For Problems Sufficiently Hard ... AI Needs CogSci

Selmer Bringsjord & Micah Clark
Department of Cognitive Science
Department of Computer Science
Rensselaer AI & Reasoning (RAIR) Lab:
http://www.cogsci.rpi.edu/research/rair/index.php
Troy NY 12180 USA
selmer@rpi.edu • clarkm5@rpi.edu
http://www.rpi.edu/∼brings

The View Sketched

Is cognitive science relevant to AI problems? Yes — but only when these problems are sufficiently hard. When they qualify as such, the best move for the clever AI researcher is to turn not to yet another faster machine bestowed by Moore's Law, and not to some souped-up version of an instrument in the AI toolbox, but rather to the human mind's approach to the problem in question. Despite Turing's (Turing 1950) prediction, made over half a century ago, that by now his test (the Turing test, of course) would be passed by machines, the best conversational computer can't out-debate a sharp toddler. The mind is still the most powerful thinking thing in the known universe; a brute fact, this.

But what's a "sufficiently hard" problem?

Well, one possibility is that it's a problem in a set of those which are Turing-solvable, but which takes a lot of time to solve. As you know, this set can be analyzed into various subsets; complexity theorists do that for us. Some of these subsets contain only problems that most would say are *very* hard. For example, most would say that an **NP**-complete problem is very hard. But is it *sufficiently* hard, in our sense? No. Let P be such a problem, a decision problem for F associated with some finite alphabet A, say. We have an algorithm $\mathcal{A}$ that solves P.[1] And odds are, $\mathcal{A}$ doesn't correspond to human cognition. The best way to proceed in an attempt to get a computer to decide particular members of $A^\star$ is to rely on computational horsepower, and some form of pruning to allow decisions to be returned in relatively short order.

What we have just described structurally, maps with surprising accuracy onto what was done in AI specifically for the problem of chess. In his famous "20 Questions" paper, written in the very early days of AI and CogSci (and arguably at the very dawn of a sub-field very relevant, for reasons touched upon later, to issues dealt with herein: computational cognitive modeling and cognitive architectures), Newell (Newell 1973) suggested that perhaps the nature of human cognition could be revealed by building a machine able to play good chess. But Deep Blue was assuredly not what Newell had in mind. Deep Blue was an experiment in harnessing horsepower to muscle through a Turing-solvable, but time-consuming, problem. Kasparov lost because he earns his livelihood playing a game that, from the standpoint of the affirmative answer defended in this brief paper, is too easy (as one of us has argued elsewhere: (Bringsjord 1998)).

Things change when you move to Turing-*un*solvable problems. Now of course if you agreed that some humans can solve such a problem, you would have to agree as well that *computationalism*, the view (put roughly) that human minds are bio-incarnated computing machines at or below the Turing Limit, is false.[2] We suspect that many of our readers will be averse to such agreement, and in the present debate that's fine. It's fine because our claim is that if AI researchers want to try to get standard computing machines to solve meaty *particular* problems from a Turing-unsolvable class of problems,[3] they need to turn for help to human cognition. Obviously, one can affirm computationalism while still maintaining that humans have the capacity to solve particular problems from a Turing-unsolvable class. For example, while it may not be true that humans can solve the halting problem, it's undeniable that, for some particular Turing machines of considerable complexity, ingenious, diligent humans can ascertain whether or not they halt.

Now, we must confess that, at least as of now, we don't have a deductive argument for the view that AI must turn to human cognition in order to produce a machine capable of solving particular problems from a set of Turing-unsolvable ones. What we do have is some empirical evidence, connected to specific challenges of the sort in question. That is, we can explain the challenges, and show that AI of a sort that explicitly turns for guidance to the human case is more successful than cognitive-science-ignoring variety in surmounting these challenges. We will *briefly* mention three such challenges here: one that researchers have explored in the Rensselaer AI & Reasoning (RAIR) Lab with indispensable help from many others (esp. Owen Kellett), and with support from the National Science Foundation; another that has in significant part catalyzed a major advance in computational logic; and thirdly, the

[1] I.e., for every $u \in A^\star$, $\mathcal{A}(u) = Yes$ iff Fu (F is true of u), and $\mathcal{A}$ returns No when not Fu.

[2] For a recent clarification of what computationalism amounts to, and an argument that the doctrine is false, see (Bringsjord & Arkoudas 2004).

