ccr: A network of worlds for research[*]

David H. Ackley
Department of Computer Science
The University of New Mexico
Albuquerque, New Mexico

Abstract

Experienced artificial life world builders know the long-term evolution of a system depends heavily on its size, and that individual computers today, still, support only modest worlds. Vastly richer worlds are possible if we can meet the challenges of designing for large-scale, distributed implementation. **ccr** is an Internet-based software system in development as a common environment for human and artificial agents. With themes such as artificial life, information access, agents, entertainment, and interactivity, **ccr** is ultimately an *evolution management* system, dedicated to enabling interdisciplinary research on—and in—a complex, open-ended, shared artifact.

1 MANIFESTO

The biggest problem facing computer science today is how to go parallel. Broadly construed, this includes not only parallel and distributed computing, networks, distributed AI, and so on, but also the evolution of protocols and standards, and cryptography and computer security generally. The classical Von Neumann architecture has delivered miracles, but as a conceptual organization it is largely tapped out. To design and build future computation and communication networks, we will either learn from or be forced to rediscover evolutionary biology, which, from the beginning, has dealt with parallel and distributed processing under resource and reliability limits, conflicting individual and collective goals, and limited trust. Artificial life research can and should, but to date has yet to, inform and unify these efforts.

The central tenet of the project is that "living systems" and "information systems" refer to the same class of systems. The research goals are both theoretical and practical: To understand better the connections between 'information' and 'life', and to design, build, deploy, and evolve experimental systems that explore and extend the 'living computation' framework.

2 OVERVIEW

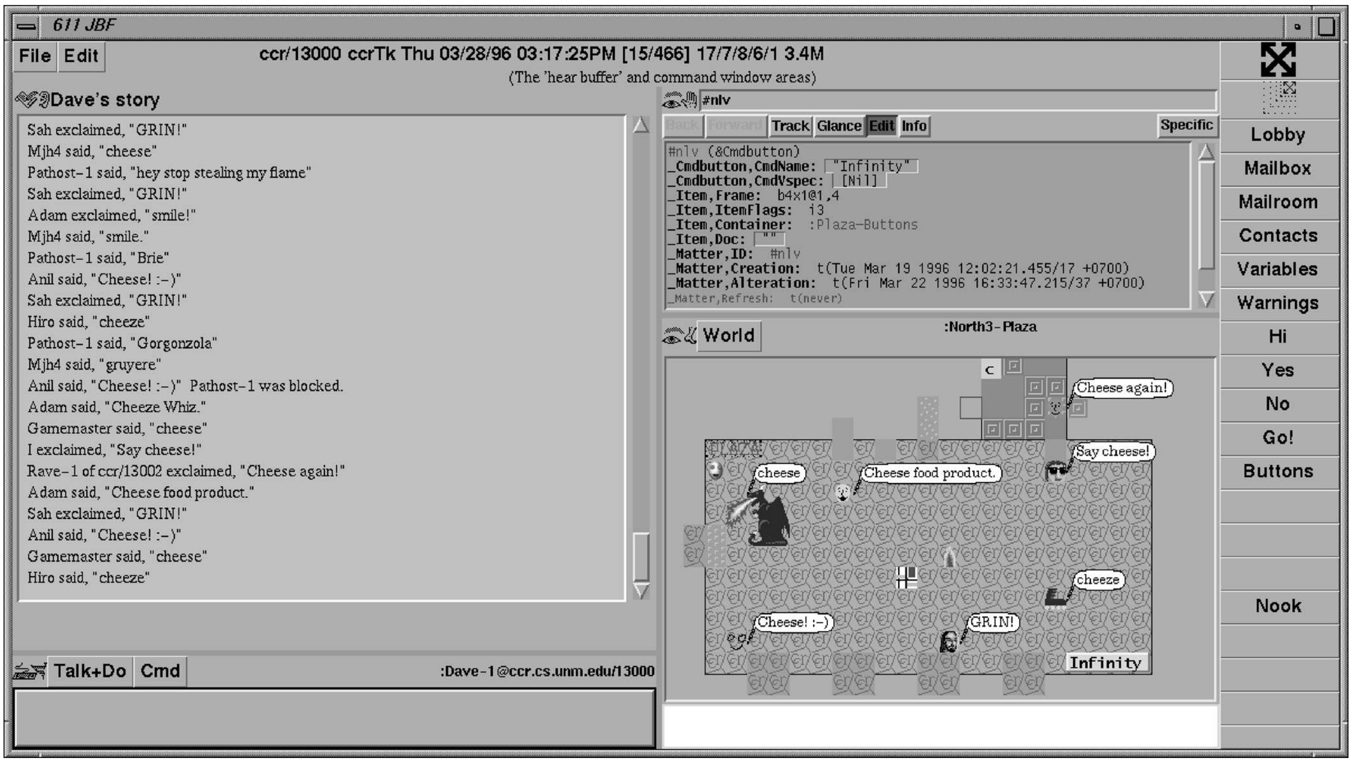
ccr is the working name[1] of a distributed environment for fostering and studying interactions between human and software life. The project focuses on a basic dilemma in distributed-system research: Even to uncover the real issues one must field a system with a significant number of machines and users, but as the 'mass' of the system grows it becomes more and more difficult to evolve the system structurally, to respond to the issues that are discovered.

