Proof-Theoretic Foundations of Indexing in Logic Programming

Iliano Cervesato
iliano@cmu.edu

Carnegie Mellon University

Supported by grant NPRP 4-341-1-059, *Usable automated data inference for end-users*

Qatar National Research Fund
Member of Qatar Foundation

LFMTP'14 Vienna, Austria, July 2014

The Two Worlds of Computational Logic

- **Logical** world
    - Universal language
    - Abstract specifications
    - Simple and natural reasoning
- **Computational** world
    - Return answers fast!
    - Pragmatics
        - Logical status?

Justifying Pragmatics Logically

- Forward/Backward proof search — focusing
- Goal/Clause selection — ordered logic
- Unification — (contextual) reasoning about equality
- WAM-style compilation — currying

Justifying Pragmatics Logically

- Forward/Backward proof search — focusing
- Goal/Clause selection — ordered logic
- Unification — (contextual) reasoning about equality
- WAM-style compilation — currying

What about indexing?

Justifying Pragmatics Logically

- Forward/Backward proof search — focusing
- Goal/Clause selection — ordered logic
- Unification — (contextual) reasoning about equality
- WAM-style compilation — currying

What about indexing?

Cuts context lookup from $O(n)$ to $O(1)$ — exponential savings!

Justifying Pragmatics Logically

- Forward/Backward proof search — focusing
- Goal/Clause selection — ordered logic
- Unification — (contextual) reasoning about equality
- WAM-style compilation — currying

What about indexing?

Cuts context lookup from $O(n)$ to $O(1)$ — exponential savings!

- Backward logic programming: *select relevant clauses*
- Forward logic programming: *identify rules affected by new facts*
- Theorem proving: *retrieve relevant lemmas*

This Work

Provide a logical justification for indexing

...in the context of backward logic programming

Punch line: Polarization + Linearity

Roadmap:

- Indexing over predicate symbols
- Indexing over first-order terms
- Beyond Horn clauses

$\left.\begin{array}{l}\text{Indexing over predicate symbols}\\ \text{Indexing over first-order terms}\end{array}\right\}$ Horn clauses

Backward Proof Search for Horn Clauses

$$\begin{array}{rrcl}
\textit{Atoms:} & a^- & ::= & p^-(\vec{t}) \qquad \text{— } \textit{negative} \\
\textit{Goals:} & G & ::= & a^- \mid \top \mid G_1 \wedge G_2 \\
\textit{Clauses:} & D & ::= & G \supset a^- \mid \forall x.\, D \\
\textit{Programs:} & \Gamma & ::= & \cdot \mid \Gamma, D
\end{array}$$

Case study: the usual `append` program

$$\begin{array}{rr}
\forall l. & \top \supset \mathrm{app}^-(\mathrm{nil}, l, l) \\
\forall x, l_1, l_2, l_3.\, \mathrm{app}^-(l_1, l_2, l_3) & \supset \mathrm{app}^-(\mathrm{c}(x, l_1), l_2, \mathrm{c}(x, l_3)))
\end{array}$$

In Prolog:

```
append([], L, L).
append([X|L1], L2, [X|L3]) :- append(L1, L2, L3).
```

Backward Proof Search for Horn Clauses

$$\frac{}{\Gamma \longrightarrow \boxed{\top}}\ \top_R \qquad \frac{\Gamma \longrightarrow \boxed{G_1} \quad \Gamma \longrightarrow \boxed{G_2}}{\Gamma \longrightarrow \boxed{G_1 \wedge G_2}}\ \wedge_R$$

$$\frac{\Gamma, D, \boxed{D} \longrightarrow a^-}{\Gamma, D \longrightarrow \boxed{a^-}}\ \mathrm{atm}_R$$

$$\left.\right\}\ \text{Inversion}$$

$$\frac{\Gamma \longrightarrow \boxed{G}}{\Gamma, \boxed{G \supset a^-} \longrightarrow a^-}\ \supset_L \qquad \frac{\vdash t \quad \Gamma, \boxed{[t/x]D} \longrightarrow a^-}{\Gamma, \boxed{\forall x.\, D} \longrightarrow a^-}\ \forall_L$$

$$\left.\right\}\ \text{Chaining}$$

Outline

Clause Selection

$$\frac{\Gamma, D, \boxed{D} \longrightarrow p^{-}(\vec{t}\,)}{\Gamma, D \longrightarrow \boxed{p^{-}(\vec{t}\,)}}\ \mathrm{atm_R}$$

