \land e \)

A. Provable by TD  
B. It depends  
C. Not Provable by TD
Standard Search vs. Specific 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:
- **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
- **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**: ….. *(next time)*
Learning Goals for today’s class

You can:

• Model a relatively simple domain with propositional definite clause logic (PDCL)

• Trace query derivation using SLD resolution rule of inference