A Logical Measure of Progress for Planning

Aarati Parmar
Department of Computer Science
Stanford University
Gates Building, 2A Wing
Stanford, CA 94305-9020
aarati@cs.stanford.edu

Abstract

Heuristic search planners are so far the most successful. Almost all use as their heuristic an estimate of the distance to a goal state. We formalize a *logical measure of progress*, defined as a predicate $P(\bar{x}, s)$ true of objects $\bar{x}$ at a situation s. Actions which increase P's extension are guaranteed to move closer to a goal situation, so that P enables us to form plans without search. One example of a measure of progress is the concept of final position used in BlocksWorld. It is not clear how to find a P for an arbitrary domain, so instead we identify three different classes of domains and conditions which allow us to construct a measure of progress.

An *obvious* P will not deliver optimal plans, but it should encode plans which are "good enough." Our paradigm is entirely within first-order logic, allowing us to extend our results to concurrent domains and those containing non-trivial state constraints. It turns out P not only encodes goal orderings, but subgoal orderings. P also gives rise to a strategy function $a(s)$ which can be used to create a universal (complete) teleo-reactive (TR) program. Given the fact that P-increasing actions will never require backtracking, this TR program can be a powerful on-line planner.

1 Introduction

Planners that use *heuristic search* have been the most successful to date, garnering four out of the top six spots in the recent AIPS 2000 planning competition (Bacchus 2000). These planners reduce planning to heuristic search, where the heuristic estimates the distance to the goal state. Most use ADL operators, which not only express what STRIPS can, but also disjunctive preconditions, conditional effects, and quantification.

The planners mentioned above all use some estimate of distance to the goal, derived from a relaxed version of the plan, where operators' delete lists are ignored. However, they use slightly different heuristics, in different ways. HSP2 (Bonet & Geffner 2001) employs a strategy of different heuristic functions, simultaneously. MIPS (Edelkamp & Helmert 2001) uses a symbolic heuristic search, where an estimate of the goal distance is associated with each proposition, and combined accordingly. If a domain looks like a route-planning or resource allocation problem, STAN4 (Fox & Long 2001) cleverly estimates a distance heuristic that takes advantage of the domain structure. FF (Hoffmann 2001a), the top heuristic planner in the competition, uses GraphPlan to compute its estimate, which automatically takes into account positive interactions between goals.

Heuristic search planners are extremely effective. However, the heuristics used are not very elucidating; they only estimate a distance to a goal, without giving any motivation in terms of the structure of the domain. In this paper we define *a logical measure of progress*, that not only produces an action leading *directly* towards the goal, but also explains *why* it works, in terms of properties of the domain. Our avenue of research is orthogonal to the current state of the art in planning; we are more interested in understanding what properties of domains lead to efficient planning rather than finding faster and better algorithms.[1]

We follow the epistemological approach proposed by (McCarthy 1959) regarding the Advice Taker: "behavior will be improvable merely by making statements to it, telling it about its symbolic environment." Intelligent robots operating in the world will need to identify and take advantage of regularities in the world in order to reason efficiently. This research is one step towards this goal.

This work can also be viewed as a bridge between domain independent and dependent planning. We share the motivation of the creators of TIM (Long & Fox 2000), in that discovering domain-specific heuristics from a domain's structure, is an important task for planning, and AI. While TIM identifies generic types, we find a logical measure of progress within the domain.

Before continuing we discuss some related research. The other top planner in AIPS 2000 is TALplanner (Doherty & Kvarnström 2001), a forward chaining planner guided by first order temporal formulas. This approach retains a high level of expressivity and control, without damaging performance. Our logical measure of progress could be used as control formulas for TALplanner, avoiding the need for a user to define them.[2] (Sierra-Santibañez 2001)

[1](Hoffmann 2001b) finds properties leading to efficiency based on the *topology* of the search space (in terms of local minima, plateaus, etc.); we want to find properties based upon the structure of the domain (in terms of local predicates).

[2]One of the reviewers points out that finding a measure of progress could be just as much work as writing a proper control

uses a declarative formalization of strategies for action selection. The approach utilizes user-defined *action selection rules* of the form $\phi(\bar{x}, a, b, s) \implies \psi(a, b, s)$, where $\psi(a, b, s) = Good(a, s) \mid Bad(a, s) \mid Better(a, b, s)$. The planner then uses these categorizations to search. Our approach will give reasons why a particular action is $Good$ or Bad.

More details, including full proofs, may be found in the technical report (Parmar 2002).

2 Preliminaries

The language of a planning domain traditionally includes some finite set of objects $Objects$ and predicates $\{p(\bar{x})\}$. A planning problem consists of predicates true of the initial state I, a goal formula G, and a set of STRIPS or ADL operator schema O. We translate this specification into a theory T of first-order logic, using the situation calculus.[3] We use first-order logic, instead of STRIPS or ADL, for the greater expressivity and because logic provides a more generalizable framework, addressed more in the Conclusions. The language of T has three sorts, $Situations$, $Actions$, and $Objects$:

