Learning efficient logical robot strategies involving composable objects

Andrew Cropper and Stephen H. Muggleton

Imperial College London
Initial state

Final state

\[\text{pos(robot,1/1), pos(ball,1/1)}\]  \[\text{pos(robot,3/3), pos(ball,3/3)}\]
move(X,Y):- p3(X,Z), p3(Z,Y).
p3(X,Y):- p2(X,Z), drop(Z,Y).
p2(X,Y):- grab(X,Z), p1(Z,Y).
p1(X,Y):- north(X,Z), east(Z,Y).

move(X,Y):- p3(X,Z), drop(Z,Y).
p3(X,Y):- grab(X,Z), p2(Z,Y).
p2(X,Y):- p1(X,Z), p1(Z,Y).
p1(X,Y):- north(X,Z), east(Z,Y).
Inefficient solution

\[
\text{move}(X,Y) :\leftarrow p3(X,Z), p3(Z,Y).
\]
\[
p3(X,Y) :\leftarrow p2(X,Z), \text{drop}(Z,Y).
\]
\[
p2(X,Y) :\leftarrow \text{grab}(X,Z), p1(Z,Y).
\]
\[
p1(X,Y) :\leftarrow \text{north}(X,Z), \text{east}(Z,Y).
\]

Efficient solution

\[
\text{move}(X,Y) :\leftarrow p3(X,Z), \text{drop}(Z,Y).
\]
\[
p3(X,Y) :\leftarrow \text{grab}(X,Z), p2(Z,Y).
\]
\[
p2(X,Y) :\leftarrow p1(X,Z), p1(Z,Y).
\]
\[
p1(X,Y) :\leftarrow \text{north}(X,Z), \text{east}(Z,Y).
\]
Inefficient solution

Efficient solution

<table>
<thead>
<tr>
<th>Action</th>
<th>drop</th>
<th>grab</th>
<th>north</th>
<th>east</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Iterative descent

1. find first consistent solution with minimal textual complexity
2. repeat until convergence:
   A. calculate resource complexity of learned solution
   B. learn new solution with a maximum resource bound that is smaller than the resource complexity of the previous solution

Theorem: guaranteed to converge to minimal resource complexity hypothesis
Implementation of meta-interpretive learning*, a form of inductive logic programming based on a Prolog meta-interpreter, which supports predicate invention and the learning of recursive theories

Actions: go_to_bottom/2, go_to_top/2, find_next_sender/2, find_next_recipient/2, take_letter/2, give_letter/2, bag_letter/2
Mean resource complexity

---

**Metagol**

- **Metagol**
- **Metagol**

**Composable tight bound**

\[ 2(n + d) \]

**Non-composable tight bound**

\[ n(2d + 2) \]

---

**No. objects**

- 2
- 4
- 6
- 8
- 10

---

**Mean resource complexity**

- 0
- 200
- 400
- 600
- 800
- 1000

---

**Graph**: Line graph showing the relationship between the number of objects and mean resource complexity, with error bars indicating variability.
Initial state
[2,5,6,1,9,7,3,4,8]

Final state
[1,2,3,4,5,6,7,8,9]

Actions:
- comp_adjacent/2
- decrement_end/2
- go_to_start/2
- pick_up_left/2
- split/2
- combine/2
List length

Mean resource complexity

- Metagol_O
- Metagol_D
- Tight bound n log n
- Tight bound n(n-1)/2

Graph showing the mean resource complexity for list lengths from 20 to 100.
Conclusions

• Suggests that we can build delivery and sorting robots which learn resource efficient strategies from examples

Future work

• Optimise the iterative descent search procedure
• Generalise to a broader class of logic programs
Thank you