The **ccr** vision is of a non-proprietary software system that retains evolvability and remains of moderate size over the long term—say, hundreds to thousands of machines and users, as a target—by drawing its users initially and principally from the interdisciplinary population of investigators whose research is facilitated by the existence of the system, and who are thus motivated both to shape the directions of, and put up with the overhead of, continued system evolution. Compared to proprietary commercial systems aimed at mass markets, **ccr** would be large enough to be relevant while small enough to avoid a competitive threat, and could serve as a vendor-independent technology and demand driver in addition to enabling basic research that could not be conducted safely or at all on commercial systems.

Version 0.1, the second prototype developed for the project, now comprises some 52K lines of C and another 11K of Tcl [16] and **ccrl**, and is in active devel-

[*] *v0.1.86˜(2)˜ release: in indep/doc/al5-ccrf.ps.gz* To appear in *Artificial Life V*, May 16–18, 1996; Nara, Japan.

[1] I.e., the pre-version 1.0 name. As of 3/31/96 the version was v0.1.86˜(2)˜. To the question "What does **ccr** stand for?" the canonical response is "Whatever you want it to stand for," to underline the goal of bottom-up, user-driven system evolution. Similarly, those who want **ccr** to be an acronym are invited to make up expansions to suit their needs, or select from *caricature cartoon clearwater communication community competitive computation connected cooperative creedence reality reconsidered realized reified research revealed revival rooms.*



ccr/13000 ccrTk Thu 03/28/96 03:17:25PM [15/466] 17/7/8/6/1 3.4M
(The 'hear buffer' and command window areas)

Dave's story

```
Sah exclaimed, "GRIN!"
Mjh4 said, "cheese"
Pathost-1 said, "hey stop stealing my flame"
Sah exclaimed, "GRIN!"
Adam exclaimed, "smile!"
Mjh4 said, "smile."
Pathost-1 said, "Brie"
Anil said, "Cheese! :-)"
Sah exclaimed, "GRIN!"
Hiro said, "cheeze"
Pathost-1 said, "Gorgonzola"
Mjh4 said, "gruyere"
Anil said, "Cheese! :-)"  Pathost-1 was blocked.
Adam said, "Cheeze Whiz."
Gamemaster said, "cheese"
I exclaimed, "Say cheese!"
Rave-1 of ccr/13002 exclaimed, "Cheese again!"
Adam said, "Cheese food product."
Sah exclaimed, "GRIN!"
Anil said, "Cheese! :-)"
Gamemaster said, "cheese"
Hiro said, "cheeze"
```

:Dave-1@ccr.cs.unm.edu/13000

#nlv

```
#nlv (&Cmdbutton)
_Cmdbutton,CmdName:  "Infinity"
_Cmdbutton,CmdVspec:  [Nil]
_Item,Frame:  b4x1@1,4
_Item,ItemFlags:  i3
_Item,Container:  :Plaza-Buttons
_Item,Doc:  ""
_Matter,ID:  #nlv
_Matter,Creation:  t(Tue Mar 19 1996 12:02:21.455/17 +0700)
_Matter,Alteration:  t(Fri Mar 22 1996 16:33:47.215/37 +0700)
_Matter,Refresh:  t(never)
```

Figure 1. A view of a **ccrTk** world.

opment and testing in a "universe" of a dozen or so interconnected worlds. Though primitive, largely undocumented, and crash-happy, pathways are clearing toward a system tamed enough for adventurous interdisciplinary researchers—and this fifth artifical life conference is a good opportunity to offer an overview as development proceeds. In addition to essentially all of the computer sciences, **ccr** needs perspectives from art, biology, ecology, economics, literature, political science, psychology, and sociology, to name a few.

This document is a first introduction to **ccr**, briefly considering both general themes and more technical issues. To anchor the discussion and provide a recurring example, Figure 1 presents a recent snapshot taken in **ccrTk**, currently **ccr**'s primary graphical interface world. Section 3 compares the **ccr** strategy to other alife research methodologies. Section 4 introduces version 0.1 specifically, touching on the interface language **ccrl**, the computation and communication model, and the object system. Section 5 concludes with riffs on methodology and the meaning of life. For more current information and plans for the system, visit `http://www.cs.unm.edu/~ackley/ccr`.

