New Generation Computing (2019) 37:141–158
https://doi.org/10.1007/s00354-019-00058-y

From Fifth Generation Computing to Skill Science

A Biographical Essay of Koichi Furukawa

Tomonobu Ozaki[1] · **Randy Goebel**[2] · **Katsumi Inoue**[3,4,5]

Received: 7 February 2019 / Accepted: 10 April 2019 / Published online: 25 April 2019

Abstract
Professor Koichi Furukawa, an eminent computer scientist and former Editor-in-Chief of the *New Generation Computing* journal, passed away on January 31, 2017. His passing was a surprise, and we were all shocked and saddened by the news. To remember the deceased, this article reviews the great career and contributions of Professor Koichi Furukawa, focusing on his research activities on the foundation and application of logic programming. Professor Furukawa had both a deep understanding and broad impact on logic programming, and he was always gentle but persistent in articulating its value across a broad spectrum of computer science and artificial intelligence research. This article introduces his research along with its insightful and unique philosophical framework.

Keywords Institute for New Generation Computer Technology · Logic programming · Inductive logic programming · Skill science

✉ Tomonobu Ozaki
tozaki@chs.nihon-u.ac.jp

Randy Goebel
rgoebel@ualberta.ca

Katsumi Inoue
inoue@nii.ac.jp

[1] College of Humanities and Sciences, Nihon University, Tokyo, Japan

[2] Department of Computing Science, University of Alberta, Edmonton, Canada

[3] National Institute of Informatics, Tokyo, Japan

[4] Department of Informatics, SOKENDAI (The Graduate University for Advanced Studies), Tokyo, Japan

[5] Department of Computer Science, School of Computing, Tokyo Institute of Technology, Tokyo, Japan

```
prodSystem(WM, FinalState) :- member(FinalState, WM).
prodSystem(WM, FinalState) :- member(Fact, WM),
                              recognize(Fact, WM, RHS),
                              act(RHS, WM, NewWM),
                              prodSystem(NewWM, FinalState).
recognize(Fact, WM, RHS) :- rule(LHS=>RHS),
                            member(Fact, LHS),
                            deduce(LHS, WM).
```

Fig. 1 Abstract specification of a production system in Prolog (meta-interpreter), extracted from [12]

Biography Summary

Professor Koichi Furukawa (1942–2017) was born in Manchuria, the present northeastern part of China, and returned to Japan when he was 3 years old. He was recognized as a talented young man from an early age, and was especially good at mathematics, science, and music. Even as a young man, he was also socially active, e.g., he participated in a choral group.

In 1961, he entered the University of Tokyo. In addition to studies in statistical mathematics, he also joined the university orchestra group, and was eagerly practicing cello playing in his college years. It is not a surprise that both mathematics and music became a consistent thread throughout his entire research career.

After graduation, he worked on a time-sharing system at the public national Electro Technical Laboratory (ETL), which eventually became part of the National Institute of Advanced Industrial Science and Technology (AIST) in 2001. He received a Ph.D. in Engineering from the University of Tokyo in 1980, based on a dissertation on the topic of query answering in deductive databases, implemented in LISP.

Furukawa spent 1976 in California at the Stanford Research Institute (now SRI International) where he first encountered the idea of logic programming in the form of a Prolog interpreter, written by Alain Colmerauer. He was very excited about this implementation of computational logic, and returned to ETL the following year, and introduced the idea to his colleagues at ETL. The response was very positive at ETL, which then acquired David H.D. Warren's DEC-10 implementation of Prolog. Furukawa wrote a Prolog program for database indexing, and quickly discovered that one could use logic programming to both specify and implement very efficient and high-level systems, including the first production system interpreter shown in Fig. 1. According to [6], Furukawa described this experience as follows.

> I wrote a program to solve the Rubik cube problem in DEC-10 Prolog [12]. It ran efficiently and solved the problem in a relatively short time (around 20 seconds). It is a kind of expert system in which the inference engine is a Production System realized efficiently by a tail-recursive Prolog program. From this experience, I became convinced that Prolog was the language for knowledge information processing.

David H. D. Warren mentioned that one can easily take this message to mean that Furukawa had fallen in love with Prolog. He was one of the earliest researchers to exploit the clarity and power of Prolog for building meta-interpreters, where

sophisticated reasoning (e.g., abduction, analogy, higher-order predicate logic) could be pursued with both ease and clarity. Since that early work on Prolog, logic programming became a foundation of both his future theoretical and application work, which carried on until the very end of his life.

The Japanese national project on *Fifth Generation Computing Systems* (FGCS) began in 1982, and Furukawa joined the project as a research director at the *Institute for New Generation Computer Technology* (ICOT). His friend and colleague from the University of Tokyo and ETL, Professor Kazuhiro Fuchi, was a "born leader" and helped convene both public and industrial funders to create the joint private–public 10 year project. The overall project had the goal of advancing research in the area of parallel inference machines, their applications and even operating systems, based on logic programming. Furukawa became the deputy director, recruited to lead the foundational research on logic programming. Further details of his role in major achievements in advancing logic programming at ICOT are described in "Research at Institute for New Generation Computer Technology".

In 1992, after the completion of the Fifth Generation Computing project, Furukawa returned to academia, to the Graduate School of Media and Governance at Shonan Fujisawa Campus (SFC), Keio University. There he became a mentor for many graduate students, and led further work on *Inductive Logic Programming* (ILP) [36] and data mining. ILP is a subfield of machine learning which uses logic programming for inductive inference. He had already started to work on ILP when he was in ICOT, and continued to conduct both theoretical and practical projects with colleagues and students while at Keio.

At this time, his strong interest in human activities was revived, especially the skill in playing music instruments. With a focus on musical instrument performance, he began to intensively develop a scientific framework for verbalization of human tacit knowledge, especially to be able to make implicit musical performance skills sufficiently explicit to enable performance improvements in performers. This subsequently led to the creation of a new research field called skill science. In this new research field, Furukawa sought the development of knowledge-based systems to acquire explicit and understandable knowledge, by exploiting abductive and analogical reasoning implemented in logic programming. In 2010, he moved to Kaetsu University, where he continued his intensive research on skill science.

