The Amorphous and the Crystalline*

P.S.Subramanian,
Theoretical Computer Science Group,
Tata Institute of Fundamental Research,
Mumbai-400 005, India.
e mail - *cayley@tifrvax.tifr.res.in*
web - *http://www.tcs.tifr.res.in/~pss*

10 August, 96.

TIFR/TCS/96 - 2.

Abstract

It is intuitively obvious that Artifacts -Engineering systems, Mathematical proofs and Software- actually created by humans have some features which *distinguish* them from other objects of a similar nature which could have been created in principle. A clearer view of this distinction may be obtained by considering the following: There is nothing in the laws of Physics which could prevent us from constructing systems which do *not* have well defined sub-systems. Similarly, laws of logic and the laws of computation do *not* require that proofs in mathematics have lemmas and computer programs have a modular structure! But, as a matter of fact, whenever these artifacts are the result of human creation, they do *inevitably* possess these features. If the task of the Natural Sciences is to provide explanations of the form, structure and properties of naturally occurring objects starting from some basic laws, then a science, if it could provide us with similar explanations about the structure of humanly created Artifacts, should be called the Science of the Artificial. H.A.Simon, in an eloquent booklet (*The Sciences of the Artificial* - MIT press, 1970) pointed out that such a science, *comparable in depth* to that of Natural Sciences, did not exist then; Nor does it exist now.

In this report, I describe my efforts in this direction. Analysis of the central concept of Hierarchies in turn brings out the importance of understanding the related concept of 'auxiliary notions'. These are known

*Mirroring '*The Dry and the Wet*' - J.A.Goguen, PRG Tech. Monograph - 100, March 92, Oxford University.

by different names depending on the subject matter - Definitions, Lemmas (in mathematics), Cuts, Types (in logic and foundations), Modules (in software engineering) etc.. The search for the answers to the questions - *What are the desirable characteristics of formal frameworks within which we can address problems of such generality? Is there a sufficiently general and mathematical characterization of the concept of auxiliary notion? What exactly is its functional role?* - takes us through a tour of a conceptual terrain involving Category theory, Galois Theory, Theory of Descent, Topos Theory, Foundations of Mathematics and Algebraic Geometry, Axiomatic theory of Radicals, and topics from Theoretical computer science like Proof theory, Type theory and Complexity theory. As a result of this 'tour', we are able to identify a set of well defined (sub) problem areas for future research. Their well-definedness should enable us to address the basic questions in depth. Moreover, these problem areas have the pleasant feature of being simultaneously fundamental as well as having the potential (in view of the fact that they are about fundamental problems of *Engineering*) to be useful in practice.

Contents

"In reply to the question 'What does mathematics study?', it is hardly acceptable to answer 'structures' or 'sets with specified relations'; for among the myriad conceivable 'structures' or 'sets with specified relations', only a very small discrete subset is of interest to mathematicians and the whole point of the question is to understand the special value of this infinitesimal fraction dotted among the amorphous masses"[1]

I.R.Shafarevich

1. Introduction

The motivation for the questions raised in this note come from **two** sources - The first one is the appearance of the book 'The Sciences of the Artificial' [1] around 1970, authored by H.A.Simon, which asks whether a Science of the Artificial can exist comparable in mathematical depth to that of the Science of the Natural (The Physical Sciences). According to the author of this book, there still does not exist a satisfactory answer to this question (*Ref*: Modelling the Tempus - Horus metaphor) [2].

The second source is a desire to understand a 'paradoxical' situation involving Theory and Practice in Computer Science. *On the one hand*, Theoretical Computer Science is full of *negative* results: NP completeness of various interesting problems, Impossibility results, and complexity (even undecidability!) results involving logics used in Formal verification. *On the other hand*, useful algorithms *are* developed and large scale systems *do* get built-which perform with more reliability than one would expect them to! Of course, the 'paradox' gets resolved by observing that the criteria and premises used in Theory and Practice are slightly different - In the area of algorithms, adopting a Probabilistic view (Simplex is Polynomial on the average, but exponential in the worst case, Distributed consensus is impossible in the Deterministic Asynchronous model, but not so probabilistically..) and exploiting structure (various kinds of BDD methods, Graph algorithms for various special classes etc...) are some of the heuristics which have been used informally long before Theory provided (and is providing) a posteriori explanations. A similar situation obtains with respect to System verification - whereas Verification environenments are developed to tackle arbitrary system descriptions, in practice only a small subset of systems having certain 'regularity' actually get built.

Let us call systems which can *in principle* be built from a given set of primitive components **Amorphous** (the 'Design Space') and the 'regular' subset **Crystalline** (the space of 'Structured' designs)

The crucial questions of a Science of The Artificial are about this 'regular'

[1] *Algebra I*, A.I.Kostrikin, I.R.Shafarevich (Eds), Encyclopaedia of Mathematical Scices, Volume **11**, Spreinger Verlag, 1990.

subset - reasons for its existence, characterisation and study of its internal structure - hierarchy[2], modularity. Naturally, then, the *precision* (= quantifiability?) with which these questions can be answered within a theoretical framework of the will determine it's *depth*. Currently, Science of the Artificial falls *far* short of Natural Sciences in this respect. Books on Software Engineering, for example, will dutifully record the statement that 'Hierarchy and Modularity are the main tools used to control the complexity of Systems' - and no further *quantitative* statements about the nature of this 'control' will appear in the book[3].

