NSF ITR-(ASE+ECS+NHS)-(dmc+soc+int)

Mathematical Foundations for Understanding and Designing Conceptualizing Strategizing Control (CONSCS) Systems

Manuel Blum (PI), Avrim Blum (Co-PI), Lenore Blum (Co-PI), Steven Rudich (Co-PI),
Research Assistants: Ryan Williams, Luis von Ahn

From Collins *English Dictionary*, 1995:

conscious adj. 4.a. denoting or relating to a part of the human mind that is aware of a person's self, environment, and mental activity and that to a certain extent determines his choices of action. b. (as n.) the conscious is only a small part of the mind. Compare *unconscious*.

1. Introduction: Proposal Objectives and Expected Significance.

There is a critical need for, and a great deal of interest in, the design of self-aware programs, robots, and systems. Robots that are aware of their own state and the state of their surroundings are better able to deal with unexpected adversity, changes in goals, new information, and to make long-term plans. Indeed, a number of areas of AI deal with different models and approaches to self-awareness, as for example the explicitly modeled belief-state of a robot in a Partially Observable Markov Decision Process (POMDP) framework [KaelLitCas, JaaSinJor, Monahan, SutBar].

Our goal in this proposal is *to initiate an automata-theoretic study of self-awareness*, and what we would mean by a truly self-aware autonomous system, in terms of both its behavior and its internal design. More ambitiously, we would like to develop a set of definitions for an automata-theoretic version of consciousness, what we call a CONceptualizing Strategizing Control System (CONSCS), that would enable us to prove theorems about CONSCSness, and would *help to support the eventual development of a truly self-aware robot.*

The high-level idea of the proposed framework is for an entity to have a simplified internal model of itself and its relationship to its environment. The internal models are required to have certain properties, and support *narratives* or *stories* that provide justifications for proposed behaviors as well as predictions for what the outcome of a behavior will be.

A separate but related goal of our work is to develop simple behavioral tests that measure the extent to which a system is CONSCS. In this context, we aim to build on our previous work on CAPTCHAs (Completely Automated Public Turing tests to tell Computers and Humans Apart; see http://www.CAPTCHA.net)[1]. A CAPTCHA is a task that is extremely easy for a human (even a child) to perform, and yet extremely difficult for computer programs. Furthermore, it is a test that can be given and graded by a computer program. Such tests have a number of practical uses. For example, Yahoo! uses our CAPTCHAs to keep software "bots" from signing up for free email accounts (e.g., see http://edit.yahoo.com/config/eval_register).

[1] Our prior work on CAPTCHAs was developed under the auspices of the NSF-ALADDIN Center (http://www.aladdin.cs.cmu.edu) at Carnegie Mellon, supported by NSF Grant CCR-0122581.

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu

Examples of CAPTCHAs.

In addition, from a scientific perspective, CAPTCHAs serve as challenges to the AI community (currently, these tests mainly involve vision and OCR tasks). We would like to develop similar sorts of tests for self-awareness in particular, and for CONSCSness in general, that can help us better understand the strengths and limitations of the current state of the art.

In short, we seek a mathematically concise automata-theoretic definition for understanding and quantifying consciousness, and for designing CONSCS robots.

To start, we state a list of axioms that we believe are essential for laying out a proper definition: the CONSCS-defining game. We then state a few basic properties of CONSCSness that should follow as theorems of the definition. Finally, we suggest a specific form that a definition of CONSCS should take, and why.

We lay down a methodology for checking that our notion is on track. Briefly, we apply our definition of CONSCS -- initially created with the human in mind -- to non-human entities. This is a reality check: we firmly require that any automata-theoretic definition of consciousness correctly classify common sense examples, e.g. humans are CONSCS, but rocks are not CONSCS. A good definition, moreover, should elucidate near misses, e.g. exactly why are minimax chess programs not CONSCS? What would it take to make them so?

Our tools will be those of theoretical computer science. In particular, the tools of automata theory, algorithms, complexity theory, machine learning theory, and cryptography should all prove useful in our research. For example, theoretical cryptography is especially good at modeling interactions between parties; in our case, we will want to model interactions between a computer and its outside world.