He finally published a summary of his skill science work, including a handbook for cello players, co-authored with Toshiki Masuda, published in 2016 [34]. He intended to translate the book into English, but that project remained uncompleted at the time of his death. He also planned a more comprehensive monograph entitled "Abduction and Induction—A logic programming approach." But in the middle of that dream, he passed away suddenly just on the day where he was to meet with a publishing company.

In the following sections, Furukawa's research achievements are introduced in chronological order, that is, (1) the results in the FGCS project ("Research at Institute for New Generation Computer Technology"), (2) the basic research on ILP ("Research on Inductive Logic Programming"), and (3) introduction and advancement of skill science ("Research on Skill Science").

A particular focus is put on the relationship with logic programming, and it is shown that all these achievements are tightly connected.

Research at Institute for New Generation Computer Technology

In 1982, the Fifth Generation Computer Systems (FGCS) Project began, motivated by a Japanese national policy of developing as a technologically advanced nation. The Institute for New Generation Computer Technology (ICOT) was established to run the project, and Furukawa joined as a research director. The main aim of the project was research and development of new computer technologies for knowledge information processing on parallel inference machines based on logic programming. Until the project finished successfully in 1992, Furukawa engaged in working on various research challenges in logic programming, including concurrent logic programming, partial evaluation, and advanced reasoning such as abduction and induction. In addition, he was in charge of international services, where he not only worked hard to deliver results to conferences and institutes around the world, but also recruited young researchers to ICOT.

Parallel Inference and Logic Programming

At the beginning of the project, ICOT decided to make logic programming the foundation principle of the project. However, at that time the programming language Prolog was the only implementation, and it could not explicitly express concurrent processes, which were required as a basic component of modern operating systems. So ICOT explored the possibility of concurrent logic programming by focusing on the concurrent programming within a framework of logic programming proposed by Keith Clark and Steve Gregory [1]. Both Parlog [2] and Concurrent Prolog [47, 48] were examined as candidates; finally, ICOT consolidated all those ideas in the development of its own concurrent logic programming language named Guarded Horn Clauses (GHC) [51]. One noticeable feature in GHC is the ability to control parallel computation with a very simple mechanism called a guard. It is of note that Furukawa's broad knowledge of computing found the relationship between Dijkstra's "guarded commands" for programming based on predicate transformation, and his vision for their implementation in GHC. This development of ICOT's own kernel language is significant, since it made possible to advance the development of hardware and software at the same time. As a consequence, the hardware design of a parallel inference machine (PIM) as well as a parallel operating system named PIMOS was developed, with one of the few projects that embraced simultaneously the development of hardware and software.

As a concrete application of parallel knowledge information processing, Furukawa encouraged Ryuzo Hasegawa and his colleagues in the development of a theorem prover for full first-order predicate logic as a powerful extension of Prolog's inference mechanism. They employed a Prolog technology theorem prover SATCHMO [33] written in only eight clauses in Prolog as a target, and created an

```
do(A) :- true | satchmo_problem:model(M), false(M, A).
false(M, A) :- true |
    satchmo_problem:nc(NC),
    satisfy_clauses(0, NC, M, cl_sat, A).

satisfy_clauses(Cn, NC, M, A2, A) :- Cn < NC |
    Cn1 := Cn + 1,
    satisfy_ante(Cn1, [], [true|M], M, A2, A1),
    satisfy_clauses(Cn1, NC, M, A1, A).
satisfy_clauses(NC, NC, _,  sat(M1),  A) :- true | A = sat(M1).
satisfy_clauses(NC, NC, M,   cl_sat,  A) :- true | A = sat(M).
satisfy_clauses(NC, NC, M, unsat(Ms), A) :- true | A = unsat(Ms).

satisfy_ante(Cn, GS, [P|M2], M, cl_sat, A) :- true |
    satchmo_problem:c(Cn, P, GS, R),
    satisfy_ante1(Cn, R, P, GS, M2, M, A).
satisfy_ante(Cn, _, [], _, cl_sat, A) :- true | A = cl_sat.
otherwise.
satisfy_ante(_, _, _, _, A1, A) :- true | A = A1.
```

Fig. 2 A part of SATCHMO interpreter in GHC

```
psolve(true, true).
psolve((G1, G2), (R1, R2)) :- psolve1(G1, R1), psolve1(G2, R2).
psolve(G, R) :- clause(G, B), psolve1(B, R).
psolve1(G, R) :- proceed(G), psolve(G, R).
psolve1(G, G) :- stop(G).
```

Fig. 3 Partial evaluator in Prolog

implementation in GHC. The resulting interpreter of SATCHMO in GHC is shown in Fig. 2. This attempt led to a parallel theorem prover named MGTP (Model Generation Theorem Prover) [16, 17]. MTGP was one of the principal research outcomes of ICOT. It achieved very high performance in theorem proving by parallel speedup (it ran 220 times faster on a parallel inference machine with 256 processors), and solved a number of open problems in finite algebra. The MGTP achievement was a promising demonstration of the potential of parallel inference for the whole world.

Partial Evaluation in Logic Programming

Furukawa and his colleagues also worked on a variety of techniques known as *partial evaluation* of logic programs [32]. The idea is that, since all computation with logic programs is inference, even partially executed logic programs can be interpreted and provide the basis for other kinds of reasoning. As noted below, this has been tried for both conventional systems and compilation, but also as an implementation technique for diagnostic, analogical, and higher-order reasoning.

Partial evaluation was first described as an optimization technique for fast computation of meta-interpreters. A simple partial evaluator in Prolog is shown in Fig. 3. As a promising application of the partial evaluator for knowledge-based systems, Furukawa developed a self-applicable partial evaluator with an incremental

compilation method in Prolog, which was used for compiler generation and compiler generator generation [3]. Using similar and extended techniques of partial evaluation, Furukawa, together with Randy Goebel and David Poole, proposed a *theory formation* system that provided a reformulation of rule-based diagnosis systems [15]. Furthermore, he developed partial evaluation techniques for GHC programs [4]. It should be noted that partial evaluators developed by Furukawa are based on meta-interpreters of Prolog or GHC programs.

