Using concrete, perceptually based representations to avoid the frame problem*

Scott B. Huffman and John E. Laird
Artificial Intelligence Laboratory
The University of Michigan
1101 Beal Ave.
Ann Arbor, Michigan 48109-2122

Abstract

Real-world planners must be able to temporally project their external actions internally. Typically, this has been done entirely at an abstract representational level. We investigate an alternative approach, performing projections on a *concrete*, property-based representation, and rederiving the abstract level from it. We show that this approach can greatly alleviate the frame problem in certain types of spatial domains, while maintaining the advantages of planning at an abstract level. We present a comparison of the approaches in various object-manipulation domains, including the results of a re-implementation of the Robo-Soar system [Laird *et al.*, 1989].

Introduction

Planners that execute actions in the "real world" require *two* implementations for each action: one for use in planning and one in execution. In execution, an action is generally implemented by issuing a command to an external peripheral; for example, issuing a *move* command to a robot. For planning, this implementation cannot be used directly: extra knowledge is needed to allow the planner to reason about the action internally. This knowledge must answer the question "How will the world change when this action is performed?"

Our work on the Robo-Soar system [Laird and Rosenbloom, 1990; Laird *et al.*, 1989] has brought us face to face with the issue of temporally projecting external actions. The original implementation of Robo-Soar performed projections entirely at an abstract level (the level of objects and symbolic relationships between them), requiring a number of fairly specific productions to update the system's model of its environment directly. However, during execution of actions, abstract relationships were derived from a relatively small amount of incoming perceptual information. This observation led us to investigate the idea of performing projections at a lower level, closer to the perceptual level.

In general, temporal projection can be a very difficult task, requiring that we predict all possible effects an action can have on possible world-states. The STRIPS assumption (that things which aren't explicitly changed stay the same) does not fully overcome the problem. We still need frame axioms to update all the possible things that executing an action *can* change (the *ramification* problem [Ginsberg and Smith, 1987]). Traditional AI systems have not had to face the full wrath of this problem because of the simplicity of their domains. They typically have only a small number of objects with a small amount of interaction between them. But for domains with many objects, and many relevant relationships between them, the amount of frame knowledge required can become exceedingly large.

However, as we will show, the amount of frame knowledge required can vary greatly depending on the *level of abstraction* at which temporal projection is performed. Typically, a system which builds a model of its environment based on perception transforms its input through a series of abstraction levels (e.g., Kaelbling [1986]). Through this process, the data is combined and abstracted, irrelevant data is dropped out, and relevant relationships are made explicit. The end result is typically an abstract, symbolic model which encodes objects and the relevant relationships between them. This "most abstract" level of representation is clearly useful for reasoning: by making all of the relevant relationships explicit, it provides a basis for proposing relevant actions, expressing goals in a general form, and learning at the proper level of generality. Although abstract in that irrelevant information has been dropped out, in Levesque's [1986] terminology, this representation is *vivid*: all relevant information is directly available without inference.

We will refer to this "most abstract" level of representation as the *abstract model*, and to the property-based representations leading up to it as *concrete* representations. Most planners perform temporal projection at the abstract model level. So, for example, dur-

*Sponsered in part by a University of Michigan Research Partnership Fellowship, and by NASA/ONR under contract NCC 2-517.

ing a look-ahead search, `move-block(A,B)` might be implemented (in part) by explicitly adding the relation `ontop(A,B)`.

An alternative is to simulate action at one of the concrete representational levels used during perceptual processing, and then from this representation to produce a new abstract model which reflects the simulated action's effects. Transforming the concrete representation into an abstract model won't require any extra machinery for the system, since this transformation must be performed during perceptual processing anyway. Also, since perception must be efficient in a real system, adding this processing to temporal projection shouldn't be computationally expensive. This approach maintains the advantages of reasoning at the abstract model level, since its final result is an abstract model reflecting the projected action, just as in the traditional approach. However, the approach requires the capability to produce a concrete representation of the current abstract model, which is a one-to-many mapping in general. In the simple spatial domains explored in this paper, this is not difficult, but in more complex domains it may be.