2. Why this is the right time for taking a theoretical computer science approach to the study of consciousness: Relation to other work.

Developments in cognitive science and in artificial intelligence have brought us to a point where a serious study of self-awareness and consciousness is no longer a pipe dream [ASSC, CCJ, AAAI, Rochat, BerBroad, McCarthy95]. Empirical developments in cognitive sciences now give us a potential wealth of data about brains and behavior, such as fMRI and simultaneous multiple-electrode brain recordings [DembDesWag, FiezPet, KanMcDChun, PohlMillMuss]. These developments permit psychologists and neuroscientists to test hypotheses about brains that are far bolder and more ambitious than ever before [HenTreyNag, Farah, JacYonJen].

Machine Learning has become highly developed in theory and in practice; in very little time it has evolved into a distinctive area of its own [Angluin, BlumBlum, Ellman,

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu

KeVaz, Mitchell, ThrunMit, Val, WatDay]. The Web, though still in its infancy, has already given us amazing connectivity. Google and other search engines have augmented the role of the personal computer-- from that of a simple task manager, to one with an incredibly knowledgeable encyclopedia of past and present. Until recently, computers were blind, deaf, and unable to move about in the world. This situation will inevitably change with (improving) access to the Web. Already, methods have been devised to use the Web for large-scale machine learning tasks. By harnessing human cycles to classify and describe images from the Web, the ESP Game (http://ESPgame.org) is implicitly building a massive knowledge base for teaching computers about the world.

Researchers in AI and software engineering have various mechanisms with a notion of "self-awareness" built into them. For example, in a POMDP an agent has bounded observational powers, and thus has a *belief state* it uses to ascertain the current state of the world (cf. [Monahan]). An agent distinguishes between its beliefs and what is actually true, and it chooses the best action to take based on its beliefs. In reinforcement learning, an agent chooses its action based on observation of its previous actions. In the area of multi-agent learning, recent works informally classify an agent algorithm as "self-aware" if it can detect the effects of its own actions on agents, e.g. [ConSand]. Similarly, work in software architectures on "self-healing" and "self-adapting" systems posit that the system has "reflective" models of its own operation and capabilities, in order to determine how to repair itself when problems arise [Wile]. In the Java programming language, there is a "Reflection API" to facilitate this; it contains methods for returning fields, methods, and constructors of a class. This addition has been repeatedly said to make classes "self-aware" (e.g. [PiddCas]), as reflection allows objects to refer to themselves. A definite need has arisen for a general theoretical framework for clarifying and quantifying this common idea of "self-awareness" among these disciplines.

Explanation based learning (EBL) is a kind of machine learning that resembles our idea of justification [Ellman, DeJMoon]. In EBL, a computer attempts to learn general concepts by witnessing events (or, examples). Using its model of the world, it tries to deduce the most general explanation for why the event occurred (or, why the example had its label). In the past, we explored inductive justification on an automata-theoretic level [BlumBlum]. What most directly contrasts our proposed approach from these is that the formation of justifications should also take into account the tradeoff between resource bounds and explanatory power.

We believe that theoretical computer science has the potential to contribute to further progress in this enterprise, particularly with regard to applications to artificial intelligence. Theory has proved useful in the past, in similar more focused projects. The proof of the existence of self-reproducing machines is the basis (for better or for worse) for the design of all self-reproducing machines, including designs for self-reproducing nanotech devices and computer viruses. In like manner, we argue that conscious bots are more likely to develop on top of an automata-theoretic definition and a mathematical version of consciousness.

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu

3. What distinguishes our approach?

Our approach is a formal mathematical one. As stated previously, our work is based on automata theoretic considerations. More significantly, it is foundational and top-down: Our definition for CONSCS is intended to capture characteristics that are arguably necessary for any reasonable automata-theoretic notion of consciousness. The definition should apply in principle to any imaginable computational device: a human, a chess-playing program, a bot.

For a kindred example, the theoretical computer science definition of a pseudorandom generator is foundational and top-down:

- It elucidates properties of the concept at an abstract level; it captures characteristics (arguably) common to all possible pseudorandom generators.