Artificial Intelligence and Logic Programming

Furukawa's work on theory formation with Goebel et al. [15] is also considered as an early work on *abductive logic programming* [27], which performs abduction in logic programming. In fact, Furukawa conducted at ICOT several projects on advanced reasoning in logic programming. Furukawa had noticed that artificial intelligence (AI) or intelligent computing systems would need these advanced reasoning methods in addition to deductive technologies.

An integrated framework emerged in the work on *knowledge assimilation*, which maintains and updates knowledge bases whenever a new piece of knowledge or information is acquired. Integrity constraints play an important role in knowledge assimilation, and all modes of reasoning, i.e., deduction, induction and abduction, are involved in the update process. The first knowledge assimilation system was developed in Prolog at ICOT [35], and since then, *hypothetical reasoning* systems have been developed in ICOT including a version of Theorist [15]. These attempts were indeed too advanced in the middle of 1980s, and people could only notice the importance much later in 1990s [27].

As a natural extension of Furukawa's depth of understanding of AI through logic programming, researchers at ICOT developed an interest in *nonmonotonic reasoning*, *constraint* (*logic*) *programming*, and *inductive logic programming* (ILP). These logic programming paradigms could be specified and implemented with some techniques of *meta-programming* and partial evaluation, providing both clarity of their scopes and exposing practical challenges in their implementation.

It is remarkable that Furukawa was involved in early work on abductive and inductive reasoning as contributions to the foundations of AI. He encouraged David Poole, Randy Goebel, Stephen Muggleton and others in their pursuit of the creation of new AI systems such as Theorist [45] and Progol [39]. Furukawa also invited many other AI researchers to ICOT, including Mark Stickel for theorem proving and abductive reasoning and Nicolas Helft for ILP and nonmonotonic reasoning. Katsumi Inoue enjoyed discussions and collaboration, which resulted in the successful development of a consequence finding system for first-order full clausal theories called SOL resolution [19]. SOL resolution was later implemented in SOLAR [42], and Furukawa used SOLAR for meta-level abduction in his later work on skill science.

In the late 1980s, researchers of logic programming and AI in the world began collaborations to extend the class of definite logic programs by introducing negation and abducibles in the bodies of program rules, to allow the users to declaratively

express defaults or hypotheses. Logic programs with negation became the basic form of nonmonotonic reasoning, and then lead to *answer set programming* in the 21st century. On the other hand, logic programs with abducibles were soon called abductive logic programs, and were integrated in ILP [41]. Around the end period of the FGCS project, Hasegawa, Inoue and their colleagues implemented both nonmonotonic reasoning [24] and hypothetical reasoning [25] on top of MGTP, thereby showing the effect of parallelism of these reasoning systems on PIM machines. Bob Kowalski admitted, in his perspective of FGCS [31], that this last achievement was very important, by saying that "ICOT has been a significant participant in these developments." Furukawa's perspectively concluded [6] as follows:

> However, most research output, including the knowledge assimilation system was not integrated into the concurrent logic programming framework. Therefore, very little was produced for final fifth-generation computer systems based on PIM. An exception is a parallel bottom-up theorem prover called MGTP, and application systems running on it. An example of such applications is hypothetical reasoning. I expect this approach will attain my original goal of "high-speed knowledge information processing based on parallel computation mechanisms and specialized hardware" in the near future.

Managing International Affairs Services

In the 1980s, Furukawa took a leadership role at ICOT, to connect the world to the Japanese initiative in logic programming and artificial intelligence. He organized many bilateral (UK–Japan, French–Japan, and USA–Japan), trilateral (Italy–Japan–Sweden) and international workshops. During his many visits to research laboratories, he invited many top-level researchers to ICOT: J. A. Robinson, Robert A. Kowalski, Ehud Shapiro, Keith Clark, Randy Goebel, David Poole, Mark Stickel, Donald W. Loveland, Wolfgang Bibel, Gerd Brewka, Ray Reiter, Vladimir Lifschitz, and Stephen Muggleton, among many others. He was always noted to be a gracious intellectual host, always seeking to find ideas and methods that could strengthen and compliment the research goals of those visitors. In his final lecture at Keio University, he said that his "total number of business trip abroad reaches to about sixty in eleven years." So in addition to providing tireless research leadership within ICOT, he was also a tireless missionary for logic programming all around the world.

Within the logic programming network he created around the world, he was a central figure in organizing the series of International Conference on Logic Programming (ICLP). He was the official program chair of the Eighth International Conference on Logic Programming (ICLP 1991) in Paris [5].

Within the research community on logic programming, he was also famous as a cello player in "Logic Programming Trio," (Fig. 4) formed with J. A. Robinson and Jacques Cohen. After their first concert at the Joint International Conference and Symposium on Logic Programming in 1992, they subsequently showed many more amazing musical performances at many international conferences.

Fig. 4 Logic Programming Trio: Jacques Cohen, J. Alan Robinson, Koichi Furukawa, circa 1996 (http://jc.cs.brandeis.edu/?gallery=music-2)

Research on Inductive Logic Programming

Inductive logic programming (ILP) [36, 40, 41] is an interdisciplinary research area that combines logic programming and machine learning methods for inductive inference. Furukawa already had a great interest in ILP while he was in ICOT, and invited Nicolas Helft as a researcher to ICOT and Stephen Muggleton as a guest researcher. Furukawa participated in the first international workshop on Inductive Logic Programming [37] held in Portugal, then organized the second workshop [38] with Fumio Mizoguchi in Tokyo in 1992.

As a theoretical research pursuit within the scope of ILP, Furukawa made an effort to make *Inverse Entailment* [39] complete. This idea is highly appealing, for it provides the basis to understand the formal relationship between facts stated about some domain, and those hypotheses which could minimally cover those facts, much like the formation of "best" or minimal hypotheses in the framework of scientific reasoning. Inverse Entailment [39] was formulated as an inductive inference rule, which effectively uses the most specific hypothesis to derive hypotheses for given positive examples and background knowledge. Right after the incompleteness of Inverse Entailment was articulated [53], Furukawa proposed a sufficient condition for existence of the most specific hypothesis [11] as well as procedures for constructing a complete most specific hypothesis for recursive programs [9, 13].

