You are employed by a transport company that delivers a variety of parcels using a variety of forms of transport to carry the parcels to varying destinations.

The transport company already has a way of doing things that has been human-designed. Your job is to improve and hopefully optimise the existing system, to make it more efficient, cost effective, ecological etc.

You've just been studying automated theorem proving using Resolution so you decide to use a logical system with an inference engine to reason about your problem. You start putting together a knowledge base of all the information you can get about your problem domain,

HOWEVER A NUMBER OF PROBLEMS ARISE
Reasoning versus projecting

Your first problem is that in the inference systems you are accustomed to using with Propositional logic you need to have a different copy of any action description for each time at which the action might be executed.

What an inference system does without reference to the effects of actions at different times is inferring information about an agent or KB's present state. This can be contrasted with computing the future states that arise because of an agent's actions. What you want to do is compute future states

So you change your inference system to use Predicate logic where you can quantify over time.

∀ t (some-outcome is the result of doing action at t)

The states that result from taking actions in this way are called situations. The approach to representing the effects of actions in Predicate Logic is called the situation calculus
The situation calculus

The *situation calculus* allows you to *project* the effects of actions into the future to see what state of the world will result.

It also allows you to choose a state of the world as your goal and then find a sequence of actions that achieves that goal state. This is the *planning* task.

In the simplest version of the situation calculus, each action is described by two axioms:

- A possibility axiom that specifies the conditions under which the action can be taken.
- An effect axiom that specifies what effect results from the action taken, that is, it describes how the state of the world should change when that action is taken.
Unforeseen side-effects mar the performance of your planning system

When you try and implement some of the plans that are derived from your new situation calculus inference system you find that the plans fail because of unforeseen side-effects.

For example:

when you fill the vehicles to their full capacity, they need more fuel and cannot reach the furthest destinations,

when you find the shortest route that only needs one truck rather than two trucks, it means the journey is so long that the driver is legally required to rest over-night

when you decrease the delivery time for fruit they have not ripened yet and then need to be stored further rather than immediately sold
Effect axioms and frame axioms in situation calculus

Unforeseen side-effects occur because actions change the world in more ways than are recorded in the *effect axioms*

How to account for the side-effects of taking actions?

One solution that allows you to consider all possible side-effects for all your actions is to use *frame axioms*

However, the inclusion of frame axioms for all actions and effect axioms is computationally costly in terms of the space needed for them in the knowledge base and the time needed to check them,

The problem of considering all side-effects through frame axioms, when a great majority of possible side-effects will not occur, is known as the *Frame Problem*
unforeseen side-effects in planning can cause what is known as the frame problem

Dennett illustrates the frame problem with a story:

“Once upon a time there was a robot, named R1 by its creators. Its only task was to fend for itself. One day its designers arranged for it to learn that its spare battery, its precious energy supply, was locked in a room with a time bomb set to go off soon. R1 located the room, and the key to the door, and formulated a plan to rescue its battery. There was a wagon in the room, and the battery was on the wagon, and R1 hypothesized that a certain action which it called PULLOUT (Wagon, Room, t) would result in the battery being removed from the room. Straightaway it acted, and did succeed in getting the battery out of the room before the bomb went off. Unfortunately the bomb was also on the wagon. R1 knew that the bomb was on the wagon in the room, but didn't realise that pulling the wagon would bring the bomb along with the battery. Poor R1 has missed the obvious implication of its planned act.

Back to the drawing board ....
Back to the drawing board. “The solution is obvious,” said the designers. “Our next robot must be made to recognise not just the intended implications of its acts, but also the implications about their side-effects, by deducing these implications from the descriptions it uses in formulating its plans.” They called the next model, the robot-deducer, R1D1. They placed R1D1 in much the same predicament that R1 had succumbed to, and as it too hit upon the idea of PULLOUT (Wagon, Room, t) it began, as designed, to consider the implications of such as course of action. It had just finished deducing that pulling the wagon out of the room would not change the colour of the room's walls, and was embarking on a proof of the further implication that pulling the wagon out would cause its wheels to turn more revolutions than there were wheels on the wagon - when the bomb exploded.

Back to the drawing board .......
Back to the drawing board. “We must teach it the difference between relevant implications and irrelevant implications” said the designers, “and teach it to ignore the irrelevant ones.” So they developed a method of tagging implications as either relevant or irrelevant to the project in hand, and installed the method in their next model, the robot-relevant-deductor, or R2D1 for short. When they subjected R2D1 to the test that has so unequivocally selected its ancestors for extinction, they were surprised to see it sitting, Hamlet-like outside the room containing the ticking bomb, the native hue of its resolution sicklied o'er with the pale cast of thought, as Shakespeare (and more recently Fodor) has aptly put it. “Do something!” they yelled at it. “I am” it retorted. “I'm busily ignoring some thousand of implications I have determined to be irrelevant. Just as soon as I find an irrelevant implication, I put it on the list of those I must ignore, and ...” the bomb went off.
Dennett's description of the THE FRAME PROBLEM contd

Dennett concludes:

“All these robots suffer from the frame problem. If there is ever to be a robot with the fabled perspicacity and real-time adroitness of R2D2, robot designers must solve the frame problem”

What 'work-arounds' might there be for the Frame Problem?

