Cooperating Reasoning Processes: More than just the Sum of their Parts *

Alan Bundy
School of Informatics
University of Edinburgh
A.Bundy@ed.ac.uk

Abstract

Using the achievements of my research group over the last 30+ years, I provide evidence to support the following hypothesis:

> *By complementing each other, cooperating reasoning process can achieve much more than they could if they only acted individually.*

Most of the work of my group has been on processes for mathematical reasoning and its applications, e.g. to formal methods. The reasoning processes we have studied include:

Proof Search: by meta-level inference, proof planning, abstraction, analogy, symmetry, and reasoning with diagrams.

Representation Discovery, Formation and Evolution: by analysing, diagnosing and repairing failed proof and planning attempts, forming and repairing new concepts and conjectures, and forming logical representations of informally stated problems.

Other: learning of new proof methods from example proofs, finding counter-examples, reasoning under uncertainty, the presentation of and interaction with proofs, the automation of informal argument.

In particular, we have studied how these different kinds of process can complement each other, and cooperate to achieve complex goals.

We have applied this work to the following areas: proof by mathematical induction and co-induction; analysis; equation solving, mechanics problems; the building of ecological models; the synthesis, verification, transformation and editing of both hardware and software, including logic, functional and imperative programs, security protocols and process algebras; the configuration of hardware; game playing and cognitive modelling.

1 Introduction

"Many hands make light work" (John Heywood)

Much research in Artificial Intelligence consists of inventing or developing a new technique: analysing and establishing its properties, implementing and testing it, comparing it with rival techniques for the same task, etc. Other work consists of combining several techniques into a complex system that solves problems, models natural systems, etc. It is commonly observed that complementary techniques can assist each other, extending their range of automation and/or effectiveness. For instance, this was the theme of a previous IJCAI paper of mine [Bundy, 1985]. It is not just that different techniques are needed to tackle different aspects of a complex task; one technique can also assist another by addressing its shortcomings. In the current paper we examine this observation in more detail, giving examples of how techniques can complement each other: achieving more than the sum of their parts. The examples are drawn from the work of my research group, over more than 30 years, aimed at automating mathematical reasoning.

This same point could be illustrated by many pieces of work. However, given the context of this paper and the space limitations of the IJCAI-07 proceedings, I have limited myself to work in which I played some role, albeit a small one

*I would like to thank the many colleagues with whom I have worked over the last 30+ years on the research reported in this paper: MECHO/PRESS (Lawrence Byrd, George Luger, Chris Mellish, Rob Milne, Richard O'Keefe, Martha Palmer (née Stone), Leon Sterling, Bernard Silver and Bob Welham) ECO (Robert Muetzelfeldt, Mandy Haggith, Dave Robertson and Mike Uschold), *Proof Planning* (Dave Barker-Plummer, David Basin, Richard Boulton, James Brotherston, Francisco Cantu, Claudio Castellini, Ewen Denney, Louise Dennis, Lucas Dixon, Jacques Fleuriot, Jason Gallagher, Jeremy Gow, Ian Green, Jane Hesketh, Christian Horn, Andrew Ireland, Predrag Janičić, Helen Lowe, Ina Kraan, Ewen Maclean, Pete Madden, Rául Monroy, Julian Richardson, Alan Smaill, Andrew Stevens, Frank van Harmelen, Lincoln Wallen and Geraint Wiggins), ISANEWT Hazel Duncan and Amos Storkey, TM/HR (Alison Pease, Simon Colton, Alan Smaill, Graham Steel, John Lee and Toby Walsh) and ORS (Fiona McNeill, Marco Schorlemmer and Chris Walton). I'm grateful for feedback on an earlier draft from David Basin, James Brotherston, Claudio Castellini, Simon Colton, Priya Gopalan, Jane Hesketh, Andrew Ireland, Predrag Janičić, Erica Melis, Chris Mellish, Rául Monroy, Jörg Siekmann and Sean Wilson. The research reported in this paper was supported by numerous grants and studentships, the most recent of which is EPSRC GR/S01771.

in some cases. My apologies to those people whose work I have not surveyed; profuse apologies to those I have not even cited.

2 Object-level and Meta-level Inference

"Not all who wander are lost." (John Ronald Reuel Tolkien)

Much of our work has been on automatic inference, especially proving mathematical theorems. Most work in this area builds on mathematical logic, which is used to represent the conjectures to be proved, the axioms of the theories in which they are to be proved, and the rules of inference with which they are to be proved. These logical formulae provide a *search space* of potential proof steps. Typically, proof search is conducted backwards, starting with the original conjecture as the main goal and applying the rules of inference backwards to exchange each goal for a set of subgoals. Navigating this search space is *the* major problem in automated theorem proving. It is called the *combinatorial explosion*, because the number of subgoals increases super-exponentially with the length of the proof, rapidly swamping the computer's memory for all but trivial problems.

Human mathematicians succeed in navigating these huge search spaces by stepping back from the low-level details and thinking at a higher-level about the nature of the problem and the suitability of their tools for solving such problems. We have been trying to emulate this high-level thinking in our automated provers. In particular, we have implemented proof search as simultaneous and cooperating inference at two levels: the object-level, in which the original conjecture, theory and rules are expressed, and the meta-level, in which the proof methods are specified and the conjectures analysed. In *meta-level inference*, the conjecture is analysed, a suitable proof method identified and applied, progress is measured and the process recurses. In this way, inference at the meta-level directs inference at the object-level, avoiding unproductive parts of the search space and making proof search computationally tractable.

