A SWOT analysis of Instruction Sequence Theory

Jan A. Bergstra
Minstroom Research,* Utrecht

February 26, 2015

Abstract

After 15 years of development of instruction sequence theory (IST) writing a SWOT analysis about that project is long overdue. The paper provides a comprehensive SWOT analysis of IST based on a recent proposal concerning the terminology for the theory and applications of instruction sequences.

Contents

*Minstroom Research BV, Utrecht, The Netherlands (hereafter called MRbv), KvK nr. 59560347. Author's email address: info@minstroomreserch.org,janaldertb@gmail.com. Appendix A provides detailed statements concerning copyright protection of this document and about its status. This paper is a nopreprint in conformance with the definition given in Appendix B. MRbv nopreprint series nr. 6 (MRbv NPP#6). The paper has MRbv document class B (see Appendix C).

1 Introduction

As in [2] I will denote with instruction sequence theory (IST) the theory of instruction sequences based on the PGA approach to program algebra of [6]. I will freely make use of the terminology proposed in [3], which an interested reader probably should consult first, and in most cases I will use that terminology without further reference.

Thread algebra, (in [6] referred to as polarised process algebra) is not considered a part of IST. It is used in IST, however. In this paper I will make an attempt to provide a SWOT analysis of IST as it stands at this moment.

In the conclusions I will discuss to what extent the picture thus obtained can be helpful for the design of an agenda for the subsequent development of IST. In particular the SWOT analysis will prove helpful for separating work in the conventional scholarly manner aiming at peer reviewed publication from work in a more freely defined format. Judgements made in this SWOT analysis are my personal opinions, which cannot simply be taken for facts.

1.1 Structure of the paper

After an extensive description of the content of IST and of its mereologic relations with other subjects, I provide an assessment of the current state of affairs. Subsequently a survey is given of options for the valorisation of IST.

An agenda for work on that topic is listed. The agenda consists of two parts, work that is preferably performed from an academic affiliation and work that is more plausibly performed from a non-academic affiliation.

2 Survey of IST

Here is a listing of themes dealt with by IST. I have included proposals on how to view the relations that IST may have to other more clearly established research avenues in informatics. This should help to position IST clearly in the context of informaticology at large.

1. IST includes PGA style program algebra.
2. IST provides an approach, called "projection semantics", to program notation semantics.
3. IST provides a detailed theory of jumps and goto's (modelling labels as instructions as well) providing comparisons between different formats of these instructions. There seems to be no theoretical framework for these control mechanisms which is more inclusive than the framework provided by IST.
4. IST is included in algebraic algorithmics conceived broadly, following the terminology of G.E. Tseitlin in [25]. This inclusion was mentioned before in [2].
5. IST provides a theory of EXIP (exclusively imperative programming), a notion that was coined in [2].
6. IST is included in the theory of imperative programming and imperative software technology, which is included in the science of computer programming, which is included in computer science, which is included in informatics, which is included in the information sciences.
7. IST has been developed with at least the following design objectives in mind:
   (a) To provide a mathematical definition of the notion of a computer program which is useful both in practice and in education, and to have this definition rooted in the well-known (but mathematically imprecise) slogan that a program is a sequence of instructions. In particular IST provides:
      i. The notion of a program object (that is a finite or infinite single pass instruction sequence) which plays the role of a mathematical entity denoted by a program (qua program). The program object is to be distinguished from its meaning, which is a thread.
      ii. The notion of an instruction sequence expression which denotes a program object.
      iii. Awareness of the fact that one is very tempted to label an instruction sequence expression simply as an instruction sequence, thereby running into an inconsistency with the terminology just mentioned.

iv. An explanation of this inconsistency (ambiguous meaning for instruction sequence) in terms of preservationist paraconsistent reasoning using the chunck and permeate paradigm (see [23, 5, 2]).

(b) To define the simplest possible program notations and to define a hierarchy of increasingly more expressive program notations in a manner comparable with the ACP style family of process algebras (see [16]).

(c) To determine a mathematically clear notion of effectuation for programs (instruction sequences), including a general story of how a program interacts with its environment.

(d) To provide a model, or the basis for the development of a model, of software engineering, where software is replaced by inseqware (see [2]). This fictional brand of engineering might be tentatively be named inseqware engineering. The notion of engineering, however, seems not to have the flexibility of being practiced in a game-like manner (that is in a toy setting) rather than in reality with realistic objectives in mind.

At this stage I feel that each of these intentions, except item 7d has been dealt with in a satisfactory manner. Moreover I see the following state of affairs concerning the elusive notion of an algorithm:

(a) Because the core of IST provides a complete and coherent account of the foundations of imperative programming, IST provides a framework for investigating the concept of an algorithm. However, within IST we have not yet been successful in providing a convincing definition of the concept of an algorithm, see [12]. Perhaps this is a mission impossible, that remains to be seen.

(b) At this stage, the IST perspective suggests that algorithm may be a mathematical (or perhaps logical) concept which semantically denotes equivalence classes of instruction sequences for an appropriate equivalence relation.

(c) Each instruction sequence in such an equivalence class constitutes a formal implementation of that algorithm. Formal implementations may be implemented in practice by means of practical implementations written in known program notations aiming for effectuation on existing computing platforms.

(d) An algorithm may be understood as resulting from the instantiation of an algorithmic invention to specific parameter values (e.g. k and l in the notation of 13a below).

