Exploiting data independence

A.W. Roscoe and R.S. Lazić
Oxford University Computing Laboratory

May 28, 1999

Part 1: Previous track record

The Oxford University Computing Laboratory has an exceptional record in the development and promotion of practically-relevant theory across a wide range of areas. Professor Roscoe has played a large part in much of this activity related in some way to CSP. He played a leading role in the collaborative efforts with INMOS on the development of the transputer and occam (via Alvey and ESPRIT funding), and this work led to the award of the Queen's Award for Technological Achievement to the Laboratory in 1990.

He has received substantial funding over the years from the US Office of Naval Research for work on CSP, which has led to the development of Timed CSP and has contributed in large measure (together with ESPRIT and industrial support) to the development of the CSP refinement checker FDR (a commercial product) which is now in use in approximately 50 academic and commercial sites worldwide. This tool has been used in practical projects in hardware design, telecommunications, safety-critical systems, distributed algorithms, real-time control and scheduling, computer security, fault tolerance and other areas.

Roscoe's book, *The theory and practice of concurrency* was recently published by Prentice Hall. This provides an extensive and up-to-date presentation of CSP, settting out the ground work for much of the research in this proposal.

For the past five years, Roscoe has devoted a significant proprtion of his research effort to computer security. His most recent EPSRC funding was a ROPA award "Establishing Security Properties of Distributed Systems" (GR/K34368) relating to this area, and was responsible for the first serious applications of model checkers to security protocols. So succesful was this work that it is now widely immitated across the entire model checking community (with Gavin Lowe's attack on the Needham Schroeder protocol discovered as an RA on this project being the standard target).

The work on the above project benefitted from parallel work being carried out by, and funded by DERA Malvern (the most important contact there being Dr P.Y.A. Ryan). The results of that project, both in terms of protocol verification and the spin-offs from the tool developments it required, are now widely used by DERA across an impressive range of practical topics for both military and civilian clients.

The work at PRG on the Parameterised Verification Problem, and data independence in particular, was inspired by the needs of the work on security protocols and other applications of model checking where one is forced to consider only "small" examples because of the limits on tools. R.S. Lazic, then a research student of Roscoe and now a Junior Research Fellow of Christ Church College, Oxford, began a programme of work that established many strong results in this area and developed considerably on the work of a diverse and growing worldwide community looking into similar problems.

This work led, in 1997, to Lazić becoming the first ever computer scientist to win first place in the Oxford University Senior Mathematical Prize competition (for graduate students).

The work on data independence has been applied manually by users of FDR for some time. DERA recently sponsored a feasibility study into related tool developments. The original motivating example of data independence in CSP has now been justified, since Roscoe has shown how security protocols may be proved via finite model checking runs using it (see [18]).

DERA has recently started to sponsor a CASE student at PRG (Sadie Creese), working with Roscoe on inductive techniques for CSP. This work has already uncovered a significant flaw in a "proved" practically important protocol (TTP, the time-triggered protocol).

Part 2

Background

Model checkers have been extremely successful over the past few years, having demonstrated their ability to decide questions about finite state machines that are well beyond human capabilities. A variety of *implicit* techniques, including BDD's, hierarchical compression and partial order methods, have been developed to enable these tools to address very large finite state machines indeed, at least in reasonably well-suited examples.

The major drawback with model checking is that these tools do not naturally prove general results about classes of circuits or systems, or establish that an algorithm works for an infinite range of inputs. Rather, they analyse a specfic and finite instance of whatever process the user is considering. But what one would frequently like to prove is that claims hold for all instances of one or more parameters, such as

- Network structure (including number of processes), within appropriate limits.
- The data types being passed around as data, and perhaps associated operations on it.
- The amount of buffering on channels.

The *Parameterised Verification Problem*, or PVP, relates to proving results independent of such parameters. This problem is certainly undecidable in general, in that there is no technique which can decide whether an arbitrary parametrised system satisfies its specification for all values of its parameters. Research is therefore concentrated on discovering classes of PVP that either are decidable, or where there are at least techniques that have a reasonable chance of resolving the issue.