Informal methods of restricting oneself to the crystalline part of a 'Design Space' are designer's thumb rules, standardisation etc... In the context of Computer Science one could conceive of *two* different mechanisms which can help the designer to stay within the crystalline part of the Design Space - The *first* possibility is by providing a *Type System*[4] - the so called idea of **Structured Design**. A *second* method could conceivably be based on some kind of *Semantics* - i.e one provides some kind of Abstract semantics which comes equipped with a test of membership w.r.t the crystalline part. If the first can be called 'design for (easy) verifiability *by construction*' then the second one should be called '*generate and test* for (easy) verifiability'.

Whereas the study of various 'Type systems' is an active[5] area of study currently, I am not aware of approaches to the problem of constructing efficiently verifiable systems by the second method - i.e by providing an abstract semantics which formalises some notion of regularity which in turn has implications on the complexity of verification.

In the rest of this note, I discuss some issues involved in setting up such a theoretical framework; during the discussion an effort is made to identify problem areas which might also prove to be interesting on their own. The note is organised as follows - In the next section, an overview of Computer Science is provided to place the aims of the current research programme in *context*. This section also contains an analysis of the *types* of interaction between Mathematics and Computer Science. In section 3 we attempt to enumerate the desirable features of a theoretical framework which can serve as the basis for the study of the Sciences of the Artificial and of Hierarchies in particular. Sections 4 and 5 look at Galois theory and it's generalisations with the intention of using it as a model for structured design. Section 6 is the conclusion[6].

---

[2]It is conjectured that hierarchy confers (evolutionary) advantage even among biological systems.

[3]This reflects the state of the art. A workshop to discuss related issues was held at DIMACS recently (July, 96)

[4]*It is perhaps not surprising that the semantics of parametric types involves the categorical concept of Natural Transformation - a formulation of a facet of regularity*

[5]There are many type theoretic approaches which provably bound the complexity of compuatation; but not (m)any which bound the complexity of verification.

[6]The Appendices contain the 'pre-history' of this report and the Abstract and Table of Contents of Version 2, which is currently under preparation.

Remarks on the title of the paper: It is often the case, that the *mere act of attaching names* (encapsulation!?) brings further clarity to a situation. Although, I had some preliminary ideas about the contents of this paper earlier[7], things became clearer after I coined this title! The title itself was arrived at after reading the report 'The Dry and the Wet' by Joseph A.Goguen, and by trying to fit my own ideas to the context of that report. In fact, this note can be regarded as a refinement of the 'Dry' part of that report. The occurrance of the phrase 'amorphous' in the title is due to the influence of the passage from Shfarevich.

2. Computer Science - an Overview

Computers are nowadays being used in many diverse areas - Artificial intelligence *to* Accounting procedures, Education *to* Encryption, Medical instrumentation *to* Missile guidance etc.. An attempt to isolate the *invariant* aspect of Computer Science from these diverse applications necessarily involves some abstract concepts. We first isolate these concepts - **Intension** and **Extension**.

Without going into definitions we illustrate these concepts by few examples:

- Formulae vs Functions
- Matrix vs Linear transformation
- Proofs vs Theorems
- Logic Circuits vs truth tables
- Realisations of Systems vs Specifications
- Syntax vs Semantics
- Operational semantics vs Denotational Semantics
- Presentations vs Algebras
- Triangulations vs Topological Spaces ...

One can easily think of many other pairs which exemplify these concepts.

In such situations, one generally will have a function, called the *semantic function*, which maps the space of Intensions to the space of Extensions. Note that if the space of Intensions happens to have canonical forms - Jordan form, Disjuntive normal form, Cut-free proofs etc one could possibly have a mechanism for testing equality, at least in principle, w.r.t this function.

Next, Computer science is **defined** to be the science of the analysis of the tension[8] between intension and extension (see Fig 2.1).. This analysis itself is

[7] *Category Theory and System Design*: TIFR/TCS/90-3

[8] Some examples of this *tension* in the context of *Presentations vs Algebras* pair : It is

Figure 2.1: The Computer Science map

carried out with the help of *Calculational models*, roughly those in which one can do loop free computations. Although many concepts and problems of computer science are already definable in this framework, we find that perhaps it is too narrow, and broaden the class of models to *Computational models*, where the computations can involve arbitrary iterations.

The transition from calculational models to computational models entails a subtle shift in the type of mathematical structures used in studying these models; In computational models the intension are denoted by 'terms' of a rather simple kind and the simplicity is reflected in the fact that the semantic categories (The space of extensions) will be merely algebraic. The simplicity also means that one can think of intensions as loop free programs and hence one can predict the number of computational steps from the structure of the terms. But, when we make the ascent to computational models, we generally lose this ability, since we will be enriching the term structure to model *iteration* and/or *interaction* in the space of Intensional descriptions. This additional complexity will be reflected on the space of extensions - the categories of extensions will now tend to have topological aspects too to model partiality and other phenomena. They also will be required to have the fixed point property.

