Bimodal Spatial Reasoning with Continuous Motion

Samuel Wintermute and John E. Laird

University of Michigan
2260 Hayward St.
Ann Arbor, MI 48109-2121
{swinterm, laird}@umich.edu

Abstract

Symbolic AI systems typically have difficulty reasoning about motion in continuous environments, such as determining whether a cornering car will clear a close obstacle. Bimodal systems, integrating a qualitative symbolic system with a quantitative diagram-like spatial representation, are capable of solving this sort of problem, but questions remain of how and where knowledge about fine-grained motion processes is represented, and how it is applied to the problem. In this paper, we argue that forward simulation of motion is an appropriate method, and introduce continuous motion models to enable this simulation. These motion-specific models control behavior of objects at the spatial level, while general mechanisms in the higher qualitative level control and monitor them. This interaction of low- and high-level activity allows for problem solving that is both precise in individual problems and general across multiple problems. In addition, this approach allows perception and action mechanisms to be reused in reasoning about hypothetical motion problems and abstract non-motion problems, and points to how symbolic AI can become more grounded in the real world. We demonstrate implemented systems that solve problems in diverse domains, and connections to action control are discussed.

Introduction

As research in cognitive architectures progresses to address spatial problems, systems containing both high-level qualitative symbolic and low-level diagram-like spatial representations have been proposed (Chandrasekaran 2006, Wintermute & Laird 2007). These bimodal systems share many similarities to psychologically based systems for visuospatial imagery, which similarly propose multiple levels (Lathrop & Laird 2007, Kosslyn et al. 2006).

Problem solving in these systems progresses by the symbolic level extracting out qualitative predicates from the diagram (through what we will call a *query*), reasoning over these extracted predicates, and creating new objects in the diagram, or *images,* by projecting qualitative predicates into it. For example, in Figure 1, the system is attempting to place a new object in the diagram such that it does not intersect existing objects. By repeatedly querying the diagram, reasoning over the extracted predicates to determine new predicates to project back into the diagram as images, and extracting further predicates that are the implications of those images, the system is able to solve the problem. The bimodal approach to spatial reasoning has many inherent advantages. Symbolic systems have well-established means for using both general and specific knowledge to dynamically control behavior, and they can easily reason using hypothetical situations and a wide variety of existing learning algorithms. Adding a spatial level changes none of this, but allows the system to leverage perceptual representations and processes in the aid of reasoning (Chandrasekaran, 2006).

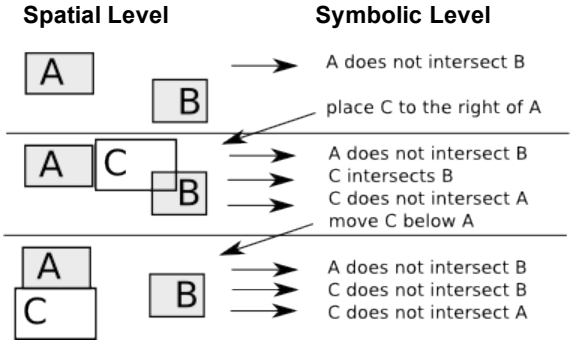

Figure 1. Solving a bimodal spatial problem. (from Wintermute & Laird, 2007).

While bimodal reasoning is powerful, it is difficult to see how the knowledge to solve spatial problems involving non-uniform motion can be encoded. For example, consider the problem of predicting if a turning car will hit an obstacle. One approach to this problem might be to project an image into the diagram that outlines the path that the car would follow around the corner, and check if this image intersects the obstacle. However, this presupposes that the system knows exactly what path the car will follow and has the ability to construct an image in that shape. For cases where the car is traveling in a straight line, this is feasible – the path image is a rectangle connecting the car to its destination. In more complex cases, the path image may be difficult to construct and represent, such as when a car transcribes a path that is partially curved and partially straight. Moreover, a car with alignment problems might traverse a path that approximates a sine wave.