Clause Selection

$$\frac{\Gamma, D, \boxed{D} \longrightarrow p^{-}(\vec{t}\,)}{\Gamma, D \longrightarrow \boxed{p^{-}(\vec{t}\,)}}\ \mathsf{atm_R} \qquad \frac{\vdash t \quad \Gamma, \boxed{[t/x]D} \longrightarrow p^{-}(\vec{t}\,)}{\Gamma, \boxed{\forall x.\, D} \longrightarrow p^{-}(\vec{t}\,)}\ \forall_{\mathsf{L}}$$

Clause Selection

$$\frac{\Gamma, D, \boxed{D} \longrightarrow p^{-}(\vec{t})}{\Gamma, D \longrightarrow \boxed{p^{-}(\vec{t})}}\ \mathrm{atm_R} \qquad \frac{\vdash t \quad \Gamma, \boxed{[t/x]D} \longrightarrow p^{-}(\vec{t})}{\Gamma, \boxed{\forall x.\, D} \longrightarrow p^{-}(\vec{t})}\ \forall_\mathrm{L} \qquad \frac{\Gamma \longrightarrow \boxed{G}}{\Gamma, \boxed{G \supset p^{-}(\vec{t})} \longrightarrow p^{-}(\vec{t})}\ \supset_\mathrm{L}$$

Clause D is selected long before a match is established

Clause Selection

$$\frac{\Gamma, D, \boxed{D} \longrightarrow p^{-}(\vec{t}\,)}{\Gamma, D \longrightarrow \boxed{p^{-}(\vec{t}\,)}}\,\mathrm{atm}_R \qquad \frac{\vdash t \quad \Gamma, \boxed{[t/x]D} \longrightarrow p^{-}(\vec{t}\,)}{\Gamma, \boxed{\forall x.\, D} \longrightarrow p^{-}(\vec{t}\,)}\,\forall_L \qquad \frac{\Gamma \longrightarrow \boxed{G}}{\Gamma, \boxed{G \supset p^{-}(\vec{t}\,)} \longrightarrow p^{-}(\vec{t}\,)}\,\supset_L$$

Clause D is selected long before a match is established

What we want:

$$\frac{\Gamma, D_p, \boxed{D_p} \longrightarrow p^{-}(\vec{t}\,)}{\Gamma, D_p \longrightarrow \boxed{p^{-}(\vec{t}\,)}}\,\mathrm{atm}'_R$$

Clause Selection

$$\frac{\Gamma, D, \boxed{D} \longrightarrow p^{-}(\vec{t}\,)}{\Gamma, D \longrightarrow \boxed{p^{-}(\vec{t}\,)}}\,\mathsf{atm_R} \qquad \frac{\vdash t \quad \Gamma, \boxed{[t/x]D} \longrightarrow p^{-}(\vec{t}\,)}{\Gamma, \boxed{\forall x.\, D} \longrightarrow p^{-}(\vec{t}\,)}\,\forall_L \qquad \frac{\Gamma \longrightarrow \boxed{G}}{\Gamma, \boxed{G \supset p^{-}(\vec{t}\,)} \longrightarrow p^{-}(\vec{t}\,)}\,\supset_L$$

Clause D is selected long before a match is established

What we want:

$$\frac{\Gamma, D_p, \boxed{D_p} \longrightarrow p^{-}(\vec{t}\,)}{\Gamma, D_p \longrightarrow \boxed{p^{-}(\vec{t}\,)}}\,\mathsf{atm'_R}$$

What is the logical status of D_p?

Internalizing Indexing

An old idea:

- Associate an index atom i_p with each predicate p^-
- Guard each clause D for p^- with i_p: $\quad i_p \supset D$
- Release i_p to start search for $p^-(\vec{t}\,)$: $\quad i_p \supset p^-(\vec{t}\,)$

But ...

Internalizing Indexing

An old idea:

- Associate an index atom i_p with each predicate p^-
- Guard each clause D for p^- with i_p: $\quad i_p \supset D$
- Release i_p to start search for $p^-(\vec{t}\,)$: $\quad i_p \supset p^-(\vec{t}\,)$

But . . .

