

Applying Lakatos-style reasoning to AI domains

1 Lakatos-style reasoning

Lakatos’s theory of mathematical evolution [3] is a tour de force of ways in which mathematicians use inconsistency and exploit ambiguity to develop ideas. We argue that Lakatos-style reasoning can be used to: (a) suggest modifications to software specifications which are incomplete, incorrect or inconsistent; (b) refine a model or constraints in a constraint satisfaction problem, and (c) aid revisions in the planning domain.

Lakatos’s heuristic approach identifies ways in which mathematicians interact in order to refine concepts, conjectures and proofs. Suppose that we have conjecture $\forall x(P(x) \rightarrow Q(x))$, supporting examples $s \in S$ such that $\forall s \in S, P(s) \wedge Q(s)$ and counterexamples $c \in C$ such that $\forall c \in C, P(c) \wedge \neg Q(c)$. Then *surrender* would entail abandoning the conjecture when the first $c \in C$ is found; *piecemeal exclusion* in determining the extension of C , generating an intensional definition $C(x)$ of a concept which covers exactly those examples in C and modifying the conjecture to $\forall x((P(x) \wedge \neg C(x)) \rightarrow Q(x))$; *strategic withdrawal* in determining the extension of S , generating an intensional definition $S(x)$ of a concept which covers exactly those examples in S , and modifying the conjecture to $\forall x((P(x) \wedge S(x)) \rightarrow Q(x))$; *monster-barring* in arguing that $\forall c \in C, \neg P(c)$, either by narrowing an already explicit definition, or by formulating a first explicit definition of P (each party in the discussion must then accept the new definition of P , and revise their theory accordingly); *monster-adjusting* in arguing that $\forall c \in C, Q(c)$ (again formulating and negotiating the definition as for monster-barring); and *lemma-incorporation* in using each $c \in C$ to identify flaws in the proof (local counterexamples) or conjecture (global counterexamples) which can then be rectified.

2 Evolving requirement specifications

Lakatos’s terminology can work on multiple levels in this domain. For instance, conjectures may be of the sort: “this set of specifications/program captures the set of user requirements” (supported by examples of desired input-output pairs); “this output will be produced, given the input” (supported by a set of specifications or a program); “this system design captures the set of specifications” (supported by formal verification methods). These could be nested, where a refined conjecture on one level might play the role of axioms on another level. A counterexample is analogous to input/output pairs where output is not the desired kind, either because it does not satisfy the specifications, or because the specifications do not satisfy the user requirements.

Given some initial functionality specification by an input-output relation R and a range of desired input values, a system may test a specification by providing input-output pairs related by R . The user may then decide that a given pair are not desirable for the task at hand, and thus some modification would be necessary. We can translate this into Lakatosian terminology as follows: given a conjecture “Let I be the set of possible inputs, $R(x, y)$ be an input-output relation (which is either a system design or a program) and O_x be the desired output range for each input x (assuming that there *is* an output for each input), then $x \in I \rightarrow \exists y(y \in O_x \wedge R(x, y))$ ”. Supporting examples would take the form (m, n) , where $m \in I \wedge \exists n(n \in O_m \wedge R(m, n))$, and counterexamples would take the form (m, n) , where $m \in I \wedge \neg \exists n(n \in O_m \wedge R(m, n))$. Reactions to these might include:

Surrender: reject the system design $R(x, y)$.

Piecemeal exclusion: find those values in the input range I which break the conjecture, form a set of negatives N and modify the conjecture to $x \in I \setminus N \rightarrow \exists y(y \in O_x \wedge R(x, y))$.

Strategic withdrawal: find those values in the input range I which support the conjecture, form a set of positives P and modify the conjecture to $x \in I \cap P \rightarrow \exists y(y \in O_x \wedge R(x, y))$.

Monster-barring: argue that the input m is *not* a member of I .

Monster-adjusting: argue that m is really m' and output n *is* a member of $O_{m'}$, or that the output n actually *is* a member of O_m , or that the input-output relation R does *not* produce n when given m .

3 Constraint satisfaction problems

A conjecture in a constraint satisfaction problem (CSP) corresponds to a current search path which is hypothesised to satisfy all the constraints. Supporting examples correspond to constraints which are satisfied by a model, and counterexamples to constraints which are violated by the model. Correspondences to Lakatos's methods include:

Surrender: Abandon the current search path (model) as soon as a single inconsistency is encountered (*i.e.*, a constraint is violated). This is most commonly used for CSPs, and triggers backtracking techniques.

