J-G Ganascia

L'époque homérique 1980 - 1990

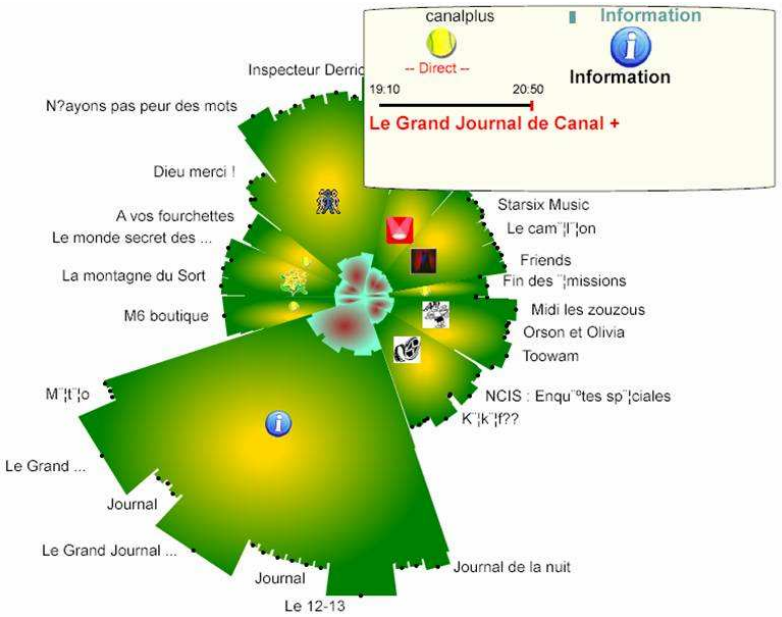
Apprentissage artificiel
La carte, le territoire et l'horizon

En l'honneur d'Yves Kodratoff

La carte en « île de mémoire »