3 A RESEARCH MACROCOSM, WITH PEOPLE

ccr breaks with most artificial life systems in a major way by presuming an *open world*, both in that humans are expected to interact with the system while it is running, and in that the underlying physical implementation—the computers and communications links—is expected to change dynamically. Three consequences of this approach are: Research strategies change because global repeatability is sacrificed, humans must be enticed to use the system, and the tradeoffs between security, efficiency, and power must be addressed, not only early and fundamentally, but continually and at many levels.

3.1 Repeatability is unscalable

A great virtue of closed-world alife models is that *every* detail is determined by the researcher, so every observation can be reproduced, and utterly controlled variations can be tested. The price is that comparatively small worlds must be studied. With ingenuity, much has been and remains to be learned from such worldlets, and the possible richness of high-isolation worlds grows with improvements in affordable individual computers, but of course single-owner systems cannot compete with large collaborative networks of systems. The irony is that as computer systems become more like living systems—more complex and articulated, more robust and interconnected—they become less suited to closed-world artificial life research. Ray's NetTierra proposal [18] offers one compromise position, in which repeatability is explicitly sacrificed but isolation is mostly preserved.

3.2 Humans are evolutionary forces

ccr approaches the dilemma another way, trading the isolated clarity of lab work for the symbiotic relevance of field work. **ccr** proposes to be a strictly non-proprietary [8] experimental platform, controlled by and for its research users, providing a venue within which software for computation and communication tasks—"agents," "brokers," "robots," etc.—can be created, copied, hybridized, allowed to cooperate and compete for "market share" and perhaps win through into the base **ccr** standards and protocols.

As in typical alife systems, the overall "fitness functions" are supplied by humans; on the other hand, in **ccr** humans are also a source of novelty and change, conflicting with the Darwinian principle of blind variation. Given the nature of complex adaptive systems [11], and the gap between the manifest *intentionality* of the individual human action and the *unintentional* effects that often ensue, it is an empirical question just how un-Darwinian the evolution of a substantial **ccr** universe would actually be.

3.3 Security is biology

Communication entails risk [1]: A message sender necessarily reveals information in the act, and a message receiver is necessarily impacted in some way by the act, or else no communication has occurred. The dual tasks—of revealing some information while hiding some, and of allowing selected influences while rejecting others—are fundamental to the structure and function of living systems, from cell walls to immune systems, crucial to the very notions of self and independent existence.

A price exacted by the oft-touted mobility of a "software agent" is that it doesn't control the physical hardware that embodies it, so to protect its integrity it must either hide from or ally with the machine owner. Computer viruses take the former route; **ccr** takes the latter, committing to reveal to the owner the tradeoffs between safety and power as obviously and intuitively as possible, and placing its source code on the table as *bona fides*. In turn, in a general release of **ccr**, the hardware owners would commit to playing within the system[2] and would place their digital signatures on the table co-signed by existing **ccr** users (the bottom-up "web of trust" approach [20]) and/or an appropriate external certification authority [19]. Various flavors of anonymity can be created within the system, but only built upon a base of identified owners, shifting risk from loss of integrity to loss of anonymity. It is an open question whether special-purpose protected hardware could in principle be "owned" by the system itself, which could allow the construction of robust "public spaces" with known and reliable rules of behavior, *inside* of independently-owned computers—and if so, under what circumstances would wise owners choose to incorporate it.

While technologies such as digital signatures [19] are necessary, security is much more than a purely technical issue, and so the purely technical power of the system must have corresponding limitations. In **ccr**, as in most high-isolation alife systems [17, 6] as well as emerging commercial systems such as Java [9], the basic approach to limiting communication risk via limiting power is to control the semantics of the *language* in which communications are expressed. Though Java significantly improves security from the "language on down", as a general purpose programming language, its approach to trust is at the level of the program and is largely boolean—you either grant a disturbing amount of power to an incoming Java 'applet' or you don't run it at all. As a research system, **ccr** sacrifices some speed and generality to gain fine-grained access control at multiple points including each function invocation, directly supporting intuitive and ccr-specific degrees and modes of trust and risk.

4 A TOUR OF VERSION 0.1

Though primitive compared to where it needs to be for widespread use, version 0.1 is already quite rich—possessing, for example, a fairly well-developed computational and communications model, a (and please excuse the jargon) multiply-inheriting runtime-extensible persistent object-oriented interpreted language with incremental network object and type cache updating, the graphical user interface world **ccrTk**, and the text-only **ccrt**—and it is difficult to know where to begin to describe it. The hands-on approach is to sit down with somebody at a pair of machines, walk them through building a "genesis" world for themselves, then show them how we build a "netdoor" between their world and mine, and then have them over for a visit. Here, an overview of the **ccrl** language serves as a backbone upon which to hang brief discussions of related topics: the user interface, the execution engine, and some basic object types—such as `&Matter` and `&Energy`—that are key elements of the **ccr** "metaphysics."

