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

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

Definition 1 *A first-order KB equivalent to a possibly infinite set of clauses [Lakemeyer and Levesque, 02]*

An example is:
$\forall x.x \neq y \supset \neg on(x,y) \vee \neg clear(y)$, $\quad \forall x.\neg on(x,x)$
$\forall x,y,z.y \neq z \supset \neg on(x,y) \vee \neg on(x,z)$ $\forall x.x \neq A \wedge x \neq D \supset clear(x)$

- Liu, Lakemeyer & Levesque, 04 proposed a logic of limited belief $\mathcal{SL}$ and showed $\mathcal{SL}$-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

Well-formed Basic Action Theories (BAT)

- Initial KB $\mathcal{D}_{S_0}$ is proper$^+$, Sensing result is quantifier-free,
- Physical actions are local-effect, only changing the truth value of fluent atoms with arguments mentioned by the actions.
- Influenced atoms $\Omega(S_0)$: those fluent atoms mentioning some constants in physical action $A(\vec{c})$,
- $\mathcal{D}_{ss}[\Omega]$: the successor state axioms (SSAs) instantiated wrt $\Omega(S_0)$.

Result I

Theorem 1 *Let $\mathcal{D}$ be a well-formed BAT and $\alpha = A(\vec{c})$ a ground physical action. Let B be the set of constants appearing in $\vec{c}$ but not Σ_p. We define $pprog(\Sigma_p, \alpha)$ as $forget(egnd(\Sigma_p, B) \cup \mathcal{D}_{ss}[\Omega], \Omega(S_0))(S_0/S_\alpha)$. Then $pprog(\Sigma_p, \alpha)$ is a finite representation of $prog(\Sigma, \alpha)$.*

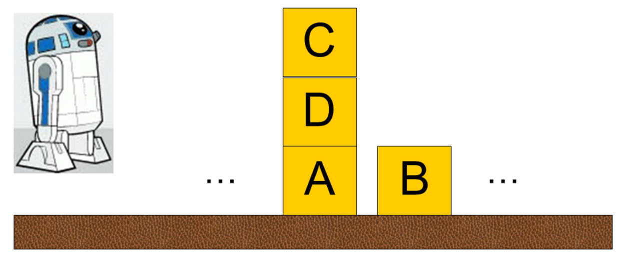
- Σ_p is a ground KB of Σ with some extra constants as representatives, and $egnd(\Sigma_p, B)$ is the extension of Σ_p with B.
- For any set Σ of clauses and any atom p, $forget(\Sigma, p)$ can be computed by adding resolutions wrt p into Σ and delete all clauses containing p.

Result II

Theorem 2 $F[\mathsf{UP}(\Sigma_p), \varphi] = 1$ *iff* $bcl(\mathbf{B}_0\Sigma) \models \varphi$.

- φ is an FO formula (subjective) whose atoms are of the form $\mathbf{B}_0\phi$ where ϕ is an FO formula (objective).
- $bcl(\mathbf{B}_0\Sigma) \stackrel{def}{=} \mathbf{B}_0\Sigma \cup \{\neg\mathbf{B}_0\phi | \mathbf{B}_0\Sigma \not\models \mathbf{B}_0\phi\}$ (DCWA on knowledge).
- $\mathsf{UP}(s)$ is the closure of s under unit resolution and F is implemented trivially on G procedure.
- G procedure reduced an FO query inductively to ground clause quries and check whether they are subsumed.

Motivating Example



- The initial KB $\mathcal{D}_{S_0}$ is: $\forall x.x \neq A \wedge x \neq B \wedge x \neq C \wedge x \neq D \supset clear(x), \forall x,y.x \neq A \wedge x \neq B \wedge x \neq C \wedge x \neq D \wedge x \neq y \supset \neg on(x,y)$, % and some domain axioms.
- Actions: $move(x,y,z), sense\_clear(x), sense\_on(x,y)$
- Goal: make clear a list of blocks: A, B, C, D
- Note that 4 blocks `a`, `b`, `c` and `d` appear in $\mathcal{D}_{S_0}$

Obviously she needs some extra blocks *e.g.* `c1` and `c2` in the infinite domain to accomplish her task.

```
?- lbGolog(make_clear_all).
sense_clear(a):no.
sense_on(b,a):no.
sense_on(c,a):no.
sense_on(d,a):yes.
sense_clear(d):no.
sense_on(b,d):no.
sense_on(c,d):yes.
sense_clear(c):yes.
move(c,d,c1)
move(d,a,c2)
sense_clear(b):yes.
true.
```