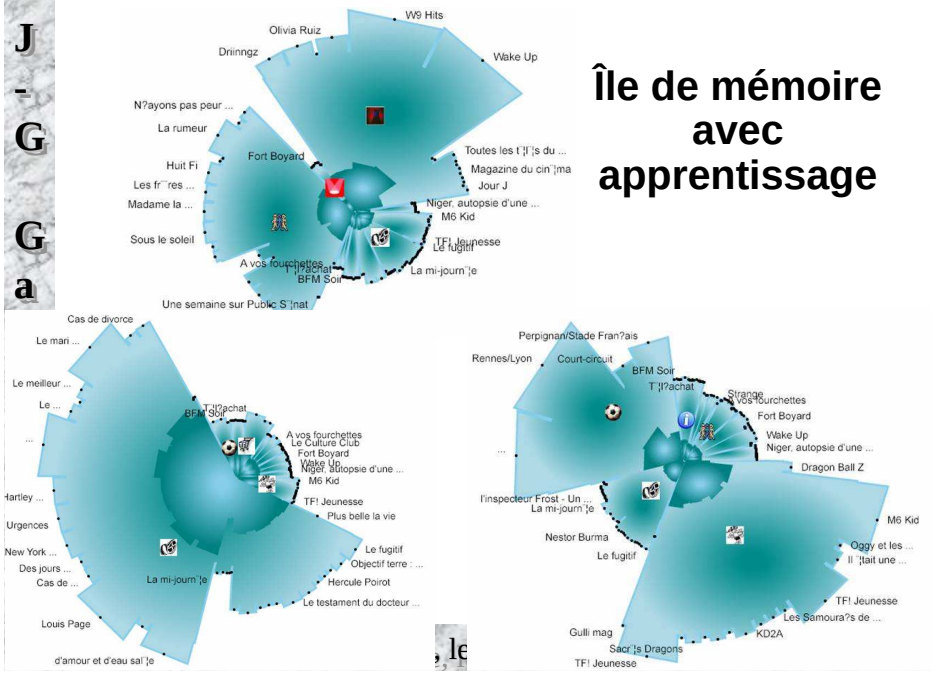
J-G Ganascia

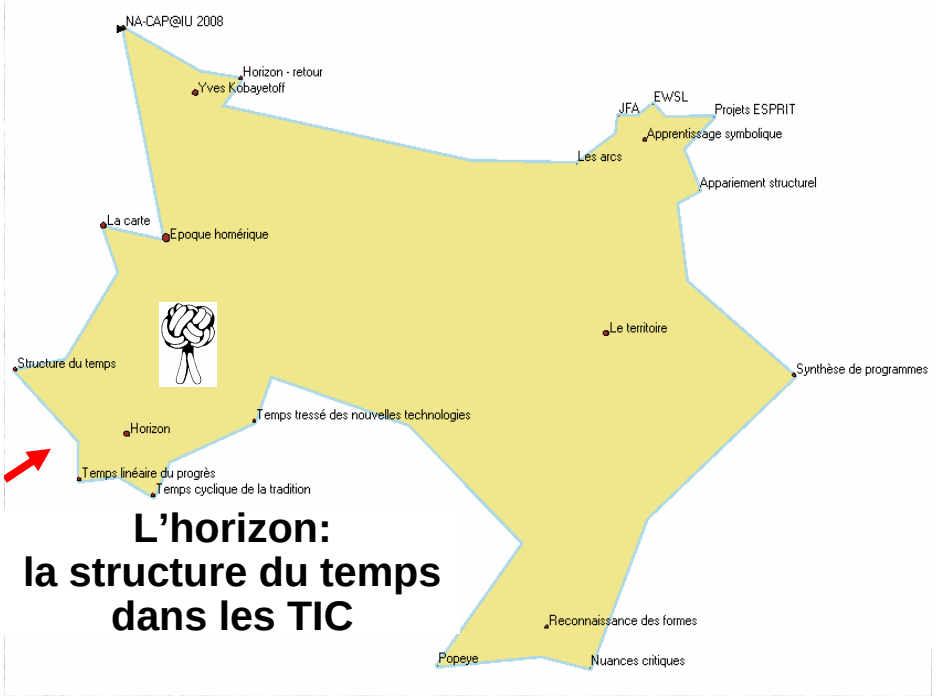
Application à la TV interactive



J-G Ga

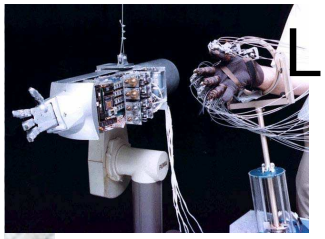
Île de mémoire avec apprentissage





L'horizon: la structure du temps dans les TIC

L'horizon…



Est-ce que le temps des nouvelles technologies est linéaire ou cyclique?

Très souvent, des choses neuves sont déjà vielles et des choses vielles redeviennent neuves!

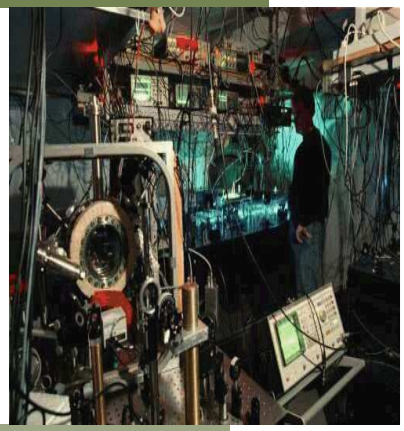
Prédictions à court terme erronées

- **Echec de la première cybernétique**
  - **PERCEPTRON de Rosenblatt 1958: limitations – Minsky & Pappert 1969**
- **Echec de la traduction automatique**
  - **Rapport ALPAC 1966**
- **Intelligence Artificielle**
  - **Echec des prédiction de H. Simon et A. Newell**
  - **Echec de l'IA sémantique et de la représentation des connaissances**
  - **Déconvenue des systèmes experts et de l'ingénierie des connaissances**
- **…**