1. $Objects$: One of the crucial assumptions made throughout this paper is that *our domain has finitely many objects.* Hence T has a domain closure axiom (DCA) of the form: $(\forall x)[x = x_1 \vee ... \vee x = x_n]$, along with a unique names axiom over these objects: $UNA[x_1, ..., x_n]$.
2. Predicates: Only one fluent relation $\Phi(\bar{y}, s)$ is used in T, assuming each original predicate can be coded by defining $\Phi(code(p), \bar{x}, s)$ for each $p(\bar{x})$, where $code(p)$ is a tuple of object constants.[4] $\Phi(\bar{y}, s)$ which do not code a particular fluent, are defined to be $\top$. Φ provides a single handle to talk about any fluent. In this way Φ is similar to the well-known situation calculus predicate $Holds(f, s)$ which asserts that fluent f is true in situation s.
3. Initial state: I, the initial state, is mapped to S_0, the initial situation. All facts true in the initial state are true of S_0.
4. Goal formula: $goal(s)$ abbreviates the goal formula G.
5. Operators: An action $a(\bar{y})$ corresponds to each operator schema $o(\bar{y})$. Any ADL operator can be translated to a successor state axiom of the form:

$$\Phi(\bar{x}, res(a(\bar{y}), s)) \iff \gamma^+_\Phi(\bar{x}, a(\bar{y}), s) \vee (\Phi(\bar{x}, s) \wedge \neg\gamma^-_\Phi(\bar{x}, a(\bar{y}), s)),$$

where $\gamma^+_\Phi(\bar{x}, a(\bar{y}), s)$ is a fluent formula abbreviating the conditions under which $\Phi(\bar{x}, \cdot)$ is true after $a(\bar{y})$, while $\gamma^-_\Phi(\bar{x}, a(\bar{y}), s)$ are those where $\Phi(\bar{x}, \cdot)$ becomes false. Successor state axioms are written such that the truth of $\Phi(\bar{x}, res(a(\bar{y}), s))$ is only a function of the fluents true at s and are thus Markovian. While we assume the number of $Objects$ is finite, we make no such restriction on the numbers of action schema (which can even be uncountable).

$Poss(a, s)$, used to abbreviate *action preconditions*, is omitted. Instead the γ_Φs are written such that when $\neg Poss(a, s)$ holds, $\Phi(\bar{x}, res(a(\bar{y}), s)) \iff \Phi(\bar{x}, s)$. Indeterminacy of inapplicable actions is replaced with inertia, so that the entire tree of situations is utilized, minimizing technical complications.

For the rest of this paper assume T is a domain theory satisfying the constraints given above.

3 A Strong Measure of Progress

We are motivated by the well-known idea of final position used in BlocksWorld planning. A block is in *final position* if the object it is on is the object it is on in the goal state, and that object is in final position. The table is always in final position. This strategy avoids the Sussman anomaly by building towers from the bottom up.

Final position is a good measure of progress, because putting something into final position will lead us closer to the goal. It also identifies objects so that putting something into final position will not prevent the completion of the other goals. We will *never* have to backtrack on any action that increases the extension of final position. Final position is a sort of Dynamic Programming for AI – it identifies optimal substructure to promote tractable solutions to a problem.

Our *logical measure of progress* is generalized as a similar predicate over objects in the domain. Increasing its extension will lead us closer to the goal, without undoing any subgoals. Note that it is a predicate over *objects* in the domain, and *not* some property relating situations to their [estimated] distance to the goal. The purpose of our construction is to force our measure of progress to be a function of the properties of the domain:

Definition 1 (*A Strong Measure of Progress*)
Let s be a situation variable, $x_1, ..., x_n$ object variables, and $P(x_1, ..., x_n, s)$ a fluent formula. Call P a strong, n-ary measure of progress *with respect to $goal(s)$ if:*

$$T \models (\forall s)[\neg goal(s) \implies (\exists a)[ext(P, s) \subset ext(P, res(a, s))]] \quad (1)$$

$ext(P, s) =_{def} \{\bar{x} \mid P(\bar{x}, s)\}$. *Hence* $ext(P, s) \subset ext(P, s')$ *abbreviates* $(\forall \bar{x})(P(\bar{x}, s) \implies P(\bar{x}, s')) \wedge (\exists \bar{y})(\neg P(\bar{y}, s) \wedge P(\bar{y}, s'))$.

P captures a very strong notion of progress. If $\neg goal(s)$, the definition guarantees that there is an action a that strictly increases the extension of P. Furthermore, when $P(\bar{x}, s)$ is true for all tuples in the domain, $goal(s)$ holds. An action which increases the extension of P not only necessarily gets us closer to the goal, but does so without undoing any other goals along the way.

Equation (1) assures us that in a domain with finitely many objects, we can reach the goal situation from any situation; if T has a strong measure of progress P, then T is

formula. If this is the case, our approach will at least provide some foundations for how to construct the control formula in the first place.

[3] We abbreviate res for $result$.

[4] We assume our domain includes at least two object constants. The reason why fluents are coded using object constants will be apparent when we define our measure of progress.

deadlock-free. Intuitively this is obvious as the predicate P above is strictly extendible in *any* situation, and if there are finitely many objects to cover, eventually they must all be covered, which means by definition the goal is achieved:

Theorem 1
Let P be a strong, n-ary measure of progress. Then, from any non-goal situation s we can reach a situation satisfying $goal$, within $|Objects|^n - |ext(P, s)|$ steps.

Proof: Clearly, at s there are $|Objects|^n - |ext(P, s)|$ object tuples not in P. We need to apply at most that many P-increasing actions to have P be true for all object tuples, at which point we are at the goal. ■

We can replace the existentially quantified variable a in Equation (1) with the skolem function $a(s)$.[5] This $a(s)$ becomes our *strategy function*, mapping situations to actions. With P, planning is straightforward – we find an action a strictly increasing the extension of P and apply it. This will guarantee that we move closer to the goal. By requiring a *strict* extension of P with every action, P leads directly to the goal – no local minima or plateaus clutter our path.