A variety of techniques have been proposed for addressing versions of the PVP, the most prominent of which are *induction* over network size/structure [20, 1, 14, 15] and *data independence* analysis (see [19, 9, 7]; for our work see below). Using the latter it can often be shown that for a data type parameter T it is sufficient to consider a given finite data type (or perhaps a finite selection of finite data types) to prove a result for all types, or that it is possible to run a model checker on a system with a type parameter directly, with values of type T being held as uninterpeted variables during the check. As explained below, data independence analysis can also prove crucial in supporting inductive proofs.

Over the past three years a rich theory of data independence has been developed for CSP by R.S. Lazić and A.W. Roscoe (e.g., [11, 12]), based on the integration of a typed λ-calculus into established semantic models of the language. Techniques such as logical relations (parametricity) [16, 5], symbolic operational semantics [6] and symmetry elimination [3] have allowed the calculation of thresholds (sufficient sizes of types) over an ever-increasing range of problems, to the extent that our methods have been used in a number of significant case studies in conjunction with the refinement/model checker FDR.

An accessible summary of this work can be found in Section 15.2[1] of [17]. It is, for example, shown there that a particular model of a cache is correct (in the sense that the natural combination of it and a memory process is equivalent to the memory by itself) if it is correct for 1 cache line, 2 data values and 5 address values (the last two of these values being obtained directly from general theorems).

We have recently been studying applications from the world of computer security. In looking at how data independence results apply to noninterference conditions (absence of information flow from one user to another) we extended the core concept of data independence to encompass further language constructs (i.e., ones not permitted in the core theory of data independence): constants of data independent type, mappings (such as predicates) from data independent types to finite fixed types, and arrays that are indexed by one data independent type and contain members of another. While some of this work [13] is at present restricted to just the class of property required for noninterference checks (i.e., determinism checking) we believe it is both possible and important to allow these extensions for the full range of checks (see the discussion of induction below).

As a second application to security we have been developing techniques (reported in [18]) by which security protocols can be proved correct on model checkers. Previously it had only been possible to analyse

[1] Chapter 15 is available on the web via the book's web site (see refereences for URL)

versions with very restricted numbers of agents and runs; data independence allows us to infer general results from similarly small but carefully constructed versions of the protocols.

Our work on data independence has generated great interest from the FDR user community, and is starting to be used in their work. Both their work and our own applications throw up new challenges to the theory – one of the main purposes of this proposal is to provide a theoretical resource which can address these challenges as they arise. This is a rare case where highly mathematical work in theoretical computer science is meeting an immediate practical need.

Proposed Research

A. Theory of data independence

We will continue the existing programme of work devoted to extending the range of systems and specifications that admit proof by data independence techniques. At present we imagine that the main vehicles for this will still be logical relations, symbolic operational semantics and symmetry elimination. The outputs of this research will both be characterisations of classes of check for which the PVP is decidable, and where possible efficient methods for deciding them.

A list of currently proposed topics is given below, and we anticipate a continuing flow of ideas from the user community.

1. We intend to incorporate constants and predicates fully into the theory, so that we can accurately calculate thresholds for checks involving them. This is anticipated to be relatively straightforward, and we expect to produce a variety of results depending, for example, on whether general equality tests are allowed on the type or not.

2. As discussed briefly above, we have shown that in the case of determinism checking data independent programs can include arrays of one data independent type Y indexed by another, X. (The program must be restricted so that its control flow is never affected by values in Y.) We expect to be able to allow these constructs in refinement checks, but anticipate that this will be significantly harder than constants and mappings.

3. We have a conjecture that the current threshold value calculated for determinism checking can be nearly halved (often giving exponential speed-up, if true). If this can be proved it would substantially increase the range of information-flow problems that could be handled practically.