2.1. Pure Computer Science

A specific computational model is fixed by the specification of the space of Intensions, the space of extensions and the associated semantic function. Since intensions are concrete objects which will have to be physically realized and manipulated by humans, one will also have certain cost measures on the space of intensions and a mechanism for specifying 'large' intensional objects in a compact way. These aspects lead to *three* types of questions one could ask about a particular computational model. These are

- What are the limitations 'in principle' of the computational model? In particular, is the semantic function onto ?
- What are the 'costs' associated with carrying out various computations in this model?
- What 'human friendly mechanisms' are availble for specification of intensions for this model?

Study of a variety of computational models with reference to the three types of questions belongs to the **Pure Computer Science**.

known that the same 'abstract' monoid can have different presentations leading to terminating as well non-terminating rewrite systems. Is the property 'having *some* presentation leading to a terminating rewrite system', intensional or extensional? This is a topic of current research[14]. Another example is related to expanding graph families; see Problem 1.1 in [15].

2.2. Applied Computer Science

The vitality of Pure computer science depends on the *variety* of computational models it studies. Where does this variety come from? Attempts to apply computers in various aspects of human activity *generate* this variety. An overview of these applications, collectively called **Applied Computer Science**, organised by function, is provided in Fig 2.1.

2.3. The relation between the Pure and Applied CS.

Thus, we have a situation, where attempts to utilise computers in various applications generate the need for the study of computational models relevant to those applications, and a set of questions about utilisation of these models in practice. This pair - *model, set of questions* - is handed 'over the counter' to the Pure part of Computer science. It, in turn, is expected to provide an analysis of the computational model **and** answers to the relevant questions. This is the ideal situation, which may not always hold! Sometimes, theoreticians may modify or simplify the model for the sake of tractability; else, they may keep the model intact, but proceed in a theoretically interesting direction, not necessarily relevant to practice.

2.4. The Three layers of Pure Computer Science

We have already described the **first** (outer) layer as the study of three types of questions with reference to a *fixed* computational model. Recall the *each* computational model has *five* components - *Intensions, Extensions, Semantic function, Cost functions and Structuring mechanisms.* Once one has a variety of compuational models, however, theoreticians naturally become interested in the the *category* of these models. Questions related to the amount of structure preserved by various possible definitions of *morphisms* in this category of computational models can be interpreted as questions currently studied in theoretical computer science - Relative computational power, P vs NP, Questions about language embeddings etc.. Studies of this kind form the **second** inner layer of Pure Computer Science. A theoretician also likes to have a conceptual toolbox of sufficient sophistication. Thus, he may observe that, Mathematical structures used in Pure computer Science have a particular flavour and might be interested in understanding these structures in their own right. This kind of study forms the **third** and innermost layer of Pure Computer Science and at this point it overlaps with certain parts of Mathematics (**MR** Review classification 3 to 8 and 18)

This is perhaps a good place to examine the relationship between Mathematics and Computer Science in general.

2.5. Mathematical Structures in Computer Science

The use of Mathematical structures in Computer Science can be classified by *three* increasingly wider interpretations of the phrase *Mathematical Structure*:

At the *first* level we have the application of structures which *already* exist in current mathematical practice. Most of such usage arises from the Applied part of Computer Science; Since one can see from Fig 2.1 that the applied part spans a variety of disciplines, *most* parts of mathematics belong here! The classical structures will have to be studied from a *computational perspective* (involving perhaps a passage back in time!), however, in contrast to the structural and axiomatic view of modern mathematics.

At the *second* level we have structures which are weakenings and generalisations of those at the first level: These are *not* arbitrary weakenings however; The particular kind of weakenings are characteristic of the transition from a Platonic view of mathematical structures to a more Constructive view. Thus, we move to Intutionistic logic instead of Classical (Heyting algebras instead of Boolean), Domains instead of Sets, Sober spaces instead of Hausdorff etc. It is fair to say that the constructivist philosophy owes a debt to Computer Science for it's revitalisation.

Lastly, we have the idea of Mathematical 'structure' as captured by Category Theory: here we refer to various *coordinatisations* of the space of mathematical structures - Stone Gamut [3], Spectral classification, Sketch hierarchy [4] etc. Is this aspect of the concept of mathematical structure useful to Computer Science? The research programme outlined in this note can be seen as an attempt answer this question positively. It can also seen to be 'interdisciplinary' within Theoretical computer science (see Fig 2.1).

3. Desiderata for a Science of the Artificial.

What are the desirable features of a formalism which can serve as a basis for the Science of the Artificial?

- Since it has to deal with a design problems in widely varying contexts, the mathematical structures appearing in it should admit as wide a class of interpretations as possible.

- Since it is intutively obvious that the notion of 'structure' plays an important role in the construction of artifacts, the theory should support a formulation of this concept in as objective way as possible. Moreover it should admit a definition of 'relative structure'.

- Artifacts have a 'stratified' structure- the theory should support the emergence of **non-transitive relations** between structures in a natural way.

- Within the theory one must be able to show that the requirement that complexity of comprehension be bounded is one of the reasons for stratification.

As a tentative step, we choose *Category Theory* as the substrate and (generalized) *Galois theory* as the model for a Science of the artificial.

4. Why Category theory?

The paper '*A Categorical Manifesto*' [11] motivates the use of Category theory in Computer Science. Here, we make some remarks on the 'structural' aspect.