Checking a guard must succeed immediately

- Make i_p into a *positive* atom — p^+
  - (convenient separation of name spaces)

$$\frac{}{\Gamma; p^+ \longrightarrow \boxed{p^+}}\,\mathrm{Init_R}$$

Internalizing Indexing

An old idea:

- Associate an index atom i_p with each predicate p^-
- Guard each clause D for p^- with i_p: $\quad i_p \supset D$
- Release i_p to start search for $p^-(\vec{t}\,)$: $\quad i_p \supset p^-(\vec{t}\,)$

But . . .

Checking a guard must succeed immediately

- Make i_p into a *positive* atom — p^+
  - (convenient separation of name spaces)

$$\dfrac{}{\Gamma; p^+ \longrightarrow \boxed{p^+}}\,\mathrm{Init_R}$$

Used guards must not linger

- Make p^+ *linear*

Indexing append

- Clauses

$$\boxed{\begin{array}{rr} \forall l. & \top \supset \mathrm{app}^{-}(\mathrm{nil}, l, l) \\ \forall x, l_1, l_2, l_3.\ \mathrm{app}^{-}(l_1, l_2, l_3) & \supset \mathrm{app}^{-}(\mathrm{c}(x, l_1), l_2, \mathrm{c}(x, l_3)) \end{array}}$$

$$\begin{array}{lll} \mathrm{app}^{+} \multimap & \forall l. & \mathbf{1} \supset \mathrm{app}^{-}(\mathrm{nil}, l, l) \\ \mathrm{app}^{+} \multimap & \forall x, l_1, l_2, l_3.\ (\mathrm{app}^{+} \multimap \mathrm{app}^{-}(l_1, l_2, l_3)) \supset & \\ & & \mathrm{app}^{-}(\mathrm{c}(x, l_1), l_2, \mathrm{c}(x, l_3)) \end{array}$$

- Goals

$$\boxed{\begin{array}{c} \mathrm{app}^{-}(\mathrm{c}(m, \mathrm{nil}), \mathrm{c}(n, \mathrm{c}(o, \mathrm{nil})), \mathrm{c}(m, \mathrm{c}(n, \mathrm{c}(o, \mathrm{nil})))) \\ \text{(?- append([m],[n,o],[m,n,o]) in Prolog)} \end{array}}$$

$$\mathrm{app}^{+} \multimap \mathrm{app}^{-}(\mathrm{c}(m, \mathrm{nil}), \mathrm{c}(n, \mathrm{c}(o, \mathrm{nil})), \mathrm{c}(m, \mathrm{c}(n, \mathrm{c}(o, \mathrm{nil}))))$$

Approach

Transform programs into a focused linear program with negative and positive atoms

- Goals: $\ulcorner G \urcorner$ Clauses: $\ulcorner\!\ulcorner D \urcorner\!\urcorner$ Programs: $\ulcorner \Gamma \urcorner$

Target logic:

$$
\begin{array}{rrcl}
\textit{Head formulas:} & H & ::= & p^{+} \multimap a^{-} \\
\textit{Goal formulas:} & \underline{G} & ::= & H \mid \mathbf{1} \mid \underline{G_1} \otimes \underline{G_2} \\
\textit{Program formulas:} & \underline{D} & ::= & \underline{G} \supset a^{-} \mid \forall x.\, \underline{D} \\
\\
\textit{Programs:} & \underline{\Gamma} & ::= & \cdot \mid \underline{\Gamma}, p^{+} \multimap \underline{D} \\
\textit{Active indices:} & \Delta & ::= & \cdot \mid p^{+}
\end{array}
$$

$$
\begin{array}{ll}
\underline{\Gamma}; \Delta \longrightarrow \boxed{\underline{G}} & \textit{Goal } \underline{G} \textit{ is derivable from } \underline{\Gamma} \textit{ and } \Delta \\
\underline{\Gamma}; \Delta, \boxed{\underline{D}} \longrightarrow a^{-} & \textit{Clause } \underline{D} \textit{ derives } a^{-} \textit{ using } \underline{\Gamma} \textit{ and } \Delta
\end{array}
$$