4. We can, at present, allow limited reasoning (allowing only specifications that do not constrain outputs of the DI type T) about programs which include one or more formal operations whose image is T. (In logical terms, these would be termed function symbols.) Thus, for example,

   $$APPLY = \mathit{left}?x \rightarrow \mathit{right}!f(x) \rightarrow APPLY$$

   inputs values in T and applies a formal function $f : T \rightarrow T$ to them.

   We would like to extend the range of properties that can be proved of such programs, through the use of symbolic operational semantics (potentially automated) [8, 2]. We anticipate that it may be hard to tell *a priori* between problems that can be decided and ones that can not be. Depending on how far this idea were taken, automation would probably require term rewriting on the data values and might require integration with theorem proving.

5. It would be extremely useful for many applications to allow orderings on DI types, or perhaps more general binary or higher order relations. We cannot expect results that calculate thresholds in every case here. For example a program that continues running provided that a series of values of type T increase strictly can do so for ever only when T is infinite. However we can hope to get partially effective techniques from symbolic operational semantics, or perhaps for some syntactically restricted class of programs.

6. Very direct use of logical relations has recently proved useful in security protocol analysis, by manipulating a program with an infinite threshold (and which demands an infinite type) to one with a small finite threshold. We hope these ideas will generalise to applications beyond security.
7. We intend to continue to seek new theoretical directions which can aid the general aims of the project. These might well involve further analysis of semantic models for CSP or other languages. We anticipate there may be connections between data independence and the 0-1 laws of finite model theory [4].

Milestones: We expect to be working on the above during the entire period of the grant, addressing them along with other related issues that may arise. We anticipate dealing with two or more such items each year.

B. Links with other approaches to the PVP

Induction One of the most common classes of PVP one encounters is proving properties of arbitrary-sized parallel networks. Data independence, by itself, does not handle such problems because it only applies to fixed topology systems. The most common technique for addressing variable-sized networks is *induction*: one formulates an appropriate property that is true of an increasing family of subnetworks.

The most obvious combination of these two ideas is straightforward: proving a property of a network of data independent processes by induction in such a way that the two theories do not seriously interact. If, for example (as in [17]) you have proved by data independence that the combination of a cache and main memory is equivalent to the memory, then induction tells you that *any number* of cache levels can be used without changing the logical behaviour.

More interesting is the case where we view the parameterisation by network size as parameterisation by a type T of the indices of network components, each of which treats T data independently. Although such programs do not come directly within the scope of our data independence results it is in many cases possible to perform the base and step cases of an inductive proof using them. Let $Spec(R(x), T)$ be a specification that we want to prove true of the fixed processes of our network plus all those indexed by members of T satisfying a predicate $R(x)$ on T. Then data independence is applicable, often with finite thresholds, to prove that, whenever k is a constant symbol assumed to satisfy R,

$$Spec(R(x), T) \sqsubseteq Spec(R(x) \wedge x \neq k, T) \parallel N(k)$$

We can thus (hopefully) establish via a finite check or checks on FDR that this lemma holds for *all* T, R and k. This proves the step case for an induction (for any fixed finite T) based on the number of values satisfying R.

Provided suitable inductions can be formulated and proved using finite thresholds, this provides a partial extension of data independence methods to include parallel networks indexed over the parameter type. We propose to develop this approach further and seek applications for it.

Handling buffered channels A system can be said to be buffer tolerant if it does not matter (in a sense that has to be defined, of course) if buffers (FIFO queues) are placed on some or all internal channels (see Chapter 5 of [17]). This is an important issue in the practical application of model checking since models of concurrent systems with arbitrary buffering on channels are (a) common, especially thanks to the success of SDL in becoming widely used in telecomunications applications, and thanks to the growth of the internet, and (b) have much larger or infinite state spaces thanks to the contents of the buffers.

It is often intuitively obvious that a system is buffer tolerant; we need a rigorous theory that will allow us to collapse buffers and hence state spaces without changing the result of a check.

As an alternative to eliminating buffers from system models as the above suggests it may be possible to find efficient ways of representing, or executing the checking of, models that still include them.

Even though it does not seem directly soluble using data independence (though inductive proofs of the style seen above sometimes raise related issues), we believe the buffering problem deserves serious attention within this project.