In this paper, we explore the possibility of performing temporal projection on concrete representations, and attempt to identify the domain characteristics that determine at which representation level projections are best performed. We find that for domains with many objects and relevant relationships between them, the concrete representation approach can give a large advantage, while for simple domains with only a small amount of interaction between objects, the approaches are nearly equivalent. In particular, the concrete approach gains an advantage when the relationships between objects at the abstract level are *implicit* in properties of individual objects at the concrete level. In this sense, it is actually the *non*-vividness of the concrete level that gives it an advantage. The concrete approach gains an additional advantage when the set of planning operators can be simulated with a smaller set of equivalent operations on the concrete representation.

Assumptions defining the problem

What is the best strategy for carrying out temporal projection of actions in spatial domains? By "best," we mean the strategy that minimizes the amount of frame knowledge required to update the system's abstract model. Our analysis is based on the following assumptions about the derivation, content, and use of abstract models by planning systems:

1. **Perceptual origin.** We assume that there are perceptual processes which transform perceptual data through some series of concrete representations, and produce the abstract model as the final result (e.g., Marr [1982]). By *concrete*, we mean representations which contain perceptually derived information that is *non-structured*; i.e., there are no inter-object relationships explicitly represented. For the object-manipulation domains we consider below, concrete representations might consist of pixel maps at the lowest levels, and objects located within a coordinate system at the highest levels. No assumption is being made about the perceptual processing itself; the point is simply that the abstract model is not primitive. The system thus contains machinery which can produce an abstract model from an underlying concrete representation.
2. **Object-based content.** The abstract model is *object-based*, containing a representation of the objects in the domain and their relevant relationships and properties. It is vivid: all relevant relationships are *explicit*, meaning that they are available without performing inference. The abstract model's usefulness derives from the availability of these relationships as a basis for action proposals, goal specifications, and learning. By *relevant* relationships, we mean those which are necessary for achieving the task(s) at hand. They appear either in a goal specification, or in a relevant intermediate state. In blocks world, `on-top-of` is a relevant relationship; `two-inches-left-of` is not. However, if a task domain required placing a two-inch space between blocks, then `two-inches-left-of` would be relevant. Also, `block-held` may be relevant in the blocks world, even if it never appears within a goal specification, if it must appear in intermediate states for goals to be reached.
3. **Planning operators based on relevant relationships.** We assume that there is a planning operator designed to achieve each of the relevant relationships. For instance, if in a particular domain `left-of` is a relevant relationship, then we assume there will be a `move-left-of(obj1, obj2)` operator. Basing operators on relevant relationships reduces the search space by reducing the set of possible actions. For the purposes of planning, our robot can't move just *anywhere*; it can only move to achieve one of a relatively small set of relationships. Of course, this is why planners use abstract representations in the first place.

Abstract vs. concrete projection

The goal of temporal projection is to produce an abstract model of the world after an action has been performed. So why even consider performing projections at a concrete level of representation? The reason is that projecting at a concrete level can allow for enormous savings on the frame knowledge required to encode an action's effects, in certain types of domains.

At first glance, this result seems counterintuitive. The most abstract level of representation has the least amount of information in it (since by abstracting, some information is discarded), so it might appear to be the easiest to update. However, although the abstract model contains the least amount of total information

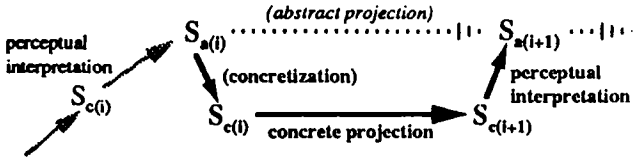

Figure 1: The concrete approach to temporal projection.

remaining from perceptual input, it may have more of the relationships which are implicit within the data represented *explicitly*. When performing temporal projection, the amount of information that is explicit is crucial: everything that is explicitly represented must be explicitly updated when an action changes it. If a large number of explicit relationships in the abstract model are derived from a smaller amount of information at a lower level, then it may be require less frame knowledge to project at the lower level. So, we see a tradeoff here. Simulating actions at lower abstraction levels may require updating irrelevant information (which will just be abstracted away); simulating at higher levels may require updating a larger number of explicit relationships. As in most tradeoffs, the right level of abstraction to perform projections at will generally fall somewhere in the middle, and will vary from domain to domain.