If a domain has a strong measure of progress then it is deadlock-free. The converse also holds:

Theorem 2
Let T be a planning domain such that, from every non-$goal$ situation s, there is a finite sequence of actions which will lead to a $goal$ situation. Then there exists an n and P such that P is a strong n-ary measure of progress.

Proof Sketch: Let $\ell(s)$ be the shortest distance from the situation s to a goal situation. The idea is to construct, for each n, a fluent formula $G(n, s)$ true of exactly those s for which $\ell(s) = n$. Since $\ell(s)$ is bounded, we can bijectively code each object tuple to each value of $\ell(s)$. Then we define $P(\bar{x}, s)$ to be true of all tuples whose code is greater than $\ell(s)$. The proof relies on the fact that we have finitely many fluents and objects (so that $\ell(s)$ is bounded), and that the progression from s to $res(a, s)$ is only a function of the fluents at s (so that $G(n, s)$ is well-defined). ■

One can see from the details of the construction that the strong measure of progress defined in this way will generate optimal plans.

Theorems 1 and 2 indicate that the planning domains which have a strong measure of progress with respect to $goal$ are precisely those which are deadlock free (with respect to $goal$). This is why P is termed as *strong* – it is equivalent to the universal reachability of a goal situation and thus will not exist in domains with deadlocks.

Another problem is that any domain that is deadlock free will have a strong measure of progress P, but it does not necessarily have to be *obvious* in the sense of (McAllester 1991).[6] That is, it may not be immediately evident from P's definition that it is a strong measure of progress with respect to the structure of the domain. A P that is obvious will be concise, clearly expressed, and meaningful, such as final position is in BlocksWorld. But unfortunately, un-obvious strong measures of progress exist, such as obscure encodings of the distance to the goal. For example, Rubik's cube is a deadlock-free domain. By Theorem 2, strong measures of progress must exist for solving the cube, but if any were obvious, Rubik's cube would not be the intriguing puzzle it is.

We believe however that most domains in the real world do exhibit an obvious strong measure of progress, as humans plan in them with little search. These domains are easy because they contain enough structure that the measure of progress is clear. In the following sections we formalize what sorts of domain structures give rise to such P. If we cannot automatically construct an adequate P for a domain, perhaps we can at least identify these structures, and from that construct a P.

4 Constructing Strong Measures of Progress

It is not clear how to automatically obtain a strong measure of progress P for a given theory T and $goal$. One could use inductive logic programming or genetic algorithms to look for definitions of P, and promote obvious solutions by using the *local rule sets* of (McAllester 1991) to generate hypotheses for P. Or, one could analyze the state-space graph of small domains for a P which satisfies the constraints, and try to generalize from there, as pointed out by (Lin 2002). We note that finding a P will be at least PSPACE-hard, since finding plans is at least that hard. But the good news is that once we find a P for a given $goal$, we can reuse it to construct any plan with the same (or logically weaker – see (Parmar 2002)) goal, so the cost of finding P is amortized over the times it is used.

Automatic construction of P is an avenue for future work which is beyond the scope of this paper. Instead we study planning domains which admit obvious strong measures of progress, and distill what properties are fundamental. The hope is that we can automatically discover these fundamental properties and perhaps combine them to construct more complicated, strong measures of progress for arbitrary domains.

4.1 Kitchen Cleaning Domain

In the Kitchen Cleaning Domain posed in (Nilsson 1998), cleaning any object makes it clean. However, cleaning the stove or fridge dirties the floor, and cleaning the fridge generates garbage and messes up the counters. Cleaning either the counters or floor dirties the sink. The goal is for all the appliances to be clean and the garbage emptied.

Regardless of the initial state, there is a natural order to cleaning the kitchen. One should clean the sink last, as we may dirty it while cleaning other objects. Cleaning the stove and fridge can safely be done first, as they cannot get dirtied by any subsequent actions. The successor state axioms of $T_{\mathcal{KC}}$ and $goal$ are shown in Equation (2). $\Phi(x, s)$ represents "x is clean," except when $x = garbage$, in which case it means the garbage is not empty.

[5] The Axiom of Choice will be required in case we have uncountably many actions.

[6] It is *obvious* that a king can traverse all squares on a chessboard; but not that a knight can.

$$
\begin{aligned}
goal(s) \quad \equiv_{abbrev} \quad & \Phi(fridge, s) \wedge \Phi(stove, s) \wedge \\
& \Phi(floor, s) \wedge \Phi(counters, s) \wedge \\
& \Phi(sink, s) \wedge \neg\Phi(garbage, s)
\end{aligned}
$$

$$
\begin{aligned}
&\Phi(fridge, res(a, s)) \iff a = c(fridge) \vee \Phi(fridge, s) \\
&\Phi(stove, res(a, s)) \iff a = c(stove) \vee \Phi(stove, s) \\
&\Phi(floor, res(a, s)) \iff a = c(floor) \vee \\
&\quad \Phi(floor, s) \wedge \neg(a = c(stove) \vee a = c(fridge)) \\
&\Phi(counters, res(a, s)) \iff a = c(counters) \vee \\
&\quad \Phi(counters, s) \wedge \neg(a = c(fridge)) \\
&\Phi(sink, res(a, s)) \iff a = c(sink) \vee \\
&\quad \Phi(sink, s) \wedge \neg(a = c(counters) \vee a = c(floor)) \\
&\Phi(garbage, res(a, s)) \iff a = c(fridge) \vee \\
&\quad \Phi(garbage, s) \wedge \neg(a = empty\text{-}garbage)
\end{aligned} \tag{2}
$$