Backward Proof Search for *Indexed* Horn Clauses

$$\frac{}{\underline{\Gamma}; \cdot \longrightarrow \boxed{\mathbf{1}}}\,\mathbf{1}_{\mathsf{R}} \qquad \frac{\underline{\Gamma}; \cdot \longrightarrow \boxed{\underline{G_1}} \quad \underline{\Gamma}; \cdot \longrightarrow \boxed{\underline{G_2}}}{\underline{\Gamma}; \cdot \longrightarrow \boxed{\underline{G_1} \otimes \underline{G_2}}}\,\otimes_{\mathsf{R}}$$

$$\frac{\underline{\Gamma}, p^{+} \multimap \underline{D}; q^{+}, \boxed{p^{+} \multimap \underline{D}} \longrightarrow a^{-}}{\underline{\Gamma}, p^{+} \multimap \underline{D}; q^{+} \longrightarrow \boxed{a^{-}}}\,\mathsf{atm}_{\mathsf{R}}$$

Inversion

. . . .

$$\frac{\underline{\Gamma}; \cdot \longrightarrow \boxed{\underline{G}}}{\underline{\Gamma}; \cdot, \boxed{\underline{G} \supset a^{-}} \longrightarrow a^{-}}\,\supset_{\mathsf{L}} \qquad \frac{\vdash t \quad \underline{\Gamma}; \cdot, \boxed{[t/x]\underline{D}} \longrightarrow a^{-}}{\underline{\Gamma}; \cdot, \boxed{\forall x.\, \underline{D}} \longrightarrow a^{-}}\,\forall_{\mathsf{L}}$$

Chaining

Backward Proof Search for *Indexed* Horn Clauses

$$\frac{\underline{\Gamma}; p^+ \longrightarrow \boxed{a^-}}{\underline{\Gamma}; \cdot \longrightarrow \boxed{p^+ \multimap a^-}} \multimap_R \quad \Bigg\} \text{Inversion}$$

. . . .

$$\frac{}{\underline{\Gamma}; p^+ \longrightarrow \boxed{p^+}} \mathrm{Init_R} \qquad \frac{\underline{\Gamma}; q^+ \longrightarrow \boxed{p^+} \qquad \underline{\Gamma}; \boxed{\underline{D}} \longrightarrow a^-}{\underline{\Gamma}; q^+, \boxed{p^+ \multimap \underline{D}} \longrightarrow a^-} \multimap_L \quad \Bigg\} \text{Chaining}$$

Does it Work?

Does it Work?

Lemma (Completeness)

- *If* $\Gamma \longrightarrow \boxed{G}$, *then* $\ulcorner\Gamma\urcorner; \cdot \longrightarrow \boxed{\ulcorner G \urcorner}$
- *If* $\Gamma, \boxed{D} \longrightarrow a^{-}$, *then* $\ulcorner\Gamma\urcorner; \cdot, \boxed{\ulcorner D \urcorner} \longrightarrow a^{-}$

Lemma (Soundness)

- *If* $\ulcorner\Gamma\urcorner; \cdot \longrightarrow \boxed{\ulcorner G \urcorner}$, *then* $\Gamma \longrightarrow \boxed{G}$
- *If* $\ulcorner\Gamma\urcorner; \cdot, \boxed{\ulcorner D \urcorner} \longrightarrow a^{-}$, *then* $\Gamma, \boxed{D} \longrightarrow a^{-}$

Does it Work?

Lemma (Completeness)

- *If* $\Gamma \longrightarrow \boxed{G}$, *then* $\ulcorner\Gamma\urcorner; \cdot \longrightarrow \boxed{\ulcorner G \urcorner}$
- *If* $\Gamma, \boxed{D} \longrightarrow a^-$, *then* $\ulcorner\Gamma\urcorner; \cdot, \boxed{\ulcorner\!\ulcorner D \urcorner\!\urcorner} \longrightarrow a^-$

Lemma (Soundness)

- *If* $\ulcorner\Gamma\urcorner; \cdot \longrightarrow \boxed{\ulcorner G \urcorner}$, *then* $\Gamma \longrightarrow \boxed{G}$
- *If* $\ulcorner\Gamma\urcorner; \cdot, \boxed{\ulcorner\!\ulcorner D \urcorner\!\urcorner} \longrightarrow a^-$, *then* $\Gamma, \boxed{D} \longrightarrow a^-$

Proof.

By simultaneous induction □

How does it Work?

Focusing forces clause D to have the same index as goal a^-

$$\underbrace{\underline{\Gamma'}, p^+ \multimap \underline{D}}_{\underline{\Gamma}}; \cdot \longrightarrow \boxed{p^+ \multimap p^-(\vec{t}\,)}$$

How does it Work?