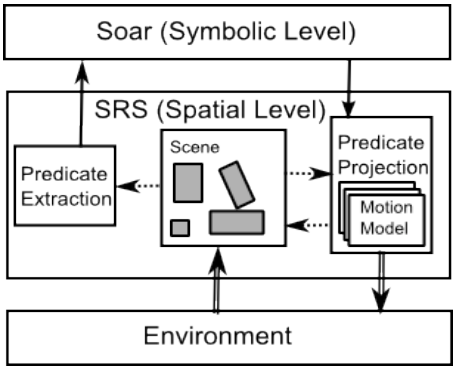

Figure 2. Our bimodal spatial reasoning system. Solid arrows show qualitative data, dotted arrows show quantitative spatial data, double arrows show raw perceptual and motor data.

For a general agent that is capable of the same range of behavior as humans, there are many diverse types of motion that the system may need to predict. To plan its own behavior, a general agent needs to predict the outcome of actions it can perform in the environment, both relative to its own body ("What would happen if I reached for object X?"), and relative to objects it controls indirectly ("What would happen if I tried to back my car into this parking spot?"). In addition, it is useful for the agent to predict the motions of others and objects not under the control of any agent, such as the path of a bouncing ball. In general, the system should be able to internally reason about motions perceived in the environment.

However, it is very difficult (if not impossible) for an agent to reason about an arbitrary motion by generating an equivalent geometric shape, as can be done with a car moving in a straight line. This is roughly equivalent to requiring the system to have a closed-form definition of the result of any movement it might reason about. A more straightforward way to solve this sort of problem is through continuous simulation: the system executes a forward model of the motion of a car, and observes whether or not the car hits the obstacle.

The general concept of combining qualitative symbolic reasoning with continuous simulation is not new (Funt, 1980 is an early example), but little work has been done to examine how, in general, continuous motion can be integrated into a bimodal system. We propose to encode this information in the form of a continuous motion model, within the spatial level of the system. By transforming one continuous spatial state to another, these motion models provide the fine-grained quantitative resolution needed for detecting qualitative interactions between objects that is critical for correct reasoning in the symbolic level, which can "observe" the simulation through predicate extraction.

The advantages of this approach at integration is that the detailed quantitative knowledge about how a particular motion unfolds in a specific spatial environment is contained within the model, independent of knowledge about when and how that motion should be invoked, and how the results should be interpreted, which is left to the symbolic system. This combines strengths of symbolic systems, such as dynamic composition, abstraction, and generalization, with strengths associated with non-symbolic AI systems: precise situational behavior.

As will be shown, the resulting system leaves most of the goal-directed behavior in the symbolic level, resulting in simple motion models with standard interfaces. Problem solving results from the interaction between the symbolic level, which doesn't need to know the details of the motion, and a motion model in the spatial level, which doesn't need to know why it is invoked or how its results will be used.

Motion Models in Soar/SRS

We have implemented a framework for using continuous motion models during spatial reasoning within SRS (Spatial Reasoning for Soar; Wintermute & Laird, 2007), an extension to the Soar cognitive architecture (Laird 2008), as shown in Figure 2. Soar is mainly a symbolic system, corresponding to the right side of Figure 1. SRS lies between Soar's input and output modules and the environment, mediating between the high-level qualitative predicates a Soar system uses to reason about space, and the lower-level representations provided by the environment. SRS consists of a quantitative spatial short-term memory (henceforth *scene*), and mechanisms to extract and project the predicates a Soar system uses to reason about the problem. The scene is represented as a set of convex polygons in a plane. Perceptions from the environment are treated identically to images created by Soar, except for a simple annotation to differentiate them, allowing abstract and hypothetical reasoning to be accomplished with the same methods used to reason about the immediate environment.

Predicate Extraction and Projection in SRS

SRS allows for predicate extraction through symbolic query structures. These structures consist of a relationship, two objects, and a value, such as (intersect A B false). To query the scene, the qualitative level creates such a structure, possibly leaving some fields unspecified, and the quantitative level responds with all true groundings of that structure. For example, for the scene at the bottom of Figure 1, the query (intersect A ? false) would return (intersect A B false) and (intersect A C false). Note that some queries might have no valid results, such as (intersect A ? true) or (adjacent A B true) in the same figure. A query *matches* if it returns at least one valid result, and does not match otherwise.