- The concept itself was so broad that it was previously considered to be incapable of proper definition; the task of classifying a program as a pseudorandom generator was at one time something of a "black art" [Knuth].

Unlike pseudorandomness, however, consciousness is a huge and broadly encompassing concept. Our hope is to discover, by way of our top-down approach, certain aspects of consciousness for which we can feasibly construct a viable but rich theory of CONSCS, based on tools from automata and complexity. We have already found a wide variety of aspects which may be explored through our approach; these will be elaborated upon in later sections.

4. Toward a high level definition of CONSCS.

We look to give a high-level (i.e. top-down) definition of CONSCS. What we give here, while still too vague and imprecise to be a formal definition, indicates what we have in mind. Our definition is further formalized in Section 8.

Definition. An *environment* is a computational device that is connected to a collection of other computational devices, called *entities*. The environment interacts with these entities and manages all communications between them; we often say an entity is in the environment to which it is connected. (The real world is an example of an environment. People, trees, and rocks are examples of entities in that environment.)

Definition. An entity is CONSCS relative to its environment iff the following five conditions hold:

1. **The entity has an *interface* with its environment.** Exactly how the interface works is something the entity learns from experiment. One may imagine the interface as the cockpit of a machine; the entity, seated in the cockpit, determines the functionality of controls by experimentation.

As a consequence of condition 1, the entity may/must model itself, in a sense, as a homunculus at the controls of a complex mechanism.

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu

2. **The entity has (creates/develops) a *model of its environment* and a *model of itself in its environment*.** These models constitute a theory of its world; they are used to create narratives and make predictions. The models are developed from inputs received from the environment, outputs to the environment, and from experiments (sequences of inputs and outputs) on the environment.

A CONSC entity is general-purpose: the less the entity has built in about its world, the better it will be at adapting to all possible worlds. The 'model of self' rather than the 'actual self,' is needed for making predictions because time (to do computation) and information (about environment and self) are limited.

3. **The models are constantly under *tactical* revision.** The entity has algorithms that are on the lookout for *unexpected events* (events contrary to what have been predicted) and *significant events* (events labeled positive or negative). After predicting the consequences of various actions, choosing and executing the *most favorable* (by some measure), the entity then compares actual observed consequences with previously predicted consequences. It learns from its mistakes, from its errors, thereby improving its models and, by some measure, its future predictions. Most importantly, the entity is universal: roughly, its learning algorithms are sufficiently powerful to enable it to acquire any model that can be acquired by a general-purpose CONSCS.

4. **The models are occasionally under *strategic* revision.** The entity has the ability to try on any *story* (that an entity could use to make decisions) for size. A *story* is a theory or model for how the world functions. It is important for supplying a value system, or guiding principle, for deciding which of several competing choices to make. It enables the entity to make decisions, when tactics alone can't decide.

A consequence of either requirement 3 or 4 is that in a sufficiently complex environment, the entity has the computational capability of a Universal Turing Machine.

5. **The entity is *motivated/driven* towards certain *goals*.** These are determined by its environment, by the significance it attaches to events, and by its current story of how the world functions.

5. The CONSCS-defining game.

Developing a coherent theory of consciousness is a daunting task with countless points of debate. We take a more modest approach of constructing a reasonable theory within the language of automata and complexity. As a reality check about the relevance of our theory, we propose to lay down some basic ground rules and assumptions along the following lines:

1. Any definition of CONSC should ensure that people are CONSCS while rocks are not (CONSCS) in the real-world environment. However, a rock may be studied as an entity interacting with the environment.

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu

2. The question whether something is or is not CONSCS is a function of its algorithms; it is not a function of what implements those algorithms. Consequently, a silicon-based CONSCS is possible. More formally, the question whether or not an entity is CONSCS in a given environment is determined (though not necessarily effectively so) by the entity's code (i.e. the rules encoded in its structure) and a description of its environment.

3. The formal definition of CONSCS should be mechanically applicable (whether positively or negatively) to any entity in a well-defined environment. One should not have to be conscious in order to test if an entity is a CONSCS. (This requirement has a similar (but weak) analogy with CAPTCHAs, which automatically distinguish humans from bots, without being human. However, we do NOT require from the start that our definition of consciousness be "black-box", i.e. based solely on communications between entities and environments. One may be forced to "open up" an entity in order to perform the test.)