Furukawa was always a fearless intellectual, and in addition to the specification of sufficient conditions for the most specific hypothesis, he also worked on that foundation as a use for higher-order concept formation, perhaps the most elaborate form of ILP to date, e.g., [44]. He also pointed out the relationship between inverse entailment of inductive inference and the consequence finding, and in 1995 he encouraged Katsumi Inoue to use a consequence finding procedure like SOL resolution for computing inverse entailment. In 2001, Inoue presented a complete algorithm called CF Induction [20] based on this suggestion.

Furukawa applied his general understanding of ILP to a variety of real-world problems including business applications. For example, he developed an expert system

```
sentence85(A) :- have(A, 'method'),
                 in_order(A, 'MFPrn', 'mode'),
                 not_have(A, 'NewMFPrn').
sentence85(A) :- in_order(A, 'MFPrn', 'automatic operation').
sentence85(A) :- in_order(A, 'turning off power supply', 'method').
```

Fig. 5 Rules in Prolog for classifying a given inquiry into prototypical question no. 85

named AUTOMAIL, to support consumer product call center operators to promptly and accurately prepare near optimal responses to their customers' questions [50]. In that system, Shimazu and Furukawa used ILP to construct rules, as shown in Fig. 5, to classify call center inquiries into 86 classes. This practical system greatly reduced the burden on the operator's work in real environments.

Furukawa was also devoted to the dissemination of ILP in Japan. He prepared a lecture course for ILP in the graduate school at Keio University, and published the first text book written in Japanese [14].

Research on Skill Science

After moving to Keio University, Furukawa's creation of the field of "Skill Science" started out as research on the "verbalization of human tacit knowledge." He believed, like with the use of logic programming for parallel systems building and for induction, that it could also provide the basis for articulating the tacit knowledge of physical skills like playing an instrument or performing as an athlete. In fact, this intellectual motivation combined well with his passionate interest in music, especially playing the cello. He always believed that the performance skills of highly skilled musicians could be captured in some logical form, and then conveyed to musicians to improve their performance skills. In this case, his focus was on cello playing, and on improving his own performance skills as a competent amateur cellist. In this regard, the scientific framework he developed included a variety of methods, including inductive logic programming, time series data mining, abduction, and analogical reasoning. He was an early advocate of research on "human-like computing," and he and his colleagues developed a new research field called skill science. Skill science is a new multidisciplinary research area with approaches including artificial intelligence, cognitive science, sports science, biomechanics and kinesiology, and ecological psychology [52]. The overall goal was to bring skills of tacit embodied expertise onto the podium of scientific exploration, to understand deeply the processes of skill acquisition, and to explore ways of designing good pedagogical environments. In this sense, Furukawa sought to focus all his research skills and background to help understand how to build the knowledge to help be a better skill science teacher.

Induction in Skill Science

In the early stages of his skill science study, Furukawa organized the "Keio International Workshop on Verbalization of Tacit Knowledge using Inductive

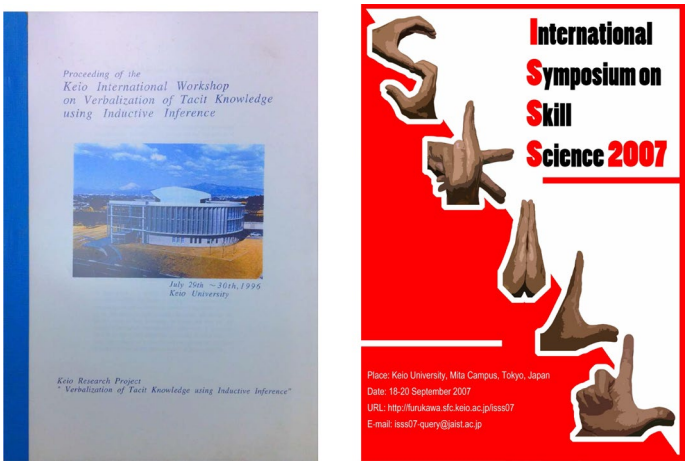

Fig. 6 Proceedings of Keio International Workshop on Verbalization of Tacit Knowledge using Inductive Inference (left), and Poster of International Symposium on Skill Science 2007 (right)

Inference" [7] (Fig. 6) by inviting experts with different backgrounds, including computer scientists, cognitive scientists, and professional musicians. As the title of the workshop suggested, he considered inductive inference, especially inductive logic programming, as a central technology of the project. This underlying thought can be confirmed in the preface of the workshop proceedings, which is concluded by the following sentence:

> In particular, we wish to discuss and explore the possibility of using inductive logic programming as a common tool to uncover the structures of tacit knowledge across different aspects of human activities.

In the research pursuit of articulating human tacit knowledge, Furukawa employed ILP for the analysis of human respiration during musical performance [18]. While respiration control is one of the most important factors for effective musical performance, it is often difficult for even experts to explain how to clearly explain the role and control of respiration during performance. So in Furukawa's research approach, he sought to capture rules that showed repeatability and regularity in a performer's respiration patterns. These rules were successfully extracted with ILP from sensor data together with musical/performance background knowledge, such as harmonic progression and bowing direction. It was also discovered that players tend to exhale at the beginning of new large musical structures, and inhale immediately before the change of keys.

Apart from the improvements of his own cello performance, Furukawa concentrated on language acquisition and concept formation as a central requirement to capture complex human tacit knowledge. For example, he proposed a computational model for children's language acquisition process using ILP [30]. His system named WISDOM (Fig. 7) succeeded in reproducing a part of a co-evolution

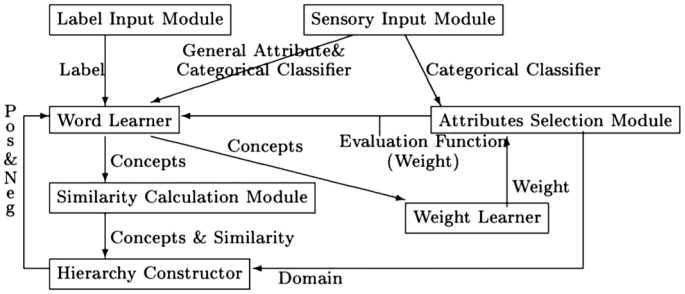

Fig. 7 Configuration of WISDOM (from Figure 1 in [30])

mechanism of acquiring concept definitions for words and in the development of concept hierarchies by incorporating cognitive biases.