Piecemeal exclusion and strategic withdrawal: Freuder *et al.*'s [1] techniques for partial constraint satisfaction are analogous to retrospective, prospective and ordering techniques for CSP. These are necessary if there is no complete solution at all (the problem is overconstrained), or we cannot find the complete solution with the resources given. Their notion of local failure is a predetermined cutoff bound, rather than a single violated constraint, *i.e.*, a certain number of counterexamples are allowed: this is called sufficient satisfaction, where the search is terminated if a sufficiently good solution is found. Freuder *et al.* also discuss more sophisticated metrics in which the importance, rather than number of constraints, is measured; where constraints might be ordered, hierarchical, or prioritised. Constraints may be weakened by enlarging a variable domain (introduce a new value that a variable might take), enlarging a constraint domain (deciding that two previously constrained values are acceptable), removing a variable (one aspect of the problem is dropped), or removing a constraint (deciding that any combination of two previously constrained variables is acceptable). Of particular interest is Freuder *et al.*'s position on alternative problems: "We suggest viewing partial satisfaction of a problem, P, as a search through a space of alternative problems for a solvable problem 'close enough' to P." [1, p. 3]. This has a very clear analogue in Lakatosian terms, where 'conjecture' is substituted for 'problem', and 'provable' for 'solvable'. Freuder *et al.* go on to argue that a full theory of partial satisfaction should consider how the entire solution set of the problem with altered constraints differs from the solution set of the original problem, as opposed to merely considering how a partial solution requires us to violate or vitiate constraints. That is, they compare problems rather than violated constraints.

As far as we know, no CSP algorithm considers the violated constraints once a satisfactory partial solution has been found. Theory exploration on these constraints may improve the solution, by grouping together the violated (or satisfied) constraints. The obvious exception-barring analogues are to find a property common to all known violated constraints, not shared by satisfied constraints, and hypothesise that the model satisfies all the constraints *except* those with this property, and to find a property common to all known satisfied constraints, not shared by known violated constraints, and hypothesise that the model satisfies all the constraints *with* this property.

Monster-barring and adjusting: Either argue that the proposed counterexample constraint is not a valid constraint, and formulate properties that a valid constraint must have, or argue that the model *does* satisfy the problem constraint. Flexible, as opposed to conventional, CSPs relax the assumption that solutions must satisfy every constraint and that constraints are either completely satisfied or else violated. In particular, fuzzy CSP represent constraints as fuzzy relations, where their satisfaction or

violation is a continuous function. Ruttkay [4] discusses the issue of soft constraint satisfaction from a fuzzy set theoretical point of view.

Unsurprisingly, an analogue of lemma-incorporation is hard to imagine since there is no obvious analogue to proof in CSPs.

4 Planning

Lakatos-style reasoning is strongly analogous to the planning domain, where a mathematical axiom corresponds to background assumptions about the world; concepts to types of action; conjectures to plans (if conditions C hold and action A is executed then effects E will result: $C \wedge A \rightarrow E$); a proof to a justification of a plan, a supporting example to an action which is executed according to a plan (preconditions are satisfied) and achieves the goal ($C \wedge A \wedge E$) and a counterexample to an action which is executed according to a plan (preconditions satisfied) but which fails to achieve the goal ($C \wedge A \wedge \neg E$).

Hayes-Roth [2] describes 5 heuristics, which are actually based on Lakatos's methods, for repairing flawed beliefs in the planning domain. He demonstrates these in terms of revising a flawed strategy in a simple card game, Hearts. In Hearts a pack of cards is divided amongst players, one player plays a card and the others must all put down a card in the same suit as the first if they have one, and otherwise play any card. The person who played the highest card in the specified suit wins that trick and starts the next. One point is awarded for each heart won in a trick, and 13 for the queen of spades (QS). The aim of the game is to get either as few points as possible ("go low") or all the points ("shoot the moon"). An example of a justification of a plan (corresponding to a mathematical proof) is "(a) the QS will win the trick, therefore (b) the player holding the QS will get the 13 points, therefore (c) this plan will minimise the number of my points"; an example of an action which is executed according to a plan (corresponding to an entity) is to "play the 9 spades"; and an example of a concept is "a spade lower than the Queen". Counterexamples correspond to moves which follow a strategy but which do not have the desired outcome. For instance, a strategy which beginners sometimes employ is to win a trick to take the lead, and then play a spade in order to flush out the QS and avoid the 13 points. Hayes-Roth represents this as shown below [2, p.230]:

Plan:	Flush the QS
Effects:	(1) I will force the player who has the QS to play that card (2) I will avoid taking 13 points
Conditions:	(1) I do not hold the QS (2) The QS has not yet been played
Actions:	First I win a trick to take the lead, and whenever I lead I play a spade

The plan (analogous to a faulty conjecture) may backfire if the beginner starts with the king of spades (KS) and then wins the trick and hence the unwanted points (this situation is a counterexample to the plan). Heuristics then provide various ways of revising the plan: we show these in terms of Lakatos's methods below.

Surrender: *retraction* - retract the part of the plan which fails, in this case effect (2).

Piecemeal exclusion: *avoidance* - rule out situations which can be predicted to fail the plan, by adding conditions to exclude them. For example by assessing why the plan failed add the condition *I do not win the trick in which the queen of spades is played*. A system can further improve its plan by negating the new condition - *I win the trick in which the queen of spades is played*, using this and its knowledge of the game to infer that it must play the highest card in the specified suit, and then negating the inference to get *I must not play the highest card in the specified suit*. This is then incorporated into the action which becomes *First I win a trick to take the lead and whenever I lead, I play a spade which is not the highest spade*.

Strategic withdrawal: *assurance* - change the plan so that it only applies to situations which it reliably predicts. In this case the faulty prediction is effect (2), and so the system looks for conditions which guarantee it. It does this by negating it, inferring consequents and then negating one of these and incorporating it into the action. For example negating effect (2) gives *I do take 13 points*, the game rules state that *the winner of the trick takes the points in the trick* so we can infer that *I win the trick*, then use this and the rule that *the person who plays the highest card in the suit led wins the trick* to infer that *I play the highest card in the suit led*. Given that *player X plays the QS* we can now infer that *I play a spade higher than the QS* and negate it to get *I play a spade lower than the QS*.

An alternative heuristic, which also relates to strategic withdrawal is *Inclusion* - this differs from assurance in that the situations for which the plan is known to hold are listed rather than a new concept being devised. Therefore instead of adding *I play a spade lower than the QS* to the action, we add *I play a spade in the set {2 of spades, 3 of spades, 4 of spades ..., 10 of spades, Jack of spades}*.

Monster-barring: *exclusion* - bar the theory from applying to the current situation, by excluding the situation. Add the condition *I do not play KS*.

We can extend this example to include monster-adjusting and lemma-incorporation as follows:

Monster-adjusting: *re-evaluation* - reinterpret the counterexample as a positive example by changing the overall strategy into shooting the moon rather than going low. In this case, getting the QS has a positive effect on the goal of winning.

Lemma-incorporation: *consider the plan* - consider the proof and try to find counterexamples to lemmas: (a) the QS will win the trick (b) the player holding the QS will get the 13 points (c) this plan will minimise the number of my points, might suggest the counterexample of the KS which violates (a) (and (b)). Analysis of the counterexample would show that it is both local and global, and so the first lemma would be incorporated into the conjecture (as a further condition). This then becomes: if (1) I do not hold the QS, and (2) The QS has not yet been played, and (3) The QS wins the trick (is the highest spade in the trick), then (1) I will force the player who has the QS to play that card, and (2) I will avoid taking 13 points.

4.1 Discussion

We have described analogies between Lakatos's theory of mathematical evolution and the fields of evolving requirement specifications, constraint satisfaction problems, and planning. Showing the relevance of Lakatos's theory to these diverse domains will highlight connections between them and suggest ways in which philosophy can inform AI domains.

References

- [1] E. Freuder and R. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, (58):21–70, 1992.
- [2] F. Hayes-Roth. Using proofs and refutations to learn from experience. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 221–240. Tioga Publishing Company, Palo Alto, CA, 1983.
- [3] I. Lakatos. *Proofs and Refutations*. CUP, Cambridge, UK, 1976.
- [4] Z. Ruttkay. Fuzzy constraint satisfaction. In *3rd IEEE Int. Conf. on Fuzzy Systems*, pages 1263–1268, 1994.