An obvious notion of progress $P_{\mathcal{KC}}$ would clean appliances in the order suggested above. But how do we come up with such a notion automatically? Here is one approach:

4.2 A Measure of Progress for Kitchen Cleaning

Assume $goal(s) \equiv_{abbrev} \Psi(x_1, s) \wedge ... \wedge \Psi(x_n, s)$, where Ψ appears positively. The successor state axioms are rewritten in terms of Ψ, and the positive and negative causes for $\Psi(x, \cdot)$ under a are denoted by $\gamma^+_\Psi(x, a, s)$ and $\gamma^-_\Psi(x, a, s)$. For the Kitchen Cleaning Domain, $\Psi(x, s) \equiv \Phi(x, s)$, except that $\Psi(garbage, s) \equiv \neg\Phi(garbage, s)$.

Definition 2 ($<_P$ ordering)

$$x <_P y \equiv_{def} (\exists a\, s)[\gamma^+_\Psi(x, a, s) \wedge \gamma^-_\Psi(y, a, s)]$$

This ordering suggests we should make $\Psi(x, s)$ true before $\Psi(y, s)$, since in accomplishing $\Psi(x, s)$ it is possible that we could make $\Psi(y, s)$ false. $<_P$ is a shortsighted, weakened version of the *reasonable* ordering relation $\leq_r$ presented in (Koehler & Hoffmann 2000).

Definition 3 (P_{simple})
P_{simple} is one of the simplest constructions we can produce:

$$P_{simple}(x, s) \iff \Psi(x, s) \wedge \bigwedge_{y <_P x} P_{simple}(y, s)$$

Definition 4 (Well-founded Relation)
A relation $\prec$ is well-founded *on a set S if every nonempty subset $A \subseteq S$ contains a $\prec$-minimal element:*

$$(\forall A \subseteq S)[A \neq \emptyset \implies (\exists x \in A)(\forall y \in A)[\neg y \prec x]]$$

With these definitions, we can prove when P_{simple} is a strong measure of progress:

Theorem 3 (P_{simple} is a Strong Measure of Progress.)
Assume that T entails that $<_P$ is well-founded over $Objects$, and for any situation s, every $<_P$ minimal element v has an action a_v such that $\gamma^+_\Psi(v, a_v, s)$. Then P_{simple} is a strong measure of progress.

Proof Sketch: If $\neg goal(s)$, $(\exists x)\neg P_{simple}(x, s)$. The $<_P$-minimal element v over the non-empty set $\{x \mid \neg P_{simple}(x, s)\}$ is the extra object moved into P_{simple} using action a_v. All other objects are guaranteed to stay in P_{simple} due to how P_{simple} is defined, with the help of the well-founded rule of induction. ■

In practice the above requirements are not too restrictive. The well-foundedness of $<_P$ is the same as requiring that every set A of objects contains one "protected" object v, such that we can make $\Psi(y, res(a, s))$ hold for any $y \in A$ without causing $\Psi(v, res(a, s))$ to be false. The well-foundedness hints at the ordering of putting objects into P so that they won't ever have to be taken out.

The additional requirement that $\Psi(v, res(a_v, s))$ be achievable is also not too restrictive. Usually we plan in spaces where actions "chain," that is, for every action a there is another action b which enables a and doesn't do much else. When actions have no prerequisites this is immediately satisfied.

From Theorem 2 we know the Kitchen Cleaning Domain has a strong measure of progress. We include it here, constructed by means of Definition 3 and Theorem 3, for the reader's benefit:

$$
\begin{aligned}
P_{\mathcal{KC}}(fridge, s) &\iff \Psi(fridge, s) \\
P_{\mathcal{KC}}(stove, s) &\iff \Psi(stove, s) \\
P_{\mathcal{KC}}(counters, s) &\iff \Psi(counters, s) \wedge P_{\mathcal{KC}}(fridge, s) \\
P_{\mathcal{KC}}(floor, s) &\iff \Psi(floor, s) \wedge P_{\mathcal{KC}}(fridge, s) \wedge \\
&\qquad P_{\mathcal{KC}}(stove, s) \\
P_{\mathcal{KC}}(garbage, s) &\iff \Psi(garbage, s) \wedge P_{\mathcal{KC}}(fridge, s) \\
P_{\mathcal{KC}}(sink, s) &\iff \Psi(sink, s) \wedge P_{\mathcal{KC}}(floor, s) \wedge \\
&\qquad P_{\mathcal{KC}}(counters, s)
\end{aligned}
$$

We have shown how to generate a strong 1-ary measure of progress, and we can generalize this for any n by replacing single variables above with tuples of variables. Note that the assumptions for Theorem 3 must be proven within T, which means extra facts (static constraints, for example) can be used to verify the assumptions.

$T_{\mathcal{KC}}$ is extremely basic. There are no preconditions for any action – each is immediately achievable. Hence none of the goals need to be regressed through their preconditions. This lack of regression is demonstrated by the fact that the measure of progress, $P_{\mathcal{KC}}$, has the same arity as Ψ – the $<_P$ relation encodes dependencies between each goal, so the only state left for $P_{\mathcal{KC}}$ to record is whether it has achieved its part of the goal or not.