The reverse process of predicate extraction, predicate projection, creates new objects in the scene, called images. SRS allows for two classes of images to be created, called *directly-* and *indirectly-described* images. Directly described images are unambiguous derivations of existing objects, such as "the convex hull of A and B," or "the line connecting the centers of C and D." Images of this type

may or may not exist (for example "the intersection of A and B" does not exist if A and B are discrete), but if they do exist, there is only one possible interpretation. In contrast, indirectly described images may be underdetermined, where multiple images match the description, such as "an object shaped like A, located entirely inside of B." These classes are discussed in detail in Wintermute & Laird (2007).

Incorporating Motion Models

Although directly- and indirectly-described images are sufficient for a broad class of problems, as discussed previously, it isn't possible to map every motion onto an equivalent geometric object. The results of simulating a motion are a third class that greatly expands the representational and reasoning power of SRS. In Figure 3, for example, we might create an image of what would happen if block B came under the influence of gravity. The desired end result of the simulation might be creating the image B' on the right of the figure: a copy of the original image, in the position it is in when it hits the floor.

Using this example, we can generate a set of issues that must be addressed by a motion representation:

1. The system must know when to terminate the simulation. In this case, when the block hits the floor.
2. The qualitative system must have some access to the intermediate states of the simulation so it can detect important interactions, such as that B hits the corner of A as it falls.
3. The simulation must have access to the spatial scene. In this case, it must know the exact coordinates of B and D, since the position of B' depends on both of those.

Outside of the block example, we can list additional desirable properties of a motion representation:

4. The methods used to invoke and reason about motion should be as general as possible. For example, the changes needed for a Soar system to reason about pathfinding with a car versus a tank in the same environment should be minimal.
5. Motions should be usable in abstract situations. A car motion should be usable when the system is considering an object it chooses to treat as a car, not only when it is considering an actual car.
6. Invoking and controlling simulated motion should be similar to invoking and controlling real motion.
7. Motion simulations should be learnable from perception of motion in the environment.

We have created a system that addresses the first five of these requirements. In our system, motions are represented through *motion models* inside the spatial level of Figure 2. Motion models can be applied to any object in the scene, satisfying requirement 5, and are invoked with a step size. The model then steps forward the state of the object it is invoked for (called the *primary object*) a small amount of time. A simulation is then a sequence of motion model steps, executed by the qualitative level. Between steps,

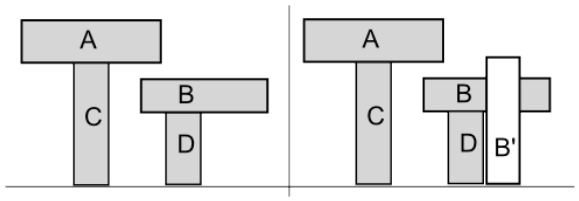

Figure 3: Falling Block Motion. Block B' on the right is the result of simulating the effect of gravity on block B. (similar to an example in Funt, 1980)

predicates can be extracted from the diagram and reasoned over, so the system can detect the collision of B and A in Figure 3, by attempting to match (intersect B ? true) between steps (requirement 2). Termination can be similarly handled. In the example, the symbolic level simply steps until (intersect B floor true) matches, satisfying requirement 1.

Note that this discretization provides a natural speed/accuracy tradeoff, with faster simulations using bigger step sizes that might miss intermediate states where queries temporarily match. Thus, the step size controls the effort an agent will expend on a given simulation.[1]

As motion models are in the spatial level, requirement 3 is met, as low-level scene information is also present there. To satisfy requirement 4, however, we should ensure that as much of the reasoning process as possible occurs in the symbolic level. In addition to the benefits gained from representing high-level reasoning symbolically, keeping the motion models themselves simple helps to keep their interfaces simple, and hence makes them more broadly applicable. Action and learning (requirements 6 and 7) are discussed later.

Implemented Examples

Several motion models have been implemented using SRS with motion simulation. Below, we describe these models and how they interact with symbolic reasoning in Soar to solve spatial problems.

Block Stability: Falling Block Model

The first implemented model is the falling block model discussed above in Figure 3. In a task inspired by those in Funt (1980), the goal of the agent is to report what would happen under the influence of gravity for each block in the scene. This motion model encodes modes of interaction for a primary object (the block that is falling), and a reference object, which it may or may not collide with. If the primary object is not touching the reference object, it moves in the direction of gravity, until the blocks touch (if ever). Once the blocks touch, the primary object may remain stationary, slide off the reference object, or rotate, depending on the configuration of the objects in relation to gravity.