The philosophy of *reductionism* is part of the methodology of Science; to understand a complex object one tries to *decompose* it into certain other primitive objects (which presumably have *already* been understood) and *then* analyse the manner in which these primitives have to be put together to reconstruct the original object. The information about the 'manner in which the primitives have to be put together' is called the 'structure' of the object (relative to the class of primitives). In this sense, the problem of understanding a complex object has been *reduced* to the problem of understanding the structure (since the primitives are already 'understood'). Various structure theories in Algebra for examples of this methodology. This methodolgy is *particularly* relevant to computer science, because *almost all* the complexities of the objects come from the *structural* part, the primitives themselves being *really* primitive! Thus, it is apparent that the concepts of 'structure' and 'structured objects' are *basic* to Computer Science and that computer science could benefit from an *axiomatic theory of structured objects*.

How will an axiomatic theory of structured objects look like? One can draw an useful analogy with geometry; Just as in axiomatic geometry, where the primitive concept of *point* is left uninterpreted and only meaningful statements about points are those which can be made with reference to a *collection* of points, in the axiomatic theory of structured objects - i.e the *Theory of Categories* - the concept of *structured object* is left uninterpreted and any statement about these objects can only be made through the relations - the morphisms - allowed in the theory. Just as in geometry, where we can get different geometries by varying the axiomatisations of the allowed relations, in the theory of Categories we get various categories by similar means. *This analogy can be extended further*: In geometry we define point sets by *equations*. In the theory of categories we have the method of specification by *diagrams*. The role of *geometric constructions* (which specify equationally defined subsets from given and/or already constructed subsets) in geometric problem solving is well known from high school geometry. This role is played by *diagram based* arguements in Category theory. Finally, in suitable geometries, one can use the technique of *linear approximation, derivatives* etc.. in the solution of certain class of problems. Similarly, in suitable categories (*Abelian*

categories - which are axiomatisations of the 'space' of structured objects possessing some crucial aspects of *linearity*), we have the method of *exact sequences, resolutions*[9] *and derived functors* (which measure the degree of departure from exactness) - the general idea behind Homology theories.

It may be mentioned, in passing, that the structural viewpoint is useful in Natural Sciences as well[10].

Is there an instance where reasoning at the 'abstract' structural level helps in solving 'concrete' problems? This question brings us to the next section.

5. Why Galois Theory?

5.1. Galois theory as a model of step-wise refinement.

Everybody knows what Galois Theory tells us - that a general equation of degree greater than four *can't* be solved by radicals. This is a *negative* result. But the theory tells us more -

- There *are* equations degree greater than four which *can* be so solved and
- There exists a *methodology* for identifying and solving them.

The phrase 'step-wise' refinement probably didn't exist when Galois developed his theory - but it would have been an accurate description!

Let us recall the situation - We have a space of equations - a space of *intensions* in our terminology - *some* of which can be solved by radicals and arithmetic operations. Galois theory provides a 'semantics' to these equations - The Galois Groups. Then we are able to *test* this semantics to ascertain whether the equation is solvable in the sense indicated above - checking it for (group theoretic!) solvability. Not only that, by successively extracting maximal normal sub-groups, we are able to obtain subproblems (resolvents) and thus providing a step-wise refinement method to solve the original equation.

It should be clear that if we think of Equations as Specifications, the classical Galois theory provides a nice role model for a 'Science of the artificial'- it not only provides for step-wise refinement, even indicates what is a *wise* step!

At this point, one might raise the objection that the classical Galois Theory operates nicely in the structured universe of Fields. Will it be useful in the weak world of Specifications? No doubt, deep and intricate theorems of the classical setting will not be available to us, but perhaps we can extract some useful part

[9] *The questions related to monoids having presentations with terminating rewrite systems- referred to in an earlier footnote- are actually being approached via their resolutions.*

[10] f*oot note added during Oct '97*: In [12] it is argued that a kind of variational principle on mathematical structures might be operating in the evolution of Physical theories. In [13], it is argued that the technical concept of *compatibility*, which plays a crucial role in the structure theory of Integrable systems, is basically a property of morphisms w.r.t bilinear structures.

of it. It should be pointed out even in the classical era, Galois Theory had been applied to a different setting - the Picard-Vessiot Theory for Differential equations - thus providing an early demonstration that the ideas behind Galois Theory *transcend* it's original goal.

Fortunately, successive generations of mathematicians have looked at Galois Theory at various levels of abstraction and expanded the settings in which it's three main ideas - *classification, correspondence and solvability* - can be formulated [9]. If the Semantic Categories used in Computer Science admit of a 'Galois structure' to be imposed on them, then the possibility of using Galois -theoretic ideas in computer science contexts is opened up[11].

What about *Complexity* aspects? It is proposed to equate 'complexity of comprehension' with the 'complexity of provability of system properties in associated formal systems' of reasoning. We face a stumbling block here however. Current methods of measuring the complexity of these logics do not take a suficiently pragmatic view of the situation - recall the remarks of the first section about satisfiability and BDD methods. So, until the time we develop a sufficiently flexible method of measuring complexity (roughly, we are looking analogues[12] of Ergodic hypothesis to bridge the microscopic and the macroscopic), it is intended to focus on the structural aspects of the situation. Perhaps there is a route through Algebraic Information Theory (of Goppa) or some version of Kolomogorov complexity. It is worth mentioning here that even the classical Galois Theory had associated notions of 'ambiguity' which definitely have information theoretic flavor.