(e) An algorithmic invention stands to an algorithm that it realizes as an invention stands to a corresponding patent description. A formal implementation of an algorithm may be compared to a complete

technical specification of an application of that invention for the same purpose. A practical implementation of the algorithm may be compared to a real application of the invention.

This comparison is far from precise but it has an explanatory value in that it helps to make sense of the distinction of four levels of abstraction which seem to be needed when attempting to give an account of the concept of an algorithm: algorithmic idea, algorithm, formally implemented algorithm, and practically implemented algorithm.

8. IST provides a family of instruction sequence notations together with a bundle of mathematical/logical facts about relationships between these notations and about expressiveness of particular notations.

9. IST provides a mathematical framework for describing the interaction between a program (instruction sequence) under execution and its environment. This framework is based on service families (captured in a so called service family algebra), and three so-called instruction sequence processing operators: use (with some variations), apply, and reply.

10. IST makes use of a special purpose theory of processes named thread algebra. Thread extraction is the primary way to determine the meaning of an instruction sequence. Strategic interleaving (rather than arbitrary interleaving) constitutes the mechanism for dealing with multi-threading which is considered to constitute the most plausible form of concurrency in the context of instruction sequences.

    Thread algebra can be developed independently from IST and is not considered a part of it, although it has been developed with application within IST in mind.

11. Following the terminology of [3], IST provides four special cases for EXIP which have been investigated in some detail already and which seem to justify much additional work: combinational instruction sequencing (CISg), objectct-oriented instruction sequencing (OOISg), reflective instruction sequencing (RISg), instruction sequencing for Maurer Machines (ISgMM). This terminology was proposed in [2]). These specific cases of instruction sequencing will be discussed in mored detail in subsequent items below.

    The key difference between these four instances lies in the services from which the operating context of an instruction sequence is made up, Boolean registers, data linkages, cellular memory network structures (with stack and Turing tape as prime examples), and Maurer machines (with or without a processor area).

12. CISg, OOISg, RISg, and ISgMM each use the same family of instruction sequence notations, the same theory of service families for linking an instruction sequence with its operating environment, the same definitions of instruction sequence processing operators, and finally the same definitions for threads, multi-threads, thread algebra, and thread extraction. These

similarities are apparent in the uniformity of the axioms that specify the key components of the setup.

Although it seems justified to view the approach of IST in these cases as an axiomatic approach, the additional correspondence in dealing with operational (and for that reason practical) aspects brings me to proposing IST in these cases as a praxiomatic approach, rather than as an axiomatic approach.

13. IST provides a theory of instruction sequence size complexity which allows the determination of precise lower bounds for very elementary algorithmic problems. A fundamental case allows for the use of Boolean registers only. That case is called the combinational case and it allows for a practically unlimited volume of questions of the following forms:

    (a) Given a definition of a function $F_{k,l}$ from k Booleans to l Booleans (with specific natural numbers k and l), and an instruction sequence X that computes $F_{k.l}$, together with some validation of the assertion that $X_{k,l}$ implements $F_{k,l}$ consider the following questions:

        i. Provide an assessment of the given validation of the assertion that $X_{k,l}$ implements $F_{k,l}$.

        ii. When possible (or necessary) in the light of the above assessment. Provide improved validation of the assertion that $X_{k,l}$ implements $F_{k,l}$.

        iii. In case of doubt: Find a counter-example to the mentioned valid implementation assertion.

        iv. Find a shorter (in terms of the number of instructions) instruction sequence Y (over the same interface of basic instructions, and allowing the same control instructions) that implements $F_{k,l}$.

        v. Or else, demonstrate that no shorter implementation of $F_{k,l}$ exists.

        vi. Suppose that an additional control mechanism is permitted (for instance backward jump instructions where $X_{k,l}$, Y are supposed to consist of finite simple pass instruction sequences only, that is no backward jumps, repetitions, or even recursion may occur), is it possible to find a shorter implementation in terms of the more inclusive instruction sequence notation admitting the mentioned control mechanism.

    (b) Given $X_{k,l}$ that implements $F_{k,l}$ (as mentioned above). Can one extract an "algorithmic invention" from this implementation and generalize that to the case of arbitrary k and l and find a uniform notation for implementations of the same "algorithmic invention". Here I adopt the viewpoint that was put forward in [12] that (in the case of finitely Boolean functions) "algorithm" is a technical concept that allows for a technical definition (which has not yet been discovered,

however) and I assume that algorithms implement algorithmic inventions the latter not being technical in nature.

(c) If only a non-algorithmic definition of $F_{k,l}$ is given (for instance $F_{k,l}$ is supposed to provide an inverse of SHA256, as specified in [10]), the following questions arise:

i. Provide an instruction sequence expression that implements $F_{k,l}$.

ii. Provide a (notation for) finite single pass instruction sequence that implements $F_{k,l}$.

iii. Now repeat the questions under 13a above.

(To motivate the relevance of this type of question: proving that SHA256 is cryptographically secure amounts to achieving a successful analysis of this kind.)

14. In the setting of OOIGs IST provides a detailed theory of the semantics of garbage collection and introduces shedding as an enhancement of full garbage collection. Shedding is a memory management technique of conceptual relevance at least. IST provides a model for so-called data linkage programming, which captures some essential aspects of object orientation and dynamic data structures in an imperative framework.

    IST provides a very flexible and precise setting in which different garbage collection mechanisms can be formally differentiated and in which questions about the computational power of garbage collection mechanisms can be posed and answered.

