AI Magazine Volume 11 Number 5 (1990) (© AAAI)

holds a Ph.D. in Computer Science and a Ed.D. in Education, both from the University of Massachusetts.

Elliot Soloway is an Associate Professor of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor, MI. He directs the "Highly-Interactive Computing Environments" project located in the AI Lab.

William J. Clancey is a Senior Research Scientist at the Institute for Research on Learning, an independent, not-for-profit organization. His current interests are relating AI programming to traditional scientific modeling, studying computer systems in the workplace, and re-examining the relation of cognitive science theories to the processes of human memory and learning.

Kurt VanLehn is an associate professor in the computer science department and a senior scientist at the Learning Research and Development Center, both at the University of Pittsburgh. Dr. VanLehn's main research interest is in cognitive simulation and its applications to education. He holds a Ph.D. degree in Computer Science from MIT.

Dan Suthers holds a M.S. in computer science from the University of Massachusetts, where he is currently completing his Ph.D. His research is concerned with the design of computer-based media for knowledge communication, including planning explanations in an interactive context, representation and use of multiple perspectives and granularities in explanation, and multimedia interfaces.

Thoughts and Afterthoughts on the 1988 Workshop on Principles of Hybrid Reasoning

Alan M. Frisch and Anthony G. Cohn

The 1988 Workshop on Principles of Hybrid Reasoning, a one-day AAAI-sponsored workshop, was held in St. Paul, Minnesota on August 21, 1988, in conjunction with the National Conference on Artificial Intelligence. This article reports on the workshop and presents some of our afterthoughts based upon prolonged discussion of the issues that arose during the workshop. To a certain extent this article can serve as a survey of research on hybrid reasoning; to aid in this purpose we include numerous citations to the literature. All references can be found in the bibliography by Alan Frisch and Richard Scherl that accompanies this report.

Researchers in Artificial Intelligence recently have been taking an increasing interest in hybrid representation and reasoning systems—systems that consist of two or more integrated subsystems, each of which may employ distinct representation languages and inference systems. Though a number of such systems have been designed, studied, constructed, and put into use, little effort has been devoted to comparing the systems or searching for common principles underlying them.

The workshop addressed this need by bringing together a small number of leading researchers on hybrid reasoning for a day of intensive interaction. The workshop was organized by Alan Frisch (Workshop Chair), Ron Brachman, and Rich Thomason.

Each participant was invited to submit a short paper that best characterized their work on hybrid reasoning. The submissions were collected into a proceedings distributed to all participants prior to the workshop. As the submissions included previously-published papers as well as early drafts of work in progress, it was agreed at the workshop that the proceedings would be distributed no further. However, since most of the draft papers have subsequently appeared in published form, it is now possible to give a virtual proceedings. In the bibliography that accompanies this article published versions of the submitted papers are indicated with an asterisk.

Overview of the Workshop

The workshop program comprised seven invited talks, two moderated discussion sessions, one dinner, and two coffee breaks. We outline the talks and discussion sessions below and leave the dinner and coffee breaks as an exercise for the reader. We concluded with an informal late-night discussion on whether the workshop had been useful to the participants and whether we would like another one. The overall feeling was that the workshop had been useful and another would be desirable, but it should be either longer or devoted to discussion of selected position papers on a specific topic.

Characterization of Hybrid Knowledge Representation and Reasoning Systems

Peter Patel-Schneider began his presentation by categorizing various types of hybrid knowledge representation and reasoning systems. A system can qualify as hybrid by employing multiple representations or by employing multiple reasoning methods, thus suggesting a characterization of hybrid systems along these two dimensions. Along the representation dimension, a system can have multiple redundant representations of the same knowledge in different media—as in the vivid reasoning system (which we shall call VIVID) presented by Brachman and Etherington (Etherington *et al.* 1989) and the multiple reasoners at a single layer of the CAKE architecture (Rich 1985)—or it can have different representations for different kinds of knowl-

edge—as in KRYPTON (Brachman *et al.* 1983) and many-sorted logics (Cohn 1987; Frisch 1989; Walther 1987). Along the reasoning dimension, a system can have different reasoners for the same representation—as in VIVID—different reasoners for different representations—as in KRYPTON, KL-TWO (Vilain 1985), theory resolution (Stickel 1985a) and many-sorted logics—or the same reasoner for different kinds of knowledge—as in Patel-Schneider's 1987a hybrid logic. Finally, it is possible to have a hybrid reasoner without a hybrid representation. In this case a system might have several independent reasoners or a general purpose, complete but slow reasoner together with one or more specialized, fast, but incomplete reasoners, all using a common representation.