6. The role of narratives: stories, principles and prediction

A Conceptualizing Strategizing Control System will be capable of forming and testing theories about its environment at a high level. The ability to compose and compare *narratives* will be an essential part of a CONSCS. Narratives are necessary for humans because while our memories are vast, the bandwidth between our memory and our perception is small. We cannot remember every pixel, tone, and sniff of our experience at any moment, but we can remember a compressed interpretation (a summary) of what happened. This interpretation or narrative allows us to recall and reconstruct essential features of the event, sometimes down to extraordinary detail, sometimes not. Concepts of narrative have been proposed in the A.I. literature; however, each proposal has depended on some proprietary formal language, developed around the situation calculus [BarGabPro, KakMil, Karlsson, McCarthy00, Miller, MilShan, Reiter]. The concept lacks a cohesive, unifying theory. (For example, Reiter's ideas only apply to a dialect of PROLOG; we are looking for something far more universal.)

Often, one finds oneself "narrating" what is happening at a given moment, or what happened in the recent past. ("Hmm, it's snowing outside." "That tree just fell over.") These narratives are typically short, simple descriptions of actions occurring within the observable surroundings; they are not precise but they can be easily parsed. Later (when trying to recall the snowfall) one remembers what one said to oneself or to others. We will call these kinds of narratives *stories* -- they constitute our interpretation of our history. In the context of reasoning, stories are most useful for being artifacts of experience: facts which we believe did hold in the past, facts from which we hope to learn.

Another kind of narrative, a *prediction*, states a proposition about what is likely to happen in the future. Predictions are useful for selecting among choices being considered, and for recognizing unexpected surprising events -- events that do not fall into the predicted class of likely events. These events are recognized as surprising only after the fact.

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu

In summary, stories narrate what happened in the past, principles are narratives on what one should do in the present (or near-future), and predictions narrate what (one thinks) will happen in the future.

7. Towards a formal theory of narratives.

We briefly outline our ideas towards developing a formal theory of *narratives*. All three types of narrative (*stories, principles, predictions*) may be construed as a kind of decision algorithm that determines a relationship between events in an environment.

Machine learning theory has been primarily concerned with the learning of *concepts*, i.e. Boolean-valued functions over some space of features [Val94, KeVaz]. One can easily imagine an extension of concept-learning that fully incorporates a notion of temporality. These "temporal concepts" will (appropriately) be called *events*. Events are building blocks from which narratives will be composed. Informally, events are actions involving objects which are captured by conventional concepts.

Narratives posit relationships between different kinds of events. *Stories* indicate that when an event of some type A occurred, it was followed by another event of some type B. *Principles* suggest that when events of type A are being witnessed, one should attempt to bring about a type B event. *Predictions* say that if a type A event occurs, a type B event will follow.

For example, one can imagine a computer learning "Leaves fall in autumn" by:

- Making its goal to find a concept-action or event A (something happening) such that leaves fall after event A in its ordered sequence of snapshots.

- Observing actions of objects over many seasons, paying special attention to unusual/surprising events that occur not long before the leaves fell. It gets colder. The wind starts being chilly. The leaves fall shortly thereafter.

What would make cold surprising? Perhaps, its rarity in past experience leads to surprise. When a relatively stable attribute undergoes a drastic change through a short sequence of snapshots, this may be seen as surprising. Also, we might think of surprising or "attention-worthy" attributes to be those which profoundly distinguish one sequence of snapshots from a previous sequence. (The "frame problem" naturally arises in this context; by our informal definition of 'surprise', anything that is "inertial" or essentially unchanging from one frame to the next is usually not surprising, cf. [Shanahan], [McCarthy86].)

8. Towards a formal definition of CONSCS.

We now outline a high-level definition for CONSCS. While considerably more formal than our earlier discussion, this is not yet a formal treatment. Much of the terminology we use is informal; ultimately it needs to be made more formal and precise --and indeed that is a major goal of this research.