15. In the setting of RISg, IST provides a detailed theory of metaprogramming for instruction sequences. Impossibility results such as the unsolvability of the halting problem, or the alleged impossibility of virus detection, are cast in a general framework allowing the analysis of variations of such problems.

    In my view IST provides the most general and flexible framework for investigating these issues available at this moment.

3 SWOT analysis of IST

An assessment of the state of affairs of IST results from combining how it looks from outside and how it looks from inside. I will use the format of a SWOT (strengths, weaknesses, opportunities, and threats) analysis to arrange the items of the assessment.

3.1 Strengths of IST

1. IST provides a reasonable definition of the notion of a program.[1]

[1] Such definitions are very rare in the literature on computer programming, as if this core notion of informatics is not worth systematic attention, and in most cases definitions clearly

2. IST provides a theory of imperative programming which is developed from first principles in such a way that it can be taught as a fits course in computer science. Experiences with that form of teaching have been quite positive. IST is sufficiently rich to provide material for more advanced courses.

3. IST primitives are independent of any particular data structure so that IST instruction sequences are very general and uniform. Service families feature a practical algebraic structure and the instruction sequence processing operators capture more directly the variety of meanings of an instruction sequence than a conventional operational semantics. Use, apply, and reply, provide three different and fundamental views on what a sequential program may deliver.

4. IST provides an attractive notion of instruction sequence size complexity which can be used to determine precise lower bounds for some, admittedly quite simple, computational problems.

5. IST allows a reasonably sophisticated, though still inconclusive analysis of the notion of an algorithm.

6. The four specialized instances of IST that have been developed thus far each provide new insights in the respective areas:

   CISg: A very comprehensible description of non-uniform complexity, and the following facts:

   (a) the fact that instruction sequences for computing parity profit in terms of length from the presence of an auxiliary boolean register,

   (b) a proposal for the/a notion of algorithmic equivalence, (or at least a suggestion on where to look for such a definition)

   (c) the proof that for multiplication of n-bit natural numbers the use of repetition allows a shorter representation of the so-called long multiplication finite algorithm than the use of single pass instruction sequences.

   OOISg: The insight that there seems not to be any formal definition of the state of affairs that for some instruction sequence X over a finite datalinkage, garbage collection provides a useful contribution to its successful effectuation. And technically:

   (a) the observation that declaring unreachable objects garbage, and having these garbage collected immediately thereafter as a consequence, has paradoxical consequences in a finite object pool.

have not been written with the idea in mind that it really matters what is written. Definitions stating that a program is a sequence of instructions without paying any attention to the notion of an instruction are of that kind. Literally an instruction is an element of a sequence so the definition is a pleonasm without further explanation of both sequence and instruction.

(b) The novel concept of shedding which resolves that paradox.

RILg: Fore metaprogramming the following insight came about:

(a) the fact that the diagonal proof of the unsolvability of the halting problem requires only a modest menu of primitives so that it may work in an environment where, seen from outside, the halting problem is computably solvable,

(b) the fact that in some contexts (that is basic instruction interfaces for a TM tape like data structure), instruction sequences are sufficiently expressive to solve their own halting problem, (that is HP is autosolvable, a novel notion as well),

(c) the observation that proof of the well-known computational impossibility of virus detection does not carry over to finite state systems,

IGsMM: Uncovering the remarkable flexibility of Maurer's machine model, which, in the real world of finite state systems, seems to be more convincing than Turing machines. Technically:

(a) a proof that larger machines are computationally stronger;

(b) a proof that the size of a CPU has impact on the expressiveness of a machine;

(c) a proof that pipelining improves the speed of execution of an instruction sequences, based on a general model of concurrent processing;

(d) a detailed analysis of the (limited) power of micro threading.

7. In all of its parts IST leaves open many detailed questions about which research may lead to useful results. This leads to a very substantial research agenda, which is a definite asset. It also requires making well-motivated choices when embarking on new projects in the context of IST.

3.2 Weaknesses of IST

When contemplating future progress on IST it us useful to have a critical perspective at hand. Here are some aspects that might shed doubt on the value and the potential of IST.

Just another program notation. Although the basics of IST are meant to provide conceptual primitives which might have the same absolute or language independent status as those of propositional logic, the fact that a notation had to be shown may suggest that IST merely provides another program notation which moreover, viewed from a practical perspective, is rather useless.[2]

[2]Remarkably this weakness suggests that it is eventually not an optimal strategy to have a standardised notation, and that only by allowing variable notations giving, the impression that we are working on a specific program notation can be avoided.

Invisibility. Given the number of published papers on IST thus far, the number (and quality) of external citations to the work on IST is remarkably low. As consequences (or signals) of that state of affairs we find that:

1. References to work in IST are made only by the members of a small team of persons belonging or or directly connected with the Section Theory of Computer Science of the University of Amsterdam (formerly called the Programming Research group).[3]
2. Ownership of IST is very local, no conferences are devoted to it, no social structure has been developed around it thus far. The parochial outlook of IST may increasingly worry potential readers, and may constitute a negative incentive for studying the details of works on IST.
3. No former PhD students play the role of advocates of IST content and approach.

Lack of external funding. Attempts to acquire funding for IST within the Dutch system of research funding have systematically failed.[4]

Lack of status. IST is not a part of the well-known framework of approaches in the science of computer programming that get invited for key-notes and so on. It is very difficult to argue for the quality of the work on IST on the basis of conventional performance metrics and performance indicators.