Milestones: We hope to make significant progress on *induction* within the first year of the project, and at least to have completed a literature search and preliminary investigations on the buffering issue. Further progress on each (and combinations of the two) can be expected in later years.

C. Broadening beyond CSP

Data independence is not unique to CSP (see citations above, for example), though the expressive power of the functional language and the process algebra, and also its extensive resources of theory and applied theory (e.g., to the specification and verification of security and fault tolerance) have led to particularly powerful and precise formulations of results, involving a wider range of language constructs than versions based on other notations. We hope, as part of this project, to define a framework under which data independence results and techniques can be moved as simply as possible from language to language.

One route to this is to give another notation a symbolic operational semantics in the same style as that we have used for CSP. Since the results about CSP are largely derived from ones about data independent symbolic transition systems, there is every likelihood that similar results can be obtained for other languages, hopefully systematically. The notations considered might be of immediate practical importance, such as SDL, or be further theoretical tools such as the pi-calculus.

The CSP/FDR approach is unusual in that specifications are cast in the same notation as implementations. In transferring results to other notations it will therefore be necessary to consider specification languages such as versions of temporal logic. We foresee few particular difficulties with this, at least in so far as they can be modelled via state machines/transition systems.

Milestones: We expect to make significant progress on this topic by the middle of year 2, and to have shown in detail (possibly in collaboration with others) how they can be used on at least one practical language such as SDL by the end of the project.

D. Tool requirements analysis

Practically all data independence work is related to the automated proof of programs on model checkers and related tools. Often its results do not themselves have to be automated: we can do a manual analysis of a specification/implementation pair to justify inferring general results from a finite instance. Nevertheless there is a definite role for tools either specifically developed to support data independence analysis or adapted to take advantage of it.

We do not anticipate any tool implementation work within the scope of this proposal, except perhaps some simple prototyping; rather we expect that tools will emerge from the related industrial/MOD projects. Our role will be to suggest potential routes for the automation of our existing and continuing work on data independence, where appropriate by specifying tools for others to implement.

The following is a rough classification of the automated techniques we expect to see emerging in connection with existing and future work on data independence.

1. Analysis of programs to determine whether they are data independent (a process closely related to type checking) and, if so, which of the various sub-classes of data independent programs they fit into.

2. Calculation of some of the quantities derived from programs that are used to calculate thresholds, and consequently calculation of these thresholds themselves from the theorems. This can be done at differing levels of sophistication.

3. The above can be regarded as stand-alone functions which, when one wishes to prove some general result about a data independent program, generate some (hopefully small and finite) proof obligations which can be discharged on a separate model checker such as FDR.

   The final category offers, potentially, a significant advance on these, at least in the most complex cases, namely actually executing the entire check on symbolic transition systems. Here one never expands the type parameter into a specific instance at all, but rather compares the normalised symbolic operational semantics of the specification against the un-normalised symbolic operational semantics of the implementation directly, using tricks such as symmetry elimination to cut the number of states visited substantially below the number that would otherwise arise.

Milestones: We will produce six-monthly reports on the automation prospects for the given period's work on the project, coupled where appropriate with specifications of tools. We expect much of our effort to be reactive – analysis and discussions prompted by one or more collaborators.

E. Case studies

Because we have an active user community already in place, we envisage that work done under this proposal will be chiefly theoretical, with appropriate liason with the other projects and bodies listed in the next section. It is clearly necessary that appropriate case studies are performed to drive and demonstrate our work, but we propose to restrict ourselves to small ones and perhaps a limited number of medium-sized ones chosen to complement work going on in the user community (though, of course, we expect to become involved in discussions concerning users' examples as well).

Resources, dissemination and beneficiaries

It is proposed to recruit a research officer and a doctoral student to work in conjunction with Roscoe and Lazić on the programme of work set out above. It would be beneficial to be able to recruit the student from October 1998, and the research officer as soon as possible thereafter for two reasons. Firstly, there is an unusually strong field of applicants for Research Studentships to the PRG for entry at that date, many of whom will not be awarded grants from the quota of 2 that is available. Secondly, Lazić's position at Christ Church runs (including its available extension of one year) to September 2001).

