Lecture 3
More Explicit Search and Version Spaces

Armando Solar-Lezama and Xiaokang Qiu
Last time

Explicit search
  • With pruning for observational equivalence
Alternative interpretation of the algorithm

The algorithm from last time can be understood in terms of partial orders
Binary search

Search for smallest element $\leq$
Binary search

Search for smallest element $\leq$
Binary search

Search for smallest element $\leq \bullet$
Binary search

Search for smallest element $\leq$
Binary search

Search for smallest element $\leq \circ$
Binary search

Search for smallest element $\leq \bullet$
Binary search

Search for smallest element $\leq \bullet$
Binary search

Search for smallest element $\leq \bullet$
Partial order for search

If $p_1 \preceq p$ then $p_1$ is always preferable.
Partial order for grammar

\[ p_1 \sqsubseteq p_2 \text{ iff } p_1 \text{ is a subprogram of } p_2 \]

Eliminate by observational equivalence

If \( p_i \) is bad, we can eliminate all \( P = \{ p \mid p_i \sqsubseteq p \} \)

- Bad == it’s equivalent to something else I am already considering
Partial order for grammar

\[ p_1 \sqsubseteq p_2 \text{ iff } p_1 \text{ is a subprogram of } p_2 \]

\begin{align*}
\text{lstExpr} &:= \text{sort(lstExpr)} \\
&\quad \text{lstExpr}[\text{intExpr},\text{intExpr}] \\
&\quad \text{lstExpr} + \text{lstExpr} \\
&\quad \text{recursive(lstExpr)} \\
&\quad [0] \\
&\quad \text{in} \\
\text{intExpr} &:= \text{firstZero(lstExpr)} \\
&\quad \text{len(lstExpr)} \\
&\quad 0 \\
&\quad \text{intExpr} + 1
\end{align*}
$p_1 \sqsubseteq p_2$ iff $p_1$ is a subprogram of $p_2$

\[
\text{lstExpr} := \text{sort}(\text{lstExpr}) \\
\text{lstExpr}[\text{intExpr},\text{intExpr}] \\
\text{lstExpr} + \text{lstExpr} \\
\text{recursive}(\text{lstExpr}) \\
[0] \\
in \\
\text{intExpr} := \text{firstZero}(\text{lstExpr}) \\
\text{len}(\text{lstExpr}) \\
0 \\
\text{intExpr} + 1
\]
Partial order for grammar

$p_1 \sqsubseteq p_2$ iff $p_1$ is a subprogram of $p_2$

lstExpr := sort(lstExpr)
lstExpr[intExpr,intExpr]
lstExpr + lstExpr
recursive(lstExpr)
[0]
in
intExpr := firstZero(lstExpr)
len(lstExpr)
0
intExpr + 1
Partial order for grammar

\[ p_1 \sqsubseteq p_2 \text{ iff } p_1 \text{ is a subprogram of } p_2 \]

\[
\begin{align*}
\text{lstExpr} &:= \text{sort}(\text{lstExpr}) \\
& \quad \text{lstExpr}[\text{intExpr}, \text{intExpr}] \\
& \quad \text{lstExpr} + \text{lstExpr} \\
& \quad \text{recursive(\text{lstExpr})} \\
& \quad [0] \\
& \quad \text{in} \\
\text{intExpr} &:= \text{firstZero}(\text{lstExpr}) \\
& \quad \text{len(\text{lstExpr})} \\
& \quad 0 \\
& \quad \text{intExpr} + 1
\end{align*}
\]
Partial order for grammar

\( p_1 \sqsubseteq p_2 \) iff \( p_1 \) is a subprogram of \( p_2 \)

\[
\text{LstExpr} := \text{sort}(\text{LstExpr})
\]
\[
\text{LstExpr}[\text{intExpr},\text{intExpr}]
\]
\[
\text{LstExpr} + \text{LstExpr}
\]
\[
\text{recursive}(\text{LstExpr})
\]
\[
[0]
\]
\[
\text{in}
\]
\[
\text{intExpr} := \text{firstZero}(\text{LstExpr})
\]
\[
\text{len}(\text{LstExpr})
\]
\[
0
\]
\[
\text{intExpr} + 1
\]
Version spaces

Programming by demonstration using version space algebra
Lau, Wolfman, Domingos, Weld, 2001
Version Space Formulation

Hypothesis space $H$
- Space of possible functions $\text{In} \rightarrow \text{Out}$

Version Space $V S_{H,D} \subseteq H$
- $H$ is the original hypothesis space
- $D$ is a set of examples $i_j, o_j$
- $h \in V S_{H,D} \iff \forall i, o \in D \ h(i) = o$

Hypothesis space provides *restriction bias*
- Defines what functions one is allowed to consider
- *Preference bias* needs to be provided independently
Partial Ordering of hypothesis

Partial order $h_1 \sqsubseteq h_2$
  • $h_2$ is “better” than $h_1$

Ex: For boolean hypothesis
  • “better” == more general
  • $h_1 \sqsubseteq h_2 \iff (h_1 \Rightarrow h_2)$

For booleans, VS forms a lattice
Partial Orders

Set P

Partial order \( \leq \) such that \( \forall x, y, z \in P \)

