Planning, by constraint satisfaction, by inference

Sam Steel
Department of Computer Science, University of Essex
sam@essex.ac.uk

Abstract  If real physical systems need to be make and execute plans, they need to be able to use complex operators and specifications. There is some support for looking again at planning by constraint satisfaction. One way of doing that in a rigorous convenient way may be to do constraint satisfaction by a form of theorem proving. This paper describes the motivation and ambitions of a programme to do that.

1  Introduction

Perhaps it is time to look again at plans as complex constraints on what happens. This paper describes the motivation of a system currently being developed to do that.

1.1  A coincidence of suggestions

At AIPS 2000 [3], I was struck by some recurrent themes in what was said.

- Suppose plans are to drive real physical robots and spacecraft. In order to do that, one needs to describe fairly complex behaviours, where one wants (say) the temperature of the lubricant to rise, the valve to operate in two stages at different speeds and the temperature to fall again; or one wants the aperture to contract whenever the luminance rises, but only if the camera is pointing at the sun.

  A picture of a plan as a series of actions producing state changes, so that the future can be seen as sequence of states, is not a bad starting point, but the more that one wants to consider actions or states with duration, or which alter a continuously varying quantity over time, or have uncertain effects, and the less one wants to avoid separating planning and scheduling into different activities, then the more that one has to talk about how a history of the world evolves.

- Much of planning can be seen as ways of imposing constraints on what is going to happen. Suppose that one wants the future history of the world to include a table being empty, then having a block standing on it for ten minutes, then being empty again. Those constraints cannot be satisfied directly, but if the future was also constrained so that it included the movement of a hand first carrying, then releasing, then again lifting and carrying the block, the desired constraints would also necessarily be met. Perhaps the constraints describing the movement of the hand can be achieved directly.

These ideas were expressed at AIPS 2000 by David Smith [10] (expanded in [11]) and Malik Ghallab, drawing on his experience with the IxTet system [8] [5]. Similar ideas were
expressed earlier by Allen and Koomen [1], and in considerable detail by Edward Tsang in his PhD thesis [12].

- Generative planning may not be so hopelessly too slow for control of real-time systems as one might think. Quite a lot of planned actions happen over minutes or hours, not milliseconds.

At ECP '99 [2], Nicola Muscettola, also working on plans to drive real physical robots and spacecraft [9], spoke in support of expressive representations and generative planning.

1.2 Plans as programs versus plans as constraints

Another strand of thought is that one can see the relation between a plan and what it is supposed to do in two complementary ways.

One way sees a plan as a complex action, which can be shown to have certain properties: most usually, that if it is started in a state where \( P \) is true, it ends in a state where \( Q \) is true. The action will typically involve several steps, which can perhaps be done in more than one order. More complex structures, for instance involving conditionals and loops, are also possible, though such actions have more often been considered when people are devising programs rather than plans. However, the way that one builds such a plan is to start with a partial description of the complex action, and elaborate it until it can be shown that, if it is carried out, the history that it will drive the world through as it is execute meets the required specification.

For instance, standard non-linear least commitment planning adds steps and causal links to a net of actions until one is sure that all executions containing those features does indeed achieve the goals.

The alternative way sees planning as a process of refining a specification. One states in very abstract terms what one wants to happen; then one finds a more detailed description of the world which would imply the specification one seeks; and one repeats that process until one has a set of specifications of the future which has the property that all of it is either things that one supposes are true of the future, or else can be made true by the executor directly. At that point one has both a description of what the executor must do - a plan — and a proof that it meets the specification.

For example,

- John wants to spend ten hours between flights at Heathrow seeing London; that happens if he travels in by tube and spends seven hours seeing London; that happens if he travels in by tube, spends two hours at St. Paul’s, and spends five hours seeing parts of London other than St. Paul’s; and so on.

- John wants never to pay interest; that happens if whenever his month’s spending with his card is non-zero, it is paid off; that happens if he sets up standing order now; that happens if he writes to his bank.

- John wants Mary to be pleased by him without great cost; that happens if he entertains her in a way she enjoys without great cost; that happens if he takes her out to things she enjoys, without great cost; that happens if he first takes her to a meal and then a film.
1.3 Teleology

A third strand of thought is teleology. Given a plan, it ought to be possible to say why it is as it is. One needs that to justify to another person why this particular plan should be chosen. One also needs it if part of the plan fails; if one knows what the failed section was for, it will be easier to look for a replacement.