In 2007, Furukawa organized the International Symposium on Skill Science [8] (Fig. 6). The symposium had three invited lectures, one of which was a professional violinist Ikuko Mizuno, who gave a lecture with the title of "How to play the violin - Controlling the body and mind." Also included was a panel discussion entitled "Three important problems of skill science for next five years," by Tsutomu Fujinami, Masaki Suwa, J. A. Robinson, Jacques Cohen, and Koichi Furukawa. This symposium demonstrated the true interdisciplinary nature of skill science and research presentations were made from a broad variety of areas. In addition, all the members of the musical group known as the Logic Programming Trio, participated in the panel, so we can see that logic programming was indeed considered as one of the fundamental techniques in skill science research. Although it is a digression, we enjoyed the concert by "Logic Programming Trio++", i.e., Logic Programming Trio with Ikuko Mizuno (violin) and Maya Okamoto (viola) in the symposium. In fact, many of Furukawa's interactions with his colleagues included some kind of musical performance.

Abduction in Skill Science

Furukawa investigated the application of abductive reasoning to skill science, because he believed that the framework of abductive reasoning for generating hypotheses was necessary to explain the basis for amazing observed performance facts; this exactly matched his idea of the mechanism of skill acquisition.

He first employed an abductive logic programming system ProLogICA [46] to find appropriate hypotheses to explain the behavior of both professional and amateur incorrect cello performances [29]. An example of Abductive Logic Programming for cello playing is shown in Fig. 8. The program in Fig. 8 is for finding a cello playing technique for the rapid position shift accomplished by two possible methods: (1) move the elbow up and down by adduction and abduction of shoulder and elbow joints, or (2) move the hand position up and down by incycloduction and excycloduction of the upper arm. In the course of his experiments, the abductive logic program derived only the second method, because of the specified integrity constraints.

```
rapidPositionShift :- rapidMove, addAbdOfShoulder, addAbdOfElbow.
repidPositionShift :- rapidMove, inExCycloOfUpperarm, addAbdOfElbow.
bigInertiaMovement :- addAbdOfShoulder.

ic :- rapidMove, bigInertiaMovement. %integrity constraints

abducible_predicate(rapidMove).
abducible_predicate(addAbdOfShoulder).
abducible_predicate(addAbdOfElbow).
abducible_predicate(inExCycloOfUpperarm).
```

Fig. 8 Abductive Logic Programming for explaining the proposition rapidPositionShift (a method of cello playing accompanied with high-speed position shift)

In fact, during his study of skill science using abduction, Furukawa experienced some sudden skill improvement in his own cello playing, after his final lecture concert at the Keio University (March 2008).[1] This improvement arose by simply keeping his right arm close, that is, to keep his elbow close to the body side. This method not only increased the sound volume, but helps keep bowing stable and supports maximum bow usage. As noted in the comments about making tacit knowledge explicit for teaching, Furukawa believed that the reproduction of good skill requires explanation, which helps makes the skill robust and tolerant of situation changes. In fact, his belief was that no explicit discovery of tacit knowledge is useful if it could not be explained. The process of explanation led further to another important finding, which connected abduction and ILP to the general notion of scientific discovery. With respect to skill science and, specifically musical instrument performance, Furukawa focused on what he called "knack discovery," and worked hard to formulate knack discovery using a bold combination of ILP, abduction, and analogical reasoning, which he called "meta-level abduction."

Meta-level abduction [21, 23] performs abduction on meta-level axioms, and abduces rules that infer hypotheses explaining empirical rules by means of hidden rules and containing unknown nodes as new predicates. The combination of rule abduction and fact abduction is possible using conditional query answering in a consequence finding system SOLAR [42, 43]. Meta-level abduction was applied to knack discovery [23] as well as biological network completion [22]. From the axioms and domain-specific background knowledge shown in Fig. 9, a hypothesis

$$\exists \mathtt{X}.(\mathtt{linked}(\mathtt{stable\_bow\_movement}, \mathtt{X}) \wedge \mathtt{linked}(\mathtt{flexible\_wrist}, \mathtt{X}) \wedge \mathtt{linked}(\mathtt{X}, \mathtt{keep\_arm\_close}))$$

is successfully inferred for explaining the goal process `caused(inc_sound, keep_arm_close)`: that goal process represents the alleged causality that keeping one's arm close (`keep_arm_close`) makes the sound increase (`increase_sound_volume`). This use of meta-level abduction provided confirmation

[1] His final lecture concert, which is also a part of his research results on skill science, can be seen at https://www.youtube.com/watch?v=r1qoyQIYt6s. A full version is available at http://gc.sfc.keio.ac.jp/cgi/video_gc/view_video_gc.cgi?2007_gc00001+10+1 by using a flash player (start from about 38:00).

```
% Axiom:
caused(X, Y) :- linked(X, Y).
caused(X, Y) :- linked(X, Z), caused(Z, Y).

% Background knowledge:
linked(inc_sound, bow_close_to_the_bridge).
linked(bow_close_to_the_bridge, stable_bow_movement)  ;
  linked(bow_close_to_the_bridge, smooth_bow_direction_change).
linked(smooth_bow_direction_change, flexible_wrist).
:- linked(inc_sound, keep_arm_close).
:- linked(stable_bow_movement, keep_arm_close).
:- linked(smooth_bow_direction_change, keep_arm_close).

% Goal:
:- caused(inc_sound, keep_arm_close).
```

Fig. 9 An example of knack discovery in meta-level abduction

```
b_caused(X, Y) :- b_connected(X, Y).
b_cuased(X, Y) :- b_connected(X, Z), t_caused(Z, Y).

t_caused(X, Y) :- t_connected(X, Y).
t_cuased(X, Y) :- t_connected(X, Z), t_caused(Z, Y).

t_connected(X, Y) :- originally_connected(X, Y).
t_connected(X, Y) :- connected_by_abduction(X, Y).
t_connected(X, Y) :- connected_by_analogy(X, Y),
                     print_connected_by_analogy(X, Y).

connected_by_analogy(X, Y) :- b_connected(XX, YY),
                              similar(X, XX), similar(Y, YY).

:- connected_by_analogy(X, Y), connected_by_abduction(X, Y).

abducible(connected_by_abduction(_, _)).
abducible(similar(_, _)).
abducible(print_connected_by_analogy(_, _)).
```

Fig. 10 Axioms in analogical rule abduction

that a knack represented by an ∃X statement can be automatically discovered and confirmed.

