# BT Expansion: supplementary material

This supplementary material contains the graphical demonstration of the BT synthesis in the simulation, the detailed description of the test set, and the discussion of the complexity.

## 1. BT GENERATION STEPS IN THE CASE STUDY

The BT generation steps for our method (BT expansion) and the baseline approach in the mobile manipulator simulation are shown graphically below.

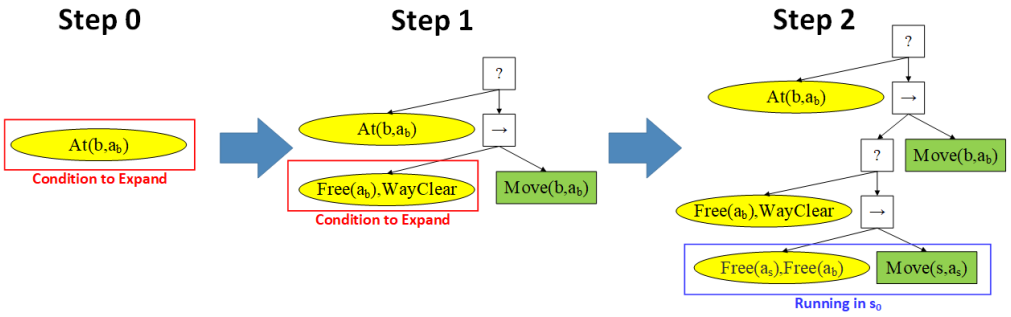


**Fig. S1.** The BT generating steps by BT expansion.

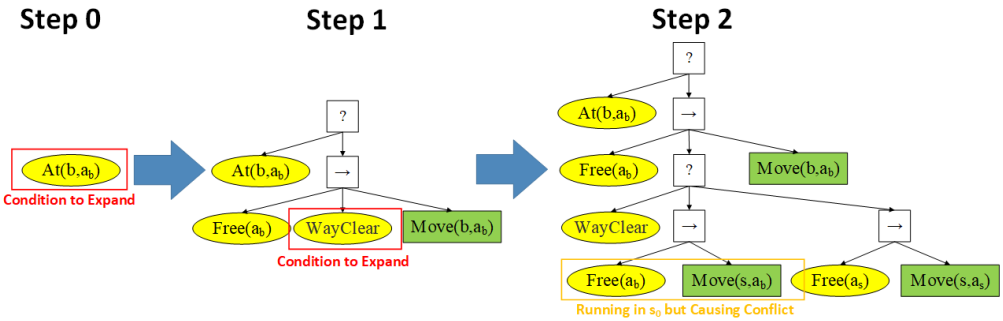


**Fig. S2.** The BT generating steps by the baseline approach.

## 2. TEST SET GENERATION

Our paper tests and compares our algorithm and the baseline in 10 test sets generated with different characteristics (cf. Table 2 in the paper). Each sets contains 1000 planning problems randomly generated. The parameters for generating a set includes *literals*, *distance*, *iterations*. *literals* is the number of (positive) literals to represent a state, e.g. 10 (positive) literals can represent a total number of $2^{10}$ states. The generation procedure of one problem is described as follows.

**Step 1**: Generate the initial state $s_0$ as a set of literals, where each literal has 50% chance to be selected.

**Step 2**: Generate an action that can be performed from the initial state as three sets, i.e. *pre*, *add* and *del*. For literals in the state, it has 50% chance to be selected in *pre*, and then 50% chance to be selected in *del*. For other literals, it has 50% chance to be selected in *add*, and otherwise 50% chance to be selected in *del*.

**Step 3**: Calculate the successor state by $s_{next} = s_{current} \cup add(a) \backslash del(a)$. According to the *distance*, iteratively generate a path with the distance, and the last state will be the goal for the problem. For example, with $distance = 10$, there will be 10 actions randomly generated and the final state is the goal.

**Step 4**: Randomly generate an action from a randomly sampled existing state to transfer to a (perhaps) new state. The generation performs for *iterations* steps.

Next, we introduce the effects of the parameters, i.e. *literals*, *distance*, *iterations*, to the generated problems (cf. Table 2 in the paper), with the experimental results.

*Literals*. More literals means exponentially larger state spaces, therefore the generated states are less likely to repeat, and each state tends to have exactly one actions resulting in the state (cf. cases 3/4/8/9 in Table 2 in the paper). More literals also means larger size of conditions like *pre*, *goal*, which can cause significant increase of tree size in the baseline method.

*Distance*. More distance means more difficulty to reach the goal from the start (or reach the start from the goal). The resulting BT size will increase since more actions are needed (cf. cases 0 vs 5, 1 vs 6, 2 vs 7, 3 vs 8, 4 vs 9). With a rough action selection, the baseline will not only add actions that destroy correctness, but also result in a much larger tree size.

*Iterations*. More iterations means more randomly generated states and actions in the problem. It indirectly affects the size of generated BTs by the number of actions connecting a state. If the actions per state do increase (cf. cases 0 vs 1 vs 2, 5 vs 6 vs 7), the generated BT size tends to increase. If the state space is very large and the actions per state remains one-to-one (cf. cases 3 vs 4, 8 vs 9), the generated BT size is unaffected.

## 3. COMPLEXITY

The complexity of the BT expansion algorithm varies with the implementation and problem settings. It is polynomial to the system size ($|A| + |S|$), just like other state space traversal algorithms. We start from the most efficient case to the crudest case.

In most planning problems, it is assumed that operating with an action requires constant time, i.e. the size of sets $pre, add, del$ for any actions are bounded by some constant. And the actions applicable in a particular state are also directly accessible though a list, whose size is bounded by $b$, often called the maximum branching factor. Then the time complexity for one-step expansion (Algorithm 1) is $O(b)$. If such a list is not given and all actions need to be checked individually for selection, the time complexity for one-step expansion (Algorithm 1) is $O(|A|)$.

As for the iterative expansion in Algorithm 2, the worst case is that the loop enumerates all the states for $|S|$ times. Inside the loop, the complexity varies according to the one-step expansion. Therefore the time complexity for BT expansion is $O(b|S|)$ or $O(|A||S|)$ accordingly.

In the crudest case, operating with an action takes $O(log|S|)$ time because the total number of literals is $log|S|$ and all actions need to be checked individually in each expansion step, the worst time complexity for BT expansion is $O(|A||S|log|S|)$.