levels of nodes on the game tree, where the leaves bring the final known (or considered) game state.

The minimax theorem states:

For every two-person, zero-sum game there is a mixed strategy for each player, such that the expected payoff for both is the same value V when the players use these strategies. Furthermore, V is the ***best*** *payoff each can expect to receive from a play of the game; that is, these mixed strategies are the optimal strategies for the two players.*

This theorem was established by John von Neumann, who is quoted as saying "As far as I can see, there could be no theory of games … without that theorem … I thought there was nothing worth publishing until the *Minimax Theorem* was proved" [2].

A simple example of minimax can be observed by building a game tree of the tic-tac-toe game. The tic-tac-toe game is a simple game which can end by the first player wining, the second player wining or a tie. There are nine positions for each of the players in which at each turn the player puts X or O sign. If the player has three adjacent signs in a row, column or the two diagonals he or she wins. This game has limited number of position and it is well suited for building the whole game tree. The leaves of this tree will be final positions in the game. A heuristics evaluation function will also need to be written to evaluate the value of each node along the way.

3. AI Planning for building Game Trees

3.1.1 AI Planning

AI Planning also referred as Automated Planning and Scheduling is a branch of Artificial Intelligence that focuses on finding strategies or sequences of actions that reach a predefined goal [3]. Typical execution of AI Planning algorithms is by intelligent agents, autonomous robots and unmanned vehicles. Opposed to classical control or classification AI Planning results with complex solutions that are derived from multidimensional space.

AI Planning algorithms are also common in the video game development. They solve broad range of problems from path finding to action planning. A typical planner takes three inputs: a description of the initial state of the world, a description of the desired goal, and a set of possible actions. Some efforts for incorporating planning techniques for building game trees have also shown up, similar to the approach explored in this effort. In addition Cased Based Reasoning [4] techniques are also gathering popularity in developing strategies based in prior knowledge about the problems in the games. One of the benefits from Hierarchical Task Network (HTN) [5] planning is the possibility to build Game Trees based on HTN plans; this method is described in the following section.

3.2 Game Trees with AI Planning

An adaptation of the HTN planning can be used to build much smaller and more efficient game trees. This idea has already been implemented in the Bridge Baron a computer program for the game of Contact Bridge [6].

Computer programs based on Game Tree search techniques are now as good as or better than humans in many games like Chess [7] and checkers [8], but there are some difficulties in building a game tree for games that have imperfect information and added uncertainty like card or games with dice. The main problem is the enormous number of possibilities that the player can choose from in making his move. In addition some of the moves are accompanied with probabilities based on the random elements in the games. The number of possible moves exponentially grows with each move so the depth of the search has to be very limited to accommodate for the memory limitations.

The basic idea behind using HTN for building game trees is that the HTN provides the means of expressing high level goals and describing strategies how to reach those goals. These goals may be decomposed in goals at lower level called sub-goals. This approach closely resembles the way a human player usually addresses a complex problem. It is also good for domains where classical search for solution is not feasible due to the vastness of the problem domain or uncertainties.

3.2.1 Hierarchical Task Networks

The Hierarchical Task Network, or HTN, is an approach to automated planning in which the dependency among actions can be given in the form of networks [9] [Figure 1].

A simple task network (or just a task network for short) is an acyclic digraph $w = (U, E)$ in which U is the node set, E is the edge set, and each node $u \in U$ contains a task t_u. The edges of w define a partial ordering of U. If the partial ordering is total, then we say that w is totally ordered, in which case w can be written as a sequence of tasks $w = \langle t_1, t_2, \ldots, t_k \rangle$.

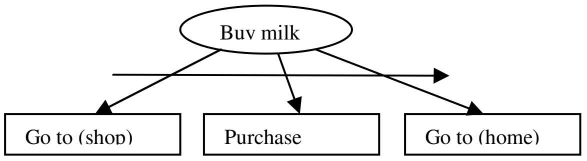

Figure 1: Simple Hierarchical Task Network

A Simple Task Network (STN) method is a 4-tuple of its name, task, precondition and a task network. The name of the method lets us refer unambiguously to substitution instances of the method, without having to write the preconditions and effects explicitly. The task tells what kind of task can be applied if the preconditions are met. The preconditions specify the conditions that the current state needs to satisfy in order for the method to be applied. And the network defines the specific subtasks to accomplish in order to accomplish the task.

A method is relevant for a task if the current state satisfies the preconditions of a method that implements that task. This task can be then substituted with the instance of the method. The substitution is basically giving the method network as a solution for the task.

If there is a task “Go home” and the distance to home is 3km [Figure 2] and there exists a method walk-to and this method has a precondition that the distance is less than 5km, then a substation to the task “Go home” can be made with this method instance.

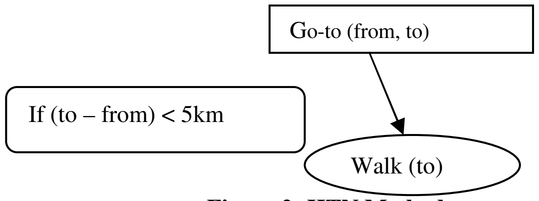

Figure 2: HTN Method