Patel-Schneider suggested two requirements that should be met by a hybrid system; these were generally accepted. Firstly, a hybrid system should have a model-theoretic semantics, and this semantics should be common to the different representations to ensure a coherent semantics for communication. Secondly, the various reasoners of a hybrid system should be tightly coupled so that information from one component affects the actions of other components. The first requirement appears to rule out blackboard systems since most of these have an ad hoc semantics at best. However, it might well be argued that a blackboard system is a hybrid architecture since the knowledge sources frequently embody specialized reasoners. The group also discussed whether a meta-level reasoning system is a hybrid system, but no conclusion was reached.

The main benefits of hybrid representation and reasoning systems include: 1) increased expressive power because a single language may not represent everything easily; 2) increased reasoning power, that is, the ability to make more inferences; and 3) increased efficiency because a specialized reasoner can make optimizations that a general purpose reasoner cannot. Each hybrid system can be evaluated by the benefits it obtains in each of these areas.

Some pitfalls are associated with the hybrid approach, particularly involving problems of integration. It is possible that certain kinds of inferences may not be performed because they "fall in the cracks" between the different reasoning modules. Even if there are no cracks between the modules, without adequate communication a set of locally complete specialized reasoners may still fail to be globally complete. Furthermore, certain problems associated with controlling multiple reasoners must be solved to obtain an efficient hybrid system. One such problem is called "overlapping." In a number of hybrid systems particular predicate and function symbols are associated with only a single special-purpose reasoner that is invoked to deal with expressions formed with that symbol. However, if a symbol is associated with many reasoners, then deciding which reasoner to invoke can be problematic.

The Substitutional Framework for Hybrid Reasoning

Alan Frisch's presentation outlined the results of his research on a class of hybrid reasoners that he has identified and dubbed "substitutional reasoners" (Frisch 1989). The distinguishing characteristic of a substitutional reasoner is that it consists of a primary reasoner that invokes an embedded special-purpose reasoner only to perform certain prescribed inferences during unification. Substitutional reasoners are so named because the embedded reasoner affects only what substitutions are made by the primary reasoner. Information flows only from the embedded reasoner to the primary reasoner. Another way to look at the architecture is to view a substitutional reasoner as a primary reasoner that uses a unifier that is extended to perform certain built-in inferences.

The language that the primary reasoner uses, and that the extended unifier must operate on, contains *restricted variables*. Unlike ordinary variables that range over the entire domain, restricted variables have information associated with them that specifies a subset of the domain over which they are to range. The restrictions may be monadic or of a higher degree. A monadic restriction constrains the values that a single variable may take on. For example, the variable x may be restricted to take on a value from the set of mammals. Variables with monadic restrictions are a prominent feature of many of the sorted logic systems that recently have been receiving great attention. A higher-degree restriction specifies dependencies among the values that two or more variables can take. For example, the variables x and y may be constrained so that x takes on a value that is greater than the value of y.

The substitutional framework has been one of the most common and most successful architectures for hybrid reasoning, being used in a wide range of reasoners including logic programming systems (Allen *et al.* 1984; Allen and Miller 1988; Frisch *et al.* 1983), a general-purpose resolution system (Walther 1983; Walther 1987), a deductive database system (Reiter 1977; Reiter 1981), a parser for logic grammars (Frisch 1986b), and a knowledge retriever (Frisch 1986a). Though all of these systems use only monadic restrictions, recent work on constraint logic programming (Jaffar and Lassez 1987) provides a broad framework for generalizing Horn-clause logic programming to incorporate higher-degree restrictions in a substitutional manner. Frisch's research identifies all of these reasoners as a single class and investigates their common properties and the general principles underlying them.

Frisch presented his main results in terms of monadic restrictions and suggested that the results could be generalized to higher-degree restrictions. The results show how one can systematically transform a unification-based non-hybrid reasoning system and its proof of completeness into a substitutional hybrid system and its proof of completeness.