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu

We first give a very general "grey-box" definition of an entity and an environment. In both entities and environments, we assume that changes in the systems take place under the timing of a discrete clock.

Definition. An *entity T* is a function implemented in a Turing-complete machine model with bit strings as inputs and outputs. The unbounded storage of *T* is modeled by the functions *get_T*(*i*) and *store_T*(*i*, *x*), where *i* is an integer index and *x* is a string. The functions obey a simple "correctness" axiom:

- For all integers *i, k* and strings *y*, if *store_T*(*i*, *x*) is called at time *t*, *get_T*(*i*) is called at time *t*+*k*, and *store_T*(*i*, *y*) is not called over all times *t*+1, …, *t*+*k*-1, then *get_T*(*i*) = *x*.

Similarly, an environment *V* is a function implemented in a Turing-complete machine model, with its unbounded storage modeled by functions *get_V* and *store_V*. Environments may be seen as the generators of inputs (the *stimuli*) for an entity *T*, and as potential modifiers of *T*.

To model the notion that an entity *T* operates *inside* an environment *V*, we allow *V* to have unlimited access to *T*'s memory (as if *T*'s memory were part of *V*'s). Furthermore, *V* dictates which of its parts *T* can perceive (stimuli) and affect.

Definition. A *stimulus* is a set of pairs of the form (*i*, *x*), where the *i*'s are memory locations (integers) of *V* that *T* can write to, and the *x*'s are the current contents of *V* at location *i*. A *response* to stimulus *s* is a finite series of *get_V* and *store_V* function calls that only access locations *i* that appear in some pair of *s*.

We say entity *T* is *interacting* in environment *V* if *V* has access to *T*'s memory (*V* has access to the functions *get_T* and *store_T*), and *T* has limited access to *V*'s memory, in a way to be defined below.

Interaction of entity *T* in environment *V* follows a protocol:

1. *V* reads from *T*'s storage.
2. *V* writes to *T*'s storage.
3. *V* outputs stimulus *s*.
4. *T* is input stimulus *s*.
5. *T* outputs a response to *s*.
6. Timestep *t* is incremented.

.

Remarks:

1. The stimulus *s* could be given as a *coded implementation*, or as an explicit set.

2. Arguably, *T* should also be able to write to locations it cannot "see", and read from locations to which it cannot write. We keep these sets the same only for simplicity in the description above.

The environment of *T* at time *t* is its input *s* from *V* at that timestep.

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu

Definition. For an entity to be CONSCS relative to a given class of environments, it must have an *environment model* and a *self model*.

- The *environment model* of a CONSCS is responsible for narratives concerning awareness of the external environment, i.e. those aspects of the entity's experience that are out of the entity's direct control (though possibly under its indirect control). The narratives dealt with by this model are predictions and stories, but not principles. One may recollect what happened in the environment today, and venture at what might happen tomorrow, but not what the environment ought to do.

Justifications for narratives of the environment would be mainly based on other narratives from the environment. More precisely, predictions are validated or invalidated through the collection of new stories, and predictions may be shown to be false in light of new evidence.

- The *self model* deals with narratives concerning awareness of the "inner environment" of an entity, sensory input coming to it from sensors of its own internal environment, principles (determining what actions should be taken) and stories about its 'sensations'. Goals are formed, evaluated, and applied here. Justifications for principles in the self model are narratives that could very well depend on past stories, predictions, and goals.

A 'model of self' is needed because time (to do computation) and information (about the environment) are limited; a specialized model for the purposes of action-taking (rather than prediction-making) is required. In order to form narratives about what should be done and why, a separate model serves to give a clear distinction between what aspects of the environment a CONSCS can control, and what it (mostly) cannot.

We believe that the self model will perhaps be one of the most useful and interesting aspects of our proposal. A computational device with a coherent self model could learn to be more fault-tolerant as it learns the consequences of its own actions (learns how to delegate blame for a system failure to itself, versus other parties). To model a fault-tolerant system on a formal level, one might conceive of online versions of 'self-correcting' and 'self-testing' of functions, where the values of a function dynamically change over time, and one's task is to maintain a black-box approximation of it that attempts to "converge" to the dynamic function. Another potential application of an entity equipped with a self model would, for example, be for automated robots traveling to and exploring Mars; many control systems have to be maintained and kept on-line, in spite of any unforeseen circumstances.