[3] We include as well problems that *appear* to be Turing-unsolvable.

challenge of getting a machine to learn in a special manner associated, in the human sphere, with learning by reading.

Challenge #1: Busy Beaver

The first challenge is produced by the attempt to "solve" Rado's (Rado 1963) Σ problem (or the "busy beaver" problem).[4] The Σ function is a mapping from $\mathbf{N}$ to $\mathbf{N}$ such that: $\Sigma(n)$ is the largest number of contiguous 1's an n-state Turing machine with alphabet $\{0, 1\}$ can write on its initially blank tape, just before halting with its read/write head on the leftmost 1, where a sequence

$$\overbrace{11\ldots11}^{m \text{ times}}$$

is regarded simply as m.[5] Rado (Rado 1963) proved this function to be Turing-uncomputable long ago; a nice contemporary version of the proof (which is by the way not based diagonalization, the customary technique used in the proof of the halting problem) is given in (Boolos & Jeffrey 1989). Nonetheless, the busy beaver problem is the challenge of determining $\Sigma(n)$ for ever larger values of n.[6]

How would one go about trying to get a computer to determine $\Sigma(n)$, for, say, 6? Well, no matter what you do, you will run smack into a nasty problem: the problem of *holdouts*, as they are called. Holdouts are Turing machines which, when simulated (starting, of course, on an empty tape) continue to run, and run, and run ... and you're not sure if they're going to stop or not.[7] So what to do? Well, smart humans develop ingenious *visual* representation and reasoning to figure out whether holdouts will terminate. Some of these representations, and the reasoning over them, are remarkable.

As a case in point, let us examine one of these representations: so-called "Christmas Trees," discussed in (Brady 1983; Machlin & Stout 1990). For purposes of this discussion, consider the following notation used to represent the contents of a tape:

$$0^*[U][X][X][X][V_s]0^*.$$

Here 0^* denotes an infinite sequence of 0's that caps each end of the tape. Additionally, $[U]$, $[X]$, and $[V]$ represent some arbitrary sequence of characters on the tape while the subscripted s indicates that the machine is in state s with its read/write head at the leftmost character of the $[V]$ character sequence.

With this in mind, we can describe the Christmas Tree pattern in the context of the transformations that it makes on the tape. Machines exhibiting this behavior are classified as non-halters due to a repeatable back and forth sweeping motion which they exhibit on the tape. Observably, the pattern of the most basic form of Christmas Trees is quite easy to recognize. The machine establishes two end components on the tape and one middle component. As the read/write head sweeps back and forth across the tape, additional copies of the middle component are inserted, while maintaining the integrity of the end components at the end of each sweep.

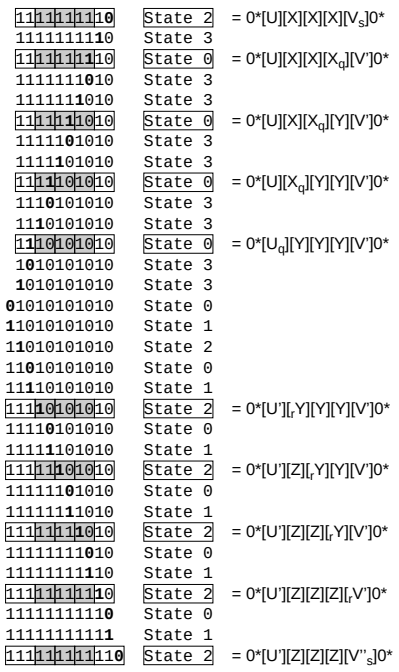

Figure 1: Christmas Tree Execution

Figure 1 displays a partial execution of a 4-state Christmas Tree machine which is representative of one sweep back and forth across the tape. As can be seen, at the beginning of this execution, the machine has established three middle or $[X]$ components capped by the $[U]$ and $[V]$ end components on each side. As the read/write head sweeps back and forth across the tape, it methodically converts the $[X]$ components into $[Y]$ components and then into $[Z]$ components on the return sweep. At the completion of the sweep, the tape is left in the state: $0^*[U'][Z][Z][Z][V_s'']0^*$ which can be shown to be equivalent to $0^*[U][X][X][X][X][V_s]0^*$. Thus each successive sweep across the tape performs similar transformations, adding an additional $[X]$ component in an infinite pattern.