4.1 ccrl

Version 0.1 speaks a quirky language called **ccrl**[3] that mediates communications between a **ccr** world and its environment, which consists principally of humans, other **ccr** worlds, files, and spawned subprocesses. **ccrl** is more akin to an application scripting language or an object-oriented MUD programming language [3] than to

[2]Which includes researching attacks to devise defenses; in that spirit **ccr** 0.1 provides an "award" system to honor and memorialize the publicizers and fixers of holes.

[3]Pronounced via spelling or as *crawl* to emphasize one of its annoying quirks.

a conventional development language, in that it developed out of and in tandem with **ccr**. In spots it carries evolutionary baggage dating back at least to 1992 (version 0.1.23); a redesign based on experiences thus far is certainly warranted, but the issues are complex and the language serves adequately for the present.

4.2 Hello world

It is a tradition in presenting programming languages to begin with the "Hello world" program, which does nothing except output "Hello world" in whatever form the system most naturally supports. In different languages the program may range from a line or two up to dozens or more, and traditionally there are minor bragging rights attendant to shorter solutions. Figure 2 presents the Hello World program in **ccrl**, and typical results of executing it. **ccrl** minimizes this program just about as far as it will go. Basic input and output are built deeply into the language, rendering the Hello World task trivial, but performing the exercise does highlight how the choice of "Hello *World*" as the traditional first task—as opposed to, say, "Hello *Dave*"—reflects the fact that typical programming systems are utterly oblivious to both *who's programming* and *who's out there listening*, while to **ccr**, for example, such matters are of paramount significance. As may be suspected from Figure 1, **ccrTk**

(a) `Hello World↵`
(b) I said "Hello World"
(c) Dave-1 said "Hello World"

Figure 2. *(a)* The "Hello world" program in **ccrl** (↵ denotes the Return key); *(b)* What I observe; *(c)* What others in the area observe.

worlds reduce the Hello World task still further, to a single mouse-click on a predefined "Hi" button—but that is indeed a cheat, since Hello World is meant to be at least a template for producing strings in general, and programming the buttons is a more complex procedure. As a template for **ccrTk**, Figure 2a is buggy in that certain initial characters are significant. Figure 3 lists some of the **ccrTk** reader's initial character shortcuts and their expansions as general **ccrl** object descriptions; a more general **ccrTk** Hello World program, similar in approach to MUD languages [3], is

```
'Hello world↵
```

In addition to "speaking" in various ways, shortcuts are available for moving around, pushing, opening, closing, and inspecting objects, and so forth.

As new users get more comfortable acting in their own world, they are sometimes irritated to discover how limited their powers are when they are visiting other worlds, until they appreciate the fundamental symmetry of the distributed **ccr** universe—every user is at once a "god" on their own world and a mere "mortal" when visiting other worlds, and to first-order if you can't do it to their world, then *they* can't do it to *your* world. This symmetry can be impacted in various ways by world-local programming, but it appears so far that the early architectural choice of a peer-to-peer universe, rather than a client-server one—so that, for example, all users are "home owners" with something to lose—was sound.

`’→[Say text "•"]`	`:→[Pose text "•"]`
`;→[Be text "•"]`	`!→[TimeOut at •]`
`?→[Focus on •]`	`*→[Teleport to •]`
`+→[Open object •]`	`<→[Get at • field ]`
`-→[Close object •]`	`>→[Set at • field  to ]`

Figure 3. ccrTk keyboard shortcuts and expansions. (• denotes the cursor.)

4.3 Building

The expansions in Figure 3 suggest the general syntax of **ccrl**:

`[`*TypeName slotname slotvalue slotname slotvalue* ...`]`

where each *slotname* names a local variable declared by *TypeName* or one of its ancestor types. When handed to a **ccrl** reader such expressions produce a *description* of an object, and an extra step is needed to produce an object so described. Rather than the "read-eval-print" loop typical of interactive languages, the **ccrl** interaction cycle is "read-build-run": First (attempt to) convert the textual form into an internal object description, then (attempt to) build an object matching the given description, and then "run" the object. This last step may amount only to performing a function call and returning a value (as `Get` does, for example), or (as in the case of `Say`) it may involve asynchronously "releasing" the object into the shared environment to have effects in a region of space-time often involving network traffic to other worlds.