Prédictions à long terme réalisées

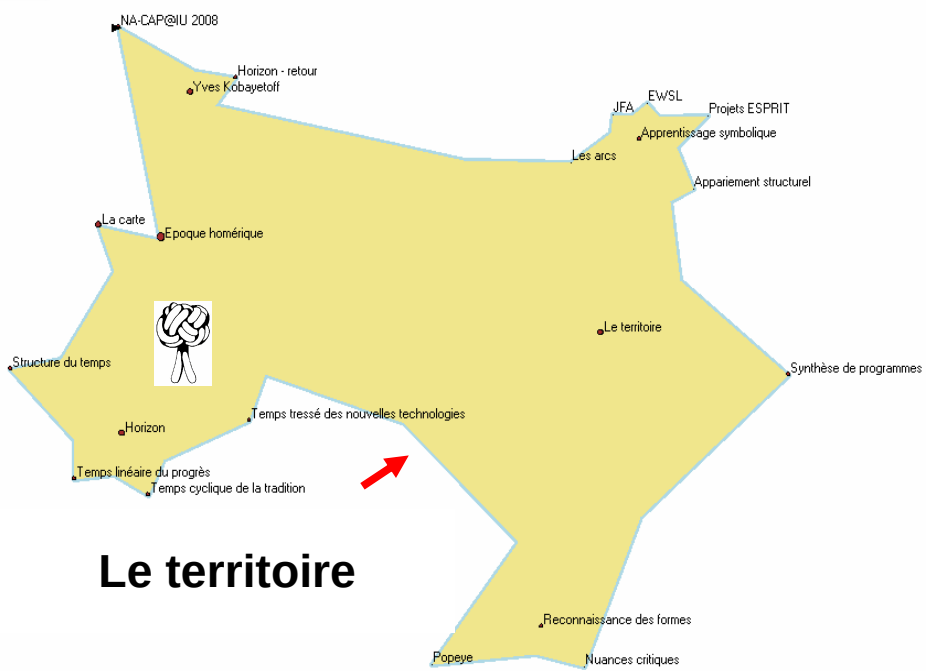
- **La MEMEX – Vanevar Bush - 1945**
- **Le test de Turing Test – 1950**
- **La loi de Moore 1965**
- **Douglas Engelbart**
  - **The Mouse: 1963**
- **Jean-François Lyotard**
  - **« La condition du savoir dans les sociétés post-modernes » - 1980**
- **Cybernétique - 1943**
  - **Les réseaux de neurones - 1982**
- **Le web sémantique**
- **La traduction automatique!**
- **…**

La structure <u>tressée</u> du temps dans les nouvelles technologies



Temps linéaire: temps du progrès

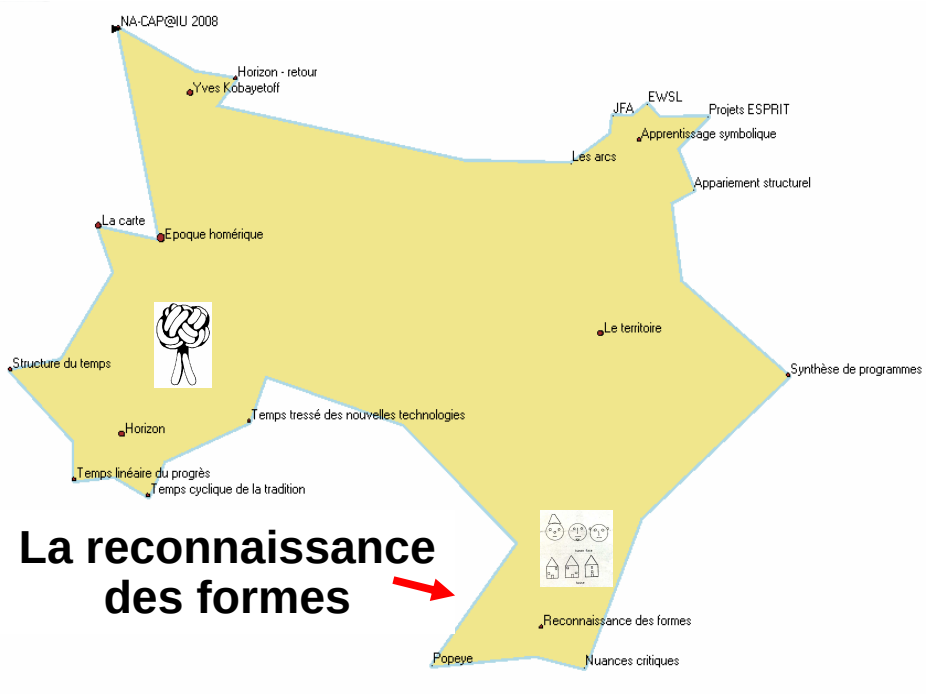
Temps cyclique: temps de la tradition



Le territoire

Les terres…

- **Synthèse de programme**
- **Reconnaissance des formes**
- **Programmation logique**
- **Traitement automatique de la langue naturelle**
- **Apprentissage automatique**
- **Sciences cognitives**
- **Appariement structurel**
- **Programmation logique inductive**
- **Acquisition des connaissances**
- **Fouille de données**
- **Fouille de textes**
- **…**



La reconnaissance des formes

Reconnaissance des formes - AISB 1978

Projet Popeye

Proc. AISB/GI Conference Hamburg July 1978

REPRESENTATION AND CONTROL IN VISION(*)

by

Aaron Sloman, David Owen, Geoffrey Hinton, Frank Birch, Frank O'Gorman
Cognitive Studies Programme,
School of Social Sciences,
University of Sussex,
Brighton, U.K.