While the above form of meta-level abduction can abduce missing links, the abduced rules must be interpreted in some external way. That is, it is not easy to find language concept equivalents in which to convey the meta-level abduced structures. As a potential solution to this hypothesis interpretation problem, Furukawa developed an extension of meta-level abduction named analogical rule abduction [10, 28] by adding axioms for analogical inference. Analogical rule abduction makes it possible to exploit a vocabulary of analogical relations to provide understandable interpretation to the introduced predicates and rules; analogical inferences across vocabularies can create appropriate cross-vocabulary language concepts. The axioms for analogical rule abduction are shown in Fig. 10. In those axioms, causalities in source and target worlds of analogy are represented in the predicate b_ - and

t_caused/3, respectively. Three kinds of connections in target worlds represented in the predicate t_connected/2 are considered: (1) observed connections (connections without assumption), (2) connections by abduction, and (3) connections by analogy. The proposed framework was applied to the problem of explaining the difficult cello playing techniques of spiccato and rapid cross strings of the bow movement.

In fact in this near penultimate work, Furukawa had finally integrated all of his background on logic programming, inductive logic programming, abduction, analogical reasoning, and higher-order reasoning, into one grand demonstration of how the knacks of performance could be identified and articulated as outputs of his skill science framework.

By this final research achievement, Furukawa was invited to the 25th International Conference on Inductive Logic Programming (ILP 2015), which was held in Kyoto, Japan, as a special invited speaker. It was his first return to an ILP conference since 2001. The editor of the special issue on ILP 2015 in *Machine Learning* journal [26] drew the following conclusion:

> Dr. Furukawa contributed to establish the field of ILP and will be missed by the entire ILP community. In ILP 2015 he was very pleased that ILP conferences were held for 25 years. We would like to dedicate this special issue to the memory of Dr. Furukawa.

Katsumi Inoue also gave a speech in memory of Koichi Furukawa at ILP 2017 held in Orléan, France, September 5th, 2017, which was organized together with memorials of Alain Colmerauer and Alan Robinson given by Frédéric Benhamou and Stephen Muggleton, respectively.

Conclusion

This article reviewed Professor Koichi Furukawa's research activities from Fifth Generation Computing to Skill Science, focusing on the logic programming point of view. As his history shows, he contributed to research on logic programming for a long time. Even in his unpublished book, mentioned in "Biography Summary", he was planning to include a small section for explaining a major advantage of logic programming, which is an ability for explicit understanding and explanation necessary for handling human knowledge and skills. We believe that this attribute was one of big reasons that Furukawa was such a strong advocate for logic programming.

Logic programming played a significant role in his research activity. He also reviewed his research history by himself in his final lecture at Keio University and special lecture in the 28th Annual Conference of the Japan Society of Artificial Intelligence, and stated the following comments (translation from Japanese):

- There always existed logic programming as the background even when changing the research greatly from the fifth generation computer systems project to skill science.

- Divergent researches from fifth generation project and data mining to skill science are converging within myself.
- Researches on meta-programming, inductive reasoning, and abductive reasoning in the fifth generation project era helped the research on skill science.

A significant personal attribute of Koichi Furukawa is that he was never afraid of tackling very difficult fundamental problems, and he always kept in mind how one should document any research progress, so that others could understand and exploit that progress. As a mentor, his style was to ask simple deep questions about how an idea was working (or not), and to encourage his students and colleagues alike, to always broaden their perspective when they reached an apparent research challenge.

We conclude this essay by noting that we expect his research legacy will continue to lead to further development of logic programming and skill science. May he rest in peace.

References

1. Clark, K.K., Gregory, S.: A relational language for parallel programming. In: Proceedings of the 1981 Conference on Functional Programming Languages and Computer Architecture, pp. 171–178. ACM, Boston (1981). https://doi.org/10.1145/800223.806776
2. Clark, K.L., Gregory, S.: PARLOG: a parallel logic programming language. Research Report DOC 83/5, Department of Computing, Imperial College, London (1983)
3. Fujita, H., Furukawa, K.: A self-applicable partial evaluator and its use in incremental compilation. New Gener. Comput. **6**(2–3), 91–118 (1988). https://doi.org/10.1007/BF03037133
4. Fujita, H., Okumura, A., Furukawa, K.: Partial evaluation of GHC programs based on UR-set with constraint solving. ICOT Technical Report, TR-344 (1988)
5. Furukawa, K.: In: Proceedings of the Eighth International Conference on Logic Programming. MIT Press, Cambridge (1991)
6. Furukawa, K.: Contribution in [49], pp. 60–65 (1993)
7. Furukawa, K.: In: Proceedings of the Keio International Workshop on Verbalization of Tacit Knowledge Using Inductive Inference. Keio University, Tokyo (1996)
8. Furukawa, K.: In: Proceedings of the International Symposium of Skill Sciences. Keio University, Tokyo (2007)
9. Furukawa, K.: On the completion of the most specific hypothesis computation in inverse entailment for mutual recursion. In: Proceedings of the First International Conference on Discovery Science. Lecture Notes in Computer Science, vol. 1532, pp. 315–325. Springer, Berlin (1998). https://doi.org/10.1007/3-540-49292-5_28
10. Furukawa, K., Kinjo, K., Ozaki, T., Haraguchi, M.: On skill acquisition support by analogical rule abduction. In: International Workshop on Information Search, Integration, and Personalization, Revised Selected Papers, Communications in Computer and Information Science, vol. 421, pp. 71–83. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08732-0_6
11. Furukawa, K., Murakami, T., Ueno, K., Ozaki, T., Shimazu, K.: On a sufficient condition for the existence of most specific hypothesis in Progol. In: Proceedings of the 7th International Workshop on Inductive Logic Programming. Lecture Notes in Computer Science, vol. 1297, pp. 157–164. Springer, Berlin (1997). https://doi.org/10.1007/3540635149_44