If a **ccr** object is an instance of the `&Matter` type, then once built it will persist across program restarts until specifically removed. Each piece of matter has an identifier that is unique across the entire **ccr** universe (such as `:North3-Plaza@ccr.cs.unm.edu/13000`, where I was when I said "Say cheese!" in Figure 1). In **ccr** all matter objects are named, but not all objects are matter. For example, objects of type `&Act` mediate computation and communication, and thereby drive the dynamics of the universe. An input like

```
You're kidding!↵
```

which **ccrTk** readers parse as the description

```
[Exclaim Text "You're kidding"]
```

eventually builds an `&Exclaim` object, which inherits from a distant ancestor called `&Energy`, which is an `&Act`.[4] Acts are somewhat like the "stack frames" of conventional programming languages, but in **ccr** they are first-class, allocated objects. `&Energy` objects, in particular, "propagate" until they visit every object in the refspace of their point of origin, or are "absorbed" somewhere along the way. In Figure 1, for example, my world was being bombarded by energy packets that had originated on some half a dozen worlds and propagated through the network to reach mine.[5]

4.4 Refspace

If for no other reasons than efficiency and security, distributed systems must somehow limit the range of possible object interactions. Often interactions are limited to one "room" at a time—so travelling through doors is an adventure, since you can't see what you're getting into until you're in it; other mechanisms such as "k-nearest neighbors" lead to a fluctuating and non-obvious "radius of interaction" as objects move.

In **ccr**, the *reference space* or "refspace" of a given `&Matter` object is the set of `&Matter` objects that are reachable from the given object by following "propagation rules" determined by the types of objects encountered. The lower right panel in Figure 1—with all the rectangles and little icons and speech balloons—is a rendering of the refspace of my **ccr** body. The rules of energy propagation are symmetric and largely hierarchical, so for example, anything said by anybody in a `&Room` propagates up to the room and then down to the other occupants. But there is one huge exception: A `&Link` object, when mated to another `&Link`, propagates to its mate the energy that reaches it hierarchically, and "reradiates" energy arriving from its mate into the hierarchy. This lateral propagation occurs whenever the connection between two `&Link` objects is open, even if the links are located in different containment hierarchies, and *even* if they are on different **ccr** worlds. Via links, **ccr** worlds weave themselves into the fabric of the **ccr** universe.

The most common subtypes of `&Link` are `&Door`, `&Body`, and `&Soul`; here we discuss only the first. A door is a link that only can be mated to another door, and then only if they "match up" in a East/West or North/South pair. In the refspace shown in Figure 1, the largest black-outlined rectangle is a `&Room`—named `:Plaza` though that is not obvious—and each of the small grey rectangles overlapping the edges of the Plaza is actually a mated pair of doors, with one end in the Plaza and the other end someplace else. Because my body was *contained* by a door, the entire refspace of the door's mate was part of my refspace, and consequently I also saw one of the smaller rooms north of the Plaza. That room has doors on three walls; the south door connects back to the Plaza, the west door and the northern east door are currently unmated. The southern east door is mated and open, and leads to some other room. The door I was standing in, which appears a cautionary orange on a color display, is actually a *netdoor*, connected through the Internet to another world, and all of my refspace there appears somewhat darker than my own world, as a reminder that that is not space that I control. On that remote world, one body is visible speaking. He "heard" everything I said, and vice versa, but he was unable to hear the clamor going on in the Plaza, because *his* refspace—determined by his location—ends at the door I was standing in. If I were to take a step south, I would leave the doorway and enter the Plaza, and I'd no longer be able to see him, and vice versa. As bodies move, their refspaces merge and separate, and the ongoing stories of their **ccr** lives—mine is visible scrolling in the large left-hand window—intertwine and diverge.

4.5 Making new physics

Though the god of a **ccr** world can travel anywhere within it in a single step, using `&Teleport` energy, visitors from other worlds must travel "overland" using doors, allowing gods to control visitor access to their worlds in a natural way. Most worlds, for example, have regions that are private simply because they are unreachable from any netdoor.[6] It is also easy to build disconnected "public" regions that each have a netdoor but have no local doors joining them, allowing visits to different areas of a world while barring transits all the way through it. Although this basic connectivity-by-doors approach is intuitive and has worked well, gods often wish to provide other sorts of access mechanisms for other purposes. The extensibility of **ccrl** offers many ways to fulfill such wishes.

As a concrete example, though with only a brief explanation, Figure 4 presents a **ccrl** type and method definition that is in common use in the current **ccr** universe, designed to provide visitors with controlled access to the powerful teleport energy. The first expression defines a new type called `&ZapSpace`, which inherits from `&Space` so it will be a visible and enterable object, and which inherits from `&Force` so it will be able to build and release (selected types of) energy. ZapSpace defines one local variable or "slot" to store a teleport destination;

[4]It was expected and initially designed so that 'Energy' was the most abstract active element of the system, but Acts turn out to be more general—in particular, `&Functions` are acts but not energy, unobservable except to their creator, since they do not spontaneously 'propagate'.