(*) The Project is funded by the U.K. Science Research Council on grant no B-RG-8688-7. Frank O'Gorman has now left the group.

Is AI vision research making good progress?

Vision work in AI has made progress with relatively small problems. We are not aware of any system in which many different kinds of knowledge co-operate. Often there is essentially one kind of structure, e.g. a network of lines or regions, and the problem is simply to segment it, and/or to label parts of it. Sometimes models of known objects are used to guide the analysis and interpretation of an image, as in the work of Roberts (1965), but usually there are few such models, and there isn't a very deep hierarchy of objects composed of objects composed of objects ... By contrast, recent speech understanding systems, like HEARSAY (Lesser 1977, Hayes-Roth 1977), deal with more complex kinds of interactions between different sorts of knowledge. They are still not very impressive compared with people, but there are some solid achievements. Is the lack of similar success in vision due to inherently more difficult problems?

Some vision work has explored interactions between different kinds

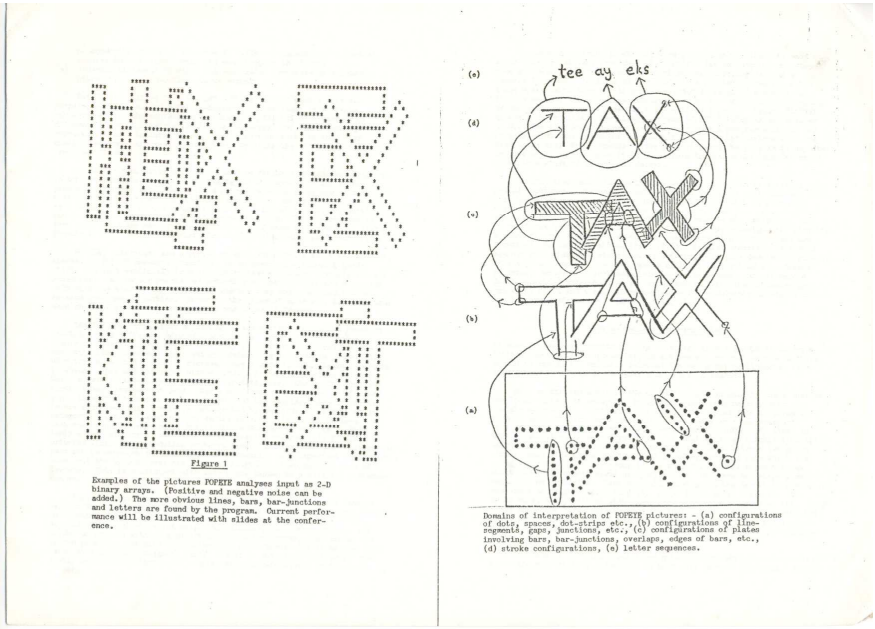


Figure 1

Examples of the pictures POPEYE analyses input as 2-D binary arrays. (Positive and negative noise can be added.) The more obvious lines, bars, bar-junctions and letters are found by the program. Current performance will be illustrated with slides at the conference.

Domains of interpretation of POPEYE pictures: - (a) configurations of dots, spaces, dot-strips etc., (b) configurations of line-segments, gaps, junctions, etc., (c) configurations of plates involving bars, bar-junctions, overlaps, edges of bars, etc., (d) stroke configurations, (e) letter sequences.