5 A Tiered Measure of Progress

In this section we present another template for domains where goals need to be regressed, but the goals are all uniform. By uniform, we mean that there is one sequence $\bar{a}(\bar{x})$ of action schema, such that each goal conjunct can be made true by applying some subsequence of $\bar{a}$, with the proper instantiation. We also assume that we can achieve some goal conjunct without negatively interfering with the rest. A domain that fits this description is the Logistics World (Veloso 1992), where packages are transported between cities by airplanes and around cities by trucks. Any sub-sequence of sending a truck to a package's location, picking it up, driving it to the airport, putting it on a plane, flying it to its destination city, loading it onto a truck, and then delivering it will put any package in its goal location. If we ignore goal locations for the trucks and planes we do have uniformity.

We formalize the properties described above in logic:

Definition 5 (Tiered Uniformity)
A planning domain T has the tiered uniformity *characteristic if there exists a collection of fluent formulas $\Theta_1(x,s),...,\Theta_n(x,s)$ such that:*

1. *$goal(s) \equiv_{def} \bigwedge_x \Theta_n(x,s)$*
2. *The $\Theta_i(x,s)$ are a partition, in particular, $(\exists k)\Theta_k(x,s)$, and $\Theta_i(x,s) \implies (\forall j \neq i)\neg\Theta_j(x,s)$.*
3. *If there is an object z such that $\neg\Theta_n(z,s)$ (goal incomplete), then $(\exists a)[(\exists x\ i\ j)[\Theta_i(x,s) \wedge \Theta_j(x, res(a,s)) \wedge j > i] \wedge (\forall y\ k\ l)[\Theta_k(y,s) \wedge \Theta_l(y, res(a,s)) \implies k \leq l]]$.*

Θ_n defines the fluent formula by which the goal may be expressed. The second requirement formalizes the notion that the sequence $\Theta_1(x,s),...,\Theta_n(x,s)$ represents the stages through which x may traverse on its way to its goal $\Theta_n(x,s)$. The final requirement says that if not all objects have reached Θ_n, then we can find a object x at state i, and we can move it up to a higher state Θ_j, without hurting the positions of other objects, and possibly making them better.

Definition 6 (P_{tiered})
Assume T has the tiered uniformity characteristic. Then define for each $i \in [1, n-1]$,

$$\begin{array}{rcl} P_{tiered}(x,i,s) & \equiv_{def} & \Theta_i(x,s) \vee P_{tiered}(x,i+1,s) \\ \text{and}\ \ P_{tiered}(x,n,s) & \equiv_{def} & \Theta_n(x,s) \end{array}$$

Each number is just a code for some of tuple of objects, thereby staying within our finite domain restrictions.

Each tier $P_{tiered}(x,i,s)$ corresponds to achieving the ith goal on each object x's path. If $\Theta_j(x,s)$ holds, that is, the object is already at level j, then $P_{tiered}(x,i,s)$ holds for all $i \leq j$. This encoding will obey the definition in Equation (1):

Theorem 4 (P_{tiered} is a strong measure of progress.)
If T has the tiered uniformity characteristic, then P_{tiered}, defined above, is a strong measure of progress.

Proof Sketch: The proof is just a straightforward application of Definitions 5 and 6 to Equation (1). ■

The problem of constructing a strong measure of progress for domains with tiered uniformity is pushed back to finding such tiers obeying the requirements of Definition 5. This is still a difficult problem, which will require the use of powerful domain analysis techniques. Note that the tiers constitute a partition which is mutually exclusive to the extent that advancing one object forward through stages will at worst leave other objects alone and at best push them forward as well. Perhaps we can take advantage of the mutual exclusion information used in planners such as GraphPlan to extract the tiers we require. Or, if we suspect a domain has the tiered uniformity property, we could run a known planner to solve $\Theta_n(x,s)$ for an arbitrary object x, and then test if the stages x proceeds through in the plan satisfy Definition 5.[7]

Theorem 5
Consider a planning problem in the Logistics World (Veloso 1992), with goal $\bigwedge_{p_i} At(p_i, l_i, s)$, where p_i are packages and l_i goal locations. Then we can find a strong measure of progress $P_{\mathcal{LW}}$ of the form in Definition 6.

Proof Sketch: We can define our Θ_is as the stages of moving an arbitrary package to its goal location (being at a non-goal location, no truck nearby; at the non-goal location, with truck nearby; in the truck; ...). Clearly these Θ_is will form a partition, and one can show that they obey requirement 3 in Definition 5 by reasoning by cases. ■

It turns out many other domains have the tiered uniformity characteristic, including the Gripper, AI-office, Ferry, and Briefcase domains.

Before proceeding, we introduce a useful method for combining two different strong measures of progress. This will ameliorate the constraint that the goals be uniform, since we can construct strong measures of progress for different kinds of goals, and then combine them:

Theorem 6 (Combining strong measures of progress)
Let P be a strong measure of progress with respect to $goal_P(s)$, and Q with respect to $goal_Q(s)$. Further assume that P-increasing actions do not affect Q's extension negatively and vice versa. Then the predicate defined as:

$$R(0,x,s) \equiv_{def} P(x,s)\ \ and\ \ R(1,x,s) \equiv_{def} Q(x,s),$$

with $R(i,x,s) \equiv \top$ for i different from 0 and 1, is a strong measure of progress with respect to $goal_P(s) \wedge goal_Q(s)$.