We have applied this two-level approach many times, but most notably in the domains of equation solving using the PRESS[1] system [Bundy and Welham, 1981; Sterling *et al.*, 1989] and inductive theorem proving using *Clam*/*Oyster*[2] [Bundy *et al.*, 1991; Bundy, 2001]. The latest realisation of this idea is *proof planning* , [Bundy, 1988; 1991], in which proof methods are specified as STRIPS-like plan operators and plan formation is used to custom-build a proof plan for each conjecture.

To illustrate these ideas I will describe *rippling*, a proof method for reducing the difference between a goal and one or more givens [Bundy *et al.*, 2005a]. It frequently occurs in mathematics that the goal to be proved is syntactically similar to a "given", i.e. a hypothesis, assumption or axiom. This occurs, most notably, in inductive proof, where the induction conclusion and induction hypothesis have strong similarities by construction. Rippling was originally developed for such inductive proofs, but we have found widespread applications in other types of problem. Typically, we want to rewrite the goal in such a way as to separate the similarities and differences while preserving the similarities. This enables the given(s) to be used to prove all or part of the transformed goal. We can visualise this process by annotating the goal and the given(s) with their similarities and differences (see Figure[3] 1).

$$t <> (y <> z) = (t <> y) <> z$$

$$\vdash \quad \boxed{h :: \boxed{t}}^{\uparrow} <> (y <> z) = (\boxed{h :: \boxed{t}}^{\uparrow} <> y) <> z$$

$$\vdash \quad \boxed{h :: \boxed{t <> (y <> z)}}^{\uparrow} = \boxed{h :: \boxed{t <> y)}}^{\uparrow} <> z$$

$$\vdash \quad \boxed{h :: \boxed{t <> (y <> z)}}^{\uparrow} = \boxed{h :: \boxed{(t <> y) <> z}}^{\uparrow}$$

$$\vdash \quad \boxed{h = h \wedge \boxed{t <> (y <> z) = (t <> y) <> z}}^{\uparrow}$$

The example shows the step case of an inductive proof of the associativity of list append, where $<>$ is the infix notation for append and :: is the infix notation for cons. $\vdash$ separates the induction hypothesis (the given) from the induction conclusion (the goal). The formulae are annotated to show the differences and similarities between the given and the goal. The grey boxes indicate the parts of the goal which differ from the given. They are called *wave-fronts*. Each wave-front has one or more *wave-holes* indicating sub-terms of the wave-fronts which correspond to parts of the given. The parts of the goal outside the wave-fronts or inside the wave-holes, are called the *skeleton*. The skeleton always matches the induction hypothesis. The arrows indicate the direction of movement of the wave-fronts — in this case outwards through the goal until they completely surround the skeleton.

The rules are also annotated and annotation must also match when rules are applied. Such annotated rules are called *wave-rules*. The main wave-rule used in this example is:

$$\boxed{H :: \boxed{T}}^{\uparrow} <> L \Rightarrow \boxed{H :: \boxed{T <> L}}^{\uparrow}$$

which is taken from the step case of the recursive definition of $<>$.

Note how the grey boxes get bigger at each wave-rule application with more of the skeleton embedded within them, until they contain a complete instance of the given.

Figure 1: The Associativity of Append using Rippling

Rippling successfully guides proof search by ensuring that the skeleton gets larger until it matches the givens. At this point the givens can be used to prove the goal. It has been successfully applied in induction, summing series, analysis and a variety of other areas. More importantly, if and when

[1] Prolog Equation Solving System.

[2] And in later proof planners, such as λ*Clam* and ISAPLANNER

[3] Tip to reader: the mathematical details are optional and have all been put in figures separated from the main text.

it fails, the nature of the failure can be used to suggest a way to patch the proof attempt and to recover from the failure (see §3).

Meta-level inference provides least-commitment devices, such as meta-variables, which can be used to postpone search decisions. For instance, in inductive proof, choosing an appropriate induction rule is an infinite branching point in the search: there is an induction rule corresponding to each well-ordering of each recursive data-structure, such as natural numbers, lists, trees, sets, etc. The key to a successful induction is often to choose an induction rule that inserts wave-fronts into the induction conclusion for which there are matching wave-rules, so that rippling is successful. We can turn this requirement on its head by postponing the choice of induction rule, inserting higher-order meta-variables into the induction conclusion to stand for the unknown wave-front, using higher-order unification to instantiate these meta-variables during rippling, and retrospectively choosing an induction rule when the wave-fronts are fully instantiated. This significantly reduces the search problem by moving it from a place (the choice of induction rule) where the branching is infinite, to a place (rippling) where it is highly constrained. Because we are effectively developing the middle of the proof before the beginning or the end, we call this search moving technique *middle-out reasoning*, in contrast to top-down or bottom-up reasoning.

We have applied middle-out reasoning to the choice of induction rule in two PhD projects. Initially, Ina Kraan used it to *select* an appropriate induction rule from a pre-determined, finite set of rules, as part of a project to synthesise logic programs from their specifications [Kraan *et al.*, 1996]. In this project, Kraan also used middle-out reasoning to synthesise the logic program. More recently, Jeremy Gow extended middle-out reasoning to create and verify new induction rules, customised to the current conjecture [Gow, 2004; Bundy *et al.*, 2005b].

Proof planning is now a thriving research area, with many applications and extensions, for instance:

- The ΩMEGA proof planner [Siekmann *et al.*, 2006], which has been applied to a wide range of areas in mathematics;
- Applications to the combination and augmentation of decision procedures [Janičić and Bundy, 2002];
- Multi-agent proof planning [Benzmüller and Sorge, 2001];
- Reasoning about feature interaction in first-order temporal logic [Castellini and Smaill, 2005]
- Multi-strategy proof planning [Melis *et al.*, 2006].

