A First-Order Interpreter for Knowledge-based Golog based on Exact Progression and Limited Reasoning

Yi Fan   Minghui Cai   Naiqi Li   Yongmei Liu

Dept. of Computer Science
Sun Yat-sen University
Guangzhou, China

July 24, 2012

AAAI-12, Toronto
Current Golog Interpreters

- Based on closed world assumption (CWA), dynamic CWA, or domain closure assumption (DCA)
- Query evaluation based on regression, with decreasing efficiency as the length of action sequences grows
- Online, offline or a combination
  - search operator for guarding successful execution
  - planning operator for improving efficiency
Proper Knowledge Bases [Lakemeyer and Levesque, 02]

Definition
A first-order KB equivalent to a possibly infinite set of clauses

Example
\[ \forall x. x \neq y \supset \neg \text{on}(x, y) \lor \neg \text{clear}(y), \quad \forall x. \neg \text{on}(x, x) \]
\[ \forall x, y, z. y \neq z \supset \neg \text{on}(x, y) \lor \neg \text{on}(x, z) \]
\[ \forall x. x \neq A \land x \neq D \supset \text{clear}(x) \]

[Liu, Lakemeyer & Levesque, 04]
proposed a logic of limited belief \( SL \) and showed \( SL \)-based reasoning with proper\(^+\) KBs is decidable.

[Liu & Lakemeyer, 09]
showed for local-effect actions and proper\(^+\) KBs, progression is not only first-order definable but also efficiently computable.
Our Contribution

An interpreter based on exact progression and limited reasoning

- Handle first-order incomplete information in the form of proper\(^+\) KBs
- Implemented progression and limited reasoning by grounding based on unique name assumption
- The search operator returns a conditional plan
- The planning operator calls a modern planner when local complete information is available
Implementing Progression and Evaluation by Grounding

- We first implemented algorithms by Liu, Lakemeyer and Levesque, but the implementations were not efficient.
- We considered implementation via grounding, but there are infinitely many individuals.
- The trick is to use an appropriate number of them as representatives of those not mentioned by the KB.

Progression and Evaluation on Proper\(^+\) KBs via Grounding
Initial Grounding

- It should be a finite representation of infinitely many clauses.

Proper$^+$ Blocks World

$\forall x. x \neq y \supset \neg \text{on}(x, y) \lor \neg \text{clear}(y), \forall x. x \neq A \land x \neq B \supset \text{clear}(x)$

- The width of the proper$^+$ KB above is 2, so we introduce 2 representatives, $u_1$ and $u_2$.

Grounding (brute-force)

- $\neg \text{on}(A, B) \lor \neg \text{clear}(B), \neg \text{on}(A, u_1) \lor \neg \text{clear}(u_1)$,
- $\neg \text{on}(A, u_2) \lor \neg \text{clear}(u_2), \neg \text{on}(B, A) \lor \neg \text{clear}(A)$,
- $\neg \text{on}(B, u_1) \lor \neg \text{clear}(u_1), \neg \text{on}(B, u_2) \lor \neg \text{clear}(u_2)$...
- clear$(u_1), \text{clear}(u_2)$
Extended Grounding

It should be extended to describe new individuals explicitly too.

Original KB with $u_1$ and $u_2$ as representatives

\[\neg on(u_1, u_2), \neg on(u_1, A), \neg on(u_1, B),\]
\[\neg on(A, u_1), \neg on(B, u_1),\]
\[clear(u_1), \neg on(u_1, u_1), \ldots\]

When an action mentions a new individual $c_1$, we add the following to the original KB:

Extension with new individual $c_1$

\[\neg on(c_1, u_2), \neg on(u_1, c_1), \neg on(c_1, A), \neg on(c_1, B),\]
\[\neg on(A, c_1), \neg on(B, c_1),\]
\[clear(c_1), \neg on(c_1, c_1), \ldots\]
Progression wrt Local-Effect Actions

Local-Effect Actions
only change the truth value of fluent atoms with arguments mentioned by the actions

Influenced Atoms of $\alpha = move(B, A, c_1)$
$on(B, A, s), on(A, c_1, s), clear(A, s), clear(c_1, s)$