This presentation concluded by mentioning a number of research projects within the substitutional framework that Frisch and his students are investigating, including a couple of reasoners that use higher-degree restrictions.

What is Hybrid Reasoning and Who Cares?

The first issue addressed in this discussion session moderated by Richard Fikes was whether every representation system is a hybrid one, since one could perversely view almost any system as being composed of specialized reasoners. For instance, in a natural deduction system one could view each inference rule as a separate reasoner. This view is unnatural since a specialized rea-

soner should be "reasonably complete'' in some sense; however it is clearly impossible to define this concept precisely and thus the boundary between hybrid and non-hybrid systems must remain somewhat arbitrary. In any case such a definition would not be profitable in any practical sense.

It was agreed that, unlike nonmonotonic reasoning systems, hybrid reasoning systems cannot be characterized by their input/output behavior; the notion of hybrid reasoning has to do with the organization of the system, not its functionality. In particular it was proposed that one might regard a system as hybrid if one can associate particular colors with particular representation schemes and specialized inferences, and thereby assign colors to every step in a derivation. One good reason for associating different colors with different components is that they have different proof-theoretic properties, such as completeness and decidability; another reason might be because the representation systems are of differing expressiveness.

It was pointed out that another way of classifying hybrid systems is according to whether the different components arise primarily because of efficiency considerations (e.g., Stickel's theory resolution) or primarily because of presenting a particular conceptual framework to the user (e.g., systems that separate definitions and assertions). An ultimate goal for a theory of hybrid reasoning might be to enable the development of systems that are "plug compatible''; Frisch's substitutional framework might be seen as working towards this end. However there was serious doubt that one could ever develop a single framework with total generality.

It was agreed that an important topic for future research, and possibly for a future workshop, is the definition of appropriate protocols for communication between cooperating inference systems.

Expressiveness of Many-Sorted Logics

Tony Cohn's presentation briefly reviewed many-sorted logic and surveyed the dimensions along which many-sorted logics vary.[1] He gave three main reasons for being interested in many-sorted logic: to increase efficiency of deduction by reducing the deductive search space, to impose structure on a "flat'' logical axiomatization, and to perform well-sortedness checking as a simple but efficient integrity check on updates and queries to a knowledge base.

Cohn outlined the main dimensions along which many-sorted logics vary: the structure of the set of sorts, the language used to describe the sortal behavior of the non-logical symbols, the method by which sorts are associated with variables, and the precise definition of what constitutes a well-sorted formula. These dimensions define a space of many-sorted logical languages of varying expressiveness.

Cohn described his extremely expressive sorted logic, known as LLAMA (Cohn 1983; Cohn 1987). Its deductive system extends a resolution system with a number of mechanisms for reasoning with sort information. Because the logic has such a rich sort system and allows sort literals to appear in ordinary formulas, it is impossible to fit LLAMA entirely within the substitutional framework. Indeed, some of the reasoning methods fall within the theory resolution scheme (Stickel 1985a). One of the main benefits of allowing sort literals to appear in ordinary formulas is that sortal knowledge known explicitly can be represented and treated specially while still allowing the possibility of implicit (e.g. disjunctive) sortal information, which is handled by general methods (Cohn 1989a).

Theory Resolution

In this presentation Mark Stickel gave an overview of theory resolution, the details of which are presented in (Stickel 1985a). Theory resolution is a framework for building a theory (i.e., a set of axioms) into the resolution rule of inference by replacing resolution's simple test for contradictory literals with a more general test for contradiction with respect to a given theory. Whereas resolution operates on clauses containing contradictory literals, theory resolution operates on clauses containing literals that form a contradiction when taken together with the built-in theory. Theory resolution employs a special-purpose reasoner to detect such contradictions, thereby gaining efficiency over an ordinary resolution system that uses the axioms in the theory directly. For example, by building in the "<'' relation theory resolution can resolve the three clauses, $a < b$, $b < c$, and $c < a$, to obtain the empty clause in a single step. Stickel presented a number of examples of theory resolution, including examples of two special cases of theory resolution, total theory resolution and total narrow theory resolution.