I have also proposed it as the basis for a "Science of Reasoning", in which proof plans provide a multi-level understanding of the structure of proofs [Bundy, 1991]

Meta-level inference, and proof planning in particular, shows how two complementary inference processes can interact to reduce the amount of search and overcome the combinatorial explosion. Meta-level analysis matches object-level goals to the proof methods best suited to solve them, cutting out a lot of legal, but unproductive, object-level, rule applications. Middle-out reasoning can rearrange the search space, postponing difficult search decisions until they are made easier as a side-effect of other search decisions. The object-level inference then provides the goals that form the ammunition for the next stage of meta-level analysis.

3 Inference and Fault Diagnosis

"The best laid plans of mice and men gang aft aglay." (Rabbie Burns)

Like all plans, proof plans are not guaranteed to succeed. When they fail, recovery requires another kind of reasoning process: *fault diagnosis*. When standard object-level proof search fails, the usual response is for the proof search to back-up to some earlier choice point and retry. Proof planning enables a more productive use to be made of failure. The mismatch between the situation anticipated at the meta-level and the situation obtaining at the object-level provides an opportunity to analyse the failure and propose a patch. Human mathematicians also make a productive use of failure. See, for instance, [van der Waerden, 1971] for an example of repeated plan failure and patching in an, eventually successful, attempt to prove a challenging conjecture.

Consider, as another example, a failure of the rippling method outlined in §2. This might occur, for instance, because there is no rule available to ripple the wave-front out one more stage. From the meta-level inference that formed the rippling-based proof plan, we can extract a partial description of the form that the missing wave-rule should take. This is best illustrated by an example. See Figure 2.

We have implemented this kind of fault diagnosis and repair within our proof planning framework using *proof critics* [Ireland, 1992; Ireland and Bundy, 1996]. The original work explored ways to recover from different kinds of failure of rippling, suggesting, variously, new forms of induction, case splits, generalisations and intermediate lemmas. Later work by Ireland and his co-workers has extended this to discovering loop invariants in the verification of imperative programs [Ireland and Stark, 2001]. They have recently applied this work to industrial verification by extending and successfully evaluating Praxis' automated prover [Ireland *et al.*, 2006]. Deepak Kapur and M. Subramaniam have also developed similar methods for lemma discovery in induction [Kapur and Subramaniam, 1996]. Rául Monroy has used critics for the correction of false conjectures [Monroy *et al.*, 1994], which requires the additional reasoning process of *abduction*. More recently, Monroy has applied his techniques to higher-order faulty conjectures, leading to the automatic formation of new theories, such as monoids [Monroy, 2001]. Andreas Meier and Erica Melis have adapted their multi-strategy MULTI system to include failure analysis and the automatic proposal of "recommendations" to overcome them [Meier and Melis, 2005].

Fault diagnosis of an earlier failure suggests the most promising alternative proof attempt. It removes the need for blind backtracking, removing legal, but unproductive choices from the search space and helping to defeat the combinatorial explosion. Moreover, the proof patches often provide inter-

$$rev(rev(l)) = l$$

$$\vdash \quad rev(rev(\boxed{h :: t}^{\uparrow})) = \boxed{h :: t}^{\uparrow}$$

$$\vdash \quad \underbrace{rev(\boxed{rev(t) <> (h :: nil)}^{\uparrow})}_{\textbf{blocked}} = \boxed{h :: t}^{\uparrow}$$

rev is a list reversing function. The example is taken from a failed proof that reversing a list twice will give you the original list. Suppose that, initially, the only rule available is:

$$rev(\boxed{H :: T}^{\uparrow}) \Rightarrow \boxed{rev(T) <> (H :: nil)}^{\uparrow}$$

The ripple attempt will fail because no rule matches the left hand side of the goal, so the wave-front cannot be moved one stage further outwards. However, by analysing the blocked ripple, fault diagnosis can work out that the missing wave-rule should take the form:

$$rev(\boxed{X <> Y}^{\uparrow}) \Rightarrow \boxed{F(rev(X), X, Y)}^{\uparrow}$$

where F stands for some unknown function, represented by a higher-order meta-variable. If we allow the proof to continue using this missing wave-rule schema, then the proof will succeed, instantiating $F(u, v, w)$ to $rev(w) <> u$ in the process. We now discover the missing wave-rule to be:

$$rev(\boxed{X <> Y}^{\uparrow}) \Rightarrow \boxed{rev(Y) <> rev(X)}^{\uparrow}$$

which must be proved as a lemma to complete the proof. Note that this lemma is a distributive law of $<>$ over rev, and is an interesting theorem in its own right, rather than just an arbitrary hack needed just to get the original proof to go through.

Figure 2: Lemma Speculation Using a Rippling Failure

esting information. As the example in Figure 2 illustrates, lemma speculation often produces lemmas that are of mathematical interest in their own right and prove to be useful in future proofs. Generalisation, which is another way of patching failed inductive proofs, also often generalises the original conjecture in a mathematically interesting way.

In the other direction, the kind of fault diagnosis illustrated in Figure 2 is only possible because of the interaction of meta-level and object-level inference. It is the discrepancy between the meta-level expectation and the object-level reality that focuses the fault diagnosis to suggest a patch that will resolve this discrepancy and allow the original proof plan to be resumed. Thus, again, we see complementary reasoning processes focus their attention to solve a problem that none of them could manage alone — nor even acting simultaneously but without interaction.