Progression of a ground KB
1. extend the ground KB if needed
2. add successor state axioms instantiated wrt influenced atoms
3. forget the influenced atoms via resolution

Theorem

Progression here is equivalent to that in [Liu & Lakemeyer, 09].
We perform unit propagation over a ground KB

For clause evaluation
- $\text{eval}(\phi(d_1, \ldots, d_n)) \rightarrow \text{eval}(\phi(u_1, \ldots, u_n))$, for $d_1, \ldots, d_n$ not mentioned by KB and $u_1, \ldots, u_n$ as representatives
- check if $\phi(u_1, \ldots, u_n)$ is subsumed by a clause in the KB

Others are reduced to clause evaluation recursively, e.g.
- $\text{eval}(\eta \lor \psi) \rightarrow \text{eval}(\eta)$ or $\text{eval}(\psi)$ returns true
- $\text{eval}(\exists x \psi) \rightarrow \text{eval}(\psi(x/d))$ returns true for some $d$ in a particular finite domain

**Theorem**

Evaluation here is equivalent to that in [Liu et al., 04] at $B_0$ level.
An Interpreter

Implemented in Prolog
- with evaluation and progression implemented in C

Search Operator $\Sigma(\delta)$
- Looking ahead to ensure that nondeterministic choices are resolved to guarantee the successful completion of $\delta$
- Sensing actions allowed in $\delta$ and a conditional plan is returned
- Automatically branching wrt sensing results, not relying on special branching actions specified by the programmer
Planning Operator $\Upsilon(\tau, \delta)$

- Based on the work of [Baier, Fritz & McIlraith, 07]
- $\tau$ explicitly specifies the domain of all related individuals
- Local complete information: for any $P(\vec{c})$ related to $\delta$, $P(\vec{c}) \in KB$ or $\neg P(\vec{c}) \in KB$
- No sensing actions are allowed in $\delta$
- Calling a modern planner to return a sequence of actions, improving efficiency and ensuring soundness and completeness
- A planner can be called multiple times efficiently because progression maintains the current KB
## Experimental Results for Wumpus World (8×8, 3000)

<table>
<thead>
<tr>
<th>Prob</th>
<th>Gold</th>
<th>IMP</th>
<th>Reward</th>
<th>Moves</th>
<th>Time</th>
<th>Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>1412</td>
<td>695</td>
<td>437</td>
<td>33</td>
<td>0.670</td>
<td>16</td>
</tr>
<tr>
<td>15%</td>
<td>890</td>
<td>917</td>
<td>275</td>
<td>22</td>
<td>0.430</td>
<td>11</td>
</tr>
<tr>
<td>20%</td>
<td>567</td>
<td>1171</td>
<td>175</td>
<td>14</td>
<td>0.254</td>
<td>7</td>
</tr>
<tr>
<td>30%</td>
<td>263</td>
<td>1581</td>
<td>82</td>
<td>6</td>
<td>0.112</td>
<td>3</td>
</tr>
<tr>
<td>40%</td>
<td>182</td>
<td>1924</td>
<td>58</td>
<td>3</td>
<td>0.064</td>
<td>2</td>
</tr>
</tbody>
</table>
An Example Program Execution in the Blocks World

Proper Initial KB

$$\forall x. x \neq A \land x \neq B \land x \neq C \land x \neq D \supset \text{clear}(x),$$
$$\forall x, y. x \neq y \supset \neg \text{on}(x, y) \lor \neg \text{clear}(y), \quad \forall x. \neg \text{on}(x, x),$$
$$\forall x, y. x \neq y \supset \neg \text{on}(x, y) \lor \neg \text{on}(y, x), \ldots$$

very little knowledge about the exact configuration

Actions: $move(x, y, z), \text{sense\_clear}(x), \text{sense\_on}(x, y)$

Goal: make clear a list of blocks: $A, B, C, D$
Conclusions

- Implemented a Golog interpreter based on exact progression of first-order incomplete information
- Implemented limited reasoning, no CWA, DCWA or DCA, but only unique name assumption and DCWA on knowledge
- Implemented progression and evaluation via grounding with theoretical foundation
- A planning problem is generated dynamically each time the planner is called during a single execution task
- Search operator returns a conditional plan not relying on special branching actions

Future Work

- Implement limited reasoning at the $B_1$ level