To compare the approaches, we will measure the amount of frame knowledge required by each in terms of "number of frame axioms," where a frame axiom is considered to update one piece of independent information. For this analysis we don't consider frame axioms with structure that would allow them to update more than one related piece of information at a time.

The knowledge required to perform projections depends on the number of actions that can be performed, and the amount of information that must be updated for each. In general, the number of frame axioms required for projecting at the abstract model level depends on:

- the number of planning operators the planner supports (call this $O_{planning}$);
- the number of objects represented in the planner's world model which these operators can manipulate (call this N_{model}), and
- the number of possible relevant relationships and properties *each object* can have (R_{am}).

In the worst case, the total number of frame axioms F_{am} required for the abstract model approach would be $F_{am} = O_{planning} \cdot N_{model} \cdot R_{am}$, since at worst, each planning operator could require updating every relationship of every object.

The concrete representation approach is diagrammed in Figure 1. The S_a's represent abstract models, and the S_c's concrete representations. $S_{a(i)}$ is produced from perception. We generate a concrete model $S_{c(i)}$ depicting the current abstract model, simulate the desired action at the concrete level to produce $S_{c(i+1)}$, and then derive a new abstract model $S_{a(i+1)}$ from the result. This process should produce the same abstract model as if we had projected the action at the abstract level directly from $S_{a(i)}$.

In some cases, $S_{c(i)}$, the original concrete representation, may be available as a result of perceptual processing. This was the case in our Robo-Soar reimplementation. In others, it may need to be rebuilt from the abstract model level. The mapping from abstract to concrete is one-to-many in general. Thus, the system needs knowledge to perform this mapping in such a way that abstract operators may be correctly simulated at the concrete level. In the simple domains dealt with in this paper, the concretization process does not appear to be difficult, but in general it may be.

Recall that the set of planning operators is determined by the set of relevant planning relationships. These relationships are not explicit within the concrete representation, so in order to project these operators on that representation, we require knowledge of each operator's effects at the concrete level. This knowledge forms a set of *concrete operators*. The mapping from planning operators to concrete operators may be one-to-one or many-to-one. In the Robo-Soar system, the physical operations provided by the PUMA arm served as our concrete operators. In some cases, many planning operators could be mapped into a single concrete operator: `move-above`, `move-left-of`, etc., all mapped into `move-to(x,y,z)` at the concrete level. In other cases, a single concrete operator was required for a single planning operator, as in `open-gripper`.

The mapping from planning operators to concrete operators will not be one-to-many, because in the worst case, a concrete operator can be specifically designed for each planning operator. Concrete operators need not correspond to physically possible operations in the world; rather, they must simply project the abstract operation within the concrete representation. For example, implementing a `pick-up` operator might take a number of robot actions, but the corresponding concrete operator need only update the final positions of the objects, and not simulate the sequence of physical actions required.

Concrete operators must simulate all effects of planning operators on the concrete representation, including any side effects. The frame problem still exists here; the difference is a matter of degree. Once an action has been simulated, the concrete representation must be transformed into a new abstract model which reflects the results of carrying out the projected action. The machinery to perform this transformation is already in place in the system, being used for perceptual processing.

For the concrete representation method, then, the number of frame axioms F_{cm} required depends on:

- the number of concrete operators required to simulate the planning operators (call this $O_{concrete}$);
- the number of objects that these operators can manipulate (this is N_{model}, from above); and
- the number of dynamic properties each object can have (P_{cm}).

$N_{model} \cdot P_{cm}$ indicates the total amount of information within the concrete representation that could require updating, so in the worst case this approach requires $F_{cm} = O_{concrete} \cdot N_{model} \cdot P_{cm}$ frame axioms.

The analysis here is rougher than the analysis of the abstract model method, since in some concrete representations "properties" may not be clearly organized by object. For example, within a pixel map, the representation of each object is spread out over many pixels (making P_{cm} very large). The examples in this paper make use of concrete representations consisting of objects at locations within an absolute coordinate system (a single dynamic property).