4 Inference and Learning

"To this day, we continue to find new techniques and methods. There's always something new to learn." (Joseph Cocking)

We have seen the value of proof methods when used by meta-level inference or fault diagnosis to guide object-level proof search, but where do such proof methods come from? Mostly, they have come from detailed analysis of a family of related proofs, and reflection on the thinking that informs experienced mathematical reasoners. But this is a time-consuming, highly-skilled and error-prone process. The proof methods that my group developed for inductive proof, for instance, took more than a decade to develop, evaluate and refine. It is necessary to maintain constant vigilance against quick, *ad hoc* fixes that will not have general utility and explanatory power. It would be much better if the process of developing new proof methods could be automated.

My group has explored various machine learning techniques to automate the construction of proof methods. These start with successful object-level proofs whose meta-level structure they try to analyse and abstract, in order to formulate new proof methods. Our earliest attempt to do this was the application of explanation-based generalisation (EBG) to the equation-solving domain [Silver, 1985]. Silver's LEARNING PRESS system was able to learn (sometimes simplified) versions of all the proof methods that had been hand-coded into our PRESS equation solving system, just by applying EBG to examples of successfully solved equations. Later, Desimone did the same for early versions of our inductive proof methods [Desimone, 1989].

Recently, we have explored the use of data-mining techniques to extract new methods[4] from large corpuses of proofs [Duncan *et al.*, 2004]. Duncan's IsaNewT[5] system first identifies frequently occurring sequences of proof steps using variable-length Markov models (VLMM) [Ron *et al.*, 1996]. Then it combines these sequences using genetic algorithms to join them with branching, repetition and macros, using a language developed in [Jamnik *et al.*, 2002]. The resulting proof methods have been evaluated successfully by comparing proof success and timings on a large test corpus of proofs with and without the newly learnt tactics. Currently, the learnt methods consist of simple, generic combinations of object-level rules. They do not approach the sophistication nor targeting of many hand-crafted methods, such as rippling. Nor do they provide dramatic search reductions. However, this use of data-mining has made a promising start and has the potential for much more development.

We have introduced a new class of reasoning processes from machine learning. These complement meta-level inference by constructing new proof methods from example proofs, thus enabling meta-level inference to guide object-level proof search. The learning methods are, in turn, informed by previous success of object-level inference in pro-

[4]Technically, it is just *tactics* that are learnt, rather than methods, since, as discussed below, we are currently unable to learn the meta-language for specifying these tactics: methods being tactics together with their meta-level specifications.

[5]Is a New Tactic.

ducing proofs. Their analysis is informed by the patterns they identify in these proofs. Thus we have a virtuous triangle of reasoning processes, each process in the triangle using information from one of its neighbours to assist the other neighbour.

However, there is a missing process in this story: a learning process that can construct the meta-level language used to specify the proof methods. Examples of such a language are the concepts of wave-fronts, wave-holes and skeletons, introduced and illustrated in Figure 1. Silver's and Desimone's EBG-based systems assumed the prior existence of an appropriate meta-level language and used it in the analysis of the object-level proofs. Duncan's IsaNewT does not assume the existence of such a meta-language, so is, thereby, restricted to an impoverished language for expressing proof methods. It cannot use conditional branching, since the conditions would have to be expressed in the missing meta-language. It uses non-deterministic branching instead. Similarly, it cannot use until-loops, since again the exit condition would have to be expressed in the missing meta-language. It just uses repetition. It cannot combine the proof methods with proof planning or analyse failures with fault diagnosis, as the method specifications necessary to do this would need to be expressed in the missing meta-language. Instead, it is reduced to conducting exhaustive search at the method level, which reduces search, but not in as directed a way as proof planning is capable of.

Current AI learning mechanisms appear not to be capable of performing the learning task required here: constructing a new ontology. There are plenty of mechanisms for defining new concepts in terms of old ones, but that is not sufficient to form a new meta-language. The concept of wave-front, for instance, cannot be defined in terms of the functions and predicates in the object-language. Here's a hard challenge for the next generation of machine learning researchers. For a start on tackling this challenge, see §6.

5 Inference and Representation Formation

"Once you know the formula, it's a guiding light that will give you everything you want to know." (Mark Lallemand)

As discussed in §2, the automation of inference requires that the axioms, rules and conjectures of a theory be represented in the computer. Where do such representations come from? In most work in automated inference, either they are adopted/adapted from a textbook or from previous research, or they are the outcome of careful and highly-skilled handcrafting. However, developing an appropriate representation is one of the most important parts of problem solving. Applied mathematics, for instance, mainly consists of exploring and developing mathematical representations of physical environments and problems. So, in a thorough investigation of automatic reasoning, the formation of formal representations ought to be a central concern.

In the MECHO[6] Project ([Bundy *et al.*, 1979]) we addressed this issue by building a program for solving mechanics problems stated in English, with examples drawn from the English GCE A-Level, applied-mathematics papers, intended for 16-18 year old pre-university entrants. MECHO took mechanics problems stated in English, parsed them, constructed a semantic representation in the form of first-order logic assertions, extracted equations from this logical representation and then solved them. This process is illustrated in Figure 3.