To formalize the above definitions, we will have to formalize a general notion of a *model* capable of *forming* narratives, *evaluating* competing narratives of the same type and *applying* existing narratives (principles and predictions) to compute the entity's response. A number of basic properties will be required of our models. They will be developed from inputs from the environment, from the entity itself, and from experiments on the environment. For example, our models must be capable of evolving over time, having

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu

mechanisms to determine if mistakes have been made and learning from their mistakes. An important property will be *model-completeness*.

- *Model-completeness* conveys the idea that we want the models of a CONSCS to be highly robust and capable in their narrative ability. Thus, a CONSCS should intuitively be universal in that it is capable, at least in principle, of producing (and comparing) *any conceivable narrative* for a large class of environments. This is necessary, as computer environments (networks, operating systems, etc.) can vary greatly in their functionality and representations. Notice that model completeness immediately rules out any table-based expert systems as being CONSCS -- such systems cannot contain preprogrammed stories for every event, assuming the environment is sufficiently expressive.

9. Towards a behavioral test for a CONSCS.

Ideally, one would like a behavioral CAPTCHA-like test for CONSCS, i.e. a test of CONSCS that can be verified by a party that has simple interactions with the entity via an environment. Of course, a notion of CONSCS that is Turing-testable may simply not be possible. In this section, we outline a weaker definition of CONSCS, in which the goal is to design a property testable through interactions.

We imagine a system (much like a challenge-response system) where a tester *R* produces a sequence of (physical) events e_1, e_2, ... in the environment *V* for the entity *T* to respond to. For each event e_i, *T* responds with a "story" s_i about what happened during the event. R uses this "conversation" to decide (algorithmically) if T is CONSCS.

We do *not* posit a common language of communication between the testing party and the entity. We do not want our notion to be language-dependent. Rather, we would prefer to utilize only the expressivity of the environment in our analysis of communications between entity and tester. The only requirement we wish to have is that of *model consistency.*

- An entity exhibits *model-consistency* if there is some efficiently determinable connection between the entity's events perceived and the entity's stories communicated. Essentially, model consistency merely requires that there are (detectable) patterns in the communicated stories that are reflected in the corresponding events, i.e. we have a direct correspondence between the model and the real world.

We say that an entity *attends to* an event e at time *t* in the environment if all or a large fraction of the properties of e that can be perceived by the entity at time *t* are perceived.

We say that an entity is *CONSC of event e at time t* iff the entity attends to e, and there is an $\varepsilon > 0$ such that the entity is capable of communicating a story *s* that is consistent with what was perceived from time $t - \varepsilon$ to time $t + \varepsilon$. That is, for some $\varepsilon > 0$, there is some connection between that perceived between $t - \varepsilon$ and $t + \varepsilon$ and story *s*.

In this behavioral notion, an entity *x* would be a CONSCS in an environment iff

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu

- For an arbitrary event e (at time t), if *x* is placed (in time and space) where it can attend to e, then it is *conscious* of event e at that time. That is, entity *x* has an environment model that is *model complete* and *model consistent.*

Note that the concept here is strictly weaker than our definition in the prior section. Note also that we do not even require a separate self model. However, model consistency does rule out some non-trivial scenarios: a pseudorandom generator would not be labeled a CONSCS by a third party, as it would not exhibit model consistency. (Small changes in the environmental seed cause large changes in the output story.) Of course, in the previous definition, the requirements on models easily banish pseudorandom generators from being a CONSCS.

10. Broader Impact: Integration of Research/Education/Diversity.

All senior members of this project have had considerable experience --and recognition-- for their work in integrating research and education in innovative ways, be it at the high school, the undergraduate or graduate level. A commitment to bringing the broadest group of students into the enterprise is manifest in all our work and will assure the broadest possible impact of this project.