Stickel cited the KRYPTON knowledge representation system (Brachman *et al.* 1983) as a successful application of theory resolution. KRYPTON divides knowledge into two classes, terminological knowledge and assertional knowledge. The definitions of the terminological component are built into the theory resolution reasoner that operates on the clauses of the assertional knowledge.

Stickel concluded by comparing theory resolution to sorted resolution. Though theory resolution could be used to build in taxonomic information, the resulting system would not be as efficient as a sorted resolution system. On the other hand, theory resolution is much more general since many other kinds of information can be built in. An operational difference between the two approaches is that the use of a theory in theory resolution can only increase the number of successful unifications, whereas the use of sorts in a sorted logic can only decrease the number of successful unifications. Stickel concluded that he wants both sorts and theory resolution in his system.

The RHET Reasoning System

In his presentation James Allen discussed the architecture of the RHET hybrid reasoning system and some of its subsystems (Allen and Miller 1988). RHET comprises a collection of separately-describable specialized reasoning systems organized around a sorted Horn-clause theorem prover. The specialized reasoners include a ground equality system, a temporal reasoner, a context mechanism for maintaining multiple interrelated knowledge bases, and a frame-like representation and reasoning system that is integrated with the sort system to obtain a hierarchy of terms. RHET has been used in a variety of applications, including systems for

natural language processing and plan reasoning. Developed at the University of Rochester by James Allen, Steve Feist, Stephanie Guez, Nat Martin, Brad Miller, and Michael McInerny, the system is a direct descendant of the HORNE system (Frisch *et al.* 1983; Allen *et al.* 1984).

Allen's presentation concentrated on three of RHET's subsystems: the sort reasoner (though he used the word "type''), the equality reasoner, and the context mechanism. Sort reasoning is integrated into the unifier according to the substitutional framework. One of the most interesting features of sorted unification in RHET is that multiple unifiers are avoided by allowing unsolved constraints to be attached to clauses. The equality reasoner uses a modified union-find algorithm to reason within the unifier about ground equalities. The context mechanism, which is used for representing beliefs and hypotheticals, maintains a tree of knowledge bases. In the context of a particular node, a reasoner has access to the information at that node and all its ancestors. Allen explained how the sort reasoner and equality reasoner can be used to encode and reason with typical frame-like representations in a straightforward manner.

When asked how all these systems were integrated, Allen simply replied, "It's hairy!'' However, he did elaborate a little. Each specialized reasoner owns particular predicates (e.g., the temporal reasoner owns the *Before* predicate) and the reasoner is invoked whenever a clause containing one of the predicates it owns is tested, added or deleted. A specialized reasoner may bind variables or may delay its action until certain variables become bound. Furthermore, a specialized reasoner must supply alternative bindings if appropriate when the system backtracks. Some reasoners are integrated into the system through hooks in the unifier. When two terms whose sorts have an intersection cannot be unified structurally, a special reasoner may be invoked to determine if the unification should be allowed anyway. One possible action a specialized reasoner may take is to tag terms with constraints, which are checked when they are further instantiated. Upon questioning, Allen acknowledged that this means that the solvability of a constraint may not be detected as early as possible, resulting in a larger search space, and that a RHET proof might be dependent upon the solvability of any remaining constraints.

Design Issues in Hybrid Reasoning

Chuck Rich, the moderator of this discussion session, opened by expressing his views on what constitute the main issues in the design of hybrid reasoners.[2] One design issue is how a system is to be divided into components. Two broad approaches to making such a decision are the top-down approach and the bottom-up approach. In the bottom-up approach one begins by identifying the reasoners that are to be included in the system and then attacks the problem of integrating them. The design of Nelson and Oppen's (1975) cooperating decision procedures is bottom-up; it starts with a set of known decision procedures for particular types of theories and then combines them into a decision procedure for the composite theory. In the top-down approach one starts with some distinction between types of knowledge and then seeks reasoning components for these. The design of KRYPTON is essentially top-down as it begins with the terminological/assertional distinction.

Another design issue involves how the components of a hybrid system are to communicate and cooperate with each other. Any hybrid system needs to have a way of representing the goals of the different subsystems and the returned results, and a public language (probably first order predicate calculus) in which to express these. There may also be languages private to particular components.