This analysis indicates which parameters play a key role in comparing the approaches. First, we can compare $O_{planning}$ and $O_{concrete}$. $O_{planning} \geq O_{concrete}$, since at worst, each planning operator will require a unique concrete operator to encode its effects at the concrete level, but in some situations one concrete operator might map to more than one planning operator.

Second, we can compare R_{am} and P_{cm}. When many relevant relationships are derived from a smaller amount of concrete information, $R_{am} > P_{cm}$, and the concrete method has an advantage. This can occur when the abstract model makes *explicit* relationships which are implicit within the concrete representation. Recent work on spatial modeling indicates that the set of relevant spatial relationships between objects can be quite large [Mukerjee and Joe, 1990].

Analysis for sample domains

Let's proceed by analyzing some sample domains that perform object manipulation tasks. We assume that the manipulator provides a `move-to(X,Y,Z)` command, which we will use as a concrete operator. We define a *concrete model* as a concrete representation which contains objects (identified symbolically) specified at locations within a coordinate system.

A hypothetical, highly interrelated domain

First we consider a domain that has N_{model} objects to be manipulated. The only relevant relationships are binary spatial relationships, such as `left-of`, `above`, `far-right-of`, etc. Assume there are enough of these spatial relationships (say, $R_{spatial}$ of them) that *exactly one* always holds between each pair of objects. This is a more interrelated domain than most.

Given this characterization, we can derive expressions for the number of frame axioms required by each approach. For the abstract model approach, each planning operator requires all R_{am} properties of the object being moved to be updated. Here, $R_{am} = N_{model} - 1$, since each object is related to every object but itself. This gives us:

$$F_{am} = O_{planning} \cdot R_{am} \approx O_{planning} \cdot N_{model}$$

Since there is a planning operator corresponding to each type of relevant relationship, $O_{planning} \approx R_{spatial}$. Thus $F_{am} \approx R_{spatial} \cdot N_{model}$.

All of the $R_{spatial}$ relationships are derived solely from the locational information in the concrete model, making P_{cm} (the number of concrete model properties per object) one. There is only one concrete operator to project (`move-to`). To project `move-to`, then, we must update the positions of the manipulator and the object being moved (if any). This means $F_{cm} = 2$.

Thus we see that for this hypothetical domain, the number of frame axioms required for the concrete approach is a small constant, while the number required for the abstract model approach is dependent on the number of objects and spatial relationships supported. In terms of our previous analysis, $O_{planning} \gg O_{concrete}$, and $R_{am} \gg P_{cm}$. Clearly, the concrete representation approach is a big win in a domain with such interrelated objects. In most domains, every object is not related to every other, but often the number of relationships is large enough to give the concrete approach an advantage, as the next example shows.

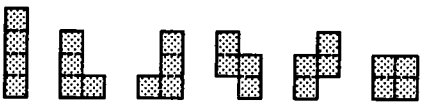

Figure 2: The set of 4-tile puzzle pieces.

A basic construction task

Consider next a puzzle task, in which the goal is to assemble a set of N-tile pieces (like those in Figure 2, for $N = 4$) into a given configuration (e.g., an $M \times M$ square). The natural abstract model to use for this problem contains each tile as an object, with four basic relationships, `left-of`, `right-of`, `above`, and `below`. Within a puzzle piece, the relationships are tagged as static. The concrete model will simply consist of tiles within a 2D coordinate system.

The planning operators will move pieces[1] to create one of the four relationships between any pair of tiles.[2] To temporally project these operators at the abstract level requires that, in addition to the four basic relationships, each tile be related directly to every tile within five "steps" of itself, where a step is a traversal of one of the four basic relationships. To see why, consider the situation shown in

[1] We ignore operators to rotate pieces. The analysis for those operators is similar to that presented here.

[2] If tiles aren't allowed to overlap, these operators will require complex pre-conditions. That is another part of the frame problem, which we won't consider here.

Figure 3: C and D must be related to project `move-left-of(A,C)` correctly.