Manuel Blum is renowned for having nurtured a large number of the most important and innovative leaders in computer science today, and continues to do so. Avrim Blum is probably the most sought after thesis advisor in the Carnegie Mellon Computer Science Department. Lenore Blum has spent more than thirty years creating and developing programs to increase the participation of women in science in technology; for the past 5 years at Carnegie Mellon, she has focused on increasing the successful participation of women in computer science (see: http://women.cs.cmu.edu). Steven Rudich, a devoted researcher, has a dual commitment to education. He has created one of the most stimulating undergraduate CS courses ever, "Great Theoretical Ideas in Computer Science" (http://www.cs.cmu.edu/~15251) and runs a summer program in CS for talented high school students, "Andrew's Leap" (http://www.cs.cmu.edu/~leap).

All members of this project -- faculty, students and postdocs -- are affiliated with the NSF ALADDIN Center (http://www.aladdin.cs.cmu.edu) that fosters "technology transfer" between theory and applications (ALADDIN = ALgorithm ADaptation, Dissemination and INtegration). For example, the CAPTCHA project, an outgrowth of ALADDIN, fostered synergy between the Carnegie Mellon Theory Group, Yahoo! and PARC. In addition, ALADDIN has fostered inter-age research teams of students (high school, undergraduates, graduates and postdocs) and faculty working together during the school year and summer. The interdisciplinary nature of much of ALADDIN research, like that proposed here, allows a range of participation.

We intend to incorporate and expand this integrative model of research and education in our proposed project. Hence we are requesting funds to support one postdoc, two graduate students, two undergraduate students and two high school students to work collaboratively on our project. We plan to facilitate the participation of underrepresented groups in our mix. To do this we will work closely with ALADDIN (of which L. Blum is co-director along with Guy Blelloch) and with the active campus organizations Women@SCS and Women@IT (of which L. Blum is faculty advisor). The high school

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu

students will participate in the Andrew's Leap program for part of the summer, and then participate more fully with our team during the rest of the summer.

In addition to the scientific and educational outcomes for the students, we believe all will benefit from the mentoring, networking and role modeling inherent in the way we structure team collaboration, weekly seminars, presentations and student colloquia.

Indeed, this has been our experience: For example, in conjunction with ALADDIN and the Pinnacle Project, M. and L. Blum (faculty), Nick Hopper (CS graduate student), Ann Lewis (CS undergraduate student) and Matt Humphrey (high school student and Andrew's Leap participant) collaborated for a year (2002-2003) on a research project to implement Nick's PhD thesis on public key steganography. As of 2/24/04, Nick is about to receive his PhD, Ann is a member of the algorithms group at Amazon.com working on their book-search project (hired by Udi Manber, Amazon's CAO (Chief Algorithms Officer)) and Matt is about to formally embark on his college career in CS.

References Cited

Machine Learning References

[Mitchell] Tom Mitchell. *Machine Learning.* McGraw Hill, 1997.

[Ellman]Thomas Ellman. Explanation-based learning: a survey of programs and perspectives. *ACM Computing Surveys* 21(2):163-221, 1989.

[KharRoth] R. Khardon and D. Roth. Learning to reason. In Proceedings of the National Conference on Artificial Intelligence, pages 682--687, 1994.

[Khardon] Khardon, R. 1996. Learning to take actions. In Proceedings of the National Conference on Articial Intelligence, pages 787-792.

[MitKelKed-C] T.M. Mitchell, R.M. Keller, and S.T. Kedar-Cabelli. Explanation—based generalization: A unifying view. Machine Learning, 1(1):47--80, 1986.

[DeJMoon] DeJong, G. F. and Mooney, R. Explanation-based learning: An alternative view. Machine Learn- ing, 1(2):145-176, 1986.

[KeVaz] M. J. Kearns and U. V. Vazirani. An Introduction to Computational Learning Theory. MIT Press, 1994.

[Monohan]G. Monahan. A survey of partially observable Markov decision processes. *Management Science 28*: 1-16, 1982.

[ThrunMit]Sebastian B. Thrun and Tom M. Mitchell. Lifelong Robot Learning. *Robotics and Autonomous Systems* 15:25-46, 1995.

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu

[WatDay]C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine Learning 8*:279-292, 1992.

[BlumBlum] Lenore Blum and Manuel Blum. Towards a mathematical theory of inductive inference. *Information and Control 28*:122-155, 1975.