Proof Sketch: The proof is a direct application of the definitions of P and Q to Equation (1). ■

If goals for two different types of objects are non-interacting, then we can formulate measures of progress for each independently, and then combine them.

6 Orthogonal Measures of Progress

In this section we show how we can construct a measure of progress based on orthogonal measures of progress. By "orthogonal" we mean we have predicates $P(x,s)$ and $C(x,s)$, and while we prefer to increase P, sometimes it is not possible and our only option is to work on C. Thankfully, continually increasing C will lead to a point where we can again

[7] There are much more complicated versions of Logistics World where the tiered uniformity is not clear, or does not exist. If we limit a truck's capacity, we still have a strong measure of progress because we can drop off packages when necessary. If we add a fuel component, we simply refine our P to fuel up when the tank gets low. But if we only provide a finite amount of fuel, we could have deadlocks in the space (Helmert 2001).

increase P. Intuitively, P-increasing actions are meant to move directly to the goal, while C-increasing ones do not, instead sidestepping or removing obstacles to the goal.

BlocksWorld exhibits this characteristic. P defined as final position cannot by itself be a strong measure of progress, because its extension cannot always be increased (blocks could be in the way). But we can always move these offending blocks out of the way, onto the table, (in C) until we can put a block into final position P. Any instance of BlocksWorld can be solved using these two kinds of actions.

We can also visualize this phenomenon as traversing a contour map on P and C. When possible, we choose actions which move directly up the P-gradient to increase its extension. However at some situations it is not possible to move "up", and instead we move "sideways" increasing C.[8] A graphical depiction of this with respect to BlocksWorld is given in the left side of Figure 1, and defined below:

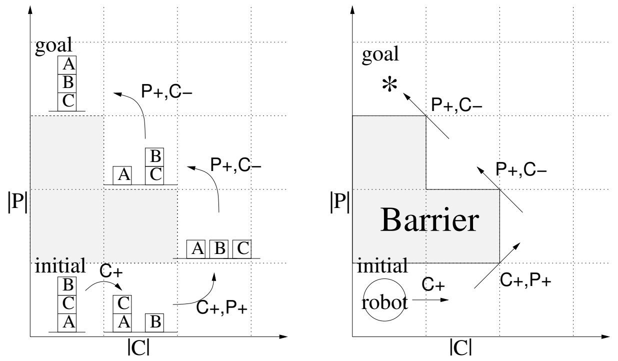

Figure 1: The left graph shows how the orthogonal measures of progress lead us to the solution. Note how they also avoid the Sussman anomaly. The right graph shows how the exact same ideas can be used to do robot motion planning.

Definition 7 (Orthogonal Measures of Progress)
P and C are orthogonal measures of progress if they satisfy:

$$\begin{aligned}
&\neg goal(s) \implies \\
&\quad (\exists a_P)[ext(P,s) \subset ext(P, res(a_P, s)) \wedge \\
&\qquad (\forall \mathbf{w})[\mathbf{C}(\mathbf{w},\mathbf{s}) \wedge \neg\mathbf{C}(\mathbf{w}, \mathbf{res}(\mathbf{a_P}, \mathbf{s})) \implies \\
&\qquad\quad \mathbf{P}(\mathbf{w}, \mathbf{res}(\mathbf{a_P}, \mathbf{s}))]] \vee \\
&\quad (\exists a_C)[ext(P,s) = ext(P, res(a_C, s)) \wedge \\
&\qquad ext(C,s) \subset ext(C, res(a_C, s)) \wedge \\
&\qquad (\forall \mathbf{w})[\neg\mathbf{C}(\mathbf{w},\mathbf{s}) \wedge \mathbf{C}(\mathbf{w}, \mathbf{res}(\mathbf{a_C}, \mathbf{s})) \implies \\
&\qquad\quad \neg\mathbf{P}(\mathbf{w},\mathbf{s})]]
\end{aligned}$$

The definition is a lexicographic ordering over P and C, except for the parts in bold, which stipulate that any loss in C due to a P-increasing action must be made up by an increase in P, and the only objects which can be moved into C in a C-increasing action must not be in P.

We can encode P and C as a binary predicate, which does in fact satisfy Equation (1):

Theorem 7
Let P and C be orthogonal measures of progress. Define:

$$\begin{aligned}
R(0,x,s) &\equiv_{def} C(x,s) \vee R(1,x,s) \\
\text{and } R(1,x,s) &\equiv_{def} P(x,s),
\end{aligned}$$

with $R(n,x,s) \equiv_{def} \top$ for all other possible values for n.

Then $R(x,y,s)$ is a strong measure of progress with respect to $goal$.

Proof Sketch: The proof is a straightforward application of Definition 7 and Equation (1). ■

The *deadlocked sets of blocks* defined in (Gupta & Nau 1991) correspond to the situation where we cannot increase P, but can increase C, thus resolving the deadlock. The form of R and Theorem 1 reassures us of the well-known result that any plan following our strategy will take at most $2m$ steps, where m is the number of misplaced blocks in our domain.[9]

This orthogonal notion can be generalized to multiple predicates. It applies to other domains as well. For example, in robot motion planning, P represents the greedy act of moving directly towards a goal. If there were no obstacles, this would be a strong measure of progress. In the presence of obstacles, C, which directs the robot to move around an obstacle, will be required as shown in the right side of Figure 1.[10]