Old style. By paying attention to jumps and goto's the IST approach qualifies as old-fashioned rather than as innovative. Putting IST in the tradition of algebraic algorithmics, which emerged from classical schematology potentially reinforces this non-forward looking appearance.

Not at home in informatics. Questions like "what is a program", "what is program testing", or "what is an algorithm" don't get much attention in current computer science. It is not even clear that informatics (here assumed to include computer science) provides an adequate "home" for this kind of work.

Theoretical simplicity. Most results in IST that were obtained thus far are quite simple in mathematical terms. I see this as a definite strength but mainstream TCS (Theory of Computer Science) has different preferences

[3] Besides myself these persons are: Inge Bethke, Bob Diertens, Marijke Loots, Kees Middelburg, Alban Ponse, Thuy Dong Vu, and Pum Walters. Chris Jesshope wrote some papers that apply aspects of IST (in particular thread algebra). The initial design of PGA evolved from joint work with Bas van Vlijmen on "Theoretische Software Engineering". A number of BSc and MSc students including Stephan Schroevers and Lars Wortel (both of whom published their MSc thesis on `arXiv`) wrote a thesis (BSc or MSc) under the supervision of either Inge Bethke, Bob Diertens, or Alban Ponse. I should mention that in many occasions we received very detailed and informative reviews upon Journal submission. These reviews were always anonymous and have been extremely helpful in the majority of cases.

[4] However, it is fair to make mention of a grant on thread algebra form NWO that funded the writing of the PhD thesis of Thuy Dong Vu.

so it seems. At the same time papers on IST are not always very accessible. Unfortunately, writing down what is in essence simple in a simple manner is not always simple.

Conceptual only. The purpose of IST is not that it finds immediate applications in hardware technology of in software technology. The purpose is to obtain conceptual clarity in areas where that is missing (in the perception of the authors of papers on IST).

Algorithm unexplained. The initial hope that IST would be helpful in a significant manner to explain the concept of an algorithm, (a question that seems currently to stand out as an open problem) has not materialized. But moderate progress can be reported, which may be a positive state of affairs after all.

3.2.1 Which positive aspects balance the negative ones (if any)?

The listing of weaknesses in Paragraph 3.2 above is rather disappointing but that is only part of the story. Is there any other explanation of the problematic performance of IST as a research project other than that it simply is of poor quality. And indeed, a different way of dealing with the mentioned facts is an option. In spite of the fact that this aspect lies outside the framework of a SWOT analysis I want to insert some observations concerning this alternative view.

A definite advantage of the mentioned state of affairs is that the small team working on IST has had the opportunity to develop these ideas for some 15 years which made it possible to slowly develop solutions for design problems which did not come easily.

Personally I prefer not to work under competitive pressure, in particular competitive time pressure. Similarly I find no pleasure in being successful in conferences and journals with high rejection rates. Usually I do not remotely share the seemingly objective value judgements that drive such schemes.

In fact as soon as I know that other people are working on a problem I start to look at other issues that may yet have been overlooked. These matters are entirely personal. For me research is about being somewhere where others have not yet been for whatever reason. The very fact that a topic is not considered to be challenging within a community which on the basis of content would be a most plausible community to pay attention to it, constitutes a major explanation of why other researchers are not on the same spot (that is working on the mentioned topic). Peer review of research proposals works against this attitude and style of work in a phenomenal manner.

Returning to the notion of competition: being a better performer in a task or a challenge that appeals to many, is only one form of competitiveness, being alone on a path that nearly all others neglect just as well manifests a form of competitive behaviour, though of a different kind: does one have the stability and strategic guidance to proceed on a long path without the interference and implicit control of a community of scholarly peers. That is not so easy, and it

is risky. One may simply fail ever to attract any attention from outside one's own microcosmos of local colleagues working on the same topics.

Remarkably, and for me somewhat unexpectedly, having worked for many years on IST without any tangible incentives from the scientific system produces a real source of pride, sense of ownership, and to some extent an awareness of responsibility.

3.2.2 Advantages of doing the work without project funding

There is a further aspect to this matter which may be personal and rather ad hoc. It is an observation in hindsight, of which I became aware only slowly. Getting your own story ("you" referring to a small research team) on paper requires time. For several steps in the development of IST a two stage process was needed, years after an initial result was obtained the same issue was revisited and a better and more useful solution to the same issue was found. Here are some specific examples:

1. Data linkages (and OOIDg) of [11] improves its precursor "molecular dynamics" of [4].

2. The first paper on RISg ([19]) was followed by a technically more systematic approach to the same issues in [9].

3. Service families and service family algebra ([9]) significantly improved its precursor in [17].

4. It took us 15 years to become aware of the technical importance of complementation as a method on a Boolean register (see [13] where a result depends on the availability of complementation) and to determine the full menu of such methods ([14]).

5. Only recently we understood the following points:

   (a) the very concept of an algorithm has a remarkably unsettled status in the informatics literature;

   (b) Robin Milner tried to define the concept of an algorithm and came up with the essential observation that most plausibly an algorithm is an equivalence class of programs;

   (c) thereafter Milner noticed that he failed in achieving his goal (of defining algorithmic equivalence) in that manner and subsequently discovered that some additional degree of abstraction creates a notion of behavioural equivalence which provides the basis for process algebra in general and CCS in particular;

   (d) IST is a useful tool for those who plan to continue where Milner "gave up" (which includes myself).