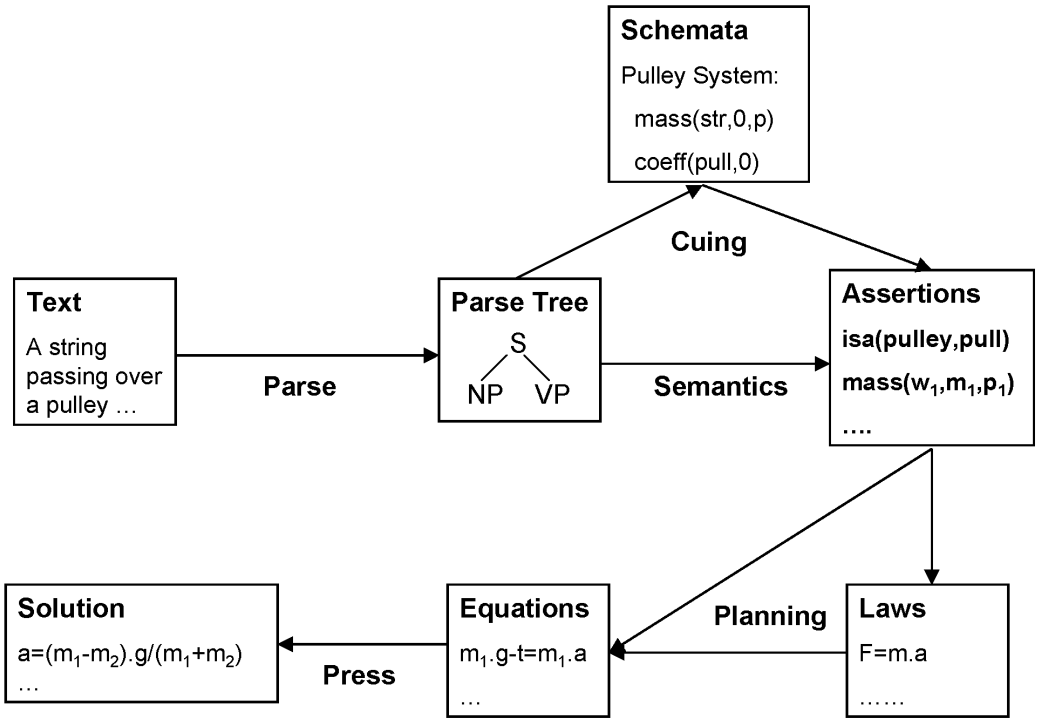
During this process, cooperation between representation formation and inference occurs at a number of levels and in several directions.

- The whole process of representation enables inference, in the form of equation solving, to solve the mechanics problem.
- A form of non-monotonic inference is needed to "flesh out" the explicit problem statement. Real-world objects must be idealised, and this can be done in a variety of ways. For instance, a ship might be idealised as a particle on a horizontal plane in a relative velocity problem, but in a specific gravity problem it will be idealised as a 3D shell floating on a liquid. The string in a pulley problem is usually idealised as inelastic and the pulley as frictionless. MECHO uses cues to recognise the problem type and then matches a problem-type schema to the extracted assertions, which then provides the implicit, unstated assertions.
- MECHO uses a form of plan formation[7] to extract a set of equations from the assertions. A physical law is identified that relates the goals to the givens, e.g., $F = m.a$. These equations may introduce intermediate variables, which must also be solved for, causing the planning process to recurse. Preconditions of the physical laws relate the variables to the assertions, enabling the laws to be instantiated to equations describing the problem situation.
- Equation solving is used to express the goal variable in terms of the givens. This inferential process is implemented using meta-level inference, as described in §2.
- Inference is also required during natural language processing, e.g., to resolve ambiguities, such as pronoun reference. The meta-level inference mechanisms used in MECHO were motivated by the problems of controlling inference both in problem-solving and semantic interpretation, and were successful in both areas. For instance, the inferential choices arising during semantic processing are limited by idealisation, especially by the framework imposed by problem-type schemata.

The ideas from the MECHO project were subsequently adapted to the ECO program, which built an ecological simulation program in response to a dialogue with a user [Robertson *et al.*, 1991]. ECO incorporated a further example of collaborative reasoning: a meta-level analysis of the user's requirements was used to propose two kinds of idealisation. ECO was able to suggest both idealisations of real world ecological objects, e.g., animals, plants, environments, etc., and idealisations of the processes by which they interacted, e.g.,

[6] Mechanics Oracle.

[7] We called it the "Marples Algorithm" in honour of David Marples, who was the first person we were aware of to describe this process in some teaching notes.



The input is a mechanics problem, posed in English, taken from an applied mathematics textbook. This is first parsed to identify its syntactic structure and then turned into a set of logical assertions by semantic analysis. These assertions must be "fleshed out" by cuing schemata to provide unstated, but implicit, default information. The givens and goals of the problem are identified and a planning process is used to instantiate general physical laws into a set of simultaneous equations from which the goals can be derived from the givens. The PRESS equation solver is then used to express the goals in terms of the givens.

Figure 3: Representations Used by the MECHO Program

logistic growth, predator-prey equation, competition equation, etc.

6 Inference, Fault Diagnosis and Ontology Repair

"I shall try to correct errors when shown to be errors, and I shall adopt new views so fast as they shall appear to be true views" (Abraham Lincoln)

MECHO and ECO formed representations with the aid of natural language processing tools and inference, but from a fixed *signature*, by which I mean the variables, functions and predicates, along with their types, that were used to form the logical formulae. However, as we saw in §4, it is sometimes necessary to invent new concepts and to create new functions, predicates, types, etc. with which to represent these new concepts.

6.1 Repairing Faulty Conjectures

Alison Pease and Simon Colton's TM[8] program, [Colton and Pease, 2005], implements some ideas of Lakatos [Lakatos, 1976] for repairing faulty mathematical theories. In particular, it takes nearly true conjectures and repairs them into theorems. TM achieves this by orchestrating: the OTTER theorem prover [McCune, 1990]; the MACE counter-example finder [McCune, 1994] and the HR[9] machine learning system [Colton *et al.*, 1999; Colton, 2002].

