Pengi: An Implementation of a Theory of Activity

Philip E. Agre and **David Chapman**
MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139

Abstract

AI has generally interpreted the organized nature of everyday activity in terms of plan-following. Nobody could doubt that people often make and follow plans. But the complexity, uncertainty, and immediacy of the real world require a central role for moment-to-moment improvisation. But before and beneath any planning ahead, one continually decides what to do *now*. Investigation of the dynamics of everyday routine activity reveals important regularities in the interaction of very simple machinery with its environment. We have used our dynamic theories to design a program, called Pengi, that engages in complex, apparently planful activity without requiring explicit models of the world.

I. Pengo

Let us distinguish two different uses of the word "planning".[1] AI has traditionally interpreted the organized nature of everyday activity in terms of capital-P Planning, according to which a smart Planning phase constructs a Plan which is carried out in a mechanical fashion by a dumb Executive phase. People often engage in lower-case-p planning. Though a plan might in some sense be mental, better prototypes are provided by recipes, directions, and instruction manuals. Use of plans regularly involves rearrangement, interpolation, disambiguation, and substitution. Before and beneath any activity of plan-following, life is a continual improvisation, a matter of deciding what to do *now* based on how the world is *now*. Our empirical and theoretical studies of activity have led us to question the supposition that action derives from the Execution of Plans and the corresponding framework of problem solving and reasoning with representations. We observe that real situations are characteristically *complex*, *uncertain*, and *immediate*. We have shown in [Chapman, 1985] that Planning is inherently combinatorially explosive, and so is unlikely to scale up to realistic situations which take thousands of propositions to represent. Most real situations cannot be completely represented; there isn't time to collect all the necessary information. Real situations are rife with uncertainty; the actions of other agents and processes cannot be predicted. At best, this exponentially increases the size of a Planner's search space; often, it may lose the Planner completely. Life is fired at you point blank: when the rock you step on pivots unexpectedly, you have only milliseconds to react. Proving theorems is out of the question.

Rather than relying on reasoning to intervene between perception and action, we believe activity mostly derives from very simple sorts of machinery interacting with the immediate situation. This machinery exploits regularities in its interaction with the world to engage in complex, apparently planful activity without requiring explicit models of the world.

This paper reports on an implementation in progress of parts of our more general theory of activity [Agre, 1985a, Agre, 1985b, Agre, in preparation, Chapman and Agre, 1987, Chapman, 1985]. We are writing a program, Pengi, that plays a commercial arcade video game called Pengo. Pengo is played on a 2-d maze made of unit-sized ice blocks. The player navigates a penguin around in this field with a joystick. Bees chase the penguin and kill him if they get close enough. The penguin and bees can modify the maze by kicking ice blocks to make them slide. If a block slides into a bee or penguin, it dies. A snapshot of a Pengo game appears in Figure 1. In the lower left-hand corner, the penguin faces a bee across a block. Whoever kicks the block first will kill the other.

Although Pengo is much simpler than the real world, it is nonetheless not amenable to current or projected Planning techniques because it exhibits the three properties of complexity, uncertainty, and real-time involvement. With several hundred objects of various sorts on the screen, some moving, representing any situation would require well over a thousand propositions, too many for any current planner. The behavior of the bees has a random component and so is not fully predictable. Real-time response is re-

---

[1]Agre has been supported by a fellowship from the Fanny and John Hertz Foundation.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research has been provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-80-C-0505, in part by National Science Foundation grant MCS-8117633, and in part by the IBM Corporation.

The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the policies, neither expressed nor implied, of the Department of Defense, of the National Science Foundation, nor of the IBM Corporation.

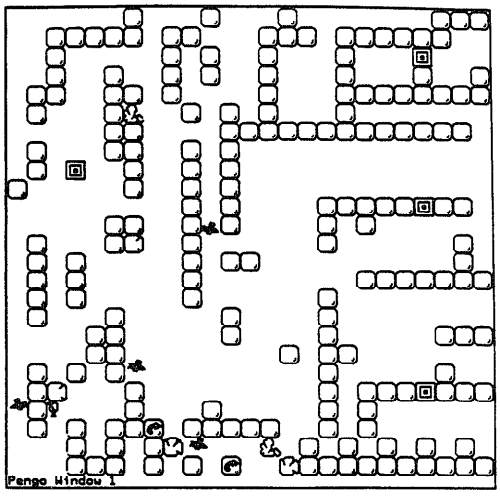

Figure 1: A Pengo game in progress.

quired to avoid being killed by bees. Still, Pengo is only a toy. There are no real vision or manipulation problems; it's a simulation inside the computer. Nothing drastically novel ever happens. This makes it a tractable first domain for demonstrating our ideas.