[1] Event-based simulation, where the notion of time and step sizes is not present, would be extremely difficult to implement because SRS would need to perform exact calculations for every type of motion represented, determining the final location of the object for any arbitrary termination condition the agent might use.

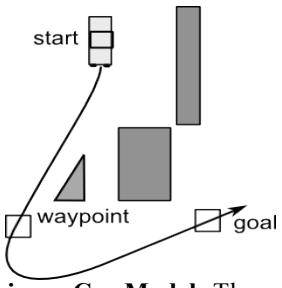

Figure 4. Using a Car Model. The system must determine a waypoint such that the car can drive to the goal without hitting any obstacles.

It is the responsibility of the symbolic level to choose an appropriate reference object for the simulation. As implemented, the system simply takes a scene as in Figure 3, and, for each block, extracts the closest object below it. It then runs the falling block motion model with this as the reference object, and reports whether it results in the block remaining stationary, falling to the floor, or colliding with another block. As a result of running simulations, Soar can gain qualitative information it did not have before: the stability of each block.

Car Path Planning: Car Model

Path planning for a car-like vehicle is a particularly hard problem, especially for symbolic AI systems like Soar, as non-holonomic properties of cars prevent the problem from being easily reduced to a graph search. Introducing a motion model for cars is then a particularly useful application of this system. As implemented, this motion model uses standard "simple car" equations (LaValle, 2006). Given a body angle, position, steering angle, and time step size, the next body angle and position can be easily calculated. As the body angle and position are present in the scene, the motion model must only determine the steering angle to use at each step, and feed it to the simple car equations to determine the next state.

A simple way for the qualitative level to provide the needed steering angle input to the model is to do it indirectly, by providing the object that is the target of the motion. The motion model can then determine the angle between the front of the car and the target, and steer in that direction, proportional to that difference, saturating at some maximum steering angle.

With this model, the qualitative system has the raw tools needed to plan a path: it determines what happens when a car tries to drive from one location to another. Then, as in Wintermute & Laird (2007) (and as will be elaborated soon), simple path planning is accomplished by decomposing the problem into a series of attempts to reach qualitatively specified waypoint images. A simple implemented example problem, and the resulting waypoint and path, is shown in Figure 4.

RTS Base Layout: Translation Models

In the computer game genre of Real-Time Strategy, one of the many sub-problems requiring spatial reasoning is

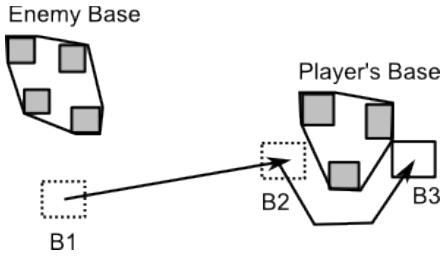

Figure 5. Placing a new building behind the player's base, relative to the enemy. Bases are the convex hulls of sets of buildings. Initially, the building is placed arbitrarily outside the player's base (B1), then it is translated toward the base (B2), then around the base to B3, where the new building is built.

laying out a clustered group of buildings with different functions, collectively called a "base." During the game, the player must strategically determine the locations of new buildings as they are added to a base. The world is viewed from the top down, so a player might describe the most desirable location for a new building with weak defensive capabilities as "near my base, but behind it in relation to the enemy's base." In Wintermute & Laird (2007), this problem was solved in SRS by creating indirect images: the system encodes an equivalent projection command to the sentence "create an image of the building, located outside of my base, near my base, and as far from the enemy's base as possible", and the location of the image is used as the location for the new building.

Internally, as it does with all indirect images, SRS determines the location by gradually ruling out potential locations for the new building. The fact that the new building is "outside of the base" means that all locations inside the polygon representing the base can be ruled out, for example. The property of being "near the base" rules out remaining locations that are not directly adjacent to the base, and "far from the enemy," when applied next, rules out all locations but those maximally far from the enemy.