Given a false conjecture, first MACE is used to find both a set of counter-examples and a set of supporting examples. TM then implements Lakatos's methods of *strategic withdrawal* and *counter-example barring* to modify the original, false conjecture into new conjectures that it hopes are true. In strategic withdrawal, HR finds a concept that describes a maximal subset of supporting examples and the theorem is specialised to just that concept. In counter-example barring, HR finds a concept that covers all the counter-examples, and a

[8]Theorem Modifier.
[9]Hardy & Ramanujan.

minimal subset of supporting examples, then makes the negation of that concept a precondition of the revised theorem. These new conjectures are given to OTTER to verify. The TM system thus exhibits the interaction of three kinds of reasoning: model generation, concept formation and inference, to do something that neither of them could do alone: correcting faulty conjectures. An example is given in Figure 4.

TM was given the following faulty conjecture in Ring Theory:

$$\forall x, y.\ x^2 * y * x^2 = e$$

where e is the multiplicative identity element.

MACE found 7 supporting examples and 6 counter-examples to this conjecture. Given these two sets of examples, HR invented a concept that can be simplified to:

$$\forall z.\ z^2 = z + z$$

and used it for strategic withdrawal. OTTER was then able to prove the original conjecture for just those rings with this new property.

Figure 4: Correcting a Faulty Conjecture in Ring Theory

HR has also been used a component of a system for the automatic generation of classification theorems for algebra models up to isomorphism and isotopism, [Colton *et al.*, 2004; Sorge *et al.*, 2006]. This system employs an intricate interplay of machine learning, computer algebra, model generation, theorem proving and satisfiability solving methods, and has produced new results in pure mathematics. For instance, it generated an isotopic classification theorem for loops of size 6, which extended the previously known result that there are 22. This result was previously beyond the capabilities of automated reasoning techniques.

6.2 Repairing Faulty Signatures

However, note that HR does not extend the *signature* of the representation language. Its new concepts are defined in terms of a fixed set of functions, relations and types. In contrast, Fiona McNeill's ORS[10] modifies the underlying signature, e.g., by changing the arity and types of functions and predicates, and by merging or separating functions and predicates [Bundy *et al.*, 2006; McNeill, 2005]. ORS is designed for a virtual world of interacting software agents. It achieves its goals by forming and executing plans that combine the services of other agents.

However, these plans are fallible and may fail on execution. Plan formation can be viewed as an automated proof of an existential theorem, where the witness of the existentially quantified variable is a plan, and the proof shows that this plan is guaranteed to achieve its goal. This plan may, nevertheless, fail to execute, because the planning agent's world model is faulty. In particular, its theory of the circumstances under which other agents will perform certain services are expressed as preconditions of action rules. However, the other agents may have different versions of these action rules. The planning agent must diagnose the fault with its version of the rule; repair the rule; and replan with the amended rule. This repair may just consist of adding or deleting a precondition, or it may go deeper, requiring a change to the signature with which the preconditions are expressed. An example is given in Figure 5.

Suppose that the planning agent's plan contains an action to pay a hotel bill, represented as

(Pay PA Hotel $200)

where *PA* is the planning agent. This action fails on execution. Just prior to failure there is a short dialogue with the *Hotel* agent. Such inter-agent dialogues are the normal way to check those preconditions of whose truth value the service providing agent is unsure. The planning agent is expecting to be asked

(Money PA $200)

However, it is surprised to be asked

(Money PA $200 credit_card)

instead. This suggests that the *Hotel* agent has a ternary version of the *Money* predicate, rather than the planning agent's binary predicate. In order to communicate successfully with the *Hotel* agent, the planning agent must change the arity of its *Money* predicate accordingly and replan.

Figure 5: Repairing an Ontology Signature

ORS employs interactions between three kinds of reasoning process: inference in the form of plan formation; fault diagnosis of the failures of plan execution and any dialogue prior to this failure; and mechanisms for repairing faults in both the theory and the signature of its ontology. The fault diagnosis works by a simple decision tree, in which the notion of being asked a surprising question (or not) is a key diagnostic cue. It assumes a context in which there is a large measure of ontological agreement between interacting agents, but some critical differences that, uncorrected, will cause the agents' plans to fail. This context might arise, for instance, where the agents have tried to adhere to some ontological standard, but different versions and local customisation have created critical differences. It can help where the ontological differences only require localised repairs, but where such repairs must be done at runtime and without human intervention, for instance, where very large numbers of software agents are interacting over the internet.

An agent can use inference to form plans to achieve its goals, but these are only guaranteed to succeed when its world model is perfect. Since perfect world models are, in practice, unattainable, a successful agent must also be able to repair a faulty world model. This requires the interaction of inference with fault detection, diagnosis and repair. Fault detection requires inference to provide a failed plan. Fault diagnosis requires a fault to be detected and the failed plan for analysis, together with further inference about the state of the world and how it differs from the world model. Fault repair requires a diagnosis of the fault. Successful inference requires the re-

[10] Ontology Repair System.

paired world model. These reasoning processes thus constitute a virtuous circle, which incrementally improves the success of the agent in attaining its goals.

7 Conclusion

"Come together" (John Lennon & Paul McCartney)

In this paper we have been defending the hypothesis that:

By complementing each other, cooperating reasoning process can achieve much more than they could if they only acted individually.

We have presented evidence in favour of this hypothesis accumulated over more than 30 years within my research group[11]. In particular, we have seen how processes of inference, at both object and meta-level, planning, fault diagnosis, learning, counter-example finding, representation formation and ontology repair can interact. It is not just that different processes deal with different parts of the task, but that the weaknesses in one process are complemented by strengths in another, leading to the more efficient and effective running of each process and of their combination. Search that would overwhelm an inference process running on its own, is effectively guided when two inference processes run in tandem. Faults that would otherwise remain undetected, are made manifest by the mismatching expectations of two inference processes. Formal representations required to perform inference are formed with the aid of inference. New proof methods needed to guide inference are created by learning from previously successful examples of inference. Faulty world models are detected, diagnosed and repaired.