The final design issue raised is how a hybrid reasoner is to control the multiplicity of reasoners and representations that it incorporates. The problem is to decide how to divide up a task, and which component of the system should deal with which subtask. The control structures of the systems may be of various kinds; the way in which resources are allocated may be heuristic or algorithmic and may be based on either syntactic or semantic criteria. The communication protocol may be directed between subsystems or broadcast generally. Control may be centralized or distributed.

The group discussed what features they would like to see in what was called an "empty machine'' for hybrid reasoning, that is, a domain independent machine with which one could implement hybrid reasoners. Among the features listed were equality, dependencies (i.e., some explicit representation of the proof structure), and a teleological vocabulary for representing the goals, prerequisites and subgoals of the machine.

Vivid Reasoning and Tractable Reasoning

Ron Brachman and David Etherington presented their preliminary work with Alex Borgida and Henry Kautz (with help from Hector Levesque and Bart Selman) on constructing and using vivid knowledge representations for commonsense reasoning.[3] The basic idea derives from Johnson-Laird's work on mental models and from Levesque's suggestion in his 1985 Computers and Thought Lecture (Levesque 1986) that some kinds of commonsense reasoning may be best modeled by simple database style lookups in an appropriately organized representation. A vivid knowledge base (KB) bears a strong structural relationship to the world being modeled. In particular, a vivid KB has a set of symbols that stand in a one-to-one correspondence to objects of interest in the world, and for every relationship of interest in the world there is a directly analogous connection between the corresponding symbols in the KB. Thus a vivid KB is an analog of the domain; everything that is represented is explicitly represented. Such a KB contains no disjunctive information so reasoning by cases is not necessary.

Since a vivid KB is expressively weaker than a first-order KB, Brachman and Etherington suggested the use of a hybrid architecture containing both representations. (It is this architecture that we are referring to as "VIVID'' within this report.) Facts represented in the first-order KB can be "vivified'' and placed into the vivid KB where they can be used efficiently. However, the vivid form of a first-order fact may contain less information, making it useful to keep the first-order representation around. If the result of querying the vivid KB yields insufficient information, one can then query the first-order KB, where more-powerful but less-efficient reasoning methods are used.

Thus, the components of their proposed hybrid system operate on redundant representations of the same information that differ in grain size, whereas the components of each of the other hybrid systems that were presented at the workshop operate on representations of different information. The problem of how to integrate vivid reasoning and more powerful forms of logical reasoning is one of the many open questions yet to be addressed at this early stage of the research.

One of the main issues under investigation is how an ordinary first-order representation can be vivified automatically. The presentation focussed on two methods under investigation. The first method eliminates universal quantifiers by replacing universally quantified atomic formulas with their ground instances. For example, a sentence such as $\forall x\ Man(x) \rightarrow Mortal(x)$ can be eliminated provided *Mortal(a)* is added whenever a sentence of the form *Man(a)* is added. The second method of vivification eliminates disjunctions by forming generalizations in a subsumption hierarchy. For example, the sentence $Teacher(Joe) \vee Professor(Joe)$ can be replaced by *Instructor(Joe)*, given that *Instructor* subsumes *Teacher* and *Professor*. This replacement can be viewed as trading indefiniteness for vagueness.

Mating Nonmonotonic Inheritance with Theorem Proving

Rich Thomason presented his work on the Linkup Project, a collaborative effort with Chuck Cross, Jeff Horty and Dave Touretzky to study the logical foundations of inheritance. In particular, he presented a Gentzen-style proof theory that combines classical logic with a default inheritance reasoner that operates on network structures.[4] The key feature of Gentzen-style systems is that they operate on sequents, expressions that state that some formula is derivable from some set of formulas. The network structures and default reasoner are brought into the proof system by an inference rule for deriving "isa'' statements in the outer logic and an axiom for reflecting the results of the inheritance module into the outer logic.

inference rule:

$$\frac{\Gamma, x \text{ isa } p \vdash x \text{ isa } q}{\Gamma \vdash p\text{'s are } q\text{'s}}$$

(where Γ is a set of formulas that does not contain x)

axiom:

$\Gamma \vdash A$, provided the inheritance reasoner can derive A from Γ

Thomason stressed that the use of sequents in the Gentzen-style system was crucial to the successful integration of the inheritance reasoner into the proof system and suggested that this approach could be used to integrate other kinds of reasoners into a logical system.