In any attempt to engineer a standard computer to solve ever greater $\Sigma(n)$, it seems necessary to model human visual reasoning. We will report on our latest attempt to model such reasoning, in a system known as Vivid.[8]

[4] For information, go to

- http://www.cs.rpi.edu/ kelleo/busybeaver

[5] As we explain below, there are a number of variations on the exact format for the function. E.g., one can drop the conditions that the output 1's be contiguous.

[6] Because the formal specification of a Turing machine varies (e.g., between the quadruple and quintuple formalisms), the problem isn't fully defined until the specification is selected. But we can leave this aside in the present white paper.

[7] Yes, the Σ problem is related to the halting problem.

[8] See

http://kryten.mm.rpi.edu/vivid_120405.pdf

Challenge #2: Provability

Our second challenge is another sort of unsolvable problem. Let Φ be an arbitrary set of first-order formulas, and let ϕ be a particular arbitrary formula of this kind. Then, as is well-known, there is no algorithm that can decide whether

$$\Phi \vdash \phi,$$

that is, whether ϕ can be deduced from Φ. So what do you do when you nonetheless have to answer this question for *particular* meaty instances of it? By 'meaty' here we mean questions of this type that the best automated theorem provers (such as Vampire: (Voronkov 1995)) completely choke on. The only option is to turn to human ingenuity, and use an *interactive* theorem prover that allows for contributions made by the human mind to be part of the proofs or disproofs. (In the longer run, the human contributions would perhaps be formalized and fully automated.) Arguably the best such system is Athena,[9] a DPL (denotational proof language) (Arkoudas 2000) responsive to the way humans reason, not simply based on what is perceived to be most convenient for a computer (e.g., resolution). We will provide specific examples at the symposium, and will demonstrate for those interested. Here, we give only a *very* simple example of a proof in NDL, a type-α DPL.[10] The example is a proof in NDL of one of the simple theorems proved in 1956 by Logic Theorist:

```
// Logic Theorist's claim to fame (reductio):
//
// (A ==> B) ==> (~B ==> ~A)

Relations A:0, B:0.

assume A ==> B
  begin
    assume ~B
      begin
        suppose-absurd A
          begin
            modus-ponens A ==> B, A;
            absurd B, ~B
          end
      end
  end
```

Readers might want to consider the corresponding proof carried out by resolution, a form of reasoning popular in automated theorem proving, but so far incapable of producing proofs that can be compared with what expert humans produce. Notice that in the NDL proof, there is a block structure that corresponds to the way humans express proofs. At the symposium, we will discuss human-structured proofs of more difficult theorems — theorems apparently beyond the reach of ordinary Turing machine operating in ways not guided by the human case.

[9] Available, with full description, at

http://www.cag.csail.mit.edu/ kostas/dpls/athena

[10] NDL is used for teaching formal logic at RPI, and is available from the course web site there:

http://www.cogsci.rpi.edu/courses/intrologic/materials.php

Challenge #3: Learning by Reading

One of the odd things about AI is that what is called learning in it isn't really what most people associate with the term. For the most part, you and I have been engaged in learning from our earliest days because we have been *reading*. But if you examine a standard AI textbook top to bottom in search of learning by reading, you will come up empty. Instead, you will find that learning means something exceedingly narrow (usually learning a mundane function through multiple trials).

The best current explanation in cognitive science of the difference between those in an academic setting who truly learn by reading, versus those who don't, is that those who do are *self-reflective* in a way that evidently facilitates their correctly answering and defending questions given on tests (Chi *et al.* 1994). This explanation is strikingly consonant with the specific new type of machine learning that distinguishes our approach. We refer to this type of learning as *poised-for* learning, or just p.f. learning. The system under construction that learns in this fashion is PFL.

At the heart of p.f. learning is the idea that if a machine is to have truly learned about (say) algebra and related matters by assimilating a book (or part of one) on the subject, then when we examine the "brain" of this machine (its knowledge and methods it possesses for processing that knowledge), we will be able to find what we call "poised-for" knowledge, or just *p.f. knowledge*. And, accordingly, we may say that the machine has *p.f. learned*. The basic idea is that: p.f. knowledge is *knowledge poised for the semantically correct generation of output that would provide overwhelming psychometric evidence that deep and durable learning has taken place*. In turn, and derivatively, we say that p.f. learning has taken place if the system in question has developed the p.f. knowledge sufficient to enable the generation of output (selecting/writing the correct answer) that provides this kind of evidence.

