UCLouvain

Life in Paris: My adventures with Hassan Aït-Kaci at DEC PRL

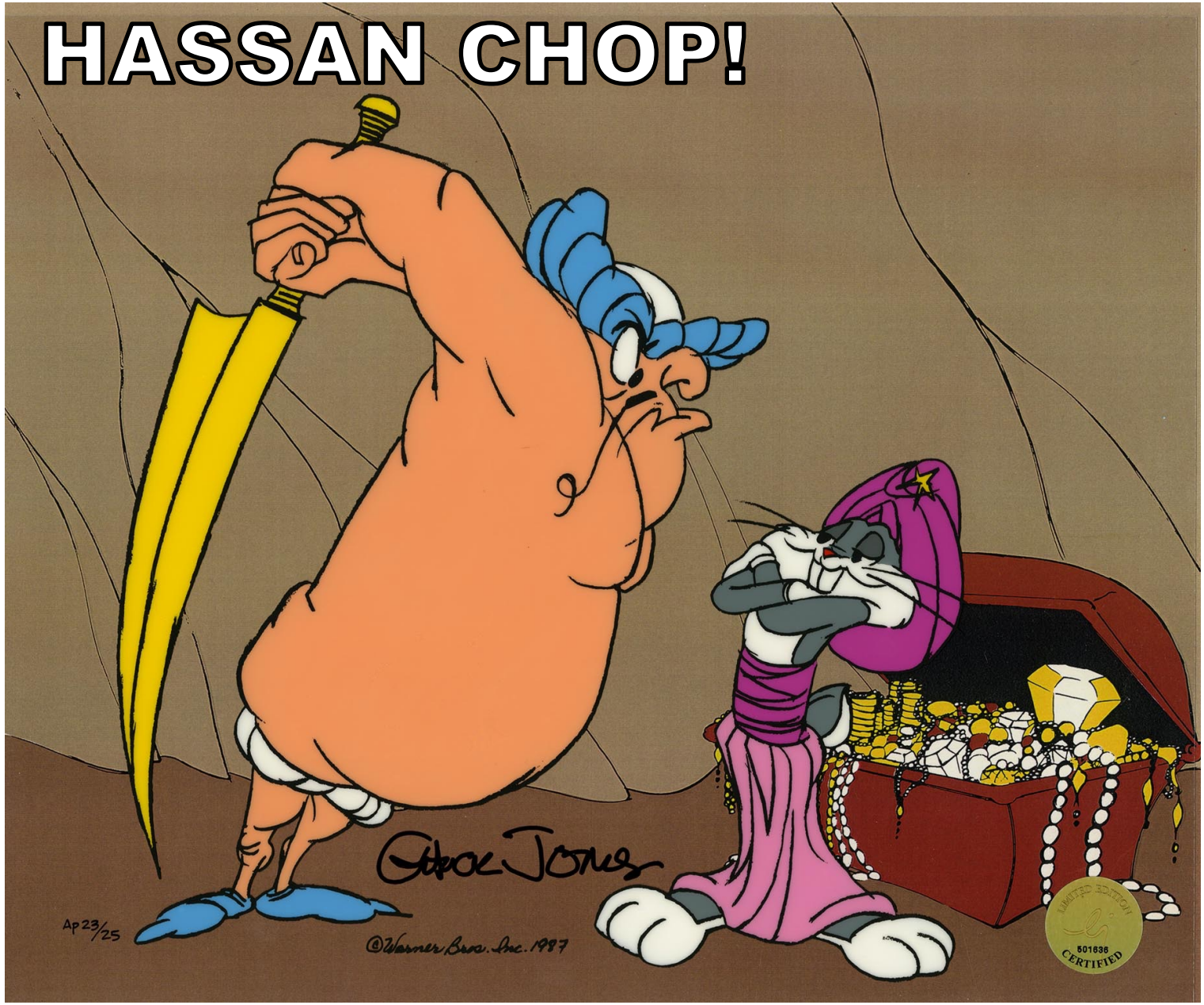
Peter Van Roy

Sep. 26, 2025

Workshop in memory of Hassan Aït-Kaci

Université Claude Bernard Lyon 1

France



Ali Baba Bunny, 1957 Warner Bros. cartoon directed by Chuck Jones

Hassan, guardian of the treasure

Bugs Bunny, Genie the light brown hare

DEC PRL

- After finishing my Ph.D. in 1990 at UCBerkeley, I joined DEC PRL in June 1991 until it closed in Sept. 1994
  - Digital Equipment Corporation Paris Research Laboratory
  - I had offers from several places but joined Hassan Aït-Kaci at PRL
  - It was a fascinating journey guided by Hassan's vision but we traveled on a bumpy road because of Digital's slow demise
  - Hassan had a strong vision for advanced logic programming
  - My work in this vision was focused on the implementation of LIFE
- The Paradise project at PRL initially consisted of Hassan Aït-Kaci, Andreas Podelski, Kathleen Milsted, and Peter Van Roy
  - Other members were Richard Meyer, Bruno Dumant, Abder Aggoun
  - Student interns were Arnaud Venet, Gerard Ellis, Seth Goldstein