In a typical Pengo game, the penguin will run to escape bees, hunt down bees when it has the advantage, build traps and escape routes, maneuver bees into corners, and collect "magic blocks" (shown as concentric squares in Figure 1) for completing the "magic square" structure that wins the game. Naturally we ascribe the player's seeming purposefulness to its models of its environment, its reasoning about the world, and its planful efforts to carry out its tasks. But as with Simon's ant the complexity of the player's activity may be the result of the interaction of simple opportunistic strategies with a complex world. Instead of sticking to a rigid plan, Pengi lives in the present, continually acting on its immediate circumstances. It happens upon each situation with only a set of goals and a stock of skills. It can take advantage of unexpected opportunities and deal with unexpected contingencies, not because they've been written into a script, but because they are apparent in the situation.

II. Interactive Routines

Routines are patterns of interaction between an agent and its world. A routine is not a plan or procedure; typically it is not represented by the agent. An agent can reliably enter into a particular routine without representing it because of regularities in the world. For example, imagine the penguin running from a bee. The penguin will run as far as it can, until it runs into a wall made of blocks. Then it will have to kick its way through the wall. Then it will run some more. Then it will hit another wall. This process could be described by a procedure with two nested loops: running until it hits something, kicking the obstacle, and repeating.

But this same pattern of activity could equally well arise from a pair of rules: (R1) when you are being chased, run away; (R2) if you run into a wall, kick through it. These rules don't represent the iteration; the loop emerges as a result of the interaction of the rules with the situation. Causality flows into the system from the world, drives the rules which chose what to do, resulting in action which changes the world, and back again into the system, which responds to the changes.

An agent executing a plan is inflexible: it has a series of actions to carry out, and it performs them one after another. But it sometimes happens that while a bee is pursuing the penguin, the bee is accidentally crushed by a block kicked by a different bee. A penguin controlled by an iterative procedure would then either continue running needlessly or have to notice that it had gone wrong and switch to executing a different procedure. An agent engaging in a routine is not driven by a preconceived notion of what will happen. When circumstances change, other responses become applicable; there's no need for the agent even to register the unexpected event. (R1) depends on being chased; if there is no bee chasing, it is no longer applicable, and other rules, relevant perhaps to collecting magic blocks, will apply instead. Thus, routines are opportunistic, and therefore robust under uncertainty. Responses can be individually very simple, requiring almost no computation; this allows real-time activity.

III. Indexical-Functional Aspects

Pengi's activity is guided by relevant properties of the immediate situation we call *indexical-functional aspects*, or "aspects" for short. Registering and acting on aspects is an alternative to representing and reasoning about complex domains, and avoids combinatorial explosions.

A traditional problem solver for the Pengo domain would represent each situation with hundreds or thousands of such representations as (AT BLOCK-213 427 991), (IS-A BLOCK-213 BLOCK), and (NEXT-TO BLOCK-213 BEE-23). These representations do not make reference to the penguin's situation or goals. Instead of naming each individual with its own gensym, Pengi employs *indexical-functional entities*, such as the following, which are useful to find at times when playing Pengo:

- the-block-I'm-pushing
- the-corridor-I'm-running-along
- the-bee-on-the-other-side-of-this-block-next-to-me
- the-block-that-the-block-I-just-kicked-will-collide-with
- the-bee-that-is-heading-along-the-wall-that-I'm-on-the-other-side-of

As we will see later, the machinery itself does not directly manipulate names for these entities. They are only invoked in particular aspects. If an entity looks like a

hyphenated noun phrase, an aspect looks like a hyphenated sentence. For example:

- the-block-I'm-going-to-kick-at-the-bee-is-behind-me (so I have to backtrack)
- there-is-no-block-suited-for-kicking-at-the-bee (so just hang out until the situation improves)
- I've-run-into-the-edge-of-the-screen (better turn and run along it)
- the-bee-I-intend-to-clobber-is-closer-to-the-projectile-than-I-am (dangerous!)
- ...-but-it's-heading-away-from-it (which is OK)
- I'm-adjacent-to-my-chosen-projectile (so kick it)

These aspects depend on the Pengo player's circumstances; this is the *indexicality* of aspects. At any given time, Pengi can ignore most of the screen because effects propagate relatively slowly. It's important to keep track of what's happening around the penguin and sometimes in one or two other localized regions. When Pengi needs to know where something is, it doesn't look in a database, it looks at the screen. This eliminates most of the overhead of reasoning and representation. (The next section will describe how Pengi can *find* an entity in, or *register* some aspect of, a situation.)

Entities and aspects are relative to the player's purposes; they are *functional.* Each aspect is used for a specific purpose: it's important to register various aspects of the-bee-on-the-other-side-of-this-block-next-to-me, because it is both vulnerable (if the penguin kicks the block) and dangerous (because it can kick the block at the penguin). Which aspects even make sense depends on the sort of activity Pengi is engaged in. For example, when running away, it's important to find the-bee-that-is-chasing-me and and the-obstacle-to-my-flight and the-edge-I'll-run-into-if-I-keep-going-this-way; when pursuing, you should find the-bee-that-I'm-chasing and the-block-I-will-kick-at-the-bee and the-bee's-escape-route. Aspects are not defined in terms of specific individuals such as BEE-69. The-bee-that-is-chasing-me at one minute may be the same bee or a different one from the-bee-that-is-chasing-me a minute later. Pengi cannot tell the difference, but it doesn't matter because the the same action is right in either case: run away or hit it with a block. Moreover, the same object might be two entities at different times, or even at the same time. Depending on whether you are attacking or running away, the same block might be a projectile to kick at the bee or an obstacle to your flight.

Avoiding the representation of individuals bypasses the overhead of *instantiation*: binding constants to variables. In all existing knowledge representation systems, from logic to frames, to decide to kick a block at a bee requires reasoning from some general statement that in every situation satisfying certain requirements there will be some bee (say SOME-BEE) and some block (SOME-BLOCK) that should be kicked at it. To make this statement concretely useful, you must instantiate it: consider various candidate bees and blocks and bind a representation of one of these bees (perhaps BEE-29) to SOME-BEE and a representation of one of the blocks (BLOCK-237) to SOME-BLOCK. With n candidate bees and m blocks, this may involve $n \times m$ work. Clever indexing schemes and control heuristics can help, but the scheme for registering aspects we present in the next section will always be faster.

Entities are not logical categories because they are indexical: their extension depends on the circumstances. In this way, indexical-functional entities are intermediate between logical individuals and categories. Aspects make many cases of generalization free. If the player discovers in a particular situation that the-bee-on-the-other-side-of-this-block-next-to-me-is-dangerous because it can easily kick the block into the penguin, this discovery will apply automatically to other specific bees later on that can be described as the-bee-on-the-other-side-of-this-block-next-to-me.

IV. Simple Machinery

We believe that a simple architecture interacting with the world can participate in most forms of activity. This architecture is made up of a *central system* and *peripheral systems.* The central system is responsible, loosely, for cognition: registering and acting on relevant aspects of the situation. The peripheral systems are responsible for perception and for effector control. Because routines and aspects avoid representation and reasoning, the central system can be made from very simple machinery.

We believe that *combinational networks* can form an adequate central system for most activity. The inputs to the combinational network come from perceptual systems; the outputs go to motor control systems. The network decides on actions that are appropriate given the situation it is presented with. Many nodes of the network register particular aspects. As the world changes, the outputs of the perceptual system change; these changes are propagated through the network to result in different actions. Thus interaction can result without Pengi maintaining any state in the central system.

V. Visual Routines

Aspects, like routines, are not datastructures. They do not involve variables bound to symbols that represent objects. Aspects are registered by routines in which the network interacts with the perceptual systems and with the world. The actions in these routines get the world and the peripheral systems into states in which the aspects will become manifest.