[ConSand] Vincent Conitzer and Tuomas Sandholm. AWESOME: A General Multiagent Learning Algorithm that Converges in Self-Play and Learns a Best Response Against Stationary Opponents. *Proceedings of 20th International Conference on Machine Learning (ICML 2003),* 2003.

[Wile] David S. Wile. Towards a Synthesis of Dynamic Architecture Event Languages. *Proceedings of the first ACM Workshop on Self-healing Systems*, 79-84, 2002.

[PiddCas] Michael Pidd and R. A. Cassel. Three phase simulation in Java. *Proceedings of the 30th ACM Winter Simulation Conference*, 367-372, 1998.

[Val] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM 27:*1134-1142.

POMDP References

[KaelLitCas] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. Artificial Intelligence, Volume 101, pp. 99-134, 1998.

[JaaSinJor]T. Jaakkola, S. P. Singh, and M. I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In *Advances in Neural Information Processing Systems 7*, 345-352, MIT Press, 1995.

[SutBar]Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 1998

AI References

[McCarthy] McCarthy, J. (1995), Making robots conscious of their mental states, in `AAAI Spring Symposium on Representing Mental States and Mechanisms'.

[AAAI] AAAI Spring Symposium on Representing Mental States and Mechanisms. ftp://ftp.cc.gatech.edu/pub/ai/symposia/aaai-spring-95/descr.html

[Russell] Stuart J. Russell. Rationality and intelligence. *Artificial Intelligence 94*:57-77, 1997.

[Val94] Leslie G. Valiant. *Circuits of the Mind.* Oxford University Press, 1994.

[Chalmers] David J. Chalmers. *Philosophy of Mind: Classical and Contemporary Readings.* Oxford University Press, 2002.

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu

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

[Minsky] M. Minsky. *The Society of Mind.* Simon and Schuster, New York, 1986.

[Vreeswijk] G. Vreeswijk (1989), An Introspective Machine, report no. IR-184, Vrije Universiteit, Amsterdam.

Frames

[McCarthy86] John McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence 26*(3):89--116, 1986.

[Shanahan] Murray Shanahan, *Solving the Frame Problem: a mathematical investigation of the common sense law of inertia*. MIT Press, 1997.

Narratives

[Reiter] Raymond Reiter. Narratives as programs. In *Proc. of the Seventh International Conference on Knowledge Representation and Reasoning* (KR2000), 99--108, 2000.

[BarGabPro] Chitta Baral, Alfredo Gabaldon, and Alessandro Provetti. Formalizing narratives using nested circumscription. *Artificial Intelligence 104*:107-164, 1998.

[McCarthy00] John McCarthy. Situation calculus with concurrent events and narrative. Manuscript, 2000. *http://citeseer.ist.psu.edu/mccarthy00situation.html*

[Miller] Rob Miller. *Formal Reasoning about Actions and Narratives.* Ph.D. Thesis, University of London, 1995.

[MilShan] RobMiller and Murray Shanahan. Narratives in the Situation Calculus. *Journal of Logic and Computation 4* (5):513-530, 1994.

[KakMil] Antonis C. Kakas and Rob Miller. A Simple Declarative Language for Describing Narratives With Actions. *Journal of Logic Programming 31*(1-3):157-200, 1997.

[Karlsson] Lars Karlsson. Anything Can Happen: On Narratives and Hypothetical Reasoning. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR 98)*, 36-47, 1998.

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu

Additional References

[Knuth] Donald Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms, 3/E*. Addison Wesley, 1998.

[Pap] Christos H. Papadimitriou. The Complexity of Knowledge Representation. In *Proceedings of the Eleventh Annual IEEE Conference on Computational Complexity*, 244-248, 1996.

[ASSC] The Association for the Scientific Study of Consciousness
http://www.assc.caltech.edu/

[CCJ] Consciousness and Cognition Journal
http://www.elsevier.com/wps/find/journaldescription.cws_home/622810/description#description

mblum @ cs.cmu.edu avrim @ cs.cmu.edu lblum @ cs.cmu.edu rudich @ cs.cmu.edu