We think of the current complexity theory as being one dimensional; Since we **do** use *structuring* to control 'complexity' even when the objects involved are completely intractable from the point of view of *standard* complexity theory, we need a *more general* complexity theory to understand this phenomenon. In particular, if we are to understand the value of structuring, then this understanding should be *independant* of the position of the class as a whole in the standard complexity theory. Consider, for example, the following hierarchies; the hierarchy of Generalised Horn formulae in the class of all Propositional formulae, Until Hierarchy in all Temporal formulae and Higher Order algebras in HOL. In each of these cases, we start with a *nice* subclass (Horn formulae, Zeroth level of the until hierarchy, Equational logic) which are 'tractable' and we build structured classes by *extensions*. Note the position of the base cases in the complexity hierarchy - in the first case are in Poly time, in the second we are in Decidability, in the last case even Undecidability!

Coming back to Galois theory, it is unreasonable to expect the full 'equation solving' machinery to carry over to arbitrary computer science settings. It is more reasonable to expect the semantic aspects of Galois theory to be more

[11]E.Engeler in 1981 has shown that Galois Theory generalised to FOL can be used prove some lower bounds. I have not seen further work in this direction since.

[12]Perhaps concepts like 'almost everywhere equivalence of logics' currently being studied in Descriptive Complexity theory is relevant here.

generally applicable; by semantic aspects, I mean the structure induced by the radical on the Category of Groups. This kind of structure is classically axiomatised by Kurosh-Amitsur theory of radicals (and its variants). Fig 5.1 shows the conceptual terrain of related aspects. Some of the related references are [[5] to [9]]

5.2. A format for application of Generalised Galois Theory.

Choose your domain of application. It would have a formalism to specify systems and implementations or realisations. Organise these into appropriate Categories. Check that these categories have enough structure to enable one to define a 'Galois structure' on them. This means we have rigorous means of identifying the Crystalline part with respect some chosen subcategory of Primitives. (Usually a set of primitive components and their specifications will be at hand.). Using some appropriate notion of Entropy associate some complexity measures with the implementations. Setting up thresholds on these measures will automatically induce a hierarchical structure on the implementations (see Fig 5.2).

One could conceive of Software tools which would aid the user in computing the maps indicated in the figure.

6. Conclusion

We have outlined a framework for application of Category Theoretic concepts (The Generalised Galois Theory and the Theory of Radicals mentioned in the note *need* category theoretic notions for their formulation). It would be interesting to fix a domain of application[13] and carry out the program according to the format described in Fig 5.2. Since I am quite familiar with the world of VLSI design, I intend to choose this as the domain of application and use the method of specification via Sketches [10] to address these issues in the future.

[13]Viewing even the classical Galois Theory in the context of this note may raise some interesting questions: Is there a *type system* for solvable polynomial equations? Is there an *information theoretic interpretation* of Galois theory?