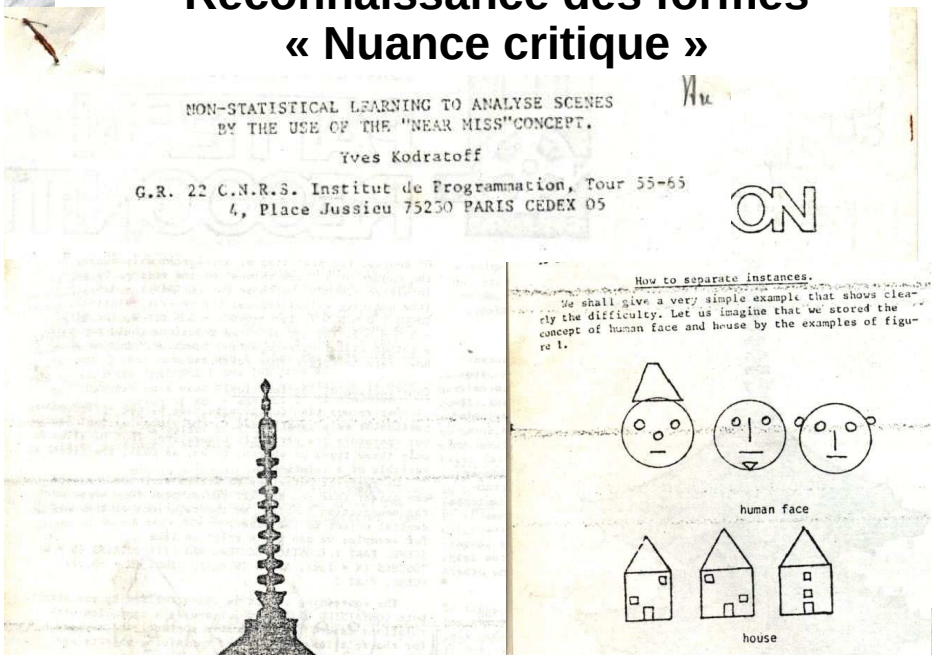
J-G Ganascia

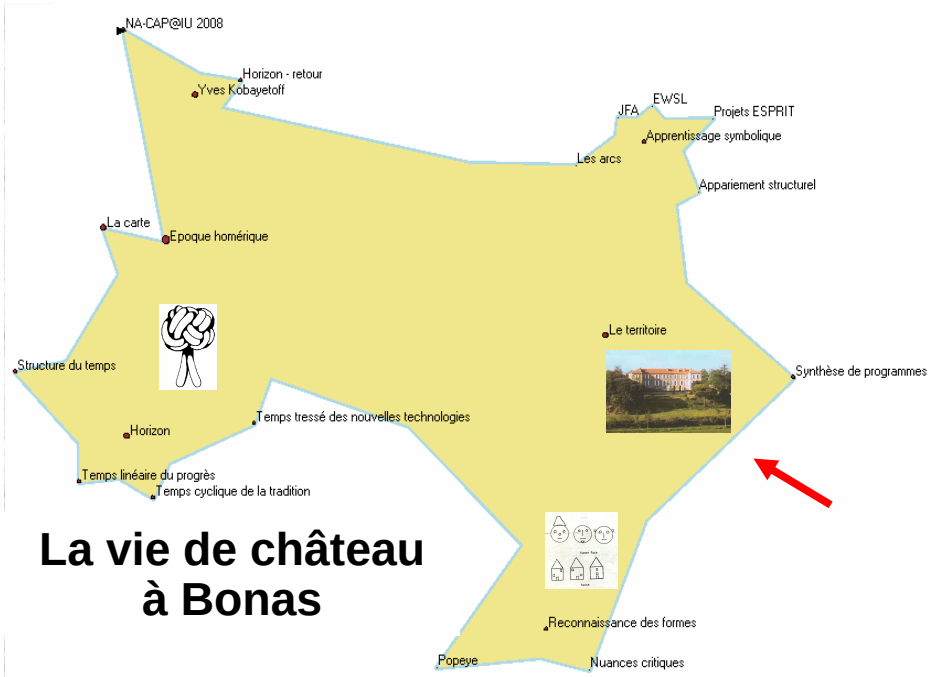
Reconnaissance des formes « Nuance critique »

NON-STATISTICAL LEARNING TO ANALYSE SCENES BY THE USE OF THE "NEAR MISS"CONCEPT.

Yves Kodratoff

G.R. 22 C.N.R.S. Institut de Programmation, Tour 55-65
4, Place Jussieu 75230 PARIS CEDEX 05





La vie de château à Bonas

J-G Ganascia

Computer Program Synthesis Methodologies

Proceedings of the NATO Advanced Study Institute held at Bonas, France, September 28 - October 10, 1981

edited by
ALAN W. BIERMANN
Duke University, Department of Computer Science, Durham, NC, U.S.A.
and
GÉRARD GUIHO
Laboratoire de Recherche en Informatique, Université de Paris-Sud, Orsay, France

D. Reidel Publishing Company
Dordrecht : Holland / Boston : U.S.A. / London : England
Published in cooperation with NATO Scientific Affairs Division

PREFACE

Ever since the early days of computing, programmers have been seeking ways to automate portions of the programming task. The first major step was the invention of assemblers which allowed mneumonic operation codes,handled certain addressing tasks and offered other functions. Next came a vast array of compiled and interpreted languages which implemented many data structures, flow-of-control features, and notational conveniences. In an attempt to continue this trend today, a number researchers are developing program synthesis methodologies which aim at making further automation possible. Most of these methodologies fall into one of the categories :

a) extensions to traditional programming methods,
b) synthesis of programs from examples,
c) synthesis from formal input-output specifications,
d) program construction from natural language dialogue.