Afterthoughts

With the benefit of hindsight, we shall compare and contrast the various hybrid reasoners presented at the workshop. In so doing we derive a partial taxonomy of hybrid reasoners that includes these systems. The taxonomy is but a first cut intended to provoke others to think about the relationships within the class of hybrid reasoners.

Leaving aside Patel-Schneider's overview, the remaining six presentations concerned particular systems or architectures for hybrid reasoning. Of the six systems or architectures only Brachman and Etherington's VIVID system contains components that use different representations of the same knowledge. In the other five, each component reasons with different knowledge. Let us consider these five systems more closely.

Cohn's LLAMA system and Allen's RHET system are complex systems, each consisting of numerous reasoning components integrated by a variety of methods. Frisch's substitutional framework and Stickel's theory resolution are architectures for integrating specialized reasoners into a resolution system. (It appears that both can be generalized beyond resolution, a point that is addressed later.) Thomason's hybrid system integrates its two reasoners via a single mechanism, which we shall assume, as he suggested, is a general technique. Therefore we shall consider his presentation to be concerned with an architecture for hybrid reasoning, henceforth called the sequent architecture.

All five of these systems or architectures have separate reasoning modules with distinct representations and all integrate the modules by a method we shall call rule embedding. As part of its execution an embedding rule invokes a subsidiary reasoner, called the embedded reasoner. After completing its assigned task the embedded reasoner passes its results back to the embedding inference rule. The embedded reasoner does nothing until invoked by the embedding rule and each time it is invoked it operates independently of all previous invocations; that is, no state is maintained from invocation to invocation. Rule-embedded inference is currently the most prevalent and best understood form of hybrid reasoning. However, other architectures currently exist, VIVID being one, and quite possibly a large variety of others are yet to be developed.

One way to categorize rule embedded systems is according to where in the embedding system the embedding takes place. Let us now illustrate this method of categorizing rule-embedded reasoners by comparing theory resolution and the substitutional framework. The comparison is particularly interesting because both are architectures for embedding special-purpose reasoners into the resolution rule of inference.

The essence of resolution comprises two fundamental ideas; one forms the foundation of theory resolution and the other forms the foundation of substitutional reasoning. The first idea is that at the ground (variable-free) level resolution looks for local evidence of unsatisfiability. The local evidence comes in the form of a pair of clauses, one containing a literal and the other containing its complement. Taken together the two literals are unsatisfiable and, indeed, a set of ground literals is unsatisfiable only if it contains a pair of complements. Theory resolution generalizes this idea; instead of merely looking for a pair of literals that are unsatisfiable by themselves, it looks for a set of literals that are unsatisfiable when taken in conjunction with a given background theory. Thus, the theory resolution rule of inference applies to a set of clauses containing a set of literals that, together with the background theory, are unsatisfiable.

The second fundamental idea in resolution is that a clause with variables is treated as schematic for the set of all its ground instances. In resolving two clauses with variables,

resolution behaves as if it were resolving all of the ground instances of the two clauses together. This simulation of ground resolution is precisely what unification achieves. The result is that every derivation involving clauses with variables is schematic for one or more ground derivations. The schematic derivation is said to lift the ground derivations. The substitutional framework extends this paradigm for handling variables by allowing constraints to be placed on expressions to restrict the set of ground instances for which they are schematic. Whether or not a particular ground expression satisfies the constraints depends on a background theory. This is handled by building into unification certain constraint satisfaction procedures that invoke the embedded reasoner to ascertain certain logical consequences of the background theory.

Thus the substitutional framework extends resolution by replacing the simple notion of an instance of an expression with one that takes a background theory into account, whereas theory resolution replaces the simple notion of unsatisfiability with one that takes a background theory into account. The two essential components of resolution are orthogonal, and hence so are the two proposed extensions. The two extensions could be combined in a single system and a theoretical understanding of the resulting system could be obtained straightforwardly from our present theoretical understanding of each.