[5]More specifically, the "17/7/8/6/1" in the status line near the top of Figure 1 reveals that at that moment my world was connected to, and was thus willing to accept energy from, eight other worlds (and had, overall, seventeen open I/O channels, seven visitors from other worlds, six open netdoors, and one active external process.)

[6]Unusually powerful items, such as the Quit button, are also protected by "point defenses."

```
[deftype at "ZapSpace" isa (&Space &Force)
  redefault ((_Space,Floor () B1,1x1:.))
  slots (("destination" (::private) @@))]
[defmethod act &enter object &ZapSpace
 class ::object
 at [let
     variable
      (($dest [get
                at [get at $self
                    field _verb,object]
                field _ZapSpace,destination])
       ($objid [get at $self field _verb,subject]))
     at [if at $dest
         then
          [progn
           at ([teleport obj $objid at $objid
                to $dest]
               [return])]]]]
```

Figure 4. Defining a new type and method.

in this case, mostly for entertainment value, the slot is marked `::Private`, meaning that values in this slot will never be shipped offworld by the network cache management system, and so visitors will be unable to tell where a given ZapSpace will take them, until they try it.

The second expression defines a method that will be executed whenever an object requests to enter a ZapSpace. If the `_ZapSpace,Destination` slot has been initialized by the local god, the method teleports the subject of the enter energy—whatever is attempting to enter the ZapSpace—to the fixed destination, and then performs a `[return]`, in effect "absorbing" the enter energy so that it has no further effects. Unlike doors as normally used, a ZapSpace is asymmetric: There may be no way back, for visitors, from the destination of a ZapSpace. Fortunately, the "bail button" (the "outward and inward arrows" icon at the top of the righthand column of buttons in Figure 1) *always* brings a user back home, come what may while out exploring the universe.

5 DISCUSSION

That brief look at version 0.1 raises far more questions than it answers, but hopefully it at least provides a flavor of the system and some of the issues involved. This final section returns to broader questions: How can we do research, particularly alife research, in **ccr**? In what sense is **ccr** alive?

5.1 ccr for alife

From a high-isolation "in the lab" perspective, having "humans in the loop" means any data on **ccr**'s evolutionary behavior is fundamentally tainted—after all, what's to stop a researcher/user from "cooking the data" by acting in some particular way solely for that purpose? This is indeed a problem, though not an insurmountable one; in addition to careful experimental design and data collection, the key is building a large enough universe that idiosyncratic individual effects wash out, much as medical research conducts large-scale studies using its own professional ranks as subjects.

It is also possible to perform more-and-less isolated alife experiments within **ccr**. For example, Hofmeyr [10] has built a small system of reproducing and evolving "robots" confined to a single room on his world, using a combination of **ccrl** code and external C++ code running in subprocesses managed by **ccr**. In such cases, **ccr** provides an infrastructure both for the experiment and for allowing visitors around the world to watch and interact with the "evolvobots" in action. A common, Internet-aware platform such as **ccr** could substantially improve the abilities of alife researchers to collaborate, and to observe, share, and duplicate results.

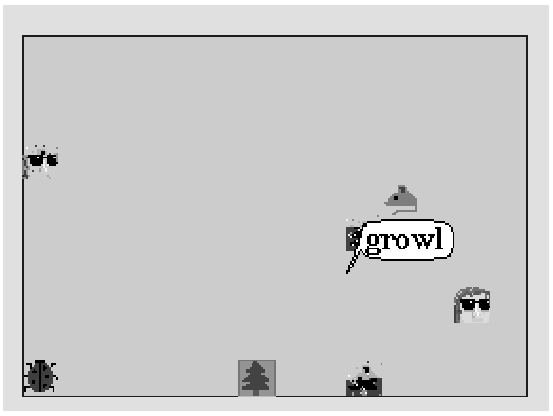


Figure 5. Evolving robots in **ccrTk**.

Figure 5 shows some early evolution in Hofmeyr's **ccr** "laboratory." The lab has no doors—the only way in, for visitors, is via an external ZapSpace destined for the lab, and there is no way out at all, short of asking Hofmeyr for a teleport, if he's around, or else bailing. The robots move about, analyzing the visual appearances of those around them, and reacting in various ways depending on their evolving genetic predispositions. Some seek visually-pleasing "mates" with which to recombine—which may in fact be a visitor to the lab, as the hybrid offspring in the middle of the west wall suggests. More aggressive robots will chase after visitors or robots with appearances they dislike—the robot initialized with a "mouse" image is evidently such an offender in Figure 5—and the aggressor will destroy the displeasing creature if it gets adjacent to it.