These program synthesis methodologies are the subject of this book.

The first three chapters examine the program synthesis issue largely without regard to the automation problem. Systematic means are described for constructing programs with emphasis on what is the nature of the synthesis task and how can it be approached. M. BROY describes a transformational technique for deriving programs from their specifications. M. SINTZOFF describes central issues in the design of concurrent programs, and R. BACKHOUSE gives a methodology for decision making in program design that accounts both for correctness and efficiency.

The later chapters of the book sequentially address the four approaches to synthesis given above. Extensions to traditional programming methods often involve the construction of very high level languages to deal with larger objects (data structures) than traditional languages with appropriately more

Éléments sous droits d'auteur

PREFACE

Ever since the early days of computing, programmers have been seeking ways to automate portions of the programming task. The first major step was the invention of assemblers which allowed mneumonic operation codes,handled certain addressing tasks and offered other functions. Next came a vast array of compiled and interpreted languages which implemented many data structures, flow-of-control features, and notational conveniences. In an attempt to continue this trend today, a number researchers are developing program synthesis methodologies which aim at making further automation possible. Most of these methodologies fall into one of the categories :

a) extensions to traditional programming methods,
b) synthesis of programs from examples,
c) synthesis from formal input-output specifications,
d) program construction from natural language dialogue.

These program synthesis methodologies are the subject of this book.

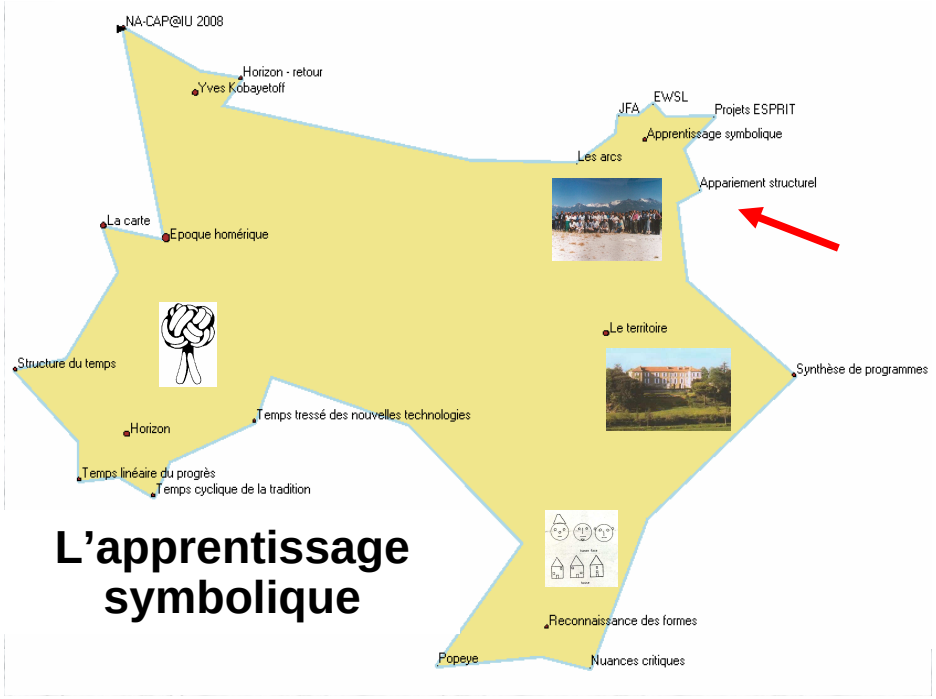
J-G Ganascia

Contenu de l'ouvrage…

- **Chapitres 1.. 3: systématisation de la construction, sans automatisation**
  - **Transformation à partir de spécification**
  - **Programmation concurrente**
  - **Conception de programmes**
- **Chapitres 4..6: langage de haut niveau**
  - **APL, SETL and BDL**
  - **PROLOG**
  - ***Programmation à partir d'exemples***
- **Chapitre 7: construction à partir de spécifications formelles**
- **Chapitre 8: interaction avec le langage naturel**