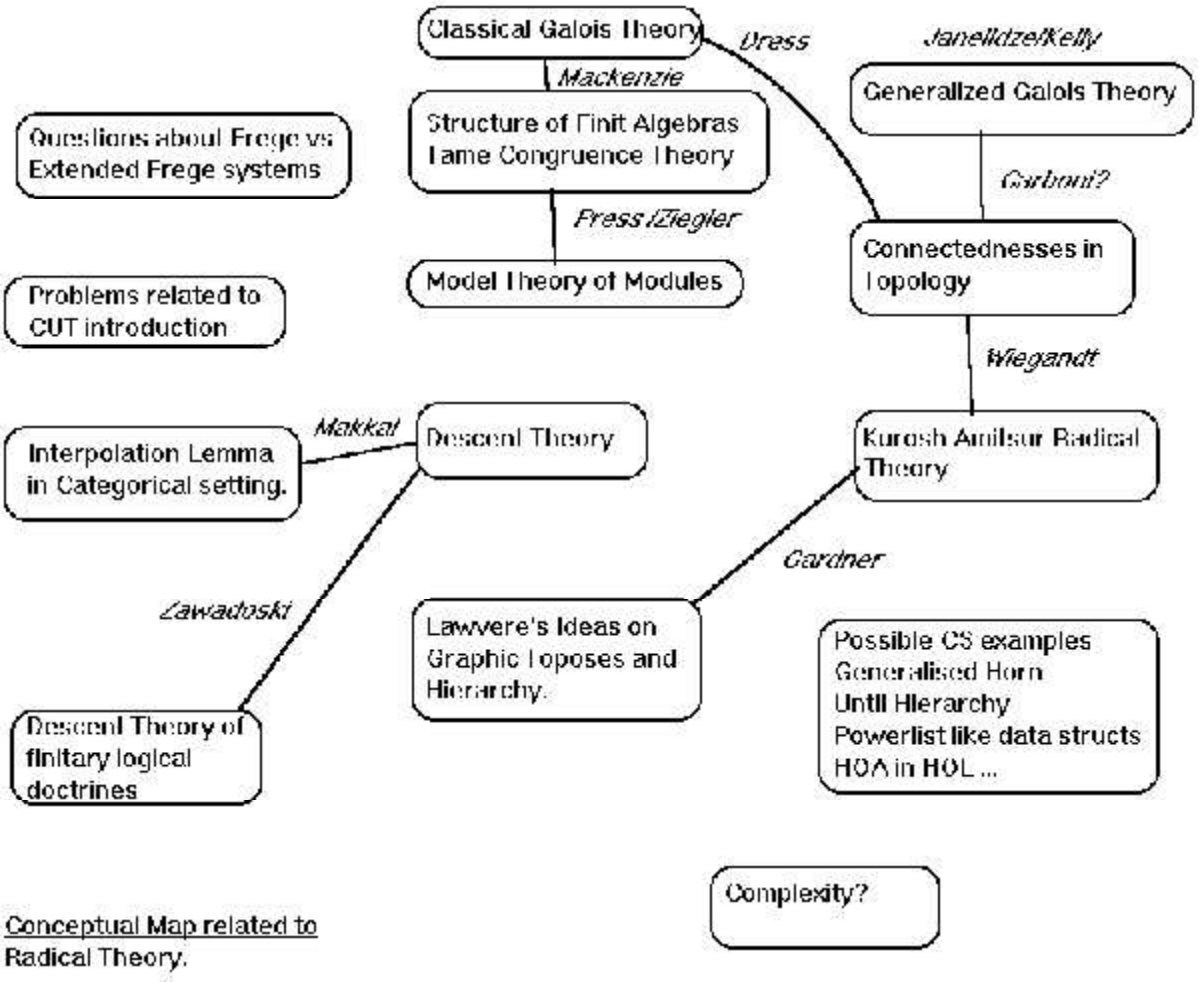
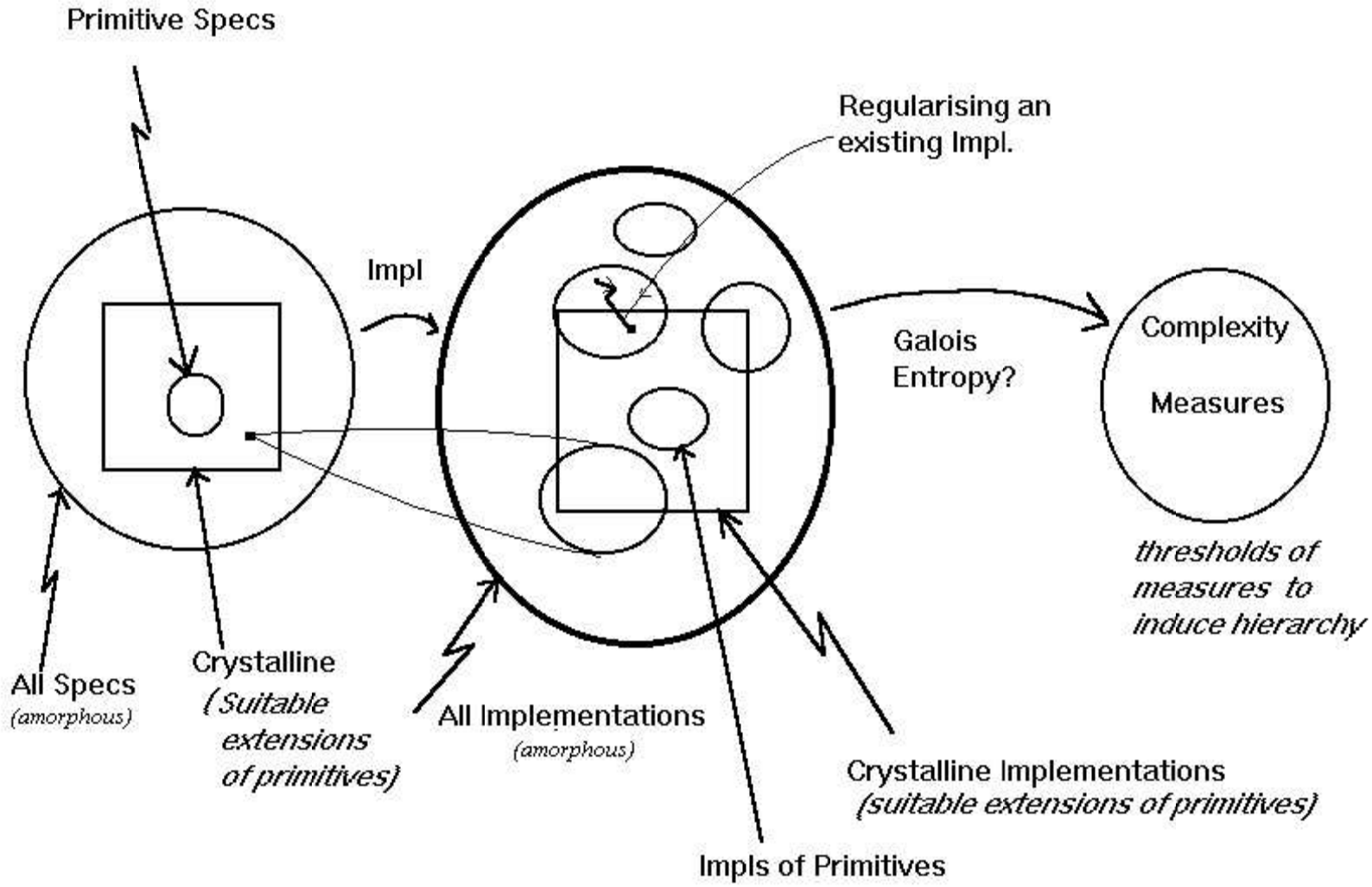


Figure 5.1: Conceptual terrain around Radical Theory



The Amorphous and the Crystalline

References

[1] Herbert A.Simon, *The Sciences of the Artificial* , M.I.T Press, 1970.