It is anticipated close contact will be maintained with Formal Systems (developers of FDR) Motorola, and DERA Malvern, who already support a CASE student working with Roscoe on inductive techniques relating to CSP and who have supported much of the computer security and safety-critical systems work on CSP and FDR in the past few years, as well as having performed a number of major practical applications of them. All three of these organisations have expressed a strong interest in the results of this research and are interested in developing and/or using associated tools. In the case of DERA this will be as part of a continuing close relationship with the Computing Laboratory and Formal Systems. Formal Systems's interest relates to potential developments to, and applications of, FDR. Motorola's interest arises primarily from telecommunications applications: it is presently negotiating a significant joint development venture with Formal Systems to adapt and extend FDR in directions which will benefit greatly from successful work on data independence.

The two investigators both propose to contribute to all the sections of work described above. We would expect the research student to concentrate on one of areas A–D (most likely A or C), depending on his or her aptitudes and preferences. (The choice made may affect the scheduling of work.) The research assistant would then be expected to concentrate on the remainder.

Powerful workstations will be required for this project because a significant proportion of the work will involve running FDR, which is intensive in both CPU and memory.

The major means of dissemination of the results of our research will be the publication of papers and participation in conferences, workshops, etc. In particular we anticipate that all tool specification work done in this project will be at the pre-competitive stage, with results being openly published, with no special rights being conferred on the collaborators. If the progress of our work dictates otherwise an appropriate collaboration agreement will be negotiated to protect the interests of all parties. Each year there will be a workshop, designed primarily for collaborators and other closely related researchers, to describe and discuss progress. This will be modelled on a successful workshop at Oxford held in March 1998 sponsored by DERA.

A project on Computer Security is presently being set up involving Oxford (Roscoe and Dr G.M. Reed), Tulane University and the US Naval Research Laboratory. We expect close contact between the research on the present proposal and this project, since as detailed earlier computer security research has close links with data independence.

References

[1] Browne, M., E.M. Clarke and O. Grumberg, *Reasoning About Networks with Many Identical Finite State Processes*, Information and Computation 81 (1), 13–31, 1989.

[2] Corella, F., Z. Zhou, X. Song, M. Langevin and E. Cerny, *Multiway Decision Graphs for Automated Hardware Verification*, 7–46, Formal Methods in System Design 10 (1), Kluwer, 1997.

[3] Emerson, E.A. (ed.), Formal Methods in System Design 9 (1–2), *Special Issue on Symmetry in Automatic Verification*, Kluwer, 1996.

[4] Fagin, R., *Finite-model Theory — A Personal Perspective*, 3–31, Theoretical Computer Science 166 (1), 1993.

[5] Fiore, M.P., A. Jung, E. Moggi, P. O'Hearn, J. Reicke, G. Rosolini and I. Stark, *Domains and Denotational Semantics: History, Accomplishments and Open Problems*, 227–256, Bulletin of the EATCS 59, 1996. (We are referring primarily to the section by P. O'Hearn on *Parametric Polymorphism*.)

[6] Hennessy, M. and H. Lin, *Symbolic Bisimulations*, 353–389, Theoretical Computer Science 138 (2), 1995.

[7] Hojati, R. and R.K. Brayton, *Automatic Datapath Abstraction in Hardware Systems*, 98–113, Proceedings of the 7$^{\text{th}}$ CAV, Springer LNCS 939, 1995.

[8] Hojati, R., A. Isles, D. Kirkpatrick and R.K. Brayton, *Verification Using Uninterpreted Functions and Finite Instantiations*, Proceedings of FMCAD, Springer LNCS 1166, 1996.

[9] Ip, C.N. and D.L. Dill, *Better Verification Through Symmetry*, 41–75, in [3].