  - There were also a number of external collaborators

LIFE: the vision

LIFE, the universe, and everything

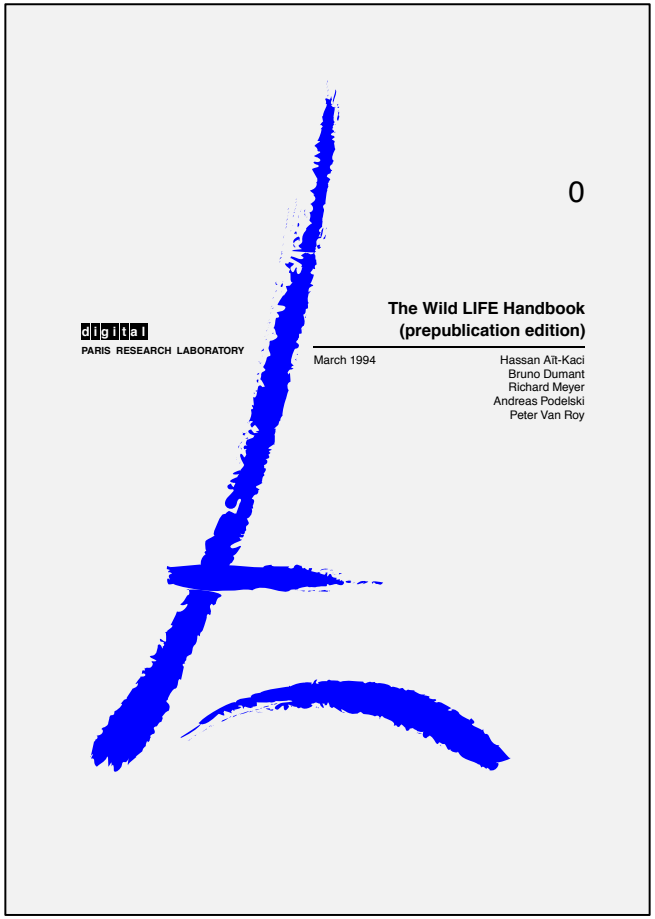
- LIFE (Logic, Inheritance, Functions, Equations) is an experimental logic programming language with a powerful facility for structured type inheritance. LIFE was conceived at MCC by Hassan Aït-Kaci. LIFE derives its syntax and resolution method from Prolog, and it is straightforward to modify Prolog programs to run under LIFE. However, the addition of functions with entailment (matching), generalized records (psi-terms), and type inheritance greatly enriches the language and allows one to formulate efficient programs more easily, more concisely, and more naturally.
- The Paradise project at DEC PRL was initiated by Hassan Aït-Kaci to develop the ideas around LIFE, including theory, implementation, and applications
- DEC PRL, represented by Paradise, joined the ACCLAIM EU project (1992-1995) to develop advanced concurrent constraint languages
  - Three languages were the core: LIFE (France), AKL (Sweden), Oz (Germany)