Shimon Ullman [Ullman, 1983] has developed a theory of vision based on *visual routines* which are patterns of interaction between the central system and a *visual routines processor (VRP).* The VRP maintains several modified internal copies of the two-D sketch produced by early vision. It can perform operations on these images such as coloring

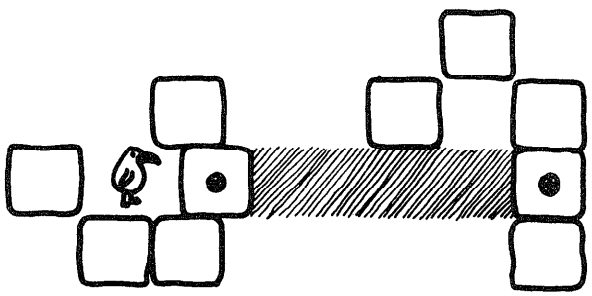

Figure 2: Finding the-block-that-the-block-I-just-kicked–will-collide-with using ray tracing and dropping a marker. The two circle-crosses are distinct visual markers, the one on the left marking the-block-that-I-just-kicked and the one on the right marking the-block-that-the-block-I-just-kicked-will-collide-with.

in regions, tracing curves, keeping track of locations using visual markers (pointers into the image), indexing interesting features, and detecting and tracking moving objects. The VRP is guided in what operations it applies to what images by outputs of the central network, and outputs of the VRP are inputs to the network. A visual routine, then, is a process whereby the VRP, guided by the network, finds entities and registers aspects of the situation, and finally injects them into the inputs of the network.

The first phase of the network registers aspects using boolean combinations of inputs from the VRP. Some visual routines are run constantly to keep certain vital aspects up to date; it is always important to know if there is a bee-that-is-chasing-me. Other routines are entered into only in certain circumstances. For example, when you kick the-block-that-is-in-my-way-as-I'm-running-away-from-some-bee, it is useful to find the-block-that-the-block-I-just-kicked-will-collide-with. This can be done by directing the VRP to trace a ray forward from the kicked block over free space until it runs into something solid, dropping a visual marker there, and checking that the thing under the marker is in fact a block. This is illustrated in Figure 2.

As another example, if the penguin is lurking behind a continuous wall of blocks (a good strategy) and a bee appears in front of the wall heading toward it, the-block-to-kick-at-the-bee can be found by extending a ray along the path of the bee indefinitely, drawing a line along the wall, and dropping a marker at their intersection. This is shown in Figure 3.

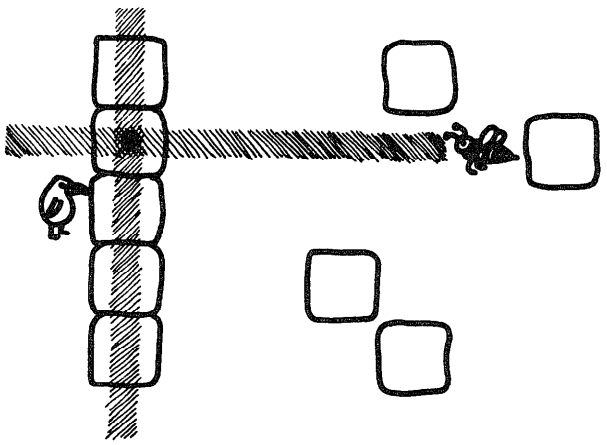

Figure 3: Finding the-block-to-kick-at-the-bee when lurking behind a wall.

VI. Action Arbitration

Actions are suggested only on the basis of local plausibility. Two actions may conflict. For example, if a bee is closing in on the penguin, the penguin should run away. On the other hand, if there is a block close to the penguin and a bee is on the other side, the penguin should run over to the block and kick it at the bee. These two aspects may be present simultaneously, in which case both running away and kicking the block at the bee will be suggested. In such cases one of the conflicting actions must be selected. In some cases, one of the actions should always take precedence over the other. More commonly, which action to take will depend on other aspects of the situation. In this case, the deciding factor is whether the penguin or the bee is closer to the block between them: whichever gets to it first will get to kick it at the other. Therefore, if the penguin is further from the block it should run away, otherwise it should run toward the block. This is not always true, though: for example, if the penguin is trapped in a narrow passage, running is a bad strategy; the ice block cannot be evaded. In this case, it is better to run toward the block in the hope that the bee will be distracted (as often happens); a severe risk, but better than facing certain death. On the other hand, if the block is far enough away, there may be time to kick a hole in the side of the passage to escape into. We see here *levels* of arbitration: an action is suggested; it may be overruled; the overruling can be overruled, or a counter-proposal be put forth; and so forth.