> The importance of these observations for IST is this: perhaps the questions "what is a program" and "what is an algorithm" must be posed together right from the start of the story. The dependence of the concept of an algorithm on the concept of a program, together with the fundamental uncertainty as to whether or not algorithm is a notion with a mathematical meaning may strengthen the wish to provide a mathematically reliable definition of the concept of program. It follows that even obtaining a stable start of the IST story from first principles may still lie ahead of us.

Now the critical point is this: had we obtained funding for a stream of young and efficiently working PhD students and postdocs then we could not possibly have been making these second steps ourselves. But having the time and stamina to get to these often hard to find improvements is a very significant reward by itself irrespective of any lack of recognition.[5]

In addition, once independent and highly creative researchers get on the move (say in IST) the task to properly incorporate their results in one's own theory development process is far from straightforward.

3.3 Opportunities

Opportunities for IST are options for additional future work with a significant chance of success, in particular involving work will probably profit decisively from what has been already achieved in IST.

3.3.1 Dissemination opportunities

There is a lack of easily accessible introductions and surveys to IST, this lack constitutes a weakness and an opportunity at the same time.

3.3.2 Research opportunities I

CISg. Both instruction sequence size complexity, and further work on algorithmic equivalence constitute challenges that qualify as significant opportunities.

The plausibility of this preference relative to the many other options is motivated by (i) this perspective providing a unique path for making progress on practical complexity problems that arise in the field of cryptography and information security, and (ii) CISg providing thus far reasonable opportunities for getting one's results peer reviewed and scholarly published.

ISgM (ISg methodology). Instruction sequencing methodology as a theme in which models of software development are investigated. This work can start in a non-academic manner, based on OOISg, say from the platform

[5] This reminds me of the paradox of mountaineering which I heard an English mountaineer formulate on TV many years ago: "we all agree that the mountains must be climbed, but we want to do it ourselves."

of MRbv, (the outcome of which can be properly documented in a preliminary manner by way of a series of nopreprints published on `viXra.org`) and may eventually progress to a degree of maturity that merits scholarly publication.

The following topics might be considered under this heading:

1. What is testing? How to provide a formally valid description of testing that takes into account that a test is a physical experiment which takes place in time and space.
2. What decision taking threads are in place in advance of taking the decision that an instruction sequence may be used for some given purpose (by some given operator, in some (perhaps partially) given environment, under some given conditions). How do verification, testing, and other means of validation, feature in this brand of decision taking?
3. How to describe relevant levels of competence and experience for single instruction sequence developers, as well as for teams of instruction sequencing engineers.
4. How to model multi-threaded teamwork involved in the production of polyadic instruction sequences that are too large to be constructed by a single programmer?

ISWE (Inseqware engineering). For ISWE a similar opportunity exists as for ISgM.

ISgM and ISWE mentioned above are plausibly studied in a non-academic environment. There are several arguments for that viewpoint:

1. no mathematically provable results are expected,
2. the main issue is in both cases how individual agents and teams of agents might go about performing certain tasks,
3. the work is neither predictive nor normative, but suggestive instead, indicating how certain issues might be approached, rather than validating the expectations for ways to approach an issue,
4. a preference for the relative methodological freedom of nonacademic work may turn out to be temporary only and it may change once a sufficient amount of ground work has been achieved,

3.3.3 Research opportunities II

Many technical open problems have been formulated in the papers on IST that were written thus far. Each of these constitutes an opportunity for further work. The main obstacle is that most remaining technical problems seem to be quite difficult, as easy problems were solved first. Here I will list some issues that

might be easy in the sense that mathematical problem solving is not the main obstacle.

What is an algorithm? Suppose that a dichotomy is introduced as follows: A) there will be a definition of algorithmic equivalence which is found within reasonable time (the approach that IST provides for this matter has been explicitly listed as a result of IST above), versus B) the assumption that algorithm is a mathematical concept that allows formalisation by way of a notion of algorithmic equivalence is denied, and a theory of algorithms is based on the negation of that assumption.

Then algorithmic equivalence becomes a matter which must be dealt with by legal and procedural means. Competent bodies need to answer specific questions about algorithmic equivalence as a service to the inseqware engineering community. This service aims at conflict resolution comparable to the legal processes around patents and IP in general. How such bodies may work (in connection with algorithms) and what the scope and impact of that work might be is a topic that may still profit from an initial drastic reduction of the scope of inseqware functionality and the availability of candidate definitions of algorithmic equivalence equivalence as in [12].

Designing alternative syntax for instruction sequences. Program algebra and IST in general are independent of the specific notations used. This implies that designing alternative syntax which is dedicated for specific purposes and contexts is useful. It is conceivable that the syntactic uniformity within CISg, OOISg, RISg, and ISgM can be "broken" in a useful manner.

Theory of computer viruses. RISg provides an angle on computer viruses which is awaiting further exploitation. It is plausible that a philosophical/qualitative approach me be meaningful, and it may be profitably connected with the qualitative approach to security issues from [7, 8].

3.3.4 Valorisation perspectives for IST

Viewed as a package of knowledge the question may be posed to what extent IST may admit valorisation. There is no protected IP about IST other than publications with various forms of intellectual ownership protection in an academic sense and corresponding copyright protection. An open source toolkit for a range of PGA instruction sequence notations has been developed by Bob Diertens (Informatics Institute, University of Amsterdam). Here are some possible perspectives.