Focusing forces clause D to have the same index as goal a^-

$$\dfrac{\underline{\Gamma}; p^+ \longrightarrow \boxed{p^-(\vec{t}\,)}}{\underbrace{\underline{\Gamma'}, p^+ \multimap \underline{D}}_{\underline{\Gamma}}; \cdot \longrightarrow \boxed{p^+ \multimap p^-(\vec{t}\,)}}\ \multimap R$$

How does it Work?

Focusing forces clause D to have the same index as goal a^-

$$
\dfrac{\dfrac{\underline{\Gamma}; p^+, \boxed{p^+ \multimap \underline{D}} \longrightarrow p^-(\vec{t}\,)}{\underline{\Gamma}; p^+ \longrightarrow \boxed{p^-(\vec{t}\,)}}\,\mathrm{atm_R}}{\underbrace{\underline{\Gamma'}, p^+ \multimap \underline{D}}_{\underline{\Gamma}}; \cdot \longrightarrow \boxed{p^+ \multimap p^-(\vec{t}\,)}}\multimap_{\mathrm{R}}
$$

How does it Work?

Focusing forces clause D to have the same index as goal a^-

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\underline{\Gamma}; p^+ \longrightarrow \boxed{p^+}}\,\text{init}_R \qquad \underline{\Gamma}; \cdot, \boxed{\underline{D}} \longrightarrow p^-(\vec{t})}{\underline{\Gamma}; p^+, \boxed{p^+ \multimap \underline{D}} \longrightarrow p^-(\vec{t})}\,{\multimap}_L}{\underline{\Gamma}; p^+ \longrightarrow \boxed{p^-(\vec{t})}}\,\text{atm}_R}{\underbrace{\underline{\Gamma'}, p^+ \multimap \underline{D}}_{\underline{\Gamma}}; \cdot \longrightarrow \boxed{p^+ \multimap p^-(\vec{t})}}\,{\multimap}_R
$$

A Macro-Rule for Indexed Clause Selection

$$\frac{\underline{\Gamma}; \cdot, \boxed{\underline{D}} \longrightarrow p^{-}(\vec{t}\,)}{\underbrace{\underline{\Gamma'}, p^{+} \multimap \underline{D}}_{\underline{\Gamma}}; \cdot \longrightarrow \boxed{p^{+} \multimap p^{-}(\vec{t}\,)}}$$

A Macro-Rule for Indexed Clause Selection

$$\dfrac{\dfrac{\underline{\Gamma}; \cdot, \boxed{\underline{D}} \longrightarrow p^{-}(\vec{t}\,)}{\underbrace{\underline{\Gamma'}, p^{+} \multimap \underline{D}}_{\underline{\Gamma}}; \cdot \longrightarrow \boxed{p^{+} \multimap p^{-}(\vec{t}\,)}}}{}$$

$$\Downarrow$$

$$\dfrac{\Gamma, D_p, \boxed{D_p} \longrightarrow p^{-}(\vec{t}\,)}{\Gamma, D_p \longrightarrow \boxed{p^{-}(\vec{t}\,)}}\ \mathrm{atm}'_{\mathrm{R}}$$

Outline

First-Order Indexing

Take information about predicate arguments into account

$$\boxed{\begin{array}{ll} \forall l. & \top \supset \mathrm{app}^{-}(\mathrm{nil}, l, l) \\ \forall x, l_1, l_2, l_3.\ \mathrm{app}^{-}(l_1, l_2, l_3) & \supset \mathrm{app}^{-}(\mathrm{c}(x, l_1), l_2, \mathrm{c}(x, l_3)) \end{array}}$$

For each predicate, fix

- position
  - l_1, l_2 or l_3 (Prolog indexes l_1, always)
  - ...or maybe a combination
- depth (1 in Prolog)

(somebody else makes the decision)

First-Order Indexing

- Associate an *indexing constant* to each function symbol
  - $\mathrm{IC}(p^-)$: the set of indexing constants in indexing position of p^-
- Parametrize p^+ with indexing constant of term in the head
  - $\ldots \supset p^-(\ldots, c(\vec{t}\,), \ldots) \quad \rightsquigarrow \quad p^+(c) \multimap \ldots \supset p^-(\ldots, c(\vec{t}\,), \ldots)$
- Use quantifier when term in head contains variable
  - $\ldots \supset p^-(\ldots, x, \ldots) \quad \rightsquigarrow \quad \forall i.\, p^+(i) \multimap \ldots \supset p^-(\ldots, x, \ldots)$