Figure 3. We would like to project the operator `move-left-of(A,C)`, but to determine that this action will also leave B `left-of` D, there must be some relationship between tiles in the two horizontal pieces. For example, the relationship between tiles C and D might be `one-step-right-two-steps-down`. Relationships within five steps are required to ensure a relationship with the tile adjacent to the far end of a moving piece. Sixty tile spaces can be reached within five steps, so we need 56 relationships in addition to the four basic ones.[3]

There are four planning operators, and each causes four tiles in one of the six possible types of pieces to move. Each of the four tiles has 60 relationships that may be updated. Thus, projection at the abstract model level requires:

$$F_{am} = 4 \text{ ops} \cdot 6 \text{ pieces} \cdot 4 \text{ tiles/piece} \cdot 60 \text{ rels/tile} = 5760$$

The reader might well complain at this point that this is just a poor way to tackle this problem. And that's exactly the point. We're trying to firm up the intuition that performing temporal projection at an abstract level, even in an only slightly complex domain such as this, is a bad idea.

Performing projections on the concrete model representation is dramatically simpler for this domain. We need frame axioms to update the locations (within the 2D grid) of the four tiles of the piece being moved. We only have a "handle" on one of the four, specified in the operator being projected, so we have to trace through paths of static relationships to find the three others (four possible directions at each of the three steps). Thus a naive implementation would require:

$$F_{cm} = 4 \cdot 4 \cdot 4 = 64$$

frame axioms. We could reduce this number by "trading time for space," taking four iterations to perform the position updates. This method would require only four frame axioms.[4]

[3]One might imagine an alternative in which there is a static structure of positions onto which pieces are placed, and the positions have all 60 relationships with other positions pre-computed. However, this is a concrete approach: there are no relationships explicit between the *objects* themselves. The approach is essentially equivalent to using a 2D coordinate system (except it's messier).

[4]This kind of tradeoff of time for space is not possible in the abstract model case, because there is no way to express the effect of the action independently of the relationship being formed between the moving tile and its destination. In the concrete model case, we can express the motion as an offset from the starting x,y position, and future iterations make use of this offset.

Planning Operators:		*Concrete Operators:*	
Operator:	Axioms:	Operator:	Axioms:
PushB(b2,b1)	3 (next-to's)	Move-to(x,y)	1
GotoBox(b)	1 (next-to)	Push-to(x,y)	2
GotoDoor(d)	1 (next-to)	Open-door	1
GoThruDr(d, r)	1 (inroom)	Close-door	1
PushToDr(b,d)	2 (next-to's)		
PushThruDr(b,d,r)	2 (inrooms)		
Open(d)	1 (status)		
Close(d)	1 (status)		
Total:	12	*Total:*	5

Table 1: Planning and Concrete Operators for STRIPS domain. (b=box; r=room; d=door)

Robo-Soar and STRIPS

These observations were motivated by our work on a re-implementation of the Robo-Soar system [Laird and Rosenbloom, 1990; Laird *et al.*, 1989]. Robo-Soar controls a PUMA robot arm, and receives perceptual feedback from an overhead camera. Its task is lining up blocks.

The original implementation of Robo-Soar used the abstract model approach for temporal projection. Even with the small number of objects and relevant relationships in the domain, 26 productions were required to encode the temporal projection knowledge (for the eight planning operators). Our re-implementation performs projections at the concrete model level. The commands provided by the PUMA arm interface were well matched to our concrete model, allowing us to use the physical operations of the robot as our concrete operators. The four PUMA operations required seven productions to simulate. After simulation, Robo-Soar's perceptual knowledge was applied to produce a new abstract model reflecting the effects of the projected action. The difference in frame knowledge required is significant, despite the domain's simplicity.

Finally, consider the STRIPS domain. The only dynamic relevant relationships in the domain are `NEXT-TO(obj1, obj2)`, `INROOM(obj, room)` and `STATUS(door, open/closed)`. We assume that the robot's environment has a 2D coordinate system imposed upon it. The set of planning operators required, taken from [Fikes *et al.*, 1972], and the concrete operators we posit to support them, are shown in Table 1.

Table 1 indicates the number of frame axioms required by each planning and concrete operator. As the table indicates, for this domain, $F_{am} = 12$, while $F_{cm} = 5$. This domain is an example of a situation where the concrete representation approach gives only a marginal benefit, because of the very limited amount

of interaction between objects (R_{am} is small).