- \( x \leq x \) (reflexive)
- \( x \leq y \) and \( y \leq x \) implies \( x = y \) (asymmetric)
- \( x \leq y \) and \( y \leq z \) implies \( x \leq z \) (transitive)

Can use partial order to define

- Upper and lower bounds
- Least upper bound
- Greatest lower bound
Upper Bounds

If $S \subseteq P$ then

• $x \in P$ is an upper bound of $S$ if $\forall y \in S. \; y \leq x$
• $x \in P$ is the least upper bound of $S$ if
  • $x$ is an upper bound of $S$, and
  • $x \leq y$ for all upper bounds $y$ of $S$
• $\lor$ - join, least upper bound, lub, supremum, sup
  • $\lor S$ is the least upper bound of $S$
  • $x \lor y$ is the least upper bound of $\{x,y\}$
• Often written as $\sqcup$ as well
Lower Bounds

If $S \subseteq P$ then

- $x \in P$ is a lower bound of $S$ if $\forall y \in S. \ x \leq y$
- $x \in P$ is the greatest lower bound of $S$ if
  - $x$ is a lower bound of $S$, and
  - $y \leq x$ for all lower bounds $y$ of $S$

- $\wedge$ - meet, greatest lower bound, glb, infimum, $\inf$
  - $\wedge S$ is the greatest lower bound of $S$
  - $x \wedge y$ is the greatest lower bound of $\{x,y\}$

- Often written as $\cap$ as well
Lattices

If $x \land y$ and $x \lor y$ exist for all $x, y \in P$
then $P$ is a lattice

If $\land S$ and $\lor S$ exist for all $S \subseteq P$
then $P$ is a complete lattice

All finite lattices are complete

Example of a lattice that is not complete

- Integers $\mathbb{I}$
- For any $x, y \in \mathbb{I}$, $x \lor y = \max(x, y)$, $x \land y = \min(x, y)$
- But $\lor \mathbb{I}$ and $\land \mathbb{I}$ do not exist
- $\mathbb{I} \cup \{+\infty, -\infty\}$ is a complete lattice
Partial Ordering of hypothesis

Partial order \( h_1 \sqsubseteq h_2 \)
- \( h_2 \) is “better” than \( h_1 \)

Ex: For boolean hypothesis
- “better” == more general
- \( h_1 \sqsubseteq h_2 \iff (h_1 \Rightarrow h_2) \)

For booleans, VS forms a lattice
- \( h_1, h_2 \in VS \Rightarrow h_1 \sqcap h_2 = h_1 \land h_2 \in VS \)

Most specific hypothesis that satisfies the observations
Boundary set representable

You can represent a VS by the pair (G, S) where
- G is most general hypothesis (i.e. $\top$)
- S is the most specific (i.e. $\bot$)

Applies in general when hypothesis space is partially ordered and version space is a lattice
Update

\[ U(VS, d) = \{ p \in VS \mid p(i) = o \text{ where } d = (i, o) \} \]

• Subset of a version space satisfying a new example \( d \)

**Ex:** For boolean HS

• VS=(G,S)

• If \( d = (i, \text{true}) \)
  \[ U(VS, d) = (G, S \lor \lambda x. \text{if } x = i \text{ then true else false}) \]

• If \( d = (i, \text{false}) \)
  \[ U(VS, d) = (G \land \lambda x. \text{if } x = i \text{ then false else true}, S) \]
Example: FindSuffix

$FS_T$: next occurrence of $T$. 
If your hypothesis space is partially ordered and your VS are boundary set representable, you can represent and search very efficiently.

If they are not?

Break them down into simpler hypothesis spaces!
Union

\[ V_{S_{H_1D}} \cup V_{S_{H_2D}} = V_{S_{H_1 \cup H_2 D}} \]
Join

\[ V_S^{H_1 D_1} \bowtie V_S^{H_2 D_2} = \{ \langle h_1, h_2 \rangle \mid h_1 \in V_S^{H_1 D_1}, h_2 \in V_S^{H_2 D_2}, C(\langle h_1, h_2 \rangle, D) \} \]

• Where \( D_1 = \{d_1^i\}_{i=0..n} \) and \( D_2 = \{d_2^i\}_{i=0..n} \) and \( D = \{(d_1^i, d_2^i)\}_{i=0..n} \)
• \( C(\langle h_1, h_2 \rangle, D) \) means that \( \langle h_1, h_2 \rangle \) is consistent with the input output pairs in \( D \)

What does \( \langle h_1, h_2 \rangle \) mean? What about \( \langle d_1, d_2 \rangle \)?

• Pair
• Composition \( \langle h_1, h_2 \rangle = h_1 \circ h_2 \) and \( \langle d_1, d_2 \rangle = (d_1.\text{in}, d_2.\text{out}) \)

Independent join: \( C \) is unnecessary

• It's a property of \( \langle . . . \rangle \)
• True for pair, not for composition
Transform

\[ VS_2 = transform_{\tau_i, \tau_o}(VS_1) \text{ iff } \forall g \in VS_2 \ \exists f \in VS_1 \text{ s.t. } g = \tau_i \circ f \circ \tau_o \]
SMARTedit version space