[10] Lazić, R.S., *Theorems for Model Checking Data Independent CSP*, Technical Report, Oxford University Computing Laboratory, to appear in 1998.

[11] Lazić, R.S. and A.W. Roscoe, *Using Logical Relations for Automated Verification of Data Independent CSP*, Proceedings of the Workshop on Automated Formal Methods (Oxford, U.K., June 1996), to appear in Electronic Notes in Theoretical Computer Science.

[12] Lazić, R.S. and A.W. Roscoe, *A Semantic Study of Data Independence with Applications to Model Checking*, to be submitted to Theoretical Computer Science in 1998.

[13] *Lazić, R.S. and A.W. Roscoe, *Verifying Determinism of Concurrent Systems Which Use Unbounded Arrays*, Proceedings of the 3$^{\text{rd}}$ INFINITY, Research Report, Technical University of Munich, 1998.

A fuller version is the Technical Report PRG-TR-2-98, Oxford University Computing Laboratory.

[14] Lesens, D., N. Halbwachs and P. Raymond, *Automatic Verification of Parameterised Linear Networks of Processes*, Proceedings of the 24$^{\text{th}}$ ACM POPL, 1997.

[15] Reed, J.N., D.M. Jackson, B. Deianov and G.M. Reed, *Automated Formal Analysis of Networks: FDR Models of Arbitrary Topologies and Flow-Control Mechanisms*, Proceedings of the European Joint Conference on Theory and Practice of Software; Fundamental Approaches to Software Engineering (ETAPS-FASE), Lisbon, Portugal, March 1998.

[16] Reynolds, J.C., *Types, Abstraction and Parametric Polymorphism*, 513–523, Information Processing '83, North-Holland, 1983.

[17] Roscoe, A.W., *The Theory and Practice of Concurrency*, Prentice Hall, 1998. (Section 15.2* is most relevant to data independence.)

[18] *Roscoe, A.W., *Proving Security Protocols with Model Checkers by Data Independence Techniques*, Proceedings of the 11$^{\text{th}}$ IEEE CSFW, 1998.

[19] Wolper, P., *Expressing Interesting Properties of Programs in Propositional Temporal Logic*, 184–193, Proceedings of the 13$^{\text{th}}$ ACM POPL, 1986.

[20] Wolper, P. and V. Lovinfosse, *Verifying Properties of Large Sets of Processes with Network Invariants*, Proceedings of the International Workshop on Automatic Verification Methods for Finite-state Systems, Springer LNCS 407, 68–80, 1989.

Items marked * are available on the web-site of *The theory and practice of concurrency*, namely
`http://www.comlab.ox.ac.uk/oucl/publications/ books/concurrency/examples/security/`

Questions on EPSRC form

Investigators

A.W. Roscoe (Principal) and R.S. Lazić.

Title

Exploiting data independence

MOD

Yes, Colin O'Halloran, DERA Malvern

Objectives

1. To enhance the theory of data independence in CSP, with the particular objective of increasing the range and power of its practical applications.

2. To provide methods by which results on data independence cast in one notation can be transferred to others without unnecessary duplication of effort.

3. To facilitate the production by others of tools supporting the theory.

4. To develop other approaches to the Parameterised Verification Problem.

Summary

A program is *data independent* in type variable T when it makes sense whatever nonempty type is used for T. Over the past few years researchers have developed methods by which model checkers can prove properties of data independent programs for all T. The Investigators have developed this theory for CSP, covering an exceptionally wide range of language constructs and types of program. Their work is beginning to find applications in MOD and industry, and also in academic work on computer security. Tools will soon be under development to support these. This proposal extends the resources available for the development of this theory, in particular by addressing issues that arise from applications and tool developers. The methods used include logical relations, symbolic operational semantics (plus the theory of labelled transition systems including explicit data variables) and symmetry elimination. We will look at other approaches to the automated proof of general results, such as induction, especially where there is a relationship with data independence. Many ideas in data independence transfer intuitively from language to language: we will develop methods, probably by using symbolic operational semantics as a bridge, enabling results to be transferred at a formal level.