Teaching. The development and delivery of educational courses and corresponding educational material about the basics of computer programming. This may take place at all levels of teaching from early primary school to academic graduates and PhD's.

Research on focus areas. Further development of CISg, OOISg, RISg, and or ISgMM to research agenda's sufficiently strong to attract research funding, preferably from industrial sources.

Praxiomatic game development and delivery. Developing CISg into a what I will call a praxiomatic construction game (PCG). A PCG is a construction game based on a praxiomatic theory, or stated differently, based on a praxiomatic approach to its foundational theory. The PCG might be viewed as a para-scientific activity in the following sense: (a) it embodies a collective effort to gain knowledge and to document both novel knowledge as well as the historic path towards its finding including proper attribution to those who made significant contributions, but (b) this knowledge is not framed and charted in terms of intellectual importance the way scientific outcomes commonly are.[6]

From the perspective of MRbv the priority is different. From that perspective my preference is making progress along the software engineering research direction which proceeds from OOISg. That work is unlikely to lead to publishable work, while many aspects need thorough investigation the outcome of which can be properly documented in a preliminary manner by way of a series of nopreprints published on `viXra.org`.

3.4 Threats

The survey of threats lacks structure. I will simply list the threats that I am aware of at this moment.

1. Different approaches to theorizing about the foundations of programming may be so distant form IST that not even the slightest form of recognition of the existence of IST by authors from other schools and approaches may materialize in say the next 15 years. This risk is far from negligible, and although it is disappointing and not so easy to understand, it needs to be taken into account as a most obvious treat.

   It is remarkable that sustained presence in scholarly journals does not lead to the remotest sign of recognition of the existence of the IST activity from other research groups. This I find somewhat disturbing and I am as yet not willing to view this as "being our own fault" only. A remarkably parochial attitude seems to have captured TCS quite widely.

2. The primary threat for IST is that the team which prefers the IST research at the moment dissolves via retirement or illness or the pressure of taking on board other tasks in part due to the mentioned lack of external funding.

[6] This situation is comparable to information gathering about chess: for some reason findings in chess (e.g. winning positions in end games) are not considered a part of science (discrete mathematics) although such findings definitely belong to finite combinatorics. Finding complex positions in chess with a determinate outcome qualifies as a praxiomatic construction game.

3. Lack of a clear image. By writing too many papers in too many different directions it may become prohibitively costly for an outsider to enter this activity and to become a novel participant to it.

4. Lack of clear naming. IST has been worked at under different names which is unhelpful for an external observer in search of a coherent picture. Here are some names that were used:

   Program algebra. This phase has invariably been used in combination with and with reference to the axiom system PGA and it extension for structural congruence.

   A definite disadvantage of the phrase program algebra is that PGA is only one of its many possible designs. Another disadvantage has been that it fails to cover quite some part of IST.

   Thread algebra. Thread algebra has been put forward as a an alternative for process algebra in the case of multi-threading, with an emphasis on strategic interleaving instead of arbitrary interleaving. The phrase thread algebra is satisfactory in that it creates no confusion and adequately names a topic of limited size and scope. Thread algebra might be viewed as a special case of process algebra in the sense that most thread algebras can be modeled as definable substructures of appropriate process algebras. Currently I don't subsume thread algebra under IST because it can stand on its own feet without a basis in terms of instruction sequences.[7]

   Projection semantics. The approach to program notation semantics that comes with program algebra is called projection semantics. That phrase is capable of carrying quite some weight just like denotational semantics, operational semantics, and axiomatic semantics. The phrase is satisfactory but the topic has ceased to draw much attention in our own work. Clearly much of IST cannot be subsumed under projection semantics.

   Instruction sequence theory. This phrase has proven quite useful in cases where the algebraic structure of some specific collection of instruction sequences played no significant role. A weakness is that "instruction sequences" seems not to work as an abbreviation of instruction sequence theory while the "theory" part stands in the way of incorporating more practical objectives. Another weakness is that it is rather unspecific a about what kind of theory might be intended.

   Instruction sequence algebra. This phrase solves both the difficulty that program algebra is to wide and that instruction sequence theory is too vague. Using this phrase does not help in cases where the algebraic structure of instruction sequences plays no role.

[7] Originally thread algebra was referred to as polarised process algebra. That is semantically fine, but it fails to express that thread algebra is a significant simplification of process algebra rather than the other way around.

Instruction sequence size complexity. This phrase is clear and adequate for what it refers too, but it is obviously too narrow for representing all of IST.

Data linkage programming. This indicates the theory of imperative programming by means of instruction sequences over finite and infinite data linkages. The phrase is adequate but of fairly limited scope.

Here are some candidate names for IST that I have rejected and some reasons why:

Instruction sequence engineering. Too serious for what is rather a toy programming exercise.

Instruction sequencing. Too much effort on the active form while explanation and understanding are of equal if not greater importance.

Instruction sequence size complexity theory. Too limited.

Axiomatic instruction sequence engineering. Seemingly inconsistent as engineering can hardly be axiomatic. The engineering attribute is very useful as it provides long lasting scope for the inclusion of aspects that are often forgotten in theories of programming.

Foundations of imperative programming. The phrase is in principle valid, but the foundational aspect will stand in the way of valorisation on the long run.

Principles of imperative programming. IST may not live up to this claim, its subject matter may be too limited and to specialized.