AI has become fragmented over the last 30 years. The development of robust, scalable AI systems to emulate the many facets of intelligence has proven to be much more difficult than envisaged by the AI pioneers of the 50s and 60s. To make the task tractable, we divided ourselves into smaller, specialised communities, organised our own conferences and journals, and developed a separate series of techniques for each of the different facets of intelligence. We have made enormous progress. Our techniques are theoretically well understood, robust and applicable to important practical problems. But, taken individually, they each suffer from severe limitations of range and autonomy.

One lesson from this paper is that perhaps the time has come to reintegrate Artificial Intelligence; to discover how the limitations of one kind of technique can be overcome by complementing it with another; to realise that together we have already achieved much more than we had realised when we were apart.

References

[Benzmüller and Sorge, 2001] Christoph Benzmüller and Volker Sorge. OANTS – an open approach at combining interactive and automated theorem proving. In M. Kerber and M. Kohlhase, editors, *8th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (Calculemus-2000)*, pages 81–97, St. Andrews, UK, 6–7 August 2001. AK Peters, New York, NY, USA.

[Bundy and Welham, 1981] A. Bundy and B. Welham. Using meta-level inference for selective application of multiple rewrite rules in algebraic manipulation. *Artificial Intelligence*, 16(2):189–212, 1981. Also available from Edinburgh as DAI Research Paper 121.

[Bundy *et al.*, 1979] A. Bundy, L. Byrd, G. Luger, C. Mellish, R. Milne, and M. Palmer. Solving mechanics problems using meta-level inference. In B. G. Buchanan, editor, *Proceedings of IJCAI-79*, pages 1017–1027. International Joint Conference on Artificial Intelligence, 1979. Reprinted in 'Expert Systems in the microelectronic age' ed. Michie, D., pp. 50-64, Edinburgh University Press, 1979. Also available from Edinburgh as DAI Research Paper No. 112.

[Bundy *et al.*, 1991] A. Bundy, F. van Harmelen, J. Hesketh, and A. Smaill. Experiments with proof plans for induction. *Journal of Automated Reasoning*, 7:303–324, 1991. Earlier version available from Edinburgh as DAI Research Paper No 413.

[Bundy *et al.*, 2005a] A. Bundy, D. Basin, D. Hutter, and A. Ireland. *Rippling: Meta-level Guidance for Mathematical Reasoning*, volume 56 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2005.

[Bundy *et al.*, 2005b] A. Bundy, J. Gow, J. Fleuriot, and L. Dixon. Constructing induction rules for deductive synthesis proofs. In S. Allen, J. Crossley, K.K. Lau, and I Poernomo, editors, *Proceedings of the ETAPS-05 Workshop on Constructive Logic for Automated Software Engineering (CLASE-05), Edinburgh*, pages 4–18. LFCS University of Edinburgh, 2005. Invited talk.

[Bundy *et al.*, 2006] A. Bundy, F. McNeill, and C. Walton. On repairing reasoning reversals via representational refinements. In *Proceedings of the 19th International FLAIRS Conference*, pages 3–12. AAAI Press, 2006. Invited talk.

[Bundy, 1985] Alan Bundy. Discovery and reasoning in mathematics. In A. Joshi, editor, *Proceedings of IJCAI-85*, pages 1221–1230. International Joint Conference on Artificial Intelligence, 1985. Also available from Edinburgh as DAI Research Paper No. 266.

[Bundy, 1988] A. Bundy. The use of explicit plans to guide inductive proofs. In R. Lusk and R. Overbeek, editors, *9th International Conference on Automated Deduction*, pages 111–120. Springer-Verlag, 1988. Longer version available from Edinburgh as DAI Research Paper No. 349.

[Bundy, 1991] Alan Bundy. A science of reasoning. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, pages 178–198. MIT Press, 1991. Also available from Edinburgh as DAI Research Paper 445.

[11] Lots of other groups have accumulated similar evidence, but surveying that is beyond the scope of the current paper (see §1).

[Bundy, 2001] Alan Bundy. The automation of proof by mathematical induction. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning, Volume 1*. Elsevier, 2001.

[Castellini and Smaill, 2005] Claudio Castellini and Alan Smaill. Proof planning for first-order temporal logic. In Robert Nieuwenhuis, editor, *Automated Deduction – CADE-20*, volume 3632 of *Lecture Notes in Computer Science*, pages 235–249. Springer, 2005.

[Colton and Pease, 2005] S. Colton and A. Pease. The TM system for repairing non-theorems. *Electronic Notes in Theoretical Computer Science*, 125(3), 2005. Elsevier.

[Colton *et al.*, 1999] S Colton, A Bundy, and T Walsh. HR: Automatic concept formation in pure mathematics. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence, Stockholm, Sweden*, pages 786–791, 1999.

[Colton *et al.*, 2004] S. Colton, A. Meier, V. Sorge, and R. McCasland. Automatic generation of classification theorems for finite algebras". In *Proceedings of the International Joint Conference on Automated Reasoning*, 2004.

[Colton, 2002] S Colton. *Automated Theory Formation in Pure Mathematics*. Springer-Verlag, 2002.

[Desimone, 1989] R. V. Desimone. Explanation-Based Learning of Proof Plans. In Y. Kodratoff and A. Hutchinson, editors, *Machine and Human Learning*. Kogan Page, 1989. Also available as DAI Research Paper 304. Previous version in proceedings of EWSL-86.

[Duncan *et al.*, 2004] H. Duncan, A. Bundy, J. Levine, A. Storkey, and M. Pollet. The use of data-mining for the automatic formation of tactics. In *Workshop on Computer-Supported Mathematical Theory Development*. IJCAR-04, 2004.