- Behaviour based robotics
- Re-inforcement learning
- Evolutionary learning
- Problem specific heuristics
How important is the Frame Problem?

The Frame Problem has a narrow version that applies to logic and a broader version that is discussed by philosophers.

We have been looking at the logic Frame Problem

We will also consider the broader epistemological Frame Problem as a DIGRESSION from our narrower consideration of planning

How can you design a system that reliably ignores what it ought to ignore under a wide variety of different circumstances in a complex action environment

McCarthy presents this as the qualification problem, and illustrates it with the famous puzzle of the Missionaries and the Cannibals
McCarty and Missionaries and Cannibals

The Frame Problem in its broader philosophical sense is linked to the issue of what to include in your considerations and what to ignore.

“Three missionaries and three cannibals come to a river. A rowboat that seats two is available. If the cannibals ever outnumber the missionaries on either bank of the river, the missionaries will be eaten. How shall they cross the river?

Obviously the puzzler is expected to devise a strategy of rowing the boat back and forth that gets them all across and avoids disaster...

Imagine giving someone the problem, and after he puzzles for a while, he suggests going upstream half a mile and crossing on a bridge. 'What bridge?' you say. 'No bridge is mentioned in the statement of the problem' and this dunce replies 'Well, they don't say there isn't a bridge.' You look at the English and even at the translation of the English into first order logic, and you must admit that 'they don't say' there is no bridge
So you modify the problem to exclude bridges and pose it again, and the dunce proposes a helicopter, and after you exclude that, he proposes a winged horse or that the others hang onto the outside of the boat while the two row.

You now see that while a dunce, he is an inventive dunce. Despairing of getting him to accept the problem in the proper puzzlers spirit, you tell him the solution. To you further annoyance, he attacks your solution on the grounds that the boat might have a leak or lack oars. After you rectify that omission from the statement of the problem, he suggests that a sea monster may swim up the river and may swallow the boat.”

There have been many suggestions from within classical AI on how to resolve these kinds of problems, including non-monotonic reasoning, scripts and frames.

However, none of these solutions would seem to the same level of flexible re-planning that humans posses when surprised
Are non-classical forms of information processing the answer?

Dennett (1987) asks whether the Frame Problem stems from the conceptual scheme engendered by the serial-processing von Neumann architecture and speculates whether fast parallel processors may bring in their wake conceptual innovations that allow the Frame Problem to be solved in artifacts.

Recent work by O'Reilly (2006) suggests that this may be the case.

Computational cognitive neuroscience simulations of the Basal Ganglia and Prefrontal Cortex suggest that the brain may operate with discrete signals that also draw upon hugely parallel inputs. These neural representations may provide fast parallel access to huge amounts of data with the compositional semantics and productivity of symbolic computation (what are compositional semantics and productivity?)
The Frame Problem and planning using STRIPS

After our DIGRESSION into the broader epistemological Frame Problem we now return to the narrower logic and planning based Frame Problem.

The STRIPS planning language was invented for the robot Shakey and it possesses an assumption that is a kind of solution to the Frame Problem.

In common with the situation calculus the STRIPS language possesses preconditions that describe when actions can be taken and the effects that result after actions are taken. In addition the STRIPS language makes the following assumption:

*that everything not mentioned in the effects list remains the same*

so the effects list has to include all relevant side-effects that would have been put in frame axioms, but with no irrelevant side-effects
What kinds of actions and effects would we want to represent in the blocks world?

Blocks can only be lifted when?

How does the state of the world change when a block is moved?
STRIPS Operators

move(X, Y, Z) {
    precondition: on(X, Y) ∧ clear(X) ∧ clear(Z)
    add: on (X, Z) ∧ clear(Y)
    delete: clear(X) ∧ on(X, Y)
}
Means-end analysis: Forward and backward searching

*If* there is a large knowledge base, but the goal can be described relatively succinctly (that is with a very small proportion of the terms in the knowledge base) *then* backward chaining from a goal may involve less branching in the search space.

(In search problems, is it easier to search from a small village (that is up a mountain with only one road in and out) to the centre of a large city, or easier to do your search the other way round?

All other things being equal, there are more ways to start going in the wrong direction from the city centre than from the mountain village)
Many planning algorithms work by creating subgoals back from a final goal state.

Goal regression involves representing the goal state and then seeing what states come before it, working back until the initial state of the problem is reached.

Goal:
- clear(A)
- on (A, B)
- on (B, C)
- on (C, fl)

Use actions and effects in reverse to 'undo' this goal back to the starting state for the problem.

Subgoals might include
A second problem with planning: the Sussman Anomaly

Demonstrating the Sussman Anomaly

Consider that your goal is to place a red block on a green block

Some of your subgoals are:

- `clear(A)`
- `on(C, fl)`
We don't want to form all states which satisfy subgoals

If subgoals never interact we say they are decomposable

Mostly subgoals in planning problems are nearly decomposable

When subgoals have strong interactions, such that the achievement of one subgoal blocks the achievement of another we have the Sussman Anomaly. This can be overcome by non-linear planning.
Conclusion

Planning with logic – Propositional

Planning with logic – Situation Calculus

Side-effects and interactions – Frame Axioms

The Frame Problem
    (narrow planning/logic problem and broad philosophical problem)

The STRIPS assumptions and add and delete lists

The Sussman Anomaly and non-linear planning

Probabilistic planning