[2] e-mail communication with H.A.Simon.

[3] V.R.Pratt, *The Stone Gamut - a coordinatisation of mathematics*, in Logic in Computer Science, 1995.

[4] Barr and Wells, *Toposes, Triples and Theories*, Springer, 1988.

[5] A.Dress, *A characterisation of solvable Groups.* , Math. Z, 1969.

[6] George Janelidze and W.Tholen, *Facets of Descent* , in Applied categorical Structures, Vol **2**,1994.

[7] F.W.Lawvere, *Graphic Toposes* - Manuscript, 1996.

[8] L.Marki et al, *A general Kurosh-Amitsur radical theory*, Comm in Algebra, Vol **16**, 1988.

[9] George Janelidze, *Pure Galois theory in categories*, J.of Algebra, Vol **132**, 1990.

[10] John Gray, *The category of sketches as a model for algebraic semantics*, Categories in Computer Science and Logic, Contemprory Mathematics, Vol **92**.

[11] J.A. Goguen, *A Categorical Manifesto*, Mathematical Structures in Computer Science, Vol **1**, (pp 46-57), 1991.

[12] L.D.Faddeev, *A mathematician's view of the development of Physics*, Frontiers in Physics, High technology and Mathematics, World Scientific Pub, 1990, (pp 238-246).

[13] B.Fuchssteiner, *Compatibility in Abstract algebraic structures*, Algebraic aspects of Integrable systems, Birkhauser, 1997.

[14] D.E.Cohen, *String Rewriting and Homology of Monoids*, Mathematical Structures in Computer Science (1997), Vol **7**, pp 207-240.

[15] A.Lubotzsky and B.Weiss, *Groups and Expanders*, DIMACS Series in Disc.Maths and TCS (1993), Vol **10**, ed- Joel Freidman, pp 95 -109.

A. The Background

A.1. A paradoxical situation in Computing

The following situation seems paradoxical - *On the one hand*, Theoretical computer science is full of *negative* results- proofs of high computational complexity of many relevant problems, undecidability of various logics applicable to verification etc. *On the other hand*, the practice of computing does **not** seem to be affected by these negative results! Simplex method is still used widely and Complex systems are built which perform with sufficient reliability. In this context, it is worth mentioning the the title of the invited talk by Prof C.A.R. Hoare in the International Conference of Software Engineering (March,96) - *The Role of Formal Techniques: Past, Current and Future or How Did Software Get so Reliable without Proof?*. After examining the situation, he concludes that use of principles which have proven effective in all other branches of engineering is responsible for the 'unreasonable' reliabilty. So, what are these 'principles' and how do they manage to avoid the complexity pitfall? Can these questions be tackled in a scientific manner?

I had started to think about these (*apparently* philosophical) questions seriously from about 1990. I had earlier put down a possible approach in an internal Technical report- *Category Theory and System Design* and have been pursuing these questions via this approach since then. Some further thoughts on this were published in the *EATCS bulletin Algebraic specification column* in 1992, in an abstract titled '*Reasonable Hierarchies*' submitted to the Mathematics of Hierarchies and Biology Workshop in 1996, and in the report *The Amorphous and the Crystalline* being writen up currently.

A.2. Basic Premise

A basic premise underlying my appraoch can be stated rather forcefully by adopting the following quotation by I.R.Shafarevich (taken from the preface of his book on Algebra).

"In reply to the question 'What does mathematics study?', it is hardly acceptable to answer 'structures' or 'sets with specified relations'; for among the myriad conceivable structures or sets with specified relations, only a very small discrete subset is of interest to mathematicians and the whole point of the question is to understand the special value of this infinitesimal fraction dotted among the amorphous masses"

This quote (suitably rephrased) could very well have been made by an engineer!

"In reply to the question 'What do engineers design?', it is hardly acceptable to answer 'systems' or 'networks of components with specified interconnections'; for among the myriad conceivable 'systems' or 'networks of components with specified interconnections', only a very small discrete subset is of interest to engineers and the whole point of the question is to understand the special value of this infinitesimal fraction dotted among the amorphous masses"

Thus a *basic premise* is that the class of systems actually built are a very small subset of systems which could have been built in principle; *moreover*, this distiction can be captured mathematically. *Almost everybody will agree* to the first part of the premise readily and *many will be highly sceptical* of the second part! However, if one beleives that objective facts can be captured mathematically, one can hardly avoid the second part.

A.3. Artifacts are piecewise (almost) linear

My current position which is eloborated in the accompanying report is (one which most engineers will empathise with) that linear and mildly nonlinear systems are much easier to reason about and that artificial systems are piecewise (almost) linear in a very general sense.

To make this point in widest possible settings however one has to *relativise* and *generalise* the concept of linearity. The use of the mathematical theory of Categories is essential to achieve this - the *generalisation* process modelled by the transitions from *Linear* to *Abelian* to *Exact* to *Regular* Categories and the *relativisation* modelled by the concept of 'object *in* a category'.

A.4. Current work

Currently, I am working on the developemnt of the theory further for measuring the degree of 'non-linearity' and showing that this measure is related to the complexity of verifying properties of the system. A future goal of the theory is to give a foundation for the design CAD tools for computing 'Verification Metrics' and use these tools in a 'Design for Verifiability' Synthesis system.