While this system works well for this problem, it requires the construction of complex indirect images, so it is worthwhile to see if the problem can be solved more simply using motion models. In order to do this, two new motion models were added to SRS: the translation model and the translation-around model. The translation model is simple: given a target object, it translates the primary object towards that object. The translation-around motion model is only a bit more complicated: given a reference object adjacent to the primary object, it translates the primary object around the perimeter of the reference object, in either the clockwise or counter-clockwise direction (as specified by Soar).

With these models in place, it is possible to solve the base-layout problem using simulation instead of indirect images (Figure 5). The new building can initially be placed at some arbitrary location in the scene outside of the base (B1 in the figure). Then, the translation model can be applied to it, to move it toward the base, until (adjacent B1

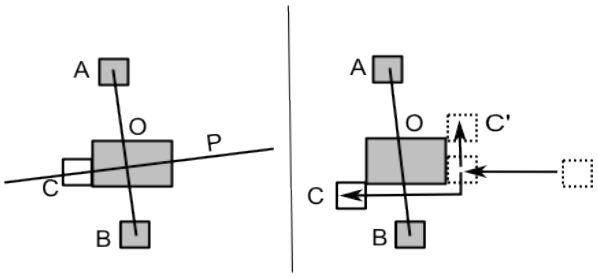

Figure 6. Two approaches for getting from A to B, around obstacle O. On the left, an indirect image is used, creating waypoint C via a temporary perpendicular line. On the right, simulation is used, where the waypoint is first placed arbitrarily, then translated towards O, then translated around O in both directions, resulting in potential waypoints C and C'.

base true) matches, resulting in B2. Then, the translation-around model can be applied, until the image is located at B3, where the building is placed.

How does the system know to stop the simulation at B3? The reason B3 is the appropriate location is that it is the furthest location from the enemy base along the perimeter of the player's base. This location can be derived in the qualitative level by monitoring the distance from the building in motion to the enemy while translating it around the base. The appropriate place to stop is where this distance reaches a local maximum, a property that can be easily detected by the symbolic system. This result is a clear-cut example of the synergy gained by using continuous models with a qualitative system. The motion model itself knows nothing about location of enemies, or the goal of its action relative to them, just how to move from one position to the next. Likewise, the qualitative level knows nothing about the specific details of the shapes it is reasoning about, which the result of the simulation critically depends upon. Between the two, though, it is easy to solve the overall problem.

Motion Models and Indirect Images

In the above example, a sequence of motions was used to create an image (the location of a new building), where previously a more complicated indirect image was needed. In general, it is possible to get much of the functionality of indirect imagery by using motion models.

Other situations also demonstrate this. In car path planning, we can decompose the problem into a sequence of generating waypoints, and attempting to reach them.[2] Attempting to reach a waypoint was discussed above, but generating a waypoint, previously accomplished through indirect imagery, can also be addressed with simulation. In Figure 6, the two approaches to waypoint generation are shown. A waypoint is simply a location on the edge of an obstacle that will help the moving object avoid it. In this case, the task is to move the object at A to B, but obstacle O blocks the direct path. Waypoint C must be generated, so a path from A to C to B avoids the obstacle.[3] One approach to generating waypoints is to first construct a line from the object to its destination (line AB), then a line perpendicular to AB, passing through the center of the obstacle (line P in the figure). An indirect image can then be used to create the actual waypoint. It is an object outside of the obstacle, near P, and near AB. This is object C on the left of the figure.

This problem can also be addressed using simulations instead of indirect images (Figure 6, right). As in the base layout problem, the potential waypoint can first be placed arbitrarily then moved towards the obstacle until (adjacent waypoint obstacle true) matches. At this point, two simulations must be run: one to translate the waypoint around the obstacle clockwise, the other counter-clockwise. Both of these should be stopped when the distance from the waypoint to the line AB reaches a local maximum. In the figure, the two end conditions for these simulations are marked C and C'. Soar can then query for which of C and C' is closest to AB, which is the appropriate waypoint (C, in this case).

As seen in the figure, these techniques produce different final waypoints. In practice, performance from using either scheme is roughly equivalent, since they are both on the correct side of the obstacle. The chief functional difference between them is how much of the process of creating the waypoint is accessible to the symbolic level of the system. In the first approach (the indirect image), the waypoint is created all at once: the projection command for it is sent to the spatial level, and in the next cycle of Soar, the image is present. The process inside the spatial level to interpret this indirect command takes multiple steps as it rules out locations (as discussed in the previous section), but Soar has no access to the intermediate states of indirect imagery.