PERT scheduler in LIFE

- The problem is to find earliest and latest starting times of all tasks such that a particular task is completed as soon as possible.
- Uses object-orientation (definition of the sort ‘task’) to organize the data
- Uses coroutining to dynamically schedule all the operations (each function waits for its arguments)
- Shows the ability of LIFE for dynamic execution order and types for program structuring
  - A major upward step from Prolog that led to further advances in concurrency and constraints

```
% Define the sort (i.e. class) 'task':
:: A:task( duration => D:real,
           earlyStart => earlyCalc(R),
           lateStart  => {1e500;real},
           prerequisites => R:{[];list} )
   | !, lateCalc(A,R).

% Determine the earliest time that task A can start:
earlyCalc([]) -> 0.
earlyCalc([B|ListOfActs]) ->
        max(B.earlyStart+B.duration, earlyCalc(ListOfActs)).

% Determine the latest time that A's prerequisites can start
% and still finish before A starts:
lateCalc(A, []) -> succeed.
lateCalc(A, [B:task|ListOfActs]) ->
        lateCalc(A, ListOfActs) |
        assign(BLS:(B.lateStart), min(BLS, A.earlyStart-B.duration)).

% Wait until B is an integer before doing the assignment:
assign(A, B:int) -> succeed | A<-B.


?- A1=task(duration=>10), A2=task(duration=>20),
   A3=task(duration=>30, prerequisites=>[A1,A2]).
...
A3=task(duration=>30,
        earlyStart=>20,
        lateStart=>Infinity,
        prerequisites=>[A1,A2]).
```

Wild_Life 1.01



- Wild_Life interpreter
  - Richard Meyer, Peter Van Roy, Bruno Dumant, Jean-Claude Hervé, Seth Goldstein, Abder Aggoun, Arnaud Venet, Kathleen Milsted
- Wild_Life 0.91 released in Dec. 1992
  - First full implementation of Life language
  - Based on Richard Meyer's original implementation Wild_Life 0.0
  - Robust for rapid prototyping and building of compiler
  - Ported to DECstations, SPARCstations, RS/6000
- Wild_Life 1.01 released in April 1994
  - Extensive testing and polishing, with help from Andreas Podelski
  - Hassan Aït-Kaci, Bruno Dumant, Richard Meyer, Andreas Podelski, Peter Van Roy. *The Wild_LIFE Handbook*. DEC PRL, March 1994.
  - SuperLint, a user-customizable lint-like C checker written by Arnaud Venet in 10,000 lines of LIFE

Wild_Life contributions

The main abilities I added to the Wild_Life interpreter:

- The interpreter correctly implements functions that have non-linear heads in their definition.
- Removing memory leaks. A timestamping mechanism to reduce the size of the trail stack. Trail garbage collection.
- The handling of external events. Events from the X window system are smoothly integrated into the mechanism of residuation. File I/O is incremental, encapsulated in psi-term.
- The interaction of predicates and functions has been made simple and consistent through the definition of *predicate positions* and *function positions*.
- Numerous bug fixes and smaller improvements: garbage collector robustness, cycle detection in sort hierarchy, adaptive polling for X events.

Side activities and context

Aquarius Prolog 1.0

- Aquarius Prolog 1.0 released in April 1993
  - Full set of built-ins, based on Aquarius prototype in Peter Van Roy's Ph.D. thesis on high-performance Prolog implementation
  - Developers: Peter Van Roy at DEC PRL, Tom Getzinger at USC, and Ralph Haygood at UCBerkeley, SICS, and DEC PRL
- At its release, the world's highest performance widely available system (Parma by Andrew Taylor had similar efficiency but never made it beyond prototype stage)
  - Direct native code with no WAM-like intermediate stage
  - The compiler was able to compile itself and its built-in predicates
  - Sustained performance two to four times that of Quintus Prolog
- Peter Van Roy. *1983-1993: The Wonder Years of Sequential Prolog Implementation*. Journal of Logic Programming, 1994. DEC PRL Research Report 36.

comp.lang.prolog (Internet in 1993)

Prolog and software engineering (May 11, 1993)

- Consider the simple append/3 predicate we all know and love:

```
append([],L,L).
append([X|L1],L2,[X|L3]) :- append(L1,L2,L3).
```

- Simple, no? Now write this in C. Be careful about how you manipulate '&', '->', '*', don't get any of the types wrong. Make sure that your dynamic memory allocation library does what you want. And then, unless you implement variable-variable unification in C, your C version will only have a subset of the Prolog version's functionality. All this hacking is not difficult, but it shows my point: because unification has a simple semantics, it lets you avoid writing low-level details explicitly.
- Smaller, simpler code. Isn't this a software engineering advantage?

prl.paradise (1993)

Infinite objects in LIFE

- In playing around with wild_life, I've tried to create potentially infinite objects in as many different ways as possible. Let me know if you find any other ways. These examples run under my own version of wild_life, which has lazy unification and incremental type constraint checking.

