If the distance is larger than 5km another method instance needs to be substituted [Figure 3].

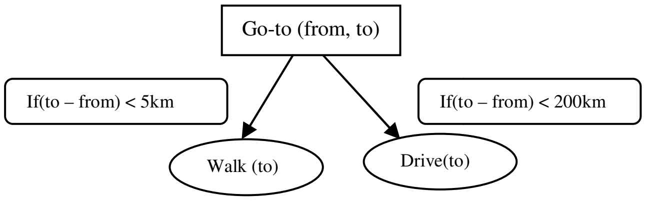

Figure 3: HTN Method 2

An STN planning domain is a set of operations O and a set of methods M. A STN planning problem is a 4-tuple of the initial state S_0, the task network w called initial task network and the STN domain. A plan $\pi = \langle a_1, \dots, a_n \rangle$ is a solution for a planning problem if there is a way to decompose w into π if π is executable and each decomposition is applicable in the appropriate state of the world. The algorithm that is capable to decompose these networks into plans is called Total-forward-decomposition (TFD) [9] or Partial-forward-decomposition (PFD). However there are cases where one does not want to use a forward-decomposition procedure. HTN planning is generalization of STN planning that gives the planning procedure more freedom about how to construct the task networks.

In order to provide this freedom, a bookkeeping mechanism is needed to represent constraints that the planning algorithm has not yet enforced. The bookkeeping is done by representing the unenforced constraints explicitly in the task network.

The HTN generalizes the definition of a task network in STN. A task network is the pair $w = (U, C)$ where U is a set of task nodes and C is a set of constraints. Each constraint in C specifies a requirement that must be satisfied by every plan that is a solution to a planning problem.

The definition of a method in HTN also generalizes the definition used in STN planning. A HTN plan is a 4-tuple of name, task, subtasks, and constraints. The subtasks and the constraints form the task network. The HTN planning domains are identical to STN planning domains except they use HTN methods instead of STN methods.

Compared to classical planners the primary advantage of HTN planners is their sophisticated knowledge representation and reasoning capabilities. They can represent and solve a variety of non-classical planning problems; with a good set of HTNs to guide them, they can solve classical planning problems orders of magnitude more quickly than classical or neoclassical planners. The primary disadvantage of HTN is the need of the domain author to write not only a set of planning operators but also a set of methods.

3.2.2 HTN Planning in building Game Trees

For a HTN planning algorithm to be adapted to build game trees we need to define the domain (set of HTN methods and operators) which is the domain of the game. This is in some sense a knowledge representation of the rules of the game, the game environments and possible strategies of game play.

In this domain the game rules as well as known strategies to tackle specific task are defined. The implementation of Game Tree building with HTN is called Tignum2 [9]. This implementation uses a procedure similar to forward-decomposition, but adapted to build up a game tree rather than a plan. The branches of the game tree represent moves generated by the methods. Tignum2 applies all methods applicable to a given state of the world to produce new states of the world and continues recursively until there are no applicable methods that have not already been applied to the appropriate state of the world.

In the task network generated by Tignum2, the order in which the actions will occur is determined by the total-ordering constraints. By listing the actions in the order they will occur, the task network can be “serialized” into a game tree [Figure 4] [Figure 5].

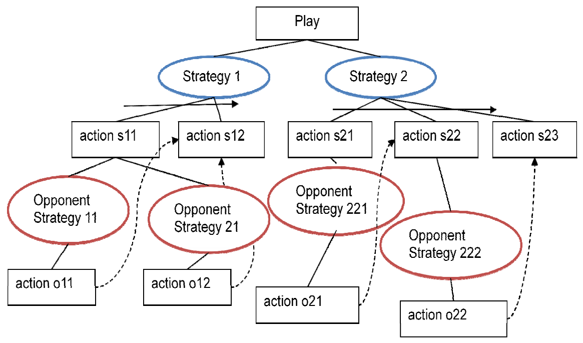

Figure 4: HTN to Game Tree Algorithm

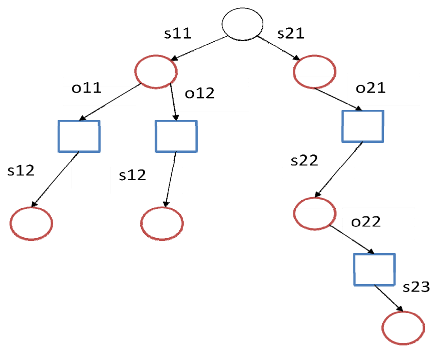

Figure 5: Game Tree built from HTN

4. Case Based Reasoning in Game Strategies

4.1 Case Based Reasoning

Case-based reasoning (CBR) is a well established subfield of Artificial Intelligence (AI), both as a mean for addressing AI problems and as a basis for standalone AI technology.

Case-based reasoning is a paradigm for combining problem-solving and learning that has became one of the most successful applied subfield of AI of recent years. CBR is based on the intuition that problems tend to recur. It means that new problems are often similar to previously encountered problems and, therefore, that past solutions may be of use in the current situation [10].

CBR is particularly applicable to problems where earlier cases are available, even when the domain is not understood well enough for a deep domain model. Helpdesks, diagnosis or classification systems have been the most successful areas of application, e.g., to determine a fault or diagnostic an illness from observed attributes, or to determine whether or not a certain treatment or repair is necessary given a set of past solved cases [11].