Conventional non-linear least commitment plans contain some teleology, in causal links. One can explain the need for a step by saying it establishes the fact which is the label of a causal link supporting a precondition of a later step. But causal links do not just express purpose. They also point out what fact is to be preserved, and over what period. Not do they express all purpose. For instance, an ordering between steps may have been added to prevent a potential clobbering, but there will be no record of that reason in the final plan.

Is there a general way of representing the reason that parts of a plan are as they are?

2 The proposal

That was motivation. Now for the proposal it is intended to motivate.

A brief summary is: plans are constraints on histories; do planning by constraint satisfaction; do constraint satisfaction by inference — that is, repeatedly replace abstract constraints by more concrete constraints that imply them.

2.1 Constraint satisfaction as theorem proving

The idea of constraint satisfaction by inference is independent of planning, so let us consider it first without reference to planning.

Constraint satisfaction by inference starts with the idea of a constraint as a sentence or proposition or description, and the things constrained as things of which the description can be true or false, or as model structures in which the sentence or proposition is true or false.

A common, more concrete, view of constraint satisfaction sees the things constrained as a set of variables, and a constraint as as condition on the assignment of values to the variables. These are clearly pretty similar. For a propositional language of constraints, the variables are propositions, the values are truth values, and the things constrained are valuations of the propositions. But the things constrained could equally well be bottles, with colour of glass, type of cap and contents. A constraint would then be a partial description of a sort of bottle.

In constraint satisfaction, one starts with a set of constraints, a description. Since it is, by definition, a description of what one wants, why does one have to do anything to it?

One reason is that the description may be inconsistent: there may be no object that satisfies it. One may need to show that the description is not empty.

The other reason is that not all descriptions are equally good. This goes somewhat against the prejudices one accepts when one accepts the importance of denotation semantics. That tells us that we know what a term means when we know what it denotes, and also that two descriptions are “equal” when they have the same denotation.
In fact, we often have a notion of a preferred or an acceptable description. For instance, if we are describing numbers, a preferred description is a simple Arabic numeral. Given a complex description such as \( X = 45 \times 93 \), we want to find a preferred equal description: here, \( X = 4185 \). Given a description such as \( X + 34 = 2 \times X + 11 \), we want a description such as \( X = 45 \).

Similarly, in planning terms, “John avoids the patrol” and “John waits 45 seconds, goes through the left-hand door, and dashes over to behind the water barrel” may describe exactly the same courses of events: but John will definitely prefer the second: it is operational.

2.2 Constraints as sentences not predicates

Note that we were explicitly mentioning the objects constrained: here, the variable \( X \). Alternatively, if we see the things constrained as model structures, we will also have a definition of when an arbitrary constraint or description is true at one of those things. If \( X \) is one of the things constrained, and \( S \) is a description which is true of \( X \), we can write \( X \models S \). However, we will not mention the \( X \) explicitly. Instead, we take the \( S \) as a sentence of a logic. All the sentences of the logic are constraints on the model structures, identifying those at which the sentence is true.

Because we have the relation \( X \models S \), this logic has an entailment relation \( \models \), defined as usual as \( S \models T \) iff for all \( X \), if \( X \models S \) then \( X \models T \). We can also hope that the logic’s inference relation \( \vdash \) is sound, so that if \( S \vdash T \) then \( S \models T \).

Now suppose that we have been given a specification which is a sentence in that logic. We want a solution; that is, a description in a preferred form of an object, a model structure, that makes the specification true.

If we are interested in finding a solution, we can replace a weaker description or specification with a stronger one. If the specification is \( S \), and our logic tells us that \( R \vdash S \), we can start looking for a model structure that makes \( R \) true, because it will also make \( S \) true.

We are replacing a sentence \( S \) with one extension by a sentence \( R \) with an extension included in the extension of \( S \). We go on doing that until we have reduced the specification to a sentence which implies the the specification, and which is also preferred, and which is true in the context of the problem: that is, a sentence which is definitely true, or which we are prepared to assume is true, perhaps because we are prepared to make it true.

In fact, one may replace the sentence \( S \) by a collection of sentences, and there may be several ways of doing it: so the proof that one builds will have a conventional and/or tree shape. At any point one will have selected a rule at one or all of the or-branch in the usual way. At that point one has reduced the original specification to the more concrete description made up of the fringe of the partial solution subtree.