Key words: Data independence, CSP, model checking, logical relations, symbolic operational semantics, symmetry elimination, parameterised verification, induction.

Beneficiaries

This theory is important over most of the application areas of model checking: communications protocols, hardware design (above gate level), safety criticial systems, computer security and distributed algorithms. The benefits are to increase the range of problems accessible and to generalise results to practical-sized data types.

The immediate beneficiaries fall into two categories: those using CSP and its tools, including defence and industrial organisations in the UK and elsewhere, as well as many academic researchers; and secondly the growing international community of workers in data independence and related fields.

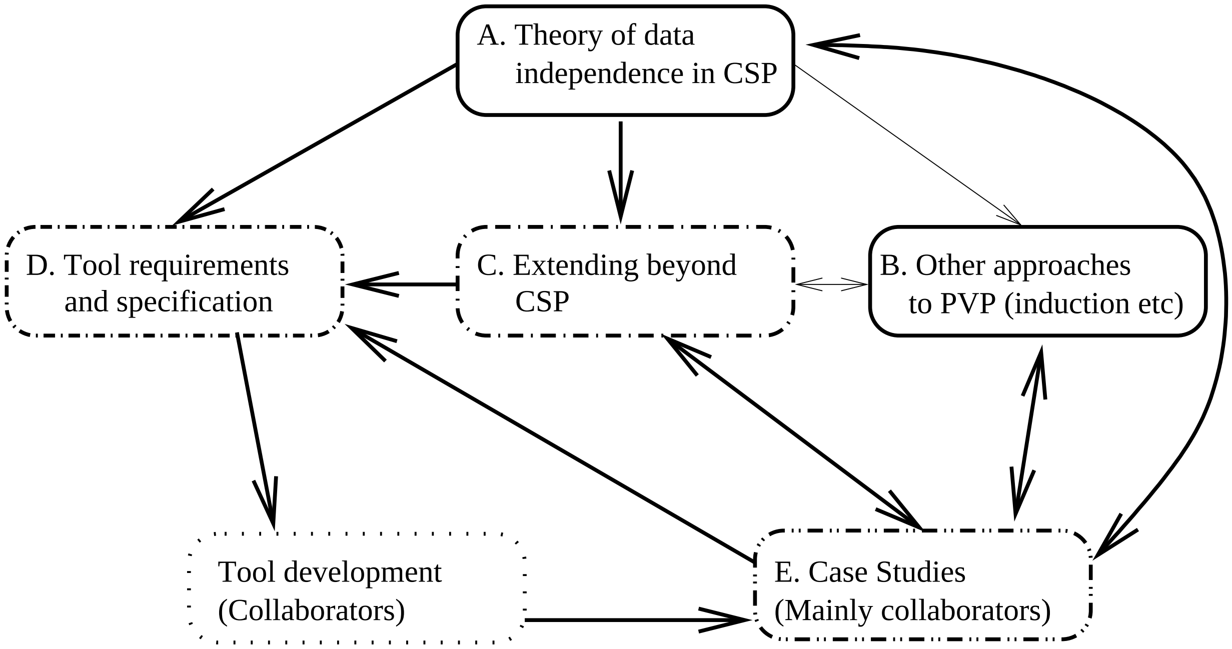
It is intended to form collaborative links with three organisations with an interest in applying our theory and developing associated tools: DERA (Malvern), Formal Systems (Europe) Ltd., and Motorola.

Key words: Computer security, Telecommunications services, Safety-critical systems, Automated verification.

Referees

S.D. Brookes
CMU
S. Abramsky
Edinburgh
A. Valmari
Tampere

Diagrammatic workplan



- Chart shows the major and intermediate level interactions between the proposed strands of work.
- Border styles indicate level of involvement of collaborators.
- The scheduling of the items depends to a significant degree on topic chosen for student (see proposal), but we expect at least some progress on each topic every year.
- Annual reports will detail progress under each of the above topics.
- Six-monthly reports on item D (see proposal)
- Workshops involving collaborators and other interested parties at least annually.