Conclusions

We have presented an alternative to performing temporal projection entirely at an abstract, relational level. Performing projections on a concrete representation, and then re-deriving the abstract level, can greatly alleviate the frame problem in certain types of domains. In particular, the method has an advantage when a larger number of planning operators can be simulated by a smaller number of concrete operators, and when relevant relationships which are explicit within the abstract representation are represented implicitly in the concrete representation. Since everything which is explicitly represented must be explicitly updated when an action affects it, in such cases the concrete approach allows projection knowledge to be encoded more compactly. The vividness of the abstract model, which makes retrieval of relevant information for reasoning and learning easy, also makes updating the model more difficult.

Others have pointed out the computational advantages of using concrete representations. Newell [1990] and Lindsay [1988], among others, have discussed the advantages of representations that are analogues of the domain for performing inference in a non-deductive fashion. In essence, the representational *medium* encodes the constraints of the domain, instead of their being axiomatized. The analysis presented here gives some indication of just how great that advantage can be, even for simple spatial domains. Other approaches to the frame problem, such as the Possible Worlds Approach [Ginsberg and Smith, 1987] and the Possible Models Approach [Winslett, 1988] have shown the advantage of encoding domain constraints axiomatically, rather than embedding them within operator definitions (as in STRIPS). In these approaches, the domain constraints are used to deductively derive a minimal set of updates when an action takes place. Similarly, in truth maintenance systems [Doyle, 1979], domain constraints are encoded by explicit dependency structures. In our approach, the domain constraints are instead encoded by the medium of the concrete representation and the perceptual knowledge of the system. Rather than using an additional deductive mechanism to compute state updates, the system's existing perceptual mechanisms are used.

One caveat of our analysis, however, is that we have not considered methods of structuring frame axioms to compact them. To the extent that it is possible, encoding frame knowledge as general domain constraints rather than axioms tied to particular operators will reduce the number of axioms. However, the number of domain constraints will still be proportional to the number of relevant relationships in the domain [Ginsberg and Smith, 1987]. In addition, there may be ways to trade space for time, structuring the model update through multiple iterations. This method might be used to compact the frame knowledge required for either the abstract or the concrete approach.

One potential weakness of the concrete approach is the cost of generating a concrete representation to perform projections on. In some cases, the representation may be available from perceptual processing, but in others it may need to be generated from the abstract model. This generation is a one-to-many mapping in general, requiring that the system have knowledge to choose the appropriate concrete model. For some types of domains and abstract operators, this could be a difficult issue. Some form of perceptual episodic memory might be useful in generating the correct concrete models.

Acknowledgments

Thanks to Dave Kortenkamp and Clare Congdon for useful comments, and to Mike Hucka, Eric Yager and Chris Tuck for implementing the original Robo-Soar system.

References

[Doyle, 1979] Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[Fikes *et al.*, 1972] R. E. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[Ginsberg and Smith, 1987] M. Ginsberg and D. Smith. Reasoning about action I: A possible worlds approach. In *The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop*, 1987.

[Kaelbling, 1986] L. Kaelbling. An architecture for intelligent reactive systems. In M. Georgeff and A. Lansky, editors, *Reasoning about actions and plans*. 1986.

[Laird and Rosenbloom, 1990] J. Laird and P. Rosenbloom. Integrating execution, planning, and learning in Soar for external environments. In *AAAI-90*, pages 1022–1029, 1990.

[Laird *et al.*, 1989] J. Laird, E. Yager, C. Tuck, and M. Hucka. Learning in tele-autonomous systems using Soar. In *Proceedings of the NASA Conference on Space Telerobotics*, 1989.

[Levesque, 1986] H. Levesque. Making believers out of computers. *Artificial Intelligence*, 30:81–108, 1986.

[Lindsay, 1988] R. Lindsay. Images and inference. *Cognition*, 29:229–250, 1988.

[Marr, 1982] David Marr. *Vision*. 1982.

[Mukerjee and Joe, 1990] A. Mukerjee and G. Joe. A qualitative model for space. In *AAAI-90*, pages 721–727, 1990.

[Newell, 1990] A. Newell. *Unified Theories of Cognition*. Harvard University Press, 1990.

[Winslett, 1988] M. Winslett. Reasoning about action using a possible models approach. In *AAAI-88*, pages 89–93, 1988.