Additional details on the DARPA-sponsored PFL system will be provided at the symposium. A key aspect of our work is to resrict our attention to English as expressed in what we call **logically controlled languages** (LCLs); we will discuss them at the symposium.[11]

The Cognitive Architecture Connection

Finally, alert readers will be waiting for us to explain why we said, in an earlier parenthetical, that cognitive architectures are relevant to the issues at hand. That explanation, at least in broad strokes, is easy to state. Cognitive architectures are relevant because they mark a key nexus between AI and CogSci: they

[11] One LCL in use is Attempto Controlled English (ACE) (Hoefler 2004). The home page for ACE is

http://www.ifi.unizh.ch/attempto

Another LCL we are exploring is CLCE, information about which is available at

http://www.jfsowa.com/clce/specs.htm

are attempts to engineer intelligent behavior by mechanizing cognition, *human* cognition. This suggests an interesting line of investigation: viz., attempt to use a cognitive architecture to tackle the kinds of "sufficiently hard" challenges we are pointing to. To our knowledge, this investigation hasn't been pursued (it has certainly been *quietly* pursued if it has been). Perhaps it should be. Accordingly, we will report on attempts to get cognitive architectures to solve some of the microcosmic challenges alluded to above. The specific cognitive architecture that we will focus on is a new one: RASCALS (Bringsjord *et al.* 2005; Bringsjord forthcoming).

Acknowledgments

We are indebted to Owen Kellett for work on the busy beaver problem, to Kostantine Arkoudas for his NDL and Athena systems on which we build, to Bettina Schimanski, Gabe Mulley, Eric Pratt, Andrew Shilliday, and Joshua Taylor for work on p.f. learning, and to NSF and DARPA for support of some of the research described herein.

References

[Arkoudas 2000] Arkoudas, K. 2000. Denotational Proof Languages. PhD dissertation, MIT.

[Boolos & Jeffrey 1989] Boolos, G. S., and Jeffrey, R. C. 1989. *Computability and Logic*. Cambridge, UK: Cambridge University Press.

[Brady 1983] Brady, A. 1983. The determination of the value of rado's noncomputable function $\Sigma(k)$ for four-state turing machines. *Mathematics of Computation* 40(162):647–665.

[Bringsjord & Arkoudas 2004] Bringsjord, S., and Arkoudas, K. 2004. The modal argument for hypercomputing minds. *Theoretical Computer Science* 317:167–190.

[Bringsjord *et al.* 2005] Bringsjord, S.; Khemlani, S.; Arkoudas, K.; McEvoy, C.; Destefano, M.; and Daigle, M. 2005. Advanced synthetic characters, evil, and E. In Al-Akaidi, M., and Rhalibi, A. E., eds., *Game-On 2005, 6th International Conference on Intelligent Games and Simulation*. Ghent-Zwijnaarde, Belgium: European Simulation Society. 31–39.

[Bringsjord 1998] Bringsjord, S. 1998. Chess is too easy. *Technology Review* 101(2):23–28.

[Bringsjord forthcoming] Bringsjord, S. forthcoming. The RASCALS cognitive architecture: Logic, top to bottom, to the rescue. In Sun, R., ed., *The Handbook of Computational Cognitive Modeling*. Cambridge University Press.

[Chi *et al.* 1994] Chi, M.; Leeuw, N.; Chiu, M.; and Lavancher, C. 1994. Eliciting self-explanations improves understanding. *Cognitive Science* 18:439–477.

[Hoefler 2004] Hoefler, S. 2004. The syntax of attempto controlled english: An abstract grammar for ace 4.0. Technical Report ifi-2004.03, Department of Informatics, University of Zurich, Zurich, Switzerland.

[Machlin & Stout 1990] Machlin, R., and Stout, Q. 1990. The complex behavior of simple machines. *Physica D* 42:85–98.

[Newell 1973] Newell, A. 1973. You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. In Chase, W., ed., *Visual Information Processing*. New York: Academic Press. 283–308.

[Rado 1963] Rado, T. 1963. On non-computable functions. *Bell System Technical Journal* 41:877–884.

[Turing 1950] Turing, A. 1950. Computing machinery and intelligence. *Mind* LIX (59)(236):433–460.

[Voronkov 1995] Voronkov, A. 1995. The anatomy of vampire: Implementing bottom-up procedures with code trees. *Journal of Automated Reasoning* 15(2).