[Gow, 2004] J. Gow. *The Dynamic Creation of Induction Rules Using Proof Planning*. PhD thesis, School of Informatics, University of Edinburgh, 2004.

[Ireland and Bundy, 1996] A. Ireland and A. Bundy. Productive use of failure in inductive proof. *Journal of Automated Reasoning*, 16(1–2):79–111, 1996. Also available from Edinburgh as DAI Research Paper No 716.

[Ireland and Stark, 2001] A. Ireland and J. Stark. Proof planning for strategy development. *Annals of Mathematics and Artificial Intelligence*, 29(1-4):65–97, February 2001. An earlier version is available as Research Memo RM/00/3, Dept. of Computing and Electrical Engineering, Heriot-Watt University.

[Ireland *et al.*, 2006] A. Ireland, B.J. Ellis, A. Cook, R. Chapman, and J Barnes. An integrated approach to high integrity software verification. *Journal of Automated Reasoning: Special Issue on Empirically Successful Automated Reasoning*, 2006. To appear.

[Ireland, 1992] A. Ireland. The Use of Planning Critics in Mechanizing Inductive Proofs. In A. Voronkov, editor, *International Conference on Logic Programming and Automated Reasoning – LPAR 92, St. Petersburg*, Lecture Notes in Artificial Intelligence No. 624, pages 178–189. Springer-Verlag, 1992. Also available from Edinburgh as DAI Research Paper 592.

[Jamnik *et al.*, 2002] M. Jamnik, M. Kerber, and M. Pollet. Automatic learning in proof planning. In F. van Harmelen, editor, *Proceedings of 15th ECAI*, pages 282–286. European Conference on Artificial Intelligence, IOS Press, 2002.

[Janičić and Bundy, 2002] Predrag Janičić and Alan Bundy. A general setting for the flexible combining and augmenting decision procedures. *Journal of Automated Reasoning*, 28(3):257–305, 2002.

[Kapur and Subramaniam, 1996] D. Kapur and M Subramaniam. Lemma discovery in automating induction. In M. A. McRobbie and J. K. Slaney, editors, *13th International Conference on Automated Deduction (CADE-13)*, pages 538–552. CADE, Springer, 1996.

[Kraan *et al.*, 1996] I. Kraan, D. Basin, and A. Bundy. Middle-out reasoning for synthesis and induction. *Journal of Automated Reasoning*, 16(1–2):113–145, 1996. Also available from Edinburgh as DAI Research Paper 729.

[Lakatos, 1976] I. Lakatos. *Proofs and Refutations: The Logic of Mathematical Discovery*. Cambridge University Press, 1976.

[McCune, 1990] W. McCune. The Otter user's guide. Technical Report ANL/90/9, Argonne National Laboratory, 1990.

[McCune, 1994] W. McCune. A Davis-Putnam program and its application to finite first-order model search. Technical Report ANL/MCS-TM-194, Argonne National Labs, 1994.

[McNeill, 2005] Fiona McNeill. *Dynamic Ontology Refinement*. PhD thesis, School of Informatics, University of Edinburgh, 2005.

[Meier and Melis, 2005] A. Meier and E. Melis. Failure reasoning in multiple-strategy proof planning. *Electronic Notes in Theoretical Computer Science*, 125:67–90, 2005.

[Melis *et al.*, 2006] E. Melis, A. Meier, and J. Siekmann. Proof planning with multiple strategies. *Artificial Intelligence*, 2006. To appear.

[Monroy *et al.*, 1994] R. Monroy, A. Bundy, and A. Ireland. Proof Plans for the Correction of False Conjectures. In F. Pfenning, editor, *5th International Conference on Logic Programming and Automated Reasoning, LPAR'94*, Lecture Notes in Artificial Intelligence, v. 822, pages 54–68, Kiev, Ukraine, 1994. Springer-Verlag. Also available from Edinburgh as DAI Research Paper 681.

[Monroy, 2001] Raúl Monroy. Concept formation via proof planning failure. In R. Nieuwenhuis and A. Voronkov, editors, *8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 2250 of *Lecture Notes in Artificial Intelligence*, pages 718–731. Springer-Verlag, 2001.

[Robertson *et al.*, 1991] D. Robertson, Alan Bundy, R. Muetzelfeldt, M. Haggith, and M Uschold. *Eco-Logic:*

Logic-Based Approaches to Ecological Modelling. MIT Press, 1991.

[Ron *et al.*, 1996] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25, 1996.

[Siekmann *et al.*, 2006] J. Siekmann, C. Benzmüller, and S. Autexier. Computer supported mathematics with ΩMEGA. *Journal of Applied Logic, Special Issue on Mathematics Assistance Systems*, 2006. To appear.

[Silver, 1985] B. Silver. *Meta-level inference: Representing and Learning Control Information in Artificial Intelligence*. North Holland, 1985. Revised version of the author's PhD thesis, Department of Artificial Intelligence, U. of Edinburgh, 1984.

[Sorge *et al.*, 2006] V. Sorge, A. Meier, R. McCasland, and S. Colton. The automatic construction of isotopy invariants. In *The proceedings of the International Joint Conference on Automated Reasoning.*, 2006.

[Sterling *et al.*, 1989] L. Sterling, Alan Bundy, L. Byrd, R. O'Keefe, and B. Silver. Solving symbolic equations with PRESS. *J. Symbolic Computation*, 7:71–84, 1989. Also available from Edinburgh as DAI Research Paper 171.

[van der Waerden, 1971] B.L. van der Waerden. *How the Proof of Baudet's Conjecture Was Found*, pages 251–260. Academic Press, 1971.