What we have done is replace a constraint satisfaction problem by an inference problem.

2.3 Empty constraints are inconsistent sentences

Constraint satisfaction demands that the constraint is not empty. We could of course overdo the reduction. We could reduce \( S \) to a sentence \( R \) whose extension was empty, so that \( \{X | X \models R\} = \{\} \) which is equivalent (in any logic which a falsity constant falsity) to
\( R \models \text{falsity} \). So, given a specification \( S \), we try to find a sentence \( R \) which is preferred, which is strong enough to imply \( S \), but not so strong that it is inconsistent.

There is a very great similarity between this approach and assumption-based truth maintenance systems (ATMSs) [7]. One difference between this and an ATMS is principally that an ATMS does not see the sentences associated with sets of nodes in an and/or tree as a constraint.

### 2.4 Application to planning

How is this account of constraint satisfaction as inference related to planning?

The idea is that the things constrained will be histories, and the sentences will be descriptions of what happens on those histories. The original description will be the specification of the plan, and the final description will be a description of what the executor does himself, or relies on happening spontaneously. Because it was constructed by a series of inferences, the extension of the final description is a set of histories which is included in the extension of the specification, and so necessarily meets the specification. Furthermore, because it is not inconsistent, there is at least one history in the extension of the final description, so it is possible.

I still have to present the sort of language used to describe plans and specifications.

The idea of a plan as denoting a set of action sequences, and of plan refinement as a restriction of that set of actions, has been extensively investigated by Kambhampati [6]. He shows that many plan formalisms and plan development processes can be systematised in the framework. He also has a notion of constraints on what is true at various points in a sequence of actions. He can then say which set of action sequences a plan denotes. The only thing he lacks is a way of saying whether an arbitrary description of state and event (as expressed by a plan) holds of an action sequence, and so whether one such description entails another. I believe that that can only be done if one takes sequences of states rather than sequences of actions as basic; especially if one wants to express time.

### 2.5 Reducing a specification

Let us now look at how a specification could be met in this way.

A history is a way the world can evolve over time. There are many possible histories. Time is a sequence of instants like the real line. The world is in a certain state at each time. This takes time and state as more primitive than, say, intervals or processes or events. I claim that all of those can be reconstructed from time and state. If though one denies that, or thinks that it may be true but is not the right level of abstraction, much of what I suggest will still go through, with a different ontology.

Claims about the past, future and present can be true or false about a history. This might suggest that modal logic is about to be proposed. It could be, but so far I have been experimenting with a very conventional language, suitable for describing conventional non-linear plans. Its ontology includes times, actions and sentences.
2.6 A simple example

It is best shown by an example.

Unfortunately, because this example should be simple, it is going to be based on a simple and familiar action representation. That is exactly what will not show this approach applied to the complex specifications that are what it is supposed to be about. For instance, it cannot handle the John and Mary example, or the credit card example.

Suppose that we are developing this fragment of a standard non-linear plan. Precedence is shown by \( \rightarrow \), causal links by \( \ldots \ldots \). Steps are associated with actions.

\[
\begin{array}{c|c|c}
\text{t1} & p & \text{tn} \\
\hline
\text{a} & & \text{to} \\
\hline
\text{p} & p & q \\
\hline
\text{t2} & q & \text{b} \\
\hline
\text{q} & \text{p} & \text{q} \\
\hline
\text{p} & & \text{p} \\
\end{array}
\]

Now let us look at that expressed as constraint satisfaction by inference. Though a merit claimed for this approach is that it can all be done very rigorously, here it will only be sketched: this paper seeks to motivate the general ideas.

The specification that such a plan is intended to support would be written in the language being used here as something like \( \text{before}(\text{tn}, \text{and}(p, q)) \) meaning “\( P \) and \( Q \) are true for a period, however short, before the step \( tn \)”. That would be reduced to a more operational description by a process of inference leading to a proof like this.

\[
\begin{align*}
\text{before}(\text{tn}, \text{and}(p, q)) & \\
\text{--------------------------} & \\
\text{before}(\text{tn}, p) & \text{before}(\text{tn}, p) \\
\text{--------------------------} & \text{--------------------------} \\
\text{after}(t1, p) & \text{preserved}(t1, tn, p) & \text{after}(t2, q) & \text{preserved}(t1, tn, q) \\
\text{--------------------------} & \text{--------------------------} \\
\text{occursat}(t1, a) & \ldots & \text{occursat}(t2, b) & \ldots
\end{align*}
\]

The fringe nodes are collectively the description to which the specification has been reduced. Whether a node is about a sentence known to be true, or whether it can be assumed — as presumably the \( \text{occursat}(\ldots) \) facts can be — or whether it needs further reduction, will depend on the formalization used and the situation in which the plan is made.