In the motion model case, the process is more explicit, and intermediate states are accessible. This is potentially advantageous, since, for example, Soar can weigh the advantages of C and second-best waypoint C' before making its decisions (maybe C' is actually closer to a location that will be moved to later, for example). In addition, with motion models, the intermediate perpendicular line is unnecessary. The conditions for indirectly constructing the waypoint are equivalent to "adjacent to O, near P, and near AB." The perpendicular is used to restrict the final condition (near AB) to choose between alternatives on the extreme left and right of the obstacle. The simulation case needs no such complication – it can reason about the extreme left and right waypoints directly, without invoking intermediate structures.

[2] This methodology is roughly similar to sampling-based motion planning (LaValle, 2006), where running the car simulation corresponds to sampling a small part of its configuration space. However, in our approach, the system controlling where to sample is a cognitive architecture, versus a specialized system as is normally used.

[3] C is actually placed adjacent to an image of a slightly larger obstacle so that it has room to clear the corner, as opposed to directly adjacent to it.

Discussion

The above examples show how continuous motion models can be integrated into a bimodal spatial reasoning system. The models exist in the spatial level and are controlled by the symbolic level issuing qualitative commands. Each model combines its qualitative command with quantitative information in the scene to transform an object in the scene to a new state. These motion models correspond to the movements of objects that the agent could have observed (in theory) in the environment, despite the fact that the system has the capability to use them in abstract situations.

This is proposed in the context of SRS, an extension to Soar. SRS shares many commonalities with SVI (Lathrop & Laird, 2007), a theory of visual imagery for Soar. SVI, in addition to supporting a quantitative spatial representation, focuses on a depictive pixel-based visual representation. This supports low-level visual operations (such as detection of visual features, and pixel-level manipulation), providing functionality not possible in a spatial representation. SRS can be considered an elaboration of the spatial level of SVI.

Motion Models and Action

The agent's own effectors are an important class of moving objects in the environment. None of the previous examples had realistic effectors, but the system could be extended to support them. In a system incorporating realistic effectors, motion models would be intimately tied to their control. Recall that objects created in the scene as images are perceived by Soar (via predicate extraction) identically to objects fed into the scene from the environment, outside of an annotation to differentiate them. This allows hypothetical reasoning to occur identically to reasoning about the actual environment. Accordingly, hypothetical actions in the scene (simulations) and real actions can have the same representation in Soar, outside of an annotation.

This observation can help direct us toward how realistic actions should be integrated into a bimodal cognitive architecture. Mainly, at a high level, the actions should be controlled in the same way motion models are. The examples above have shown that only relatively simple continuous models are needed to solve complex problems, since the qualitative system can easily take on the higher-level aspects of the problem that would be difficult to reason about in a continuous system. For example, in the car driving problem, the motion model itself needs only to know how to seek towards a location, not how to do a backtracking search for a path, which objects in the scene are obstacles and which are not, etc. An equivalent car controller, while still faced with the difficult task of reducing movement to actuator signals, would similarly not have to be concerned with those higher-level concepts.

The idea of tightly binding motor control and imagery was explored in detail by Grush (2004), in his emulation theory of representation. He points out that maintaining a prediction of the outcome of motor commands (as a Kalman filter) can aid the controller itself, shortening the feedback loop through the environment. If this predictor is present for the controller, it can be used "offline" by itself to drive imagery – as a motion model.

Although motion models correspond to movements of environmental objects, it is important to note that they can be re-used for reasoning about problems not directly involving motion. In the above RTS base layout example, the motion models were not directly simulating objects that move in the environment, as the car and block models do. The system was manipulating a potential placement position, an entity that has no direct counterpart in the actual game – buildings are not placed by sliding them around in the environment, but by issuing a command specifying the building's location.

