Extended Goal Recognition Design with First-Order Computation Tree Logic

Tsz-Chiu Au
chiu@unist.ac.kr

Ulsan National Institute of Science and Technology (UNIST)
South Korea
Goal Recognition Design (GRD)

• **Goal recognition** – an observer infers the goal of an agent from a sequence of observations of agents’ actions.

• **Goal recognition design**\(^1\) – modify an environment to help observers to recognize the goal of an agent.

\(^1\)Keren et al. Goal Recognition Design. AAAI 2014
Worst Case Distinctiveness (WCD)

- **Worst case distinctiveness** – a popular objective function for GRD
  - The highest number of observations that an observer needs to observe *before* it can be certain of the agent’s goal in the worst case.

Before redesign, WCD = 4
Minimizing WCD

- GRD aims to find a sequence of modifications to an environment in order to minimize the WCD.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Before redesign, WCD = 4

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After redesign, WCD = 0
Weakness of WCD

• When there exist two paths to two different goals but share a long common prefix, it is difficult to reduce the WCD even if other goals can be recognized easily.

![Diagram of WCD](image)
Goal Condition

- Instead of asking exactly which goal an agent aims for, an observer asks whether the agent aims for a **goal condition**
  - e.g., one of any two goals but not any other goals
  - It is weaker than recognizing a goal exactly, but still useful.

\[ g_1 g_2 \quad g_3 \quad g_4 \quad g_5 \quad g_6 \quad g_7 \quad g_8 \]

WCD_1 \[\rightarrow\] WCD_2

\[ s_0 \]
Extended Goal Recognition Design (EGRD)

- **Goal sequence** – an agent can aim for more than one goal.
Our Contributions

• A framework of extended goal recognition design
  » Use first-order computation tree logic (FO-CTL) to express goal conditions
  » The definition of WCD based on goal conditions.
  » Finding WCD by model checking

• A graphical representation of FO-CTL sentences for extended goal recognition
  » A translation algorithm from goal query graphs to FO-CTL sentences

• The EGRD search algorithm
  » A caching mechanism for speeding up the search algorithm
First-Order Computation Tree Logic (FO-CTL)

- FO-CTL = first-order logic with path quantifiers (A and E) and temporal operators (F, G, X, and U)
  » A ψ means ψ holds on all paths
  » E ψ means ψ holds on at least one path where ψ is either
    - F φ means φ eventually has to hold
    - G φ means φ always holds
    - X φ means φ holds at the next state
    - (φ₁ U φ₂) means φ₁ has to hold at least until φ₂ holds
  » We assume no function symbol, and there is only one predicate symbol Goal(g)
    - The predicate symbol Goal will be omitted.
- For example,
  \[\phi_{unique} = \exists x \{AF (x \land \forall x'[(x' \neq x) \Rightarrow AG \neg x'])\}\]
  which checks whether a goal \(g\) exists such that an agent must eventually achieve \(g\) while the agent will not achieve any other goal after achieving \(g\).
The WCD of a Goal Condition

- The WCD of a goal condition $\phi$ is

$$\left\{ \max_{p \in P_{leg}} \min_{s_i \in S_\phi(p)} [dist(s_0, s_i)] \right\} - 1$$

where

- $P_{leg}$ is the set of all legal paths
- $S_\phi(p)$ is the set of states on a legal path $p \in P_{leg}$ such that $\phi$ is true in these states
- $dist(s_0, s_i)$ is the distance between $s_i$ and the initial state $s_0$
Finding WCD by Model Checking

- Given a goal condition $\phi$, evaluate $\text{EF } \phi$ at the initial state $s_0$ by model checking.
- For example,
  \[
  \text{EF } \phi_{\text{unique}} = \text{EF } \exists x \{ \text{AF } (x \land \forall x'[(x' \neq x) \Rightarrow \text{AG } \neg x']) \}
  \]
- Attach a cost function to each node in a sentence. For example, the cost function of Node 1 is $\max$, and the cost function of Node 2 is $\text{dist}(s_0, s_i)$.
- The costs, along with the truth values, are propagated to the root node during the execution of the model checking algorithm.
Goal Query Graph (GQG)

- **Goal query graph** – a graphical representation of goal conditions

  For example, the GQG of $\exists x_1 \exists x_2 [AF [x_1] \land AX AF x_2]$ is

![Diagram of GQG example]

- Directed acyclic graph:
  - 3 vertex types: state vertices, nil vertices, and choice vertices
  - 5 edge types: AP edges, EP edges, AX edges, EX edges, and choice edges

- State vertices can have **state conditions** (e.g., $(x_2 \lor \neg x_1)$)
- AP edges and EP edges can have **edge conditions** (e.g., $XA = \forall x [\neg x]$)
- Choice vertices and choice edges:
Translating GQGs into FO-CTL Sentences

- A depth-first search in the goal query graph.
  - The FO-CTL sentence is constructed in a bottom-up fashion.
  - Each vertex/edge type has its own rule for translation.
  - Insert existential qualifiers for the free variables.
  - Optimization techniques for shortening the sentence.

- Running time: $O(|V| + |E|)$
The EGRD Search Algorithm with Caches

• A depth-limited, best-first search
  » Store unexpanded transition systems in an open list.
  » Repeat the following steps until the open list is empty or the time limit
    ▪ Remove a transition system $M$ from an open list
    ▪ Use a model checking algorithm to evaluate $M$ and compute WCD.
    ▪ If the evaluation is true and the WCD is lower than the best WCD
      › set this transition system as the best solution.
    ▪ If the search depth of $M$ is less than a threshold
      › apply modifications to $M$ to insert the generated models into the open list.
  » Return the best solution
The Caching Mechanism

- **Caching mechanism** – store the evaluation results of the recursive calls in the model checking algorithm in a cache.
  - Reuse the results in subsequent runs of the model checking algorithm.
  - Need a succinct encoding of transition systems’ states.
Empirical Evaluation

- The goal query graph:

- The running times increase as the number of goals increases.
- The caching mechanism can greatly reduce the running time of the EGRD search algorithm.

<table>
<thead>
<tr>
<th>Table 1: Execution times (in sec.) vs. the number of goals.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>LOGISTICS</td>
</tr>
<tr>
<td>DEPOTS</td>
</tr>
<tr>
<td>GRID</td>
</tr>
<tr>
<td>BLOCK-WORLD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2: Execution times (in sec.) with and without cache.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>LOGISTICS</td>
</tr>
<tr>
<td>DEPOTS</td>
</tr>
<tr>
<td>GRID</td>
</tr>
<tr>
<td>BLOCK-WORLD</td>
</tr>
</tbody>
</table>
Summary and Future Work

• Extended goal recognition design
  » Weaker goal conditions
  » Agents can aim for a sequence of goals

• Express goal conditions in FO-CTL
  » Finding WCDs by model checking
  » Goal query graphs

• Caching mechanism to speed up the EGRD search

• Future work: Partial observability

• The source code with additional examples:
  https://github.com/chiuau/AAAI22-egrd
Thank you!