12. Furukawa, K., Nakajima, R., Yonezawa, A., Goto, S., Aoyama, A.: Problem solving and inference mechanisms. In: Moto-oka, T. (ed.) Fifth Generation Computer Systems, pp. 131–138. Elsevier (1982). https://doi.org/10.1016/B978-0-444-86440-6.50008-6
13. Furukawa, K., Ozaki, T.: On the completion of inverse entailment for mutual recursion and its application to self recursion. In: Work-in-Progress Reports of the 10th International Conference on Inductive Logic Programming, CEUR Workshop Proceedings 35 (2000)
14. Furukawa, K., Ozaki, T., Ueno, K.: Inductive Logic Programming. Kyoritsu Shuppan, Tokyo (2001) **(in Japanese)**
15. Goebel, R., Furukawa, K., Poole, D.: Using definite clauses and integrity constraints as the basis for a theory formation approach to diagnostic reasoning. In: Proceedings of the Third International Conference on Logic Programming. Lecture Notes in Computer Science, vol. 225, pp. 211–222. Springer, Berlin (1986). https://doi.org/10.1007/3-540-16492-8_77
16. Hasegawa, R., Fujita, H., Fujita, M.: A parallel theorem prover in KL1 and its application to program synthesis. ICOT Technical Report, TR-588 (1990)
17. Hasegawa, R., Koshimura. M., Fujita, H.: MGTP: a parallel theorem prover based on lazy model generation. In: Proceedings of the 11th International Conference on Automated Deduction. Lecture Notes in Computer Science, vol. 607, pp. 776–780. Springer, Berlin (1992). https://doi.org/10.1007/3-540-55602-8_223
18. Igarashi, S., Ozaki, T., Furukawa, K.: Respiration reflecting musical expression: analysis of respiration during musical performance by inductive logic programming. In: Proceedings of the Second International Conference on Music and Artificial Intelligence. Lecture Notes in Computer Science, vol. 2445, pp. 94–106. Springer, Berlin (2002). https://doi.org/10.1007/3-540-45722-4_10
19. Inoue, K.: Linear resolution for consequence finding. Artif. Intell. **56**(2–3), 301–353 (1992)
20. Inoue, K.: Induction, abduction, and consequence-finding. In: Proceedings of the 11th International Conference on Inductive Logic Programming. Lecture Notes in Computer Science, vol. 2157, pp. 65–79. Springer, Berlin (2001). https://doi.org/10.1007/3-540-44797-0_6
21. Inoue, K.: Meta-level abduction. IfCoLog J. Log. Appl. **3**(1), 7–36 (2016)
22. Inoue, K., Doncescu, A., Nabeshima, H.: Completing causal networks by meta-level abduction. Mach. Learn. **91**(2), 239–277 (2013). https://doi.org/10.1007/s10994-013-5341-z
23. Inoue, K., Furukawa, K., Kobayashi, I., Nabeshima, H.: Discovering rules by meta-level abduction. In: Proceedings of the 19th International Conference on Inductive Logic Programming (ILP '09). Lecture Notes in Computer Science, vol. 5989, pp. 49–64. Springer, Berlin (2009). https://doi.org/10.1007/978-3-642-13840-9_6
24. Inoue, K., Koshimura, M., Hasegawa, R.: Embedding negation as failure into a model generation theorem prover. In: Proceedings of the 11th International Conference on Automated Deduction (CADE-11). LNCS, vol. 607, pp. 400–415. Springer, Berlin (1992). https://doi.org/10.1007/3-540-55602-8
25. Inoue, K., Ohta, Y., Hasegawa, R., Nakashima, M.: Bottom-up abduction by model generation. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93), pp. 102–108 (1993). http://ijcai.org/Proceedings/93-1/Papers/015.pdf
26. Inoue, K., Ohwada, H., Yamamoto, A.: Special issue on inductive logic programming. Mach. Learn. **106**(12), 1863–1865 (2017). https://doi.org/10.1007/s10994-017-5679-8
27. Kakas, A.C., Kowalski, R.A., Toni, F.: The role of abduction in logic programming. Handb. Log. Artif. Intell. Log. Program. **5**, 235–324 (1998)
28. Kinjo, K., Ozaki, T., Furukawa, K., Haraguchi, M.: On skill acquisition support by analogical rule abduction. Trans. Jpn. Soc. Artif. Intell. **29**(1), 188–193 (2014). https://doi.org/10.1527/tjsai.29.188 **(in Japanese)**
29. Kobayashi, I., Furukawa, K.: Modeling physical skill discovery and diagnosis by abduction. Trans. Jpn. Soc. Artif. Intell. **23**(3), 127–140 (2008). https://doi.org/10.11185/imt.3.385
30. Kobayashi, I., Furukawa, K., Ozaki, T., Imai, M.: A computational model for children's language acquisition using inductive logic programming. In: Progress in Discovery Science, Final Report of the Japanese Discovery Science Project. Lecture Notes in Computer Science, vol. 2281, pp. 140–155. Springer, Berlin (2002). https://doi.org/10.1007/3-540-45884-0_7
31. Kowalski, R.A.: Contribution in [49], pp. 54–60 (1993)
32. Lloyd, J.W., Shepherdson, J.C.: Partial evaluation in logic programming. J. Log. Program. **11**(3–4), 217–242 (1991). https://doi.org/10.1016/0743-1066(91)90027-M