1. Getting infinite objects through lazy unification:
   > f(N) -> [N|f(N+1)].
2. Getting infinite objects with quote and eval:
   > f(N) -> [N| `(f(M))] | M=N+1.
3. Getting infinite objects with type disjunctions:
   > integer -> {0;1+integer}.
4. Getting infinite objects with predicates:
   > integer(0).
   > integer(N+1) :- integer(N).
5. Getting infinite objects through type constraints: (TO BE DONE)
   > :: obj(level=>N,next=>obj(level=>N+1)).

Navigating the storms

Changes in the group (May 21, 1993)

- Digital's problems start to descend on PRL
  - DEC PRL has a reduction in budget
  - Hassan Aït-Kaci and Kathleen Milsted are dismissed
  - Peter Van Roy, Richard Meyer, Bruno Dumant, Abder Aggoun, Andreas Podelski, are kept
- Work on LIFE
  - Hassan's Paradise project becomes Peter's Proteus project, work on LIFE implementation and applications continues
  - Wild_Life interpreter 1.0 to be released
  - Life compiler in the works
  - Bruno and Richard working on their thesis
  - ACCLAIM deliverables to be changed

Acts of the Disciples (March 1994)

Email from Peter to Hassan:

- Most honorable prophet,
  Here's an update on the Acts of the Disciples

Compiler

- We have a LIFE interpreter written in LIFE, compiled under the current version of the compiler. So far, it runs at the same speed as Wild_Life. This is without indexing in either the interpreter or the compiler. The next step is adding indexing.
- The compiler is starting to be rather robust: it compiles the parser and writer, and a large set of sample programs. Richard is doing a great job.
- Still a lot to do: residuation--we have it all worked out, just need to add the instructions to the compiler, indexing—under development, sort encoding, analysis interface, fast built-ins, C interface.
- For built-ins, we have a single "master routine" for each built-in. When executed, it does two things: update the current type information and generate specialized code for the built-in. This master routine is used to generate code in the compiler and as the abstract operation in the analyzer. This generalizes nicely the scheme used in Aquarius.

Interpreter

- Andreas has been working on the formal operational semantics for LIFE. We are now knocking our heads over persistent terms. The manual's explanation of this is quite inscrutable--it should get better soon, and then they will deserve the epithet "clean".
- Wild_Life has now been ported to Alpha/OSF-1, MIPS/Ultrix, SPARC/SunOS, SPARC/Solaris, SGI/IRIX, PC/Linux, and PA-RISC/HPUX. We have "almost ports" (that is, most but not all works) to RS/6000/AIX, PC/NetBSD, and DOS (!!). Lots of friendly users are helping us out on this. Thirty users have taken the trouble to send us comments on the installation, and eight of them have done serious porting work for us, out of the blue! I am going to make a 1.01 "portability" release soon.
- Bruno's Emacs LIFE mode will be part of the 1.01 release.

Misc.

- Andreas and I are giving a two-day "LIFE days" seminar at the University of Nice on April 18-19. This includes a general talk, a foundations talk, and a hands-on course using Wild_Life 1.01.

Peter

Proteus/Paradise events (Apr. 1994)

Email from Peter to Michel Gangnet:

- Sep. 1993: Digital PRL develops Superlint
  - The Superlint tool allows arbitrary user-customizable checking rules for checking C programs. Superlint was developed with the Proteus project's prototyping system through an outside request from General Electric. This demonstrates feasibility of providing complex user-customizability in a program checker. Superlint was documented in Chip Nylander's Internal CASE report.
- Sep. 14, 1993: Digital obtains contract for multimedia project with Italian government
  - The Italian government signs a $800K contract for the development of a multimedia tutoring tool in a project called Merlin, lead by Carlo Tarantola (Consulting Engineer, Digital Valbonne). This project is done in collaboration with Peter Van Roy and Andreas Podelski (PRL) who helped with the specification of the tool and consulted on the use of high-level programming techniques.
- April 1994: Digital begins field test of TeMIP and DECss7 networking products
  - TeMIP is a network management platform, developed by the Telecommunications Engineering group. A key component of TeMIP Security was built with Aquarius, which was developed by Peter Van Roy at PRL. The DECss7 product is an implementation of the CCITT Signaling System Protocol for Intelligent Networks. The test environment of DECss7 was built with Aquarius.