Indexing $\mathsf{app}(l_1, l_2, l_3)$

$$\boxed{\begin{array}{rr} \forall l. & \top \supset \mathsf{app}^-(\mathsf{nil}, l, l) \\ \forall x, l_1, l_2, l_3.\ \mathsf{app}^-(l_1, l_2, l_3) & \supset \mathsf{app}^-(\mathsf{c}(x, l_1), l_2, \mathsf{c}(x, l_3)) \end{array}} \quad \mathrm{IC}(\mathsf{app}^-) = \{\mathsf{nil}, \mathsf{c}\}$$

Each clause head starts with a function symbol in indexing position

Indexing $\mathsf{app}(l_1, l_2, l_3)$

$$\boxed{\begin{array}{rr} \forall l. & \top \supset \mathsf{app}^-(\mathsf{nil}, l, l) \\ \forall x, l_1, l_2, l_3.\, \mathsf{app}^-(l_1, l_2, l_3) & \supset \mathsf{app}^-(\mathsf{c}(x, l_1), l_2, \mathsf{c}(x, l_3)) \end{array}} \quad \mathrm{IC}(\mathsf{app}^-) = \{\mathsf{nil}, \mathsf{c}\}$$

Each clause head starts with a function symbol in indexing position

$$\begin{array}{lll} \mathsf{app}^+(\mathsf{nil}) \multimap & \forall l. & \mathbf{1} \supset \mathsf{app}^-(\mathsf{nil}, l, l) \\ \mathsf{app}^+(\mathsf{c}) \multimap & \forall x, l_1, l_2, l_3.\, \ulcorner \mathsf{app}^-(l_1, l_2, l_3) \urcorner \supset & \\ & & \mathsf{app}^-(\mathsf{c}(x, l_1), l_2, \mathsf{c}(x, l_3)) \end{array}$$

Indexing $\mathsf{app}(l_1, l_2, l_3)$

$$\boxed{\begin{array}{rr} \forall l. & \top \supset \mathsf{app}^-(\mathsf{nil}, l, l) \\ \forall x, l_1, l_2, l_3.\, \mathsf{app}^-(l_1, l_2, l_3) & \supset \mathsf{app}^-(\mathsf{c}(x, l_1), l_2, \mathsf{c}(x, l_3)) \end{array}} \quad \mathrm{IC}(\mathsf{app}^-) = \{\mathsf{nil}, \mathsf{c}\}$$

Each clause head starts with a function symbol in indexing position

$$\begin{array}{lll} \mathsf{app}^+(\mathsf{nil}) \multimap & \forall l. & \mathbf{1} \supset \mathsf{app}^-(\mathsf{nil}, l, l) \\ \mathsf{app}^+(\mathsf{c}) \multimap & \forall x, l_1, l_2, l_3.\, \ulcorner \mathsf{app}^-(l_1, l_2, l_3) \urcorner \supset & \\ & & \mathsf{app}^-(\mathsf{c}(x, l_1), l_2, \mathsf{c}(x, l_3)) \end{array}$$

The body has a variable in indexing position: we must be prepared for any constant in $\mathrm{IC}(\mathsf{app}^-)$:

Indexing $\mathrm{app}(l_1, l_2, l_3)$

$$\boxed{\begin{array}{rr} \forall l. & \top \supset \mathrm{app}^-(\mathrm{nil}, l, l) \\ \forall x, l_1, l_2, l_3.\, \mathrm{app}^-(l_1, l_2, l_3) & \supset \mathrm{app}^-(\mathrm{c}(x, l_1), l_2, \mathrm{c}(x, l_3)) \end{array}} \quad \mathrm{IC}(\mathrm{app}^-) = \{\mathrm{nil}, \mathrm{c}\}$$

Each clause head starts with a function symbol in indexing position

$$\begin{array}{lcr} \mathrm{app}^+(\mathrm{nil}) \multimap & \forall l. & \mathbf{1} \supset \mathrm{app}^-(\mathrm{nil}, l, l) \\ \mathrm{app}^+(\mathrm{c}) \multimap & \forall x, l_1, l_2, l_3. \ulcorner \mathrm{app}^-(l_1, l_2, l_3) \urcorner \supset & \\ & & \mathrm{app}^-(\mathrm{c}(x, l_1), l_2, \mathrm{c}(x, l_3)) \end{array}$$