The basic ideas embodied in theory resolution and in the substitutional framework are applicable outside of the resolution setting. The substitutional framework could be used to extend any reasoner that treats variables schematically by using unification. Theory resolution could be used to extend any inference rule that operates by recognizing unsatisfiable formulas. Furthermore, one could imagine extending relations other than unsatisfiability—subsumption, for example—to be relative to a background theory.

Another distinction that can be made between rule-embedding systems involves the theory (i.e., knowledge base) that the embedded reasoner uses. The theory used by the embedded reasoner in a substitutional reasoner or in a theory resolution reasoner remains fixed throughout the course of an entire deduction. Consequently this theory may be compiled in some manner in order to speed up the embedded reasoner. The theory may not even exist explicitly within the embedded reasoner. The embedded reasoner may be some procedure whose operation is functionally equivalent to a procedure manipulating an explicit theory. A hybrid system that uses an embedded reasoner with an implicit theory is often said to have a built-in theory.

In the sequent architecture, unlike the theory resolution and substitutional architectures, the embedded reasoner does not have a fixed theory. Each time the embedded reasoner is invoked it is passed the theory that it is to use. Because the embedded reasoner may be using a different theory on every invocation, it is difficult to see how one could reap any benefits from compilation. However, as Thomason points out, the ability to use different theories on different invocations is important if the embedded reasoner is non-monotonic.

The distinction between embedded reasoners that use a fixed theory and those that use a varying theory is based on the amount of information that is passed to the embedded reasoner when it is invoked. In a similar way we can categorize rule-embedded systems according to the information that is passed back when an invocation of the embedded reasoner terminates. In all systems considered here the embedded reasoner at least indicates success or failure. In those systems that we call "strict" no other information is returned that affects the operation of the embedding system. The distinction between the strict systems and the non-strict systems can be explained best by examining an application of a non-strict inference rule, partial theory resolution.

Suppose that partial theory resolution uses a theory that encodes the information that the set of persons divides into the mutually-exclusive subsets of males and females. Then theory resolution can perform the following deduction:

$$\neg Female(kim) \vee A \qquad (1)$$
$$Person(kim) \vee B \qquad (2)$$
$$\overline{Male(kim) \vee A \vee B} \qquad (3)$$

The justification for this inference is that if Kim is not a male then $\neg$ *Female(kim)* and *Person(kim)* are contradictory and theory resolution could derive $A \vee B$. Hence, either Kim is a male or $A \vee B$ is true.

This kind of inference is non-strict because the embedded reasoner provides a condition—namely that $\neg$ *Male(kim)* is needed in order to obtain a contradiction—to the embedding reasoner. The condition, which Stickel calls a residue, is used in forming the resolvent, and thus formulas produced by the embedded reasoner become part of the derivation produced by the embedding rule. Thus in partial theory resolution, unlike ordinary resolution, a resolvent may contain a literal that is not an instance of any literal occurring in its parents. Total theory resolution, on the other hand, does not use residues, and therefore does not introduce new literals; it is a strictly-embedded inference rule.

The distinction between partial theory resolution, which is non-strict, and total theory resolution, which is strict, is not as big as it may first seem because total theory resolution can do anything partial theory resolution can do. In the foregoing example, for instance, total theory resolution could derive (3) from (1), (2), and the tautology *Male(x)* $\vee \neg$ *Male(x)*.

Substitutional reasoners, by definition, use strict rule embedding. The special predicates that the embedded reasoner uses are not manipulated by the embedding reasoner and do not enter the representation used by the embedding reasoner except as annotations on variables. The reasoner embedded in a substitutional system can return large amounts of information. For example, in RHET and in constraint logic programming, complex constraints on multiple variables can be returned. Nonetheless, these are examples of strict embedding because the constraints do not affect the operation of the embedding reasoner; the embedding reasoner simply stores the constraints with the associated formulas and passes them back to the embedded reasoner on future invocations.