The claim that an action \( A \) has an effect \( E \) and a precondition \( P \) is expressed by an axiom

\[
\text{after}(T, \text{Effect}) \subset \\
\text{occursat}(T, \text{Action}) \land
\]
achieves(Effect, Action, Precondition) \land
before(T, Precondition)

The meaning of the variables is as one would expect. All variables are implicitly universally quantified.

One cannot freely postulate these axioms, because the meaning of the predicates is fixed by a semantics defined in terms of histories. For instance, after(T, Effect) is true on a history if T names a time, and there is a set of states, however short, immediately after T at all of which Effect is true. achieves(Effect, Action, Precondition) means

if Action occurs at time t on a history, then
  if there is a period before t when Precondition is true, then
    there is a period after t when Effect is true

The detail of this is not important. A different language would use different predicates with different semantics.

2.7 What is clobbering in this approach?

Now, how though does one establish that most important sort of fact in non-linear planning, that a sentence is in fact preserved?

The rule that is used to guaranteed that the effect of an earlier action persists until a later action is

before(Later, P) \subset after(Earlier, P) \land preserved(Earlier, Later, P)

What happens is that there are, in this formulation of planning, no rules whose conclusion has the form preserved(Earlier, Later, P). Goals in the proof tree of that form can only be assumed. So what about clobbering?

It is here, and in similar places, that the importance of demanding that the description of the plan is consistent, that its extension is non-empty, appears.

The description of a plan may contain many nodes. It could contain nodes that collectively assert that some action occurs at a time T1, that it has the effect \(\neg P\), and that T1 is between Earlier and Later. In that case, the inference process should notice that the description to which is has reduced the specification is inconsistent. Collectively, those claims and the claim that preserved(Earlier, Later, P) entail contradiction, and so no history can make all of them true.

For instance, in this example, given the assumptions that are made, it is also possible to construct a proof like this. (I am pulling some rules out of a hat, but the general idea, not the exact set of rules, is what matters here.)

\[
\begin{align*}
\text{contradiction} \\
\hline
\text{preserved}(t_1, t_n, P) & \text{after}(t_2, \neg P) & t_1 < t_2 & t_2 < t_n \\
\hline
\text{occursAt}(t_2, b) & \ldots
\end{align*}
\]
Note though that that proof itself makes assumptions: $t_1 < t_2$ and $t_2 < t_n$. Those assume that $t_2$ is between $t_1$ and $t_n$ — that is, that the clobbering step occurs during the period over which $P$ is to be preserved. That of course need not be true. Negating one of those assumptions is how conventional promotion and demotion work.

2.8 Avoiding inconsistent descriptions

Doing constraint satisfaction by inference requires that one prove the specification without also proving contradiction. What must happen when a possible proof of contradiction is found is that one of its assumptions must be assumed false. The proof will of course remain valid but its conclusion, contradiction, will not be entailed.

As a result, the complete description of the future will contain not just parts that are there to prove that the main goal is obtained — for instance, $\text{occursat}(t_2, b)$ — but also to parts that are there to avoid proofs of contradiction — for instance, the negation of $t_1 < t_2$.

Note that there either of $t_1 < t_2$ and $t_2 < t_n$ could be assumed false. Just as the choice of which rule to use to reduce a node that should be true introduces choice-points in the search space, so does the choice about which antecedent of a rule leading to contradiction to assume false.

Furthermore, if a sentence is required to be false, but is proved by one or more rules, each of those rules must itself fail to entail its conclusion, so one of its antecedents must be false. No node in the proof and/or graph may be required to be both true and false. It may suggest the sort of search process required to find a labelling for the nodes in the graph if one thinks of falsity or unprovedness seeping up from the fringe of the graph just as truth does, except with and-nodes for provedness being or-nodes for unprovedness and vice versa.