Basics of imperative programming. This name will carry too much of the suggestion that by leaning about IST one's imperative programming skills will be elevated to a level of some reasonable competence. That elevation is not intended and it is probably not taking place either, however.

Praxiomatic instruction sequence development. This is a conceivable option, though the virtually unknown term praxiomatic constitutes a definite disadvantage and a risk.

The viewpoint that a new name for a collection of topics temporarily named as IST is needed emerges from a non-academic perspective rather than from an academic perspective. Indeed, each topic that is best studied in the tradition of conventional academic research has been labeled with a plausible name. The wish to specify an extension of the range of themes with themes that are assumed to be more appropriate for a non-academic approach that provides the incentive to look for more inclusive naming. There is an asymmetry implicit in this issue: reliably speaking about academic activity and results from academic research from a non-academic perspective is much more straightforward than speaking reliably about non-academic work from an academic perspective.

5. Another threat is that the IST team changes its research priorities and places its focus somewhere else. The main alternative for the team is moving its focus entirely towards the theory of meadows (e.g. [22, 20, 15]), and its container project on arithmetical datatypes based on [21].

6. Maintenance and further development of the PGA ToolKit may come to a halt due to a lack of continuity of UvA internal funding for both activities.

7. IST provides the basis for a course in the BSc informatics at UvA, that position is precious and may get lost.

4 Concluding remarks

Based on a survey of work that has been done on PGA style program algebra and the related theory of instructions five subthemes have been specified and named and an an overal name for the theme has been proposed. For two of the five subthemes it is argued that investigating the subtheme may be more plausibly carried out as a non-academic valorisation activity, while the other three subthemes are more plausibly studied within the conventional methods and paradigms of academic research.

References

[1] J.A. Bergstra. Putting instruction sequences into effect. `arXiv:1110.1866 [cs.PL]`, (2011).

[2] J.A. Bergstra. A nopreprint on algebraic algorithmics: paraconsistency as an afterthought. `viXra:1501.0203.1866`, (2015).

[3] J.A. Bergstra. A terminology for instruction sequencing. `viXra:1502.0228`, (2015).

[4] J.A. Bergstra and I. Bethke. Molecular dynamics. J. Logic and Algebraic Programming 51 (2), 193–214, (2002).

[5] J.A. Bergstra and I. Bethke. Note on paraconsistency and the logic of fractions. `arXiv:1410.8692[cs.LO]`, (2014).

[6] J.A. Bergstra and M.E. Loots. Program algebra for sequential code. *Journal of Logic and Algebraic Programming*, 51 (2): 125-156, (2002).

[7] Jan A. Bergstra and Karl de Leeuw. Bitcoin and Beyond: Exclusively Informational Money. `arXiv:1304.4758v2 [cs.CY]` (2013).

[8] Jan A. Bergstra and Karl de Leeuw. Questions related to Bitcoin and other Informational Money. `arXiv:1305.5956v2 [cs.CY]` (2013).

[9] J.A. Bergstra and C.A. Middelburg. Instruction sequence processing operators. *Acta Informatica*, 49: 139-172, (2012).

[10] J.A. Bergstra and C.A. Middelburg. Instruction sequence expressions for the secure hash algorithm SHA-256. `arXiv:1308.0219[cs.PL]`, (2013).

[11] J.A. Bergstra and C.A. Middelburg. Data linkage algebra, data linkage dynamics, and priority rewriting. *Fundamenta Informaticae*, 128 (4): 367–412, (2013).

[12] J.A. Bergstra and C.A. Middelburg. On algorithmic equivalence of instruction sequences for computing bit string functions. `arXiv:1402.4950[cs.LO]`, (2014).

[13] J.A. Bergstra and C.A. Middelburg. Instruction sequence size complexity for parity. `arXiv:1412.6787[cs.CC]`, (2014).

[14] J.A. Bergstra and C.A. Middelburg. On instruction sets for Boolean registers in program algebra. `arXiv:1502.00238[cs.PL]`, (2015).

[15] J.A. Bergstra and C.A. Middelburg. Division by zero in non-involutive meadows. *Journal of Applied Logic*, 13(1): 1–12 (2015).

[16] J.A. Bergstra and A. Ponse. Register machine based processes. *J. ACM*, 48 (6): 1207–1241, (2001).

[17] J.A. Bergstra and A. Ponse. Combining programs and state machines. *Journal of Logic and Algebraic Programming*, 51 (2): 175–192, (2002).

[18] J.A. Bergstra and A. Ponse. An instruction sequence semigroup with involutive anti-automorphisms. *Scientific Annals of Computer Science*, 19: 57–92, (2009).

[19] J.A. Bergstra and A. Ponse. Execution architectures for program algebra. *J. Applied Logic*, 5 (1): 170–192, (2007).

[20] J.A. Bergstra and A. Ponse. Division by zero in common meadows. `arXiv:1406.6878 [math.RA]`, (2014).

[21] J.A. Bergstra, and J.V. Tucker. Equational specifications, complete term rewriting systems, and computable and semicomputable algebras. *Journal of the ACM,* 42(6):1194–1230, (1995).

[22] J.A. Bergstra and J.V. Tucker. The rational numbers as an abstract data type. *Journal of the ACM*, 54 (2), Article 7 (2007).

[23] B. Brown and G. Priest. Chunk and Permeate, a paraconsistent inference strategy. Part I: the infinitesimal calculus. *Journal of Philosophical Logic*, 33:379–388 (2004).