As Frisch's results show, the consequence of using strict rule embedding in the substitutional framework is that a certain class of theories cannot be used by the embedded reasoner without sacrificing completeness. Cohn's sorted logic, LLAMA, reasons about sorts during unification but, because the logic is sufficiently rich, the system's architecture must go

- Hybrid reasoning systems
  - Non rule-embedded
    example: VIVID
  - Rule-embedded
    - Instantiation-based
      examples: LLAMA's unifier (fixed theory, non-strict)
      RHET's sort and constraint system (fixed theory, strict)
      Substitutional framework (fixed theory, strict)
    - Contradiction-based
      examples: Partial theory resolution (fixed theory, non-strict)
      Total theory resolution (fixed theory, strict)
    - Other rule embedded systems
      example: Sequent architecture (varying theory, strict)

Figure 1: A partial taxonomy of hybrid reasoners

beyond the substitutional framework in order to obtain completeness. Consequently, the LLAMA design has sort reasoning non-strictly embedded within unification. For example, given the same information as in the theory resolution example—that the set of persons divides into the mutually-exclusive sets of males and females—the individual symbol *kim* of sort *Person* and the variable *x* of sort *Male* can be unified provided that Kim is a male. Thus the unifier gives back the literal *Male(kim)*, and the formula inferred by the embedding rule is of the form $Male(kim) \rightarrow \psi$. This returned literal, which Cohn calls a prosthetic literal, is much like a residue in that it can appear in a resolvent even though it appears in neither parent. Thus some sort information enters the representation used by the embedding reasoner, something that does not happen in a substitutional reasoner. Once these prosthetic literals are in the formulas used by the embedding reasoner, what is known about sorts must be used to make further inferences. LLAMA does this with some inference rules that can be viewed as special cases of theory resolution. Further discussion of prosthetic literals can be found in (Cohn 1989a).

Putting this all together yields the partial taxonomy of hybrid reasoners displayed in Figure 1. The six systems or architectures presented in the talks are shown as examples of the categories in the taxonomy.

Conclusions

This workshop was organized as a step to help discover the principles underlying hybrid reasoning. As a result of this workshop and subsequent discussions we have come to believe that the class of hybrid reasoners is far too diverse for there to be a grand theory of them all. Rather, we will see a set of architectures developed, each with its own theory. Particular systems will be built that are instances of a particular architecture or, like RHET and LLAMA, combine elements two or more architectures. Additional work is needed to better understand existing architectures, and to develop new, more diverse architectures. Developers of different architectures will need to communicate to share solutions to common problems and to work toward an understanding of the relationships and tradeoffs among the various architectures.

Acknowledgements

We thank the AAAI for their financial support of this workshop and Claudia Mazzetti and the AAAI staff for their assistance with arranging the workshop. Joseph Katz, the organizer of the entire AAAI-88 workshop program, was also instrumental in this regard. While writing this report the first author was partially funded by NASA under grant number NAG 1-613 and spent some of the time on leave at IBM T. J. Watson Research Center, and the second author was partially funded by the Science and Engineering Research Council. We thank Fran Evelyn for her expert editorial assistance and Peter Patel-Schneider for his comments.

Notes

1. Cohn has published a more detailed survey along these lines (Cohn 1989b).

2. Many of the issues that Rich raised are discussed at greater length in a paper that he co-authored (Brotsky and Rich 1985).

3. Subsequent work on vivid representations and reasoning has been reported by Borgida and Etherington (1989) and by Etherington, Borgida, Brachman and Kautz (1989).

4. Thomason and Aronis (1989) have subsequently reported their results on this research.

About the Authors

Alan M. Frisch is an assistant professor in the Department of Computer Science and the Beckman Institute for Advanced Science and Technology at the University of Illinois (405 N. Mathews Ave., Urbana, IL 61801). He was awarded a Ph.D. in computer science by the University of Rochester in 1986 and has taught artificial intelligence at Rochester and the University of Sussex. His dissertation work on knowledge retrieval as specialized inference led him to other investigations of specialized deduction and hybrid reasoning, and resulted in his development of the substitutional framework for hybrid reasoning.

Tony Cohn is a Senior Lecturer in the Division of AI at the University of Leeds (Leeds LS2 9JT, England) having previously been at the University of Warwick. He gained BSc and Phd degrees from the University of Essex. Presently, he is vicechairman of AISB and treasurer of the European Coordinating Committee on AI (ECCAI). His research interests center on knowledge representation, the control of reasoning, and naive and qualitative physics. A special research interest has been computational logic and many sorted logic in particular.