2.9 Teleology

Note that once one has a labelled and/or graph one has a teleology of why the plan is as it is. Take any part of the description of the plan. Ask why it is as it is. If its truth is part of the proof of the main specification, one has one’s answer. If its negation is part of the proof of contradiction, where the rest of the proof is either true or undetermined, one again has one’s answer.

This is a very general representation of a plan’s teleology.

3 Making plans in this way using other formalisms

This example has assumed an ontology and a set of rules which are obviously inspired by non-linear planning. However, the same approach should work for something very different — say, hierarchical task-net (HTN) planning. Of course that is already well explained as a sort of constraint satisfaction by Erol, Nau and Hendler [4]. However, constraints alone do not help with stating the teleology of an HTN plan.

The rules required to express HTN plans would look very different from those used above. They would say such things as
the complex action \( Z \) occurs between \( t_0 \) and \( t_2 \) if
the simple action \( A \) occurs between \( t_0 \) and \( t_1 \) and
the simple action \( B \) occurs between \( t_1 \) and \( t_2 \)

contradiction if
the simple action \( A \) occurs between \( t_0 \) and \( t_1 \) and
the simple action \( B \) occurs between \( u_0 \) and \( u_1 \) and
\((t_0,t_1)\) and \((u_0,u_1)\) overlap

But the process of reducing a specification (stated perhaps as the claim that some complex action occurs between two times, without also proving contradiction) would be exactly the same.

4 Planning by theorem proving is different

Let us compare this proposal with planning as theorem proving. Go back to the very earliest days of the situation calculus. There one started with a specification \( \text{Specification}(A) \) of an action, where \( \text{Specification}(A) \) was something such as

\[
\text{if } P \text{ is true in situation } S, \\
\text{then } Q \text{ is true is the situation reached by doing action } A \text{ in situation } S
\]

The situation calculus approach, and many like it in different notations, then looked for a constructive proof of \( \text{Specification}(A) \). Once such a proof was found, one could say

there is a proof that \( \text{Specification}(A) \) which assumes nothing

Since the proof was constructive, and constructed in the right way, the term \( A \) that was constructed can be executed.

That approach has been rejected for many years, because, while beautiful and clean, it was too hard to control — as theorem proving so often is — because it needed endless frame axioms. (Its commitment to linear plans is incidental).

Is what is is being proposed here just the same thing all over again? No, not at all. That is not to say it is a good idea — that still has to be shown — but it is quite different.

Here, there is no action term. Instead, one finds a description of a history which implies the specification. One ends with

there is a proof of \( \text{Specification} \) which assumes \( \text{Description} \)

\( \text{Description} \) is facts about what the executor does and about how the environment behaves. There is no expectation that what is assumed is what is true at the start of the execution. It could be assumptions about what happens spontaneously part way through the execution.
5 Prospects and questions

At this point I have I hope made clear the main ideas of a way of planning by constraint satisfaction by proving. An implementation of what is described here has been started. Many things have been left out; many things are not clear.

- What is a good strategy for growing and labelling the proof trees relevant to a plan?
- Will it work fast enough to be interesting? Much of the success of practical planning is bought by building specialised decision making into control flow of the planning programs. For instance, in complex non-linear planners, there may be a special module for answering questions about the necessary order of time points. One might say that in this framework one could still use such a module, and could recover the relevant input facts about the order of the times from the fringe of the proof tree: but is that true?
- It is very likely to be true that many interesting planning formalisms can be expressed as rules in this form, but can they all? And is this done at a ruinous necessary cost in efficiency?
- When is a description of a history sufficiently refined to be given to an executor? How does one in general tell when one has reduced a specification far enough?
- This approach undertakes to record why a part of a plan is as it is. It is similarly possible to record why an alternative way of reducing a specification was not used?
- Perhaps the most embarrassing question: what about alternatives? Suppose
  - a plan prescribed different actions to be done in different cases, as in contingent planning;
  - or prescribes a single action to be done in different cases, because it will work in any of them, as in conformant planning;
  - or claims that one of several actions will meet a specification, but is not exact about which one, as in a partly formed disjunctive plan made by GraphPlan;

How are these represented and reasoned about in this framework?

6 Conclusion

Despite these serious — possibly fatal — questions, I am confident this remains a worthwhile project. Increasingly there seems to be a need for a generative planner capable of dealing with highly expressive operators and specifications. Given that need, it would be absurd not to investigate this approach.
References