Peter

Death and rebirth of LIFE project (June 1994)

Email from Peter to everyone:

- DEC is closing PRL. The LIFE project at PRL is discontinued. PRL will close end of Sep. 1994.
- There is a new project starting up at Simon Fraser University in Vancouver under the leadership of Hassan Aït-Kaci, the originator of LIFE
- We have a partial LIFE compiler, Half-Life, available to anyone upon request. Half-Life runs under Wild_Life and itself and handles the Login subset of LIFE (i.e., Prolog with psi-terms). It is self-compiling, generates high-quality C code, and is competitive in performance with emulated SICStus Prolog. Half-Life is primarily the work of Richard Meyer, aided by Bruno Dumant and Peter Van Roy. Half-Life has several remarkable abilities:
  - It uses a novel technique, the RISC-macro technique, to generate portable C code that approaches the speed of assembly code.
  - It implements psi-terms and psi-term unification with performance competitive to the best implementations of Prolog terms.
  - It is written using an accumulator preprocessor that much improves on the Extended DCG notation used in the Aquarius compiler.
- I'd like to thank all the past and present members of PRL's Paradise and Proteus groups for their contributions and enthusiasm: Hassan Aït-Kaci, who started it all, Andreas Podelski, Kathleen Milsted, Richard Meyer, Bruno Dumant, Arnaud Venet, Jean-Claude Hervé, Seth Goldstein, Gerard Ellis, Dinesh Katiyar, Jacques Garrigue, and all people interested in LIFE.

Peter

Transfer of ownership

- Request that Digital officially hand over the ownership and copyright of Wild_Life and Half_Life, the software and the documentation, to the following persons:

  Hassan Ait-Kaci, Bruno Dumant, Richard Meyer,
  Andreas Podelski, Peter Van Roy

(June 23, 1994)

Closing remarks

ACCLAIM project and its legacy

- DEC PRL, represented by Hassan Aït-Kaci's Paradise project, was a partner in the EU ACCLAIM project
  - ACCLAIM: Advanced Concurrent Constraint Languages, Applications, Implementation, and Methodology (1992–1995), Esprit project
- ACCLAIM was a key project for advancing concurrent constraint languages
  - LIFE: designed by Hassan Aït-Kaci (DEC PRL, France)
  - AKL: designed by Sverker Janson and Seif Haridi (SICS, Sweden)
  - Oz: designed by Gert Smolka (Saarland University, Germany)
- Eventually, the Oz language combined most of the good ideas of AKL and LIFE and led to a robust system, Mozart, that was widely used (and is still used today, mostly for teaching)
  - SICS dropped AKL and joined Saarland University to work on Oz
  - Peter Van Roy joined Saarland University after the closing of DEC PRL
  - Peter Van Roy *et al*. *A History of the Oz Multiparadigm Language*. 4th ACM SIGPLAN HOPL conference. PACMPL, 4(83), June 2020.

My habilitation and debt to Hassan

- “Habilitation à Diriger des Recherches” (HDR) is a degree obtained after the Ph.D. in France
  - It is required in France in order to supervise Ph.D. students
- Hassan Aït-Kaci encouraged me to work on this degree during my stay at DEC PRL
  - Hassan’s influence was instrumental in transforming the naïve fresh Ph.D. that I was into a solid computer scientist
- In 1996 I finally obtained the HDR
  - Peter Van Roy. *On the high-performance implementation of logic languages*. Habilitation à Diriger des Recherches, Université Paris VII Denis Diderot, Oct. 1996.
  - In Oct. 1996 I officially joined the faculty of the Université catholique de Louvain (Belgium), where I am still today