The body has a variable in indexing position: we must be prepared for any constant in $\mathrm{IC}(\mathrm{app}^-)$: $\ulcorner \mathrm{app}^-(l_1, l_2, l_3) \urcorner =$

$$\begin{array}{rl} (& l_1 = \mathrm{nil} \otimes (\mathrm{app}^+(\mathrm{nil}) \multimap \mathrm{app}^-(l_1, l_2, l_3)) \\ \oplus & (\exists y, z.\, l_1 = \mathrm{c}(y, z) \otimes (\mathrm{app}^+(\mathrm{c}) \multimap \mathrm{app}^-(l_1, l_2, l_3))) \end{array}$$

Indexing $\mathrm{app}(l_1, l_2, l_3)$

$$\boxed{\begin{array}{rl} \forall l. & \top \supset \mathrm{app}^{-}(\mathrm{nil}, l, l) \\ \forall x, l_1, l_2, l_3.\ \mathrm{app}^{-}(l_1, l_2, l_3) & \supset \mathrm{app}^{-}(\mathrm{c}(x, l_1), l_2, \mathrm{c}(x, l_3)) \end{array}} \quad \mathrm{IC}(\mathrm{app}^{-}) = \emptyset$$

There are no function symbols to index on!

We cannot do better than indexing on the predicate symbol

- We get our first encoding

Indexing $\mathrm{app}(l_1, l_2, l_3)$

$$\boxed{\begin{array}{rl} \forall l. & \top \supset \mathrm{app}^{-}(\mathrm{nil}, l, l) \\ \forall x, l_1, l_2, l_3.\ \mathrm{app}^{-}(l_1, l_2, l_3) & \supset \mathrm{app}^{-}(\mathrm{c}(x, l_1), l_2, \mathrm{c}(x, l_3)) \end{array}} \quad \mathrm{IC}(\mathrm{app}^{-}) = \{\mathrm{c}\}$$

First clause has a variable in indexing position

Indexing $\mathrm{app}(l_1, l_2, l_3)$

$$\boxed{\begin{array}{lr} \forall l. & \top \supset \mathrm{app}^-(\mathrm{nil}, l, l) \\ \forall x, l_1, l_2, l_3.\, \mathrm{app}^-(l_1, l_2, l_3) & \supset \mathrm{app}^-(\mathrm{c}(x, l_1), l_2, \mathrm{c}(x, l_3)) \end{array}} \quad \mathrm{IC}(\mathrm{app}^-) = \{\mathrm{c}\}$$

First clause has a variable in indexing position

$$\begin{array}{lll} \forall i.\, \mathrm{app}^+(i) \multimap & \forall l. & \mathbf{1} \supset \mathrm{app}^-(\mathrm{nil}, l, l) \\ \quad \mathrm{app}^+(\mathrm{c}) \multimap & \forall x, l_1, l_2, l_3.\, \ulcorner \mathrm{app}^-(l_1, l_2, l_3) \urcorner \supset & \\ & & \mathrm{app}^-(\mathrm{c}(x, l_1), l_2, \mathrm{c}(x, l_3)) \end{array}$$

Does it Work?

- Sound and complete
- Yields indexing macro-rule

Outline

Hereditary Harrop Formulas

- **Minimal presentation**

$$\begin{array}{rrcl} \textit{Formulas:} & A & ::= & a^- \mid A_1 \supset A_2 \mid \forall x.\, A \\ \textit{Programs:} & \Gamma & ::= & \cdot \mid \Gamma, A \end{array}$$

  Direct adaptation of technique for Horn clauses

- **Conjunctive presentation**

$$\begin{array}{rrcl} \textit{Formulas:} & A & ::= & a^- \mid A_1 \supset A_2 \mid \forall x.\, A \\ & & & \mid \top \mid A_1 \wedge A_2 \\ \textit{Programs:} & \Gamma & ::= & \cdot \mid \Gamma, A \end{array}$$

  Significantly more complex

These presentations are *equivalent*, but not in the linear case

Conjunctive Hereditary Harrop Formulas

- Clauses can have multiple heads
  - $A = a^- \supset (b^- \wedge (c^- \supset d^-))$ has two heads, b^- and d^-
  - $B = a^- \supset \top$ has no head

Conjunctive Hereditary Harrop Formulas

- Clauses can have multiple heads
  - $A = a^- \supset (b^- \wedge (c^- \supset d^-))$ has two heads, b^- and d^-
  - $B = a^- \supset \top$ has no head
- Clause A can be triggered by either b^- or d^-.