7 Conclusions and Discussion

In this paper not only have we demonstrated a new paradigm for expressing a logical (as opposed to numeric) measure of progress for planning, but have shown how to actually construct one for certain kinds of domains. The intuitions behind our measure of progress reflects those of a human's, and by the construction proving Theorem 2, we know strong measures of progress can express optimal plans. We prefer obvious strong measures of progress however because they are more likely to elucidate the structure of domain. But for non-trivial domains, obvious measures of progress cannot be optimal. This is for the same reason an NP-complete problem is so difficult; if it were obvious how to exploit its structure to generate efficient solutions, the problem would be easy to solve. Nevertheless, we believe obvious measures of progress will generate "good enough" plans, for two reasons. First, by definition they divide up the domain to promote efficiency; subgoals are never undone. Secondly, they

[8] It is possible to have available at a situation both a P-increasing action a_P and a C-increasing action a_C. If we stubbornly prefer a_C over a_P we will still reach a (less efficient) solution; eventually we will get to a point where C's extension is full and cannot be increased, in which case a_P will be the only action available.

[9] Remember that $R(\cdot, s)$ can only be false on the tuples $\langle 0, x\rangle$ and $\langle 1, x\rangle$.

[10] Technically, Figure 1 is incorrect for the robot example; the P-gradient instead of pointing up should always point towards the goal, while the C-gradient will be orthogonal to it. Pictorially, the P-gradient will be rays pointing toward the goal location, while the C-gradient is a set of concentric circles centered around the goal. Clearly P and C in BlocksWorld play analogous roles in the robot motion planning domain.

are concisely represented and therefore not too complicated. This generally means that P has a low arity, so that by Theorem 1 the plans will not get too long. In short, obvious measures of progress will take advantage of enough salient structural features to remain efficient, without getting too complicated. They will give rise to self-explanatory strategies.

Furthermore, it should be easy to improve measures of progress, to make them "good enough," when they do generate inefficient plans. In general the reason why a plan is terribly inefficient (such as moving only one ball at a time in Gripper), is easily articulated, as are the immediate improvements. Since our measure of progress is declaratively specified, encoding these improvements as a more efficient P is straightforward.

There are a number of restrictions which apply to our paradigm. The most pressing is the fact that a strong measure of progress exists only in those domains without deadlocks. Clearly, some measure of progress exists in domains with deadlocks; if anything humans discover a measure of "regress" that is used to avoid deadlock-causing actions. Equation (1) must be weakened if we want a measure of progress for such domains. One way is to find a formula $R(s)$ true only of those states reachable from the goal, and apply Equation (1) over those situations:

$$(R(s) \wedge \neg goal(s)) \implies (\exists a)[ext(P, s) \subset ext(P, res(a, s))].$$

However we must be careful because this essentially expresses a tautology and may lack any structural content.

Another issue is the apparent lack of concurrency in situation calculus. The situation calculus can be extended to handle multiple actions in parallel using the techniques in (Reiter 1996), while still staying within our framework. Then we can define progress in terms of groups of parallel actions, leading to even more efficient plans.

Another minor extension would handle domains with more expressive ramifications. The current ADL formalizations of domains can handle some ramifications, either by artifice of the language, or by constructs such as conditional effects. (McIlraith 2000) provides successor state axioms that include the effects of ramifications, if essentially no circularities exist between the ramification fluents. These successor state axioms are syntactically adaptable to our paradigm.

The insightful reader will recognize that the strong measure of progress encodes a goal ordering. Our work is parallel to (Koehler & Hoffmann 2000), in that we both want to steadily increase the truth of our set of goals. (Koehler & Hoffmann 2000) derives a partition $G_1, ..., G_k$ over the set of goals, and then achieves G_1, and then from that state achieves $G_1 \cup G_2$, etc. In practice this works well. (Korf 1985)'s approach is similar. Korf learns efficient strategies for solving problems such as Rubik's Cube by searching for *macro-operators*, a sequence of primitive operators, and an ordering of the goal conjuncts $g_1, ..., g_n$ such that every goal conjunct g_i has a macro m_i which accomplishes g_i, without changing the truth of conjuncts $g_1, ..., g_{i-1}$ (although they may change during the course of the macro).

However, in these above approaches, intermediate subgoals are hidden away in the intermediate plans/macros. On the other hand, (Porteous, Sebastia, & Hoffmann 2001) extracts *landmarks* (subgoals true on every path to the goal) which can be used to break down a planning task into many smaller ones. (Porteous, Sebastia, & Hoffmann 2001) also approximates orderings which are *natural* (necessary orderings of landmarks) and *weakly reasonable* (orderings on subgoals which prevent unnecessary actions). Our P encodes similar information. For example, the typical final position heuristic for BlocksWorld will avoid the Sussman anomaly by noting that B is not really in its goal position until C is, as depicted in the left side of Figure 1. In fact, we can interpret each instance of $P(\bar{x}, s)$ as a fluent formula encoding a subgoal and the degree it has been achieved. For Logistics World, it encodes the package's progress through the various transports. For BlocksWorld it encodes whether a block is in clear or in final position.

The strategy function $a(s)$ briefly referred to has important uses. (Bryson 2001) points out that $a(s)$ is a teleoreactive (TR) program (Nilsson 1994). In the examples we have encountered, proving that a predicate P is a strong measure of progress leads to construction of rules of the form:

$$\neg goal(s) \wedge \phi_i(s) \implies a(s) := a,$$

where ϕ_i is some fluent condition on the situation, $\bigvee_i \phi_i(s) \equiv \top$, and a is the action that increases $ext(P, s)$. Therefore from $a(s)$ we can construct a *universal* TR program that will achieve $goal$.