Action arbitration has many of the benefits of Planning, but is much more efficient, because it does not require representation and search of future worlds. In particular, a designer who understands the game's common patterns of interaction (its "dynamics") can use action arbitration to produce action sequencing, nonlinear lookahead to re-

solve goal interactions, and hierarchical action selection. Unfortunately, space does not permit us to describe the boundaries of the large but restricted set of dynamics in which this sort of machinery can participate. We should comment, though, on Pengi's central system's lack of state. We do *not* believe that the human central system has no state. Our point is simply that state is less necessary and less important than is often assumed.

VII. Status; Other Work

Currently, Pengi has a network of several hundred gates and a VRP with about thirty operators. It plays Pengo badly, in near real time. It can maneuver behind blocks to use as projectiles and kick them at bees and can run from bees which are chasing it. We expect to expand the network sufficiently that the program will play a competent if not expert game.

Pengi is an implementation of parts of a theory of cognitive architecture which will be described in greater detail in [Agre, in preparation]. In constructing the theory we learned from the cognitive-architectural theories of Batali, Drescher, Minsky, Rosenschein, and the SOAR group, among other sources. Rosenschein's is the most similar project. His situated automata use compiled logic gates to drive a robot based on a theory of the robot's interactions with its world. But these use the ontology of first-order logic, not that of aspects. The robot does not use visual routines and its networks contain latches.

We chose Pengo as a domain because it is utterly unlike those AI has historically taken as typical. It is one in which events move so quickly that little or no planning is possible, and yet in which human experts can do very well. Many everyday domains are like this: driving to work, talking to a friend, or dancing. Yet undeniably other situations do require planning. In [Agre, in preparation] we will outline a theory of planning that builds on the theory of activity that Pengi partly implements. Planning, on this view, is the internalization of social communication about activity.

We are wiring Pengi's central system by hand. Evolution, similarly, wired the central system of insects. But humans and intelligent programs must be able to extend their own networks based on experience with new sorts of situations. This will be a focus of our next phase of research.

VIII. Acknowledgments

This work couldn't have been done without the support and supervision of Mike Brady, Rod Brooks, Pat Hayes, Chuck Rich, Stan Rosenschein, and Marty Tenenbaum.

We thank Gary Drescher, Leslie Kaelbling, and Beth Preston for helpful comments, David Kirsh for reading about seventeen drafts, and all our friends for helping us develop the theory.

References

[Agre, 1985a] Philip E. Agre. The Structures of Everyday Life. MIT Working Paper 267, February 1985.

[Agre, 1985b] Philip E. Agre. *Routines.* MIT AI Memo 828, May 1985.

[Agre, in preparation] Philip E. Agre. *The Dynamic Structure of Everyday Life.* PhD Thesis, MIT Department of Electrical Engineering and Computer Science, forthcoming.

[Chapman, 1985] David Chapman. *Planning for Conjunctive Goals.* MIT AI Technical Report 802, November, 1985. Revised version to appear in *Artificial Intelligence.*

[Chapman and Agre, 1987] David Chapman and Philip E. Agre. Abstract Reasoning as Emergent from Concrete Activity. In M.P. Georgeff and A. L. Lansky (editors), *Reasoning about Actions and Plans.* Proceedings of the 1986 Workshop at Timberline, Oregon, pages 411–424. Morgan Kaufman Publishers, Los Altos, California (1987).

[Rosenschein and Kaelbling, 1986] Stanley J. Rosenschein and Leslie Pack Kaelbling. The Synthesis of Digital Machines with Provable Epistemic Properties. In Joseph Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowlege.* Proceedings of the 1986 Conference, pages 83–98. Morgan Kauffman Publishers (1986).

[Ullman, 1983] Shimon Ullman. *Visual Routines.* MIT AI Memo 723, June, 1983.