J-G Ganascia

There are many situations in machine programming where it is more natural to give examples of what is wanted than to give programs. As an illustration, the operator of an automatic machine tool would probably prefer to manually direct the device through an example machining sequence and have it self program to do that sequence than to code the sequence using a symbolic language. The machine operator can be sure in the example sequence that all processes occur as desired and will probably produce a more reliable control program than if he or she had written the program directly. As a second illustration, a person formating a piece of text would probably much rather organize the page using a manual graphics device of some kind and then have an automatic editor create the command sequence to actually space out and format the text items correctly. A fascinating domain for these kinds of studies has been in the synthesis of LISP programs from examples of their input-output behaviors. J.P. JOUANNAUD and Y. KODRATOFF have included a chapter describing some of this work.



L'apprentissage symbolique

J-Ganascia

Apprentissage symbolique

- **Ecole de printemps d'apprentissage 1983**
- **Notion d'appariement structurel**
- **…**
- **Programmes européens sur l'apprentissage**
- **EWSL, JFA, …**
- **Ecole d'été des Arcs**
- **…**

J-Ganascia

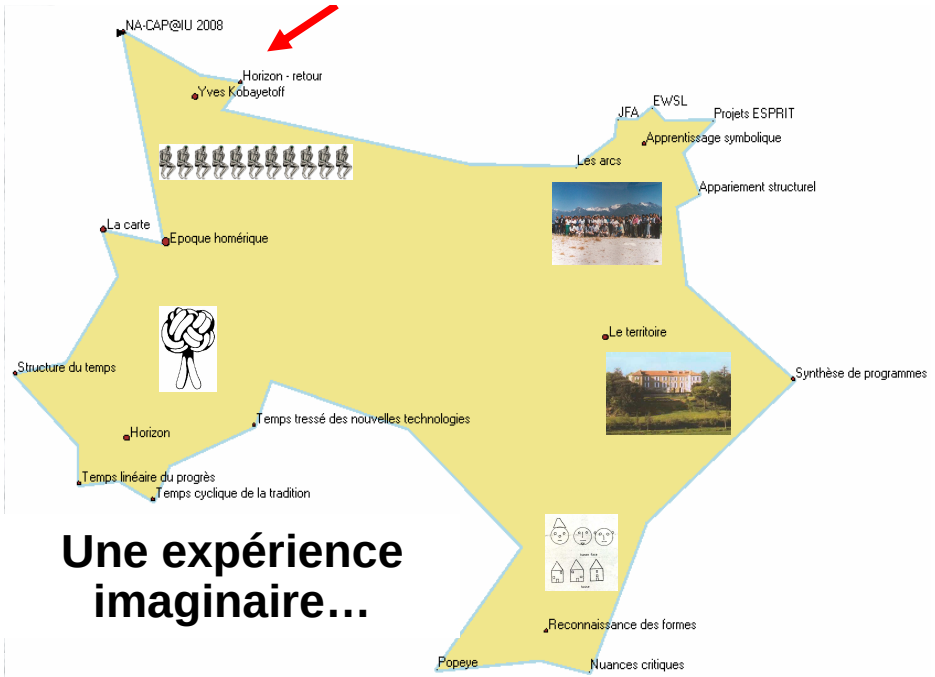
L'horizon – suite

PROLOG

+

synthèse de programmes à partir d'exemples

Programmation logique inductive



Une expérience imaginaire…

J-G Ganascia

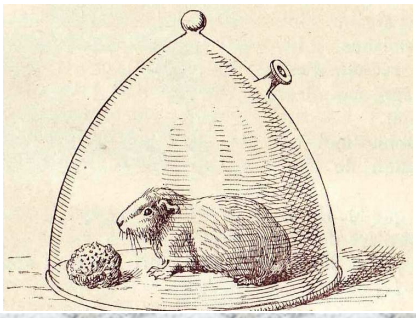
L'horizon – fin

Une expérience de pensée

Yves Kodratoff
en cobaye

→

Yves Kobayetoff
Chercheur libre



NA-CAP@IU The Limits of Computation 2008

Home Program Submissions Registration Travel & Lodging Local Attractions

The 2008 North American Conference on Computing and Philosophy

NA-CAP@IU 2008: The Limits of Computation

Submission Instructions

Deadline: March 1st, 2008

This year, NACAP will be using a system for electronic submission and review. Please submit your papers and proposals online at http://ia-cap.org/na-cap08/openconf/openconf.php. Specific guidelines for content appear below.