Finally, we have shown how to construct measures of progress for some simple domains, and even how to combine them (Theorem 6). Interesting future work would extend this arsenal of measures, and create an algebra to combine them. We should also look into refining the measures themselves – clearly an action that moves multiple objects into P is better than one which moves only one, especially for domains such as Logistics World or Gripper. We can construct more elaborate versions of Equation (1) which will select these better actions and avoid inefficient plans.

8 Acknowledgments

The author would like to thank John McCarthy and Tom Costello for first pointing out this problem, and members of the Formal Reasoning Group and the Logic Group for intriguing observations, comments, and feedback. We also thank the anonymous referees for their valuable suggestions and directions for improvement. This research has been partly supported by SNWSC contract N66001-00-C-8018.

References

Bacchus, F. 2000. AIPS 2000 Planning Competition Webpage[11].

Bonet, B., and Geffner, H. 2001. Heuristic Search Planner 2.0[12]. *AI Magazine*.

[11]http://www.cs.toronto.edu/aips2000/

[12]http://www.ai.ldc.usb.ve/˜ hector/software/hsp2.ps

Bryson, J. 2001. Personal communication.

Doherty, P., and Kvarnström, J. 2001. TALplanner: A Temporal Logic Based Planner[13]. *AI Magazine*.

Edelkamp, S., and Helmert, M. 2001. The Model Checking Integrated Planning System (MIPS)[14]. *AI Magazine* 67–71.

Fox, M., and Long, D. 2001. Hybrid STAN: Identifying and managing combinatorial optimisation sub-problems in planning[15]. In *IJCAI-01*.

Gupta, N., and Nau, D. S. 1991. Complexity results for blocks world planning. In *AAAI-91*.

Helmert, M. 2001. On the Complexity of Planning in Transportation Domains[16]. In *ECP'01*, Lecture Notes in Artificial Intelligence. New York: Springer-Verlag.

Hoffmann, J. 2001a. FF: The fast-forward planning system. *AI Magazine*.

Hoffmann, J. 2001b. Local Search Topology in Planning Benchmarks: An Empirical Analysis[17]. In *IJCAI*, 453–458.

Koehler, J., and Hoffmann, J. 2000. On Reasonable and Forced Goal Orderings and their Use in an Agenda-Driven Planning Algorithm[18]. *Journal of Artificial Intelligence Research* 12:338–386.

Korf, R. 1985. *Learning to Solve Problems by Searching for Macro Operators.* Ph.D. Dissertation, Carnegie Mellon University.

Lin, F. 2002. Personal communication.

Long, D., and Fox, M. 2000. Automatic synthesis and use of generic types in planning. In *AIPS-00*, 196–205.

McAllester, D. 1991. Some Observations on Cognitive Judgements[19]. In *AAAI-91*, 910–915. Morgan Kaufmann Publishers.

McCarthy, J. 1959. Programs with Common Sense[20]. In *Mechanisation of Thought Processes, Proceedings of the Symposium of the National Physics Laboratory*, 77–84. London, U.K.: Her Majesty's Stationery Office.

McIlraith, S. 2000. An axiomatic solution to the ramification problem. *Artificial Intelligence* 116:87–121.

Nilsson, N. J. 1994. Teleo-Reactive Programs for Agent Control[21]. *Journal of Artificial Intelligence Research* 1:139–158.

Nilsson, N. 1998. *Artificial Intelligence: A New Synthesis*. Morgan-Kaufman.

Parmar, A. 2002. A Logical Measure of Progress for Planning (Technical Report)[22]. Technical report, FRG.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the Extraction, Ordering and Usage of Landmarks in Planning[23]. In *Proceedings of ECP'01*.

Reiter, R. 1996. Natural Actions, Concurrency and Continuous Time in the Situation Calculus[24]. In Aiello, L.; Doyle, J.; and Shapiro, S., eds., *Proceedings KR96*, 2–13.

Sierra-Santibañez, J. 2001. Heuristic planning: a declarative forward chaining approach. In *Working Notes of Common Sense 2001*, 228–234. Fifth Symposium on Logical Formalizations of Commonsense Reasoning.

Veloso, M. 1992. *Planning and Learning by Analogical Reasoning*. Ph.D. Dissertation, School of Computer Science, Carnegie Mellon University.

[13] ftp://ftp.ida.liu.se/pub/labs/kplab/people/patdo/www-aimag.ps.gz

[14] http://citeseer.nj.nec.com/edelkamp00model.html

[15] http://www.dur.ac.uk/~dcs0www/research/stanstuff/Papers/sigpaper.ps

[16] http://www.informatik.uni-freiburg.de/~helmert/publications/ECP01_Complexity.pdf

[17] http://www.informatik.uni-freiburg.de/~hoffmann/papers/ijcai01.ps.gz

[18] http://www.informatik.uni-freiburg.de/~hoffmann/papers/jair00.ps.gz

[19] http://www.autoreason.com/aaai91a.ps

[20] http://www-formal.stanford.edu/jmc/mcc59.html

[21] http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume1/nilsson94a.ps

[22] http://www-formal.Stanford.edu/aarati/techreports/aaai-2002-tr.ps

[23] http://www.dur.ac.uk/~dcs0www/research/stanstuff/Papers/PorteousSebastiaHoffmann_ecp_01.ps.gz

[24] http://www.cs.toronto.edu/cogrobo/natural.ps.Z