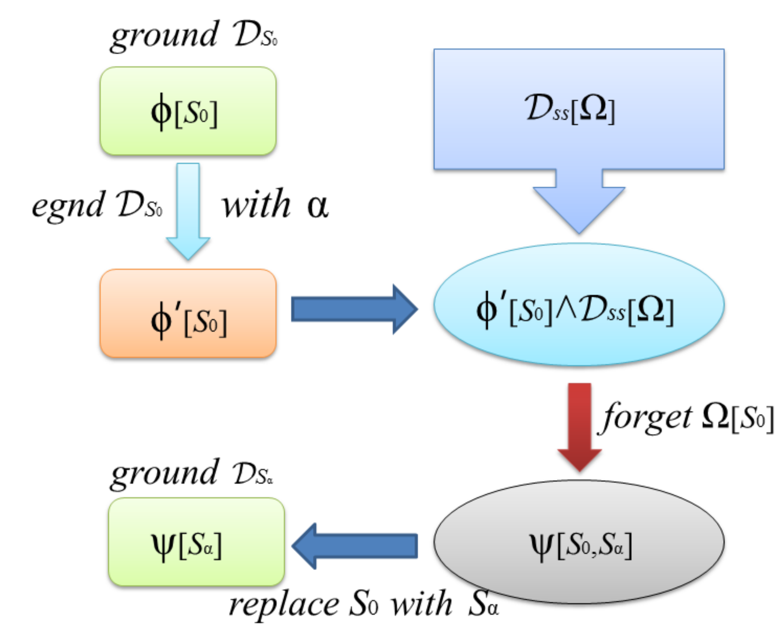
General Picture

On the right is the progression procedure shown in the figure.

Input:
$\mathcal{D}_{S_0}$: a ground KB with reserved constants as representatives.
α : a ground action that is executed.

Output:
$\mathcal{D}_{S_\alpha}$: the current KB with reserved constants.

1. Ground the initial KB and compute unit propagation (UP).
2. Compute extended grounding if α mentions new constants, then progress the KB.
   - add successor state axioms wrt influenced atoms
   - forget the influenced atoms via resolution
3. Answer any query φ in the current KB.



Progression wrt $\alpha = move(A,B,C)$

- Initial KB: $\mathcal{D}_{S_0} = \{\forall x,y.x \neq y \supset \neg on(x,y,S_0) \vee \neg clear(y,S_0), on(A,B,S_0), clear(A,S_0), clear(C,S_0)\}$.
- We are using brute-force grounding. The width of $\mathcal{D}_{S_0}$ is 2, so we introduce u_1 and u_2 as representatives, then $\Sigma_p = prop(\mathcal{D}_{S_0}, N) = \{\, on(A,B,S_0), clear(A,S_0), clear(C,S_0),$ $\neg on(A,B,S_0) \vee \neg clear(B,S_0), \neg on(A,C,S_0) \vee \neg clear(C,S_0), \ldots,$ $\neg on(C,u_1,S_0) \vee \neg clear(u_1,S_0), \neg on(C,u_2,S_0) \vee \neg clear(u_2,S_0),$ $\neg on(u_1,C,S_0) \vee \neg clear(C,S_0), \neg on(u_2,C,S_0) \vee \neg clear(C,S_0)$ % similar sentences for A and $B \ldots \}$.
- Influenced atoms: $\Omega(S_0) = \{clear(B,S_0), clear(C,S_0), on(A,B,S_0), on(A,C,S_0)\}$.
- Instantiated SSAs: $\mathcal{D}_{ss}[\Omega] = \{clear(B,S_\alpha), \neg clear(C,S_\alpha), \neg on(A,B,S_\alpha), on(A,C,S_\alpha)\}$.
- $pprog(\Sigma_p, \alpha) = \{clear(A,S_\alpha), \neg on(B,A,S_\alpha) \vee \neg clear(A,S_\alpha),$ $\neg on(B,C,S_\alpha), \neg on(C,A,S_\alpha) \vee \neg clear(A,S_\alpha), clear(B,S_\alpha),$ $\neg clear(C,S_\alpha), \neg on(A,B,S_\alpha), on(A,C,S_\alpha),$ $\neg on(u_1,C,S_\alpha), \neg on(u_2,C,S_\alpha)\,\}$.