Other **ccr** creatures survive and make a living "in the wild." The egg-shaped face visible near the upper-left corner of the Plaza in Figure 1, for example, is a "Floyd" robot [15]. Floyds can't move or reproduce without the intervention of the world owner, but they use the **ccr** infrastructure to keep in touch with each other, forming a 'Floyd-species-level' communication network within the **ccr** universe. Especially in these early days, when the universe is small, asking one's Floyd "Who is up?" is a handy way to check activity around the universe.

At a pedagogical level, getting a software agent to function effectively in a system as rich as **ccr** is itself an insightful process. For example, since **ccr** must view external subprocesses as "non-self", there are "filters" attached to their representations inside the system: A "senses" filter determining what propagated energy should be transmitted the subprocess, and "effector" filters determining what the subprocess is allowed to build and run within **ccr**. Examining the filters allows a **ccr** user to "size up" an unfamiliar robot, to tell precisely what it is capable of sensing and doing, and to act accordingly. Novice robot builders often set all the filters to `(([Anything] ::Allowed))`, both with high hopes of constructing a powerful robot that is keenly aware of its environment, and to avoid figuring out which types are actually relevant. They quickly find their robot buried under a flood of energy of which as users, due to the default filters on their souls, they had been blithely unaware—communications protocols, timing signals, incremental database updates, and so forth—a firehose of data that is both computationally and conceptually hard to handle. It takes some experience to appreciate just how deeply coupled are the senses, effectors, and brains of a well-adapted creature, and how being *less* aware of the environment can both radically reduce the brains required to handle a given task and lead to far more elegant solutions.

5.2 Living computation

Managing distributed computations across large networks of separately administered resources is in important ways more akin to managing a human society than to marshalling the closely-held resources of a single digital computer. Viewing living systems, especially living ecosystems, as computational systems, provides many insights into what the successful architecture may look like [2]. Though object-oriented programming, for example, is a step in the right direction, fundamental issues—some quite obvious in the context of living systems—remain largely unrecognized.

Consider, for example: Relatively "complex" living organisms such as, say, mammals, are *always* "designed, built, and tested" on a *whole system* basis—there are (until *very* recently) no "plug-ins," "patches," or "upgrades" to an individual's genetic code. The germ line code storage and transmission mechanisms defend against external alterations in many ways—physical, biochemical, developmental, immunological, instinctive, and cultural. Such elaborate and expensive defenses are sensible given the relatively high cost of producing a system—if the results could be easily "hijacked," the capital investment would be unwarranted.

On the other hand, relatively "simple" living organisms such as bacteria are capable of incorporating "stray" bits of code from their environment into their "operating systems"—as when a gene coding for drug resistance is observed to "jump species"—to the dismay, at least, of the complex organisms that produce the drug. Such promiscuity is sensible given the relatively low cost of producing a system, combined with the potential gains to be had by "stealing code."

Is a computer system more like a collie or an *E. Coli*? On the one hand, even a personal computer is an expensive investment, and if it is used productively its value rises much higher than its capital cost. On the other hand, personal computers today are a motley patchwork of code from dozens of sources, with essentially no "sense of self" [7], most of them still lacking even the most basic immunologic mechanisms such as protected kernel mode.

We can be quite confident that this embarrassing combination of traits is not an evolutionarily stable strategy in Maynard-Smith's sense [14], as the essentially immediate explosion of computer viruses following the rise of personal computers attests. Today, the appellation "personal computer" is in important ways a misnomer. The personally-owned computer does not "know" its owner in any significant way, and that's just as well because it is fundamentally unable to distinguish between what is "inside itself" and to be trusted with sensitive information (merely beginning with passwords) and what is not. If future personal computers do not make a credible case for *loyalty* to their owners, all the interface and ease-of-use improvements in the world will not get people to use them for serious work, but if a useful system demonstrates it is watching out for its owner's best interests, first, last, and always, from hardware to software to data to communications, people would clap rocks together in Morse code, if they had to, to interact with it.

5.3 ccr as life

I have talked about objects and processes within **ccr**, and about **ccr** itself, in a variety of "life-like" terms, without qualifying such descriptions as figurative or metaphorical. Although this is typical in alife research, it is worth some consideration. Developing **ccr** over the last five years or so has led me to think that Dawkins' suggestion, that the origin of *life* on Earth also marked the origin of *information* on Earth [5], is so deeply correct that I must suggest that we may view

Life preserves information

as *defining both* "life" and "information" in terms of each other. It is certainly plausible to claim that "All living systems do an effective job preserving information". Is it completely absurd to say that "All effective information-preserving systems are living systems"? We ground out the mutual recursion in cases where we have other *prima facie* reasons to describe a system as "living" or as "preserving information"—and our understanding of the dual role of DNA, as both an active catalytic controller of chemistry during the existence of a cell and as a passive reaction product during cell copying, provides one such base case.