Please note that proposals for the **Special Session on Automatic Programming and Human Creativity** should not be submitted via this Openconf link. Guidelines for submission to this session are included in the CFP below.

CFP for the Special Session on Automatic Programming and Human Creativity

From session organizers Selmer Bringsjord (selmer@rpi.edu) and Konstantine Arkoudas (konstantine@alum.mit.edu):

Please consider submitting to present in the special track "Automatic Programming and Human Creativity," at the North American Computing and Philosophy Conference, to be held at Indiana University July 10-12, 2008.

The conference theme is "The Limits of Computation," and we are particularly interested in how study of human creativity might allow for the current limits on automatic programming (a venerable area of AI devoted to producing computer programs that automatically write significant computer programs) to be exceeded. Prior techniques (such as deductive program automation and genetic programming) have had little success.

This track is partially supported by a grant from the National Science Foundation, through its CreativeIT program. This grant makes possible an exploratory investigation of human creativity in the area of computer programming, with the hope of exploiting study of human creativity in order to eventually make significant contributions to automatic programming. In addition, this investigation should pave the way toward educational technology to better teach computer programming, and to encourage a greater percentage of talented youth to pursue careers in, and based on, programming. Our perception is that many young students often incorrectly conceive of computer programming as the sterile antithesis to "creative" careers in, say, entertainment and the arts.

In order to try to break through present limits, we are particularly concerned with understanding what conditions are conducive to discovering highly innovative programming solutions. Provided with a description of the sorting problem (arranging a given list of numbers in increasing order), for instance, most computer science undergraduates could easily come up with the naive algorithm for solving the problem. But what does it take to come up with an algorithm as brilliant as quicksort, or as clever as mergesort? Is it mostly a mixture of luck and unanalyzable - almost mystical - genius? Or are there specific enabling patterns and conditions that usually obtain and which can facilitate creative programming work?

Sample Topics

The following is not an exhaustive list, but gives some samples.

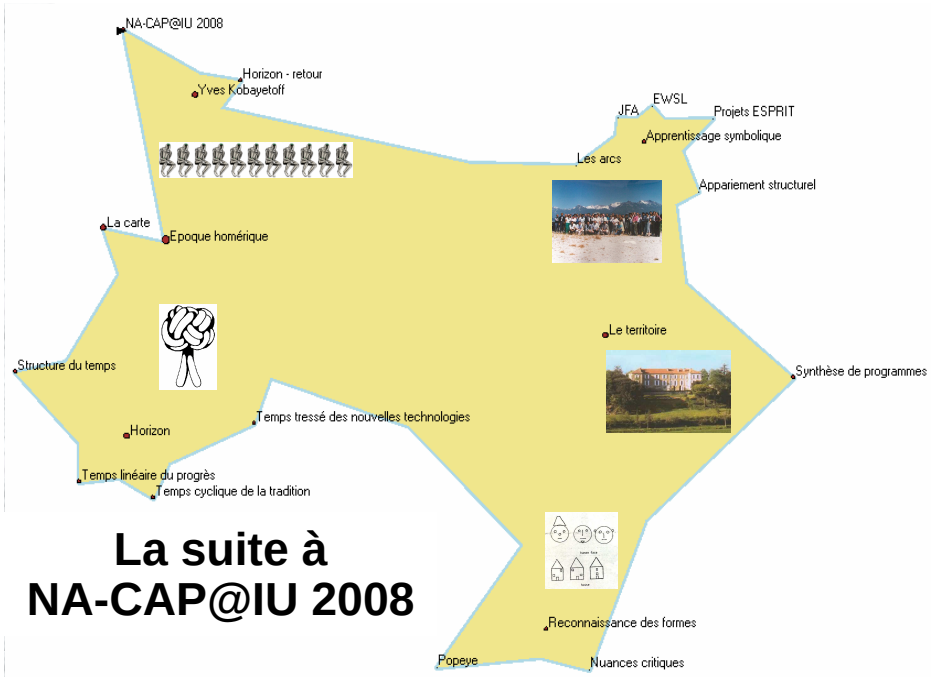
Case studies in human creativity and computer programming.

Application of prior work in AI and creativity to the automatic programming problem.

New approaches to automatic programming based on systematic study of human ingenuity, discovery, and creativity.

What is the role of diagrammatic thinking and reasoning in visualizing data structures and transformations on such data structures during the creative/exploratory part of programming?

What is the role of non-deductive reasoning in programming creativity?



La suite à NA-CAP@IU 2008