Research Review

the STRIPS (Stanford Research Institute Problem Solver) automatic problem solver and A* search methods started in 1969. These were to help developed an automated robot called Shakey. This robot had multiple systems to guide it around obstacles such as computer vision and Action planners.

The STRIPS is a way of tackling the classical planning problem. The classical planning problem is a way of looking at the wold in static states that can be transformed into another static state. The transformed can only be achieved by a single agent that is able to perform a set of given actions. With this you can search through a state space, with different search algorithms, to find an optimal path to a given goal. This idea of searching though the state space would become the center piece of the STRIPS design.

For STRIPS it is based on an assumption that an action only affects the parts of the world that are specifically mentioned in the deletion and addition lists. They first started off with GPS as their initial problem solving architecture but needed to define meaningful differences between the predicate sentences and a goal situation predicate is True.

The next task for them was to figure out what operators would be relevant and reducing to the difference. This in the end did not work since Individual states and true goal states. To get around this problem, they the made it so there were operators that could assert clauses that can get resolved with the clauses at the leaf nodes. With this where the idea of going through a list of actions that get a goal state came up with out using difference tables. The over all structure of this idea was

> If [precondition]
> Then (retract [deletion]) , (assert [additions])

This was still simple compared to systems of today since STRIPS had not addressed issues of scope and complexity of problems. For example of this would be multi agent systems as with STRIPS, the assumption is that one agent effects the world; but most people in the field felt that this was a viable framework to be improved on.

Another big problem with panning in general is sometimes what actually happens difference from the outcome of the planned outcome due to external factors. So to counter this there needed to be a check for whether or not the Shakey was still on course or if the planning needed to be recalculated. The first way was to just check if a kernel, which was a set of sentences. It would regress back through the state spaces until one held true and all those sentences were put into the kernel to be checked again. This was essentially a sync method.

The second way of solving synchronization is checking if the state is true if it is, Shakey will move to the next state if not it will recalculate where to go. The problem with this is that it can do moves follow up moves which are redundant; It also wont redo moves that might be good to do.

To solve both of these issues what they did was check if the final goal state and check the previous step to that is true and the previous to that. For example if E was the final step. It would check if E was true. If no then check if D is true, if no then check C and so on until they get a true statement. The true state would be the state Shakey is in. If no stat was true then Shakey would re-plan it roughing.