# Adding AND/OR Plan Library Support to CSQ Algorithm

Gal Varon, Advised by Prof. Gal A. Kaminka

Bar-Ilan University

Research Summary for Robotic Workshop Class

## *Introduction*

Plan recognition is the process of inferring another agent's plans, based on observations of its interaction with its environment. In general, plan recognition relies on a *plan library* of plans potentially executed by the observed agent. Typically, the plan library is composed of top-level plans that are hierarchically decomposed. The recognizer matches observations to specific plan steps in the library. It may then infer answers to plan recognition queries, such as the currently selected top-level plans (*current state* query - CSQ), or the ordered sequence of (completed or interrupted) selected plans (*state history* query). This paper focuses on improving the CSQ algorithm offered by Dorit Avrahami-Zilberbrand and Gal A. Kaminka [1]. We show how to modify the algorithm to support AND/OR graph plan library representation. Then we prove the algorithms equivalence. Finally, we will discuss the repercussions of our discoveries during this research.

## *Definitions and Notation*

The Plan Library – The plan library we present is based on the plan library presented in [1]: A single-root directed acyclic graph, where vertices denote plan steps and edges can be of two types: Hierarchical edges and decomposition edges. In this paper, each vertex belongs to either OR vertices group (denoted $V_O$) or AND vertices group (denoted $V_A$).

We also add the following definitions and notation:

AND/OR graph G, is defined as follows:
$G = \langle V_O, V_A, E_H, E_S, R \rangle$

$V_O$ = OR vertices, $V_A$ = AND vertices
$E_H$ = Hierarchical edges, $E_S$ = Decomposition edges

R = Root vertex, $R \in V_O \cap V_A$
$V_O \cap V_A = \emptyset$, $E_H \cap E_S = \emptyset$

We define the following operators:

$\forall\, v \subseteq V_O \cup V_A$:

Parent(v) = {u | (u, v) is a hierarchical edge}
Note that Parent(v) is a singleton function.

$\forall\, v \subseteq V_O$:

Prev(v) = {w | (w, v) is a sequential edge}

$\forall\, v \subseteq V_A$:

Prev(v) is implicitly defined – can be each of its siblings or nothing.
Sibling(v) = {w | {u, w} is a hierarchical edge and u $\in$ Parent(v)}

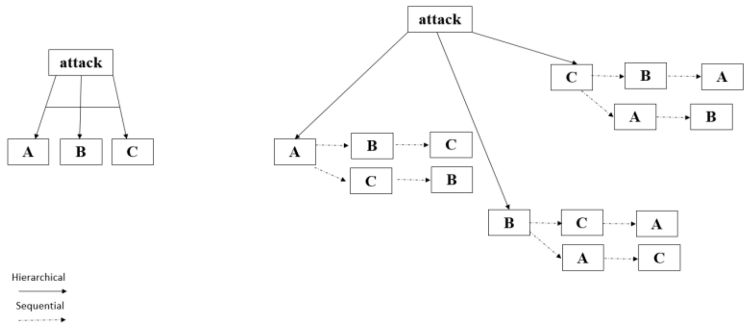

*Figure1: Equivalent AND/OR and Normal Plan Library Representation*

## *CSQ Algorithm With AND/OR Plan Library Support*

The following algorithm is a modified version of the **PropagateUp** algorithm from [1]. Using this version of PropagateUp in the CSQ algorithm, allows handling both regular and AND/OR graph representation of the Plan Library.

**Algorithm 1** PropagateUp(Node $w$, Plan Library $g$, Time-stamp $t$)

```
1: $Tagged \leftarrow \emptyset$
2: $propagateUpSuccess \leftarrow true$
3: $v \leftarrow w$
4: while $v \neq root(g) \wedge propagateUpSuccess \wedge \neg tagged(v, t)$ do
5:  if $tagged(Parent(v), t) \vee features(parent(v)) = \emptyset$ then
6:  if (Parent(v) $\in V_O \wedge (tagged(v, t-1) \vee$
       $ExistsPreviousSeqEdgeTaggedWith(v, t-1) \vee$
       $NoSeqEdges(v))) \vee$ Parent(v) $\in V_A$ then
7:        $tag(v, t)$
8:        $Tagged \leftarrow Tagged \cup \{v\}$
9:        $v \leftarrow parent(v)$
10:       $propagateUpSuccess \leftarrow true$
11:     else
12:       $propagateUpSuccess \leftarrow false$
13:   else
14:     $propagateUpSuccess \leftarrow false$
15: if $\neg propagateUpSuccess$ then
16:   for all $a \in Tagged$ do
17:   $delete\_tag(a, t)$
```

The simple modification in line 6, separates the condition into two: 1. If Parent(v) is an OR node – the condition remains the same as in the original algorithm. 2. If Parent(v) is an AND node – the condition of the original algorithm is always true (see claim 1), we continue the propagation process.

## *Algorithm Equivalence*

Algorithm 1. differs from the original algorithm only in the case of an AND vertex. We will prove that the algorithms are equivalent by proving the following claim:

*Claim 1.* Parent(v) $\in V_A$ => tagged (v, t - 1) V ExistsPreviousSeqEdgeTaggedWith (v, t -1) V NoSeqEdges(v) $\equiv$True

Proof: Parent(v) $\in V_A$, by definition of Prev(v) for v $\in V_A$, there is always a case in which NoSeqEdges(v) is true. Therefore, we can assume: Parent(v) $\in V_A$ => NoSeqEdges(v) $\equiv$ True. Under this assumption it holds that: Parent(v) $\in V_A$ => tagged (v, t - 1) V ExistsPreviousSeqEdgeTaggedWith (v, t -1) V NoSeqEdges(v) $\equiv$ tagged (v, t - 1) V ExistsPreviousSeqEdgeTaggedWith (v, t -1) V **True** $\equiv$ True.

Claim 1. shows that if Parent(v) is an AND vertex the condition always evaluates as true. If Parent(v) is not an AND vertex, we follow the logic of the original algorithm. Therefore, the condition is equivalent to the original condition. Since it is the only modification – the algorithms are equivalent.

## *Results Meaning*

This algorithm modification and the result of claim 1., raised an interesting issue that was overlooked in the original research. If there is a first child for each possible observation and the parent is tagged with time stamp t, or has no observable features – CSQ will except every possible observation. Meaning that the CSQ algorithm can be tricked by an agent that changes plans rapidly, or only does operations that are considered plan beginnings, not producing any valuable prediction.

## References

[1] Dorit Avrahami-Zilberbrand and Gal A. Kaminka, 2005, **Fast and Complete Symbolic Plan Recognition**