Such a high-handed approach, while riding roughshod over all sorts of important issues, offers a way to unite "life is selfish gene-copying" advocates [4] with "life is self-production" advocates [13], viewing copying and maintenance as *the* two fundamental strategies for preserving information. On some of the standard challenges for definitions of life, the view would include mules, exclude fire, and probably leave crystals on the margin, depending on how much "information-theoretic" information we expect to find in a crystal—which won't be much, if the crystal is pure. Impure or semi-crystalline materials, such as, say, integrated circuits, are of course another matter.

On the prospects for life in manufactured computers, the view is that, rather than being an esoteric research topic, that is a prosaic, long-established fact. Popular software programs today are preserving their information spectacularly, with population sizes in the millions and booming; malicious computer viruses are harder to measure though detections of new strains are booming as well [12]. The emergence first of affordable personal computers and now of mass-market computer networking adds urgency to the real research question: With the great flexibility of programmable computers laying before us, *what kind* of artificial life do we want?

ACKNOWLEDGMENTS

I thank the research management: Tom Landauer, Mike Lesk, George Furnas, & Jim Hollan; the gods of version 0.0: Steve Abney, George, Michael Littman, & Scott Stornetta; and the primordial gods of version 0.1: Ben Bederson, Patrik D'haeseleer, George, Mark Handler, Steve Hofmeyr, Lee Jensen, Craig Jorgensen, Nigel Kerr, Michael, Adam Messinger, Nelson Minar, Anil Somayaji, Jason Stewart, & Jeff Zacks, the creator of **ccrTk**. All screwups, of course, are my fault.

REFERENCES

1 Ackley, D.H., & Littman, M.L. (1994b) Altruism in the evolution of communication. In *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems.* edited by R. A. Brooks & P. Maes. A Bradford Book, The MIT Press: Cambridge, MA..

2 Belew, R.K., Mitchell, M., & Ackley, D.H. (1996, in press). Computation and the natural sciences. In R.K. Belew and M. Mitchell (editors), *Adaptive individuals in evolving populations: Models and algorithms.* Reading, MA: Addison-Wesley.

3 Curtis, P. LambdaMOO Programmer's Manual. At `ftp://parcftp.xerox.com/pub/MOO/Programmers-Manual.txt`

4 Dawkins, R. (1989) *The Selfish Gene.* Oxford University Press: New York.

5 Dawkins, R. (1995) *River out of Eden: A Darwinian View of Life.* BasicBooks, HarperCollins Publishers: New York.

6 Dewdney, A.K. (1984, May) In the game called Core War hostile programs engage in the battle of bits. *Scientific American.* See also, e.g., at `ftp://ftp.csua.-berkeley.edu/pub/corewar`.

7 Forrest, S., Hofmeyr, S.A., Somayaji, A., and Longstaff, T.A. (1996, in press). A sense of self for Unix processes. *1996 IEEE Symposium on Computer Security and Privacy.*

8 Free Software Foundation, Inc. The GNU General Public License Version 2. At `http://www.cygnus.-com/doc/license.html`.

9 Gosling, J. and McGilton, H. (1995, May). *The Java Language Environment: A White Paper.*, Sun Microsystems Computer Company. At `http://-java.sun.com/`.

10 Hofmeyr, S. A. (1995, November). *Human and alife interactions in* **ccr**. At `http://www.cs.unm.edu/-~steveah/ccr.ps`.

11 Holland, J. H. (1995) *Hidden Order: How Adaptation Builds Complexity.* Addison-Wesley: Reading, MA.

12 Kephart, J. O. A biologically inspired immune system for computers. In *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems.* edited by R. A. Brooks & P. Maes. A Bradford Book, The MIT Press: Cambridge, MA., 130–139.

13 Maturana, H. R., & Varela, F. J. (1980). *Autopoiesis and Cognition.* Reidel: Dordrecht.

14 Maynard Smith, J. (1982). *Evolution and the theory of games.* Cambridge University Press: Cambridge.

15 Minar, N. (1995). *Floyd, a global network of communicating agents.* At `http://www.santafe.edu/-~nelson/ccr/floyd/`.

16 Ousterhout, J. K. (1994) *An introduction to Tcl/Tk.* Addison-Wesley: Reading, MA.

17 Ray, T. S. (1991) An approach to the synthesis of life. In *Artificial Life II, SFI Studies in the Sciences of Complexity,* vol. X, edited by C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen, Addison-Wesley, 371–408.

18 Ray, T. S. (1995) A proposal to create a network-wide biodiversity reserve for digital organisms. ATR Technical Report TR-H-133. Also available at `http://www.hip.atr.co.jp/ray/-pubs/reserves/node1.html`.

19 Schneier, B. (1994) *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley: New York.

20 Zimmerman, P. R. (1995) *The Official PGP User's Guide.* MIT Press: Cambridge, MA.