B. Version - 2.

B.1. Abstract

It is intuitively obvious that Artifacts - Engineering systems, Mathematical proofs and Software - *actually* created by humans have some distinguishing features relative to objects of a similar nature which could have been created in *principle*. A clearer view of this distinction may be obtained by considering the following - There is nothing in the laws of Physics preventing the construction of systems which do **not** have well defined sub-systems. Similarly, laws of logic and the laws of computation do **not** require that proofs have lemmas and programs have a modular structure! But, as a matter of fact, whenever these artifacts are the result of human creation, they do *inevitably* possess these features. If the task of the Natural Sciences is to provide explanations of the form, structure and properties of naturally occurring objects starting from some basic laws, then, a science, playing a similar role with respect to artifacts, should be called the *Science of the Artificial*. H.A.Simon, in an eloquent booklet (The Sciences of the Artificial- MIT press, 1970) pointed out that such a science, comparable in depth to that of Natural Sciences, did not exist then. Nor, as far as I am aware, does it exist now.

In this report, I describe some of my efforts in this direction[14]. Analysis of the central concept of *Hierarchies* in turn brings out the importance of understanding the related concept of *auxiliary notions*. These are known by different names depending on the subject matter- *Definitions, Lemmas* (in mathematics), *Cuts, Types* (in logic and foundations), *Modules* (in software engineering) etc.. The search for the answers to the questions - *What are the desirable characteristics of formal frameworks within which we can address problems of such wide scope? Is there a suitably general and mathematical characterization of the concept of auxiliary notion? What exactly is its functional role?* - takes us through a tour of a conceptual terrain involving Category theory, Galois Theory, Theory of Descent, Topos Theory, Foundations of Mathematics and Algebraic Geometry, Axiomatic theory of Radicals, and topics from Theoretical computer science like proof systems, type theory and complexity theory. As a result of this 'tour', we are able to identify a set of well defined (sub) problareas for future research. Their well- definedness should enable us to address the basic questions in depth. Moreover, these problem areas have the pleasant feature of being simultaneously fundamental as well as having the potential (in view of the fact that they are about fundamental problems of *Engineering*) to be useful in practice.

[14] An preliminary talk on this topic was given to the members of the Categories and Combinatorics group, Univesity of Sydney, Nov 96.

An article, authored by E.W.Dijkstra, which also served as an inspiration for this work should be mentioned here. It is titled '*On a methodology of Design*'. It appeared slightly later than book of Simon (and in fact, refers to it) in the 'MC-25 Informatica Symposium' of the Mathematical Centre tracts series from Amsterdam. The notion that *both* mathematics and computer science belong to the species of the Sciences of the Artificial, and hence the *methodology* of mathematics, interpreted in a *general* sense as the methodology of *abstraction*, should be useful in Computer science, is stressed there. In **this** article, however, we have interpreted the phrase 'methodology' in a *technical* sense as the methodology of structural linearisation.

B.2. Table of contents

1. Hierarchies as generators of Crystalline
2. Auxiliary Concepts as generators of Hierarchies.
    1. The importance of being Non-transitive.

8. **Basic Question - What are 'auxiliary concepts' ?**

    1. Auxiliary Concepts in Logic
    2. Combinatorial models for interpolation
    3. Complexity of Mathematical concepts
    4. Concept modelling via distributive lattices
    5. Concept lattices

9. **Mathematics of Hierarchies**

    1. Galois Theory
        1. Classical Galois Theory
            1. Review of Classical Galois theory.
            2. Classical Galois theory - a Software Engineering view
        2. Generalised Galois theory-
            1. The Three facets - Classification, Correspondence and Solvability.
        3. Lower bounds, Recursion theory
        4. Abstract Galois Theory
        5. Galois Structure in Categories
        6. Universal Galois theory
    2. Radical Theory
        1. Axiomatising Radical and Semisimple classes
        2. Radicals and Interpolation
        3. Radicals and Verbals
    3. Graphic Toposes, Bands of Semigroups and Hierarchies
    4. Descent Theory
    5. Theory of Canopies
    6. Hierarchies in mathematics

10. **Why do auxiliary concepts 'exist' ?**

    1. The Complexity of Comprehension

2. Complexity Theory in CS.
    1. Review of Complexity theory.
    2. Suitability of current Complexity Theory to address the relevant issues
    3. Making semantics more intensional
    4. Complexity of Verifying Programs
3. The 'Complexity of Comprehension'= 'Complexity of Verifiability' assumption
4. The 'Mathematical structure implies Complexity' assumption
    1. Descriptive complexity Theory and Finite Model Theory
    2. Structure of Finite algebras
    3. Model Theory of Modules
    4. Stability Theory
    5. Homological methods in Model Theory
    6. Theory of ambiguities
5. Expressive power of Category Theory for structural questions
6. Unifying Galois and Kolomogorov

11. **Some CS instances of the theory**

    1. Extended Horn, G-n logics
    2. Until hierarchy
    3. Higher order algebras

12. **Applications**

    1. Galois theory in algebraic logics and applications to model checking
    2. Factorisation in the Category of Sketches
    3. Qualitative Reasoning in Design and rational design of Design Metrics.
    4. Design Space Explorer for VLSI.

13. **Conclusion**