[24] A. Doroshenko, G. Tseitlin, O. Yatsenko and L. Zacharyia. A theory of clones and formalised design of programs. *in: Proc. Int. Conf. "Concurrency, Specification, and Programming" (CS&P) 2006*, 27–29, (2006).

[25] G.E. Tseitlin. Algebraic algorithmics: theory and applications. *Cybernetics and System Analysis*, 39 (1), 1–15, (2014).

A Properties of this particular paper

The first Appendix contains information which is specific for this paper, the subsequent Appendices provide the necessary explanation.

A.1 Licencing

A.2 MRbv Nopreprint Series Number

This is #6 from the MRbv Nopreprint Series (in brief MRbv NPP#6). The other papers in the series are:

1. MRbv NPP#1: "Decision taking avoiding agency", `http://vixra.org/abs/1501.0088` (2015); this paper is not explicitly labeled as a nopreprint but it sufficiently meets the criteria as listed below, though it lacks a defensive novelty analysis which admittedly is a deficiency,

2. MRbv NPP#2: "A nopreprint on algebraic algorithmics: paraconsistency as a afterthought", `http://vixra.org/abs/1501.0203` (2015),

3. MRbv NPP#3: "A nopreprint on the pragmatic logic of fractions", `http://vixra.org/abs/1501.0231` (2015),

4. MRbv NPP#4: "Personal multithreading, account snippet proposals and missing account indications", `http://vixra.org/abs/1502.0204` (2015).

5. MRbv NPP#5: "Terminology for instruction sequencing", `http://vixra.org/abs/1502.0204` (2015).

A.3 MRbv Document Class

This paper has MRbv document class B in the MRbv Document classification scheme (MRbv DCS). This scheme is detailed in Appendix C. This classification refers to the body of the paper with the exclusion of the Appendices.

A.3.1 Justification of the MRbv document classification

In this particular case the classification in class B has the following motivation:

1. There is no immediate or even intended vision of application or valorisation of the content of this nopreprint. (This rules out categories C and D). The content consists of proposals only which still need to be accepted by being incorporated in a subsequent version of the accounts on PMTh.

2. The terminology proposed in the paper wil be used as a point of departure for further work on instruction sequences and instruction sequence theory within MRbv. There is a commitment from MRbv to al design decisions involved. This indicates class B.

3. The nopreprint status is intentional, submission to a (selectively) peer reviewed publication outlet is not intended. (This indicates MRbv as an appropriate affiliation bringing with the need for classification in A B, C, or D). Agreement of any peer review system with the design decisions in the paper is not sought.

4. Subsequent academic research on the basis of the content of this work is not foreseen by the author. Subsequent non-academic research, however, is expected and intended.

A.4 Defensive novelty analysis

A nopreprint ought to be equipped with a so-called defensive novelty analysis. An explanation of this this notion as well as an explanation of why it is needed in the case of a nopreprint is given in Appendix B below). For this paper I put forward the following arguments:

- The choice of acronyms Ing, CINg, OOINg, INgMM, INgM, IS:T&A is not amenable for criticism by a reviewer. Most acronyms have multiple uses and to the best of my knowledge each of these has other uses as well, though not in the context of informatics and in particular not in the context of software development.

- For new terminology (e.g. inseq, insets, inseqware) by means of appropriate queries in several search engines it has been secured that the proposed use of these terms does not adversely interfere with know use (if any).

- The paper provides a design of new terminology in the context of existing terminology. In practice each of the terms and phrases needs to stand on its own feet, however. The mere fact that a term or phrase fits in an overall design of terminology cannot compensate for problems encountered with its use. Thus there is no substance in the paper other than that it provides a collection of non overlapping and mutually consistent proposals on terms and phrases for particular concepts and notions. Each of these terms and phrases needs to survive in future use on its own, which may

or may not be the case. Approval of the overal design by any other agent, reader, or body is not sought, and would not contribute to the defensibility of individual terms and phrases as proposed in the paper.

B Formalities and policy statements I: about nopreprints

This Appendix begins with brief historical remarks concerning the possibly novel ideas that are put forward in this Appendix as well as and in the following Appendix. The remaining part of this Appendix spells out the details an rational of nopreprints as a novel class of papers and publications.

B.1 Remarks on micro-history

The development of the concept of a nopreprint document category as well as of of the MRbv document classification scheme including the form of presentation of such matters in appendices of MRbv nopreprints steadily evolves.

This Appendix and the following Appendix constitute a minor adaptation of essentially the same content that was included in the Appendices B and C of MRbv NPP#4 (http://vixra.org/abs/1502.0204), which in turn derives from the Appendices of two earlier nopreprints (MRbv NPP#2 and MRbv NPP#3) that were posted as `http://vixra.org/abs/1501.0231` and `http://vixra.org/abs/1501.0203` respectively, which in turn have been derived from the final Section of MRbv NPP#1 (which is `http://vixra.org/abs/1501.0088`).

I apologise for the length of these considerations. I will include similar texts in further documents (either having nopreprint status or written from my MRbv affiliation) expecting that some gradual evolution to a mature, stable and compact form wil result in due time.

B.2 Nopreprints and micro-institutions

This Paragraph and subsequent Paragraphs with are identical to the corresponding Paragraphs of [3], and are not repeated here for that reason.

C Formalities and policy statements 2: using a private micro-institution as an affiliation

This Appendix is identical to Appendix C of [3], it will not be repeated here for that reason.