Completeness and Correctness

Viewed from the symbolic level, SRS can be considered to be an inference mechanism for determining the qualitative implications (extracted predicates) of a given sequence of qualitative commands (predicate projections). There is a fundamental issue here, in that any given command might have multiple valid, but qualitatively distinct results. For example, in Figure 1, based on how the system interprets the first command, the resulting situation could be qualitatively different: the new shape might or might not intersect shape B, based on how the system chooses to interpret "to the right of A." Qualitative simulation systems (e.g., Kuipers, 1986, Forbus et. al., 1991) encounter similar issues, which are typically resolved by allowing states to have multiple successor states. In our case, though, it is difficult to see how those multiple successors could be efficiently inferred and represented. Since multiple successors are not generated by SRS, as an inference mechanism, SRS is inherently incomplete. In the example in Figure 1, the system deals with this problem by refining its commands until the interpretation returned by SRS meets its requirements, but this strategy might not be possible in all cases. The problem is particularly evident when motion simulation is used. Simulations are very sensitive to the initial quantitative state of the diagram and the continuous properties of the motion model: a minor change in either can change the qualitative result of the simulation.

In many cases, qualitative descriptions given to SRS have only one possible quantitative interpretation, so in these cases, incompleteness is not a problem. It would be simple for the system to report to Soar when it must choose arbitrarily amongst multiple interpretations of a projection command (when the command is under-constrained). The agent could then adjust its reasoning process accordingly. However, consider the base layout algorithm described above (Figure 5). No matter where the initial building is placed by the first, under-constrained command (to place the building "outside the base"), the final result of the algorithm will always converge to the same region: the place adjacent to the base, far from the enemy. So it is not always desirable to avoid reasoning processes that are based on under-constrained situations. Further study of this

completeness problem is an area for future work.

Note that this issue is a different problem from cases where a simulation might result in incorrect qualitative inferences due to inaccuracies in the model itself. Since models are learned from the environment (at least conceptually), it is expected that results of simulations will vary based on how well the model matches reality, and the results can be incorrect if the match to reality is not good.

Learning Motion Models

Learning motion models from the environment has yet to be addressed. However, studying how models are used allows us to say precisely what they do and do not do, which is a first step in determining how they can be learned. All of the models used so far have involved a primary object that moves in some way in relation to a stationary reference object. The model, given a time step, maps the spatial configuration of these objects to a new spatial configuration. The motions are represented as the agent perceives them (i.e., not necessarily in terms of underlying physics). The models should not perform any reasoning that could be done at the qualitative level. The examples above have shown some of the range of reasoning this can encompass.

Conclusion

Spatial reasoning is a problem with aspects that are both precise and grounded, and abstract and qualitative. Bimodal systems attempt to unify all of these aspects into a common framework, a qualitative symbolic system interacting with a continuous spatial system.

One important type of spatial reasoning is reasoning about motion. We have shown how a bimodal system can incorporate motion, by using simple forward models in the spatial level. The interaction between these models and qualitative symbolic reasoning can solve complex problems involving motion that were previously very difficult for general purpose cognitive architectures. In addition to allowing new problems to be solved, this work sets the stage for learning and using realistic action in a cognitive architecture. The approach is also very economical, as it allows perception and action systems to be reused during problem solving, even for problems that aren't directly about motion.

References

Chandrasekaran, B., Multimodal Cognitive Architecture: Making Perception More Central to Intelligent Behavior, AAAI Conference on Artificial Intelligence, 2006.

Forbus, K.D., Nielsen, P., Faltings, B. Qualitative Spatial Reasoning: the CLOCK Project. *Artificial Intelligence* 51, 1991.

Funt, B., Problem-solving with diagrammatic representations. *Artificial Intelligence* 13, 1980.

Grush, R., The emulation theory of representation: Motor control, imagery, and perception. *Behavioral and Brain Sciences* 27, 2004

Kosslyn, S.., Thompson, W., Ganis, G., *The Case for Mental Imagery*. Oxford University Press, 2006.

Kuipers, B. Qualitative Simulation. *Artificial Intelligence* 29, 1986.

Laird, J. E., Extending the Soar Cognitive Architecture, Artificial General Intelligence Conference, 2008.

Lathrop, S. D., and Laird, J.E., Towards Incorporating Visual Imagery into a Cognitive Architecture, International Conference on Cognitive Modeling, 2007.

LaValle, S. M., *Planning Algorithms*, Cambridge University Press, 2006.

Wintermute, S., and Laird, J.E., Predicate Projection in a Bimodal Spatial Reasoning System, AAAI Conference on Artificial Intelligence, 2007.