Conjunctive Hereditary Harrop Formulas

- Clauses can have multiple heads
  - $A = a^- \supset (b^- \wedge (c^- \supset d^-))$ has two heads, b^- and d^-
  - $B = a^- \supset \top$ has no head
- Clause A can be triggered by either b^- or d^-. Accept both:

$$(b^+ \oplus d^+) \multimap <\textit{rest of } A>$$

Conjunctive Hereditary Harrop Formulas

- Clauses can have multiple heads
  - $A = a^{-} \supset (b^{-} \wedge (c^{-} \supset d^{-}))$ has two heads, b^{-} and d^{-}
  - $B = a^{-} \supset \top$ has no head
- Clause A can be triggered by either b^{-} or d^{-}. Accept both:

$$(b^{+} \oplus d^{+}) \multimap \textless \textit{rest of } A \textgreater$$

- Doing so consumes the trigger!

Conjunctive Hereditary Harrop Formulas

- Clauses can have multiple heads
  - $A = a^- \supset (b^- \wedge (c^- \supset d^-))$ has two heads, b^- and d^-
  - $B = a^- \supset \top$ has no head
- Clause A can be triggered by either b^- or d^-. Accept both:

$$(b^+ \oplus d^+) \multimap \textit{<rest of A>}$$

- Doing so consumes the trigger! We need to reassert it

$$\forall i.\, idx^+(i) \mathbin{\&} (idx^+(b) \oplus idx^+(d)) \multimap idx^+(i) \otimes \textit{<rest of A>}$$

Conjunctive Hereditary Harrop Formulas

- Clauses can have multiple heads
  - $A = a^- \supset (b^- \wedge (c^- \supset d^-))$ has two heads, b^- and d^-
  - $B = a^- \supset \top$ has no head
- Clause A can be triggered by either b^- or d^-. Accept both:

$$(b^+ \oplus d^+) \multimap \langle \textit{rest of } A \rangle$$

- Doing so consumes the trigger! We need to reassert it

$$\forall i.\, idx^+(i) \,\&\, (idx^+(b) \oplus idx^+(d)) \multimap idx^+(i) \otimes \langle \textit{rest of } A \rangle$$

- Use of $\otimes$ breaks focus!

Conjunctive Hereditary Harrop Formulas

- Clauses can have multiple heads
  - $A = a^- \supset (b^- \wedge (c^- \supset d^-))$ has two heads, b^- and d^-
  - $B = a^- \supset \top$ has no head
- Clause A can be triggered by either b^- or d^-. Accept both:

$$(b^+ \oplus d^+) \multimap <\!\mathit{rest\ of\ A}\!>$$

- Doing so consumes the trigger! We need to reassert it

$$\forall i.\, idx^+(i) \mathbin{\&} (idx^+(b) \oplus idx^+(d)) \multimap idx^+(i) \otimes <\!\mathit{rest\ of\ A}\!>$$

- Use of $\otimes$ breaks focus! Use nested implication instead

Indexed Conjunctive Hereditary Harrop Formulas

The encoding of our two examples:

- $$\begin{array}{ll} \forall i.\, idx^{+}(i) \,\&\, (idx^{+}(b) \oplus idx^{+}(d)) \multimap & \\ \quad (idx^{+}(a) \multimap a^{-}) \supset & a^{-} \supset \\ \quad\quad (\quad (idx^{+}(i) \multimap idx^{+}(b)) \multimap b^{-} & \quad (\quad b^{-} \\ \quad\quad \& \; (idx^{+}(i) \multimap idx^{+}(d)) \multimap (idx^{+}(c) \multimap c^{-}) & \quad \wedge \; (c^{-} \\ \quad\quad\quad\quad \supset d^{-})) & \quad\quad \supset d^{-})) \end{array}$$
- $\mathbf{0} \multimap (idx^{+}(a) \multimap a^{-}) \supset \top \qquad a^{-} \supset \top$

Outline

Conclusions

A logical foundation of indexing for backward logic programming

Indexing on:

- predicate symbol
- terms, in any position and at any depth

for

- Horn clauses (Prolog)
- Hereditary Harrop formulas (λProlog)

and their linear variants

Future Work

- Two-stage indexing
  - First, on predicate symbol
  - Then, on terms
- Beyond backward logic programming
  - Forward logic programming
  - Theorem proving