  Here the last 2 sentences are resolutions between $\neg on(u_1,C,S_0) \vee \neg clear(C,S_0)$, $\neg on(u_1,C,S_0) \vee \neg clear(C,S_0)$ and $clear(C,S_0)$, wrt the influenced atom $clear(C,S_0)$, with their situation arguments replaced.

Evaluation in the Context of Grounding

- We perform unit propagation over a ground KB
- For clause evaluation
  - $eval(\phi(d_1,\ldots,d_n)) \rightarrow eval(\phi(u_1,\ldots,u_n))$, 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.
  - $eval(\eta \vee \psi) \longrightarrow eval(\eta)$ or $eval(\psi)$ returns true
  - $eval(\exists x\psi) \longrightarrow eval(\psi(x/d))$ returns true for some d in a particular finite domain

An Interpreter

The interpreter is implemented in Prolog with evaluation and progression programmed in C.

The search operator $\Sigma(\delta)$:

- looking ahead to ensure that nondeterministic choices are resolved to guarantee the successful completion of δ.
- sensing actions allowed in δ and a conditional plan is returned.
- automatically branching wrt sensing results, not relying on special branching actions specified by the programmer.

The planning operator $\Upsilon(\tau, \delta)$:

- based on the work of [Baier, Fritz & McIlraith, 07].
- τ explicitly specifies the domain of all related individuals.
- local complete information: for any $P(\vec{c})$ related to δ, $P(\vec{c}) \in KB$ or $\neg P(\vec{c}) \in KB$.
- no sensing actions are allowed in δ.
- control structure is compiled into the planning problem.
- calling a modern planner to return a sequence of actions.

Experimental Results for Wumpus World (8×8, 3000)

- Prob: the probability of a location containing a pit
- Gold: the number of golds the agent has picked up
- IMP: the number of maps for which it is impossible to explore
- Calls: the number of times a planner is called in a game
- Time: the time spent in one game on average in seconds

Prob	Gold	IMP	Reward	Moves	Time	Calls
10%	1412	695	437	33	0.670	16
15%	890	917	275	22	0.430	11
20%	567	1171	175	14	0.254	7
30%	263	1581	82	6	0.112	3
40%	182	1924	58	3	0.064	2

Future Directions

- Implement limited reasoning at $\mathbf{B}_1$ level
- Support of state constraint
- Incorporating procedure calls in the scope of planning operators

References

[Baier, Fritz, and McIlraith 2007] Baier, J. A.; Fritz, C.; and McIlraith, S. A. 2007. Exploiting procedural domain control knowledge in state-of-the-art planners. In *Proc. ICAPS-07*, 26–33.

[De Giacomo, Lespérance, and Levesque 2000] De Giacomo, G.; Lespérance, Y.; and Levesque, H. J. 2000. Congolog, a concurrent programming language based on the situation calculus. *Artif. Intell.* 121(1-2):109–169.

[De Giacomo, Levesque, and Sardiña2001] De Giacomo, G.; Levesque, H. J.; and Sardiña, S. 2001. Incremental execution of guarded theories. *ACM Trans. Comput. Log.* 2(4):495–525.

[Hoffmann and Nebel 2001] Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res.* 14:253–302.

[Lakemeyer 1999] Lakemeyer, G. 1999. On sensing and off-line interpreting in Golog. In *Logical Foundations for Cognitive Agents, Contributions in Honor of Ray Reiter*.

[Liu and Lakemeyer 2009] Liu, Y., and Lakemeyer, G. 2009. On first-order definability and computability of progression for local-effect actions and beyond. In *Proc. IJCAI-09*.

[Liu, Lakemeyer, and Levesque 2004] Liu, Y.; Lakemeyer, G.; and Levesque, H. J. 2004. A logic of limited belief for reasoning with disjunctive information. In *Proc. KR-04*, 587–597.

[Reiter 2001b] Reiter, R. 2001b. On knowledge-based programming with sensing in the situation calculus. *ACM Trans. Comput. Log.* 2(4):433–457.

[Sardina 2001] Sardina, S. 2001. Local conditional high-level robot programs. In *Proc. LPAR-01*.