33. Manthey, R., Bry, F.: SATCHMO: a theorem prover implemented in Prolog. In: Proceedings of the 9th International Conference on Automated Deduction. Lecture Notes in Computer Science, vol. 310, pp. 415–434. Springer, Berlin (1998). https://doi.org/10.1007/BFb0012847
34. Masuda, T., Furukawa, K.: You Approach the Cello, and the Cello Approaches You—An Approach Based on Skill Science. Doremi Publishing, Tokyo (2016)
35. Miyachi, T., Kunifuji, S., Kitakami, H., Furukawa, K., Takeuchi, A., Yokota, H.: A knowledge assimilation method for logic databases. New Gener. Comput. **2**(4), 385–404 (1984). https://doi.org/10.1007/BF03037329
36. Muggleton, S.: Inductive logic programming. New Gener. Comput. **8**(4), 295–318 (2001). https://doi.org/10.1007/BF03037089
37. Muggleton, S.H.: In: Proceedings of the First International Workshop on Inductive Logic Programming (1991)
38. Muggleton, S.H.: In: Proceedings of the Second International Workshop on Inductive Logic Programming (1992)
39. Muggleton, S.: Inverse entailment and Progol. New Gener. Comput. **13**(3–4), 245–286 (1995). https://doi.org/10.1007/BF03037227
40. Muggleton, S., De Raedt, L.: Inductive logic programming: theory and methods. J. Log. Program. **19**(20), 629–679 (1994). https://doi.org/10.1016/0743-1066(94)90035-3
41. Muggleton, S., De Raedt, L., Poole, D., Bratko, I., Flach, P., Inoue, K., Srinivasan, A.: ILP turns 20—biography and future challenges. Mach. Learn. **86**(1), 3–23 (2012). https://doi.org/10.1007/s10994-011-5259-2
42. Nabeshima, H., Iwanuma, H., Inoue, K.: SOLAR: a consequence finding system for advanced reasoning. In: Proceedings of International Conference on Automated Reasoning with Analytic Tableaux and Related Methods. Lecture Notes in Computer Science, vol. 2796, pp. 257–263. Springer, Berlin (2003). https://doi.org/10.1007/978-3-540-45206-5_22
43. Nabeshima, H., Iwanuma, K., Inoue, K., Ray, O.: SOLAR: an automated deduction system for consequence finding. AI Commun. **23**(2–3), 183–203 (2010). https://doi.org/10.3233/AIC-2010-0465
44. Padmanabhuni, S., Goebel, R., Furukawa, K.: Curried least general generalization: a framework for higher order concept learning. In: Learning and Reasoning with Complex Representations, PRICAI'96 Workshops on Reasoning with Incomplete and Changing Information and on Inducing Complex Representations, Selected Papers. Lecture Notes in Computer Science, vol. 1359, pp. 45–60. Springer, Berlin (1998). https://doi.org/10.1007/3-540-64413-X_27
45. Poole, D., Goebel, R., Aleliunas, R.: Theorist: a logical reasoning system for defaults and diagnosis. In: Cercone, N., McCalla, G. (eds.) The Knowledge Frontier: Essays in the Representation of Knowledge, pp. 331–352. Springer, Berlin (1987) **[Also Research Report CS-86-06, Department of Computer Science, University of Waterloo (1986)]**
46. Ray, O., Kakas, A.C.: ProLogICA: a practical system for abductive logic programming. In: Proceedings of the 11th Workshop on Nonmonotonic Reasoning, pp. 304–314. Institut für Informatik, Technische Universität Clausthal, Clausthal-Zellerfeld (2006)
47. Shapiro, E.Y.: A subset of concurrent Prolog and its interpreter. ICOT Technical Report, TR-003 (1983)
48. Shapiro, E., Takeuchi, A.: Object oriented programming in concurrent Prolog. New Gener. Comput. **1**, 25–48 (1983). https://doi.org/10.1007/BF03037020
49. Shapiro, E., Warren, D.H.D. (eds.): Launching the new era—personal perspectives of Fifth Generation Computer Systems project. Commun. ACM **36**(3), 47–101 (1993)
50. Shimazu, K., Furukawa. K.: DAISY, an RER model based interface for RDB to ILP. In: Proceedings of the 22nd International Conference on Conceptual Modeling. Lecture Notes in Computer Science, vol. 2813, pp. 390–404. Springer, Berlin (2003). https://doi.org/10.1007/978-3-540-39648-2_31
51. Ueda, K.: Guarded horn clauses. ICOT Technical Report, TR-103 (1985)
52. Ueno, K., Furukawa, K., Bain, M.: Motor skill as dynamic constraint satisfaction. Linköp. Electron. Artic. Comput. Inf. Sci. **5**(2000), nr36 (2000). http://www.ep.liu.se/ea/cis/2000/036/. Accessed 10 Apr 2019
53. Yamamoto, A.: Which hypotheses can be found with inverse entailment? In: Proceedings of the 7th International Workshop on Inductive Logic Programming. Lecture Notes in Computer Science, vol. 1297, pp. 296–308. Springer, Berlin (1997). https://doi.org/10.1007/3540635149_58

Tomonobu Ozaki Ph.D. is a Professor in the College of Humanities and Sciences, Nihon University. His areas of interest include logic programming, skill science, data mining and artificial intelligence.

R.G. (Randy) Goebel is currently a Professor of Computing Science in the Department of Computing Science at the University of Alberta, Associate Vice President (Research) and Associate Vice President (Academic), and Fellow and co-founder of the Alberta Machine Intelligence Institute (AMII). He received B.Sc. (Computer Science), M.Sc. (Computing Science), and Ph.D. (Computer Science) from the Universities of Regina, Alberta, and British Columbia, respectively. Professor Goebel's theoretical work on abduction, hypothetical reasoning and belief revision is internationally well known, and his recent research is focused on the formalization of visualization and explainable artificial intelligence (XAI). He has worked on optimization, algorithm complexity, systems biology, and natural language processing, including applications in legal reasoning and medical informatics.

Katsumi Inoue is a Professor of Principles of Informatics Research Division, National Institute of Informatics, and is a Professor of Department of Informatics, School of Multidisciplinary Sciences, SOKENDAI (The Graduate University for Advanced Studies). He has also been a Specially Appointed Professor of Department of Computer Science, School of Computing, Tokyo Institute of Technology, since 2015. He received the Doctor of Engineering degree from Kyoto University in 1993. His research interests include artificial intelligence, logic programming, and computer science in general.