**Seminar F4: Future directions and projects**

Research areas: Programming Paradigms, Applications, Foundations, SE

Slide for different areas + James' quote + Brodner quote

1. Software Engineering

- In what sense do we have a software development method? Our work so far oriented towards reactive systems; work elsewhere addresses data intensive applications. [Many existing tools being developed along these lines in the UIMS, data-base and spreadsheet culture.] Paul Ness collaborated with IBM on OOA comparative study. PhD thesis: Creative Software Development: an Empirical Modelling Framework. Adzhiev and Richlinskey's LSD Engine (1997).

  Concerns to connect with formal specification. Connections with Z, CSP, CCS etc still hazily understood. Also origins of LSD in SDL. Would be good to evaluate in comparison with Lotos, SDL and Estelle. Other work related in the real-time and concurrency area: French school, StateCharts, UNITY, (Numerical) Petri Nets. Relationship to protocol specification and testing.

2. Programming Paradigm

- The potential applications of our approach to abstract programming have yet to be fully explored. Our approach doesn't put the emphasis on control and algorithmic issues in the first instance. Is state-based though, and can be applied to modelling real/abstract machine models. In particular, intend to explore its relevance to portability concerns and to parallel machine models. Prototyping from empirical models is of particular interest here. Potential as development method for general-purpose programs to be considered.

  Relevance to complexity theory, where there is a significant lack of appropriate ways to measure performance formally. Want to count abstract operations generally, so declarative languages are useless here, but procedural languages do not necessarily have the right expressive power. Not enough to construct abstract data types and calculate numbers of abstract operations. Of particular interest are complexity measures that concern maintenance of dynamic data structures. Possible interest in exploring definitive models in that area.

  Abstractions to be more fully analysed and exploited

  Higher-order definitions (EG UK'96 + Gehring)

  Data structures (cf heapsort, ARCA, CADNO + R I Cartwright)

Examples of interesting challenges for definitive programming:

1) write an LR parser in such a way that the finite state machine associated with the grammar is specified by a definitive script.

2) write a text editor (developing Edward Yung's Unconventional Text Editor, the first EDEN program to be devised)

3) write definitive implementations of definitive software tools

3. AI and Applications

Design (especially for engineering) is an area of particular interest.
Geometric Modelling for VR

Neural networks.

Semantics of semantics of programs: form vs content. Reference information.

Cognitive technology: the empiricist perspective on learning (Aizu 97)
Cognitive artefacts

New architectures: DAM and JaM
(DoNaLD-DAM compiler Allderidge, see Allderidge et al 1997)

Generic templates for interfaces (Yvonne Lui 1997)

Business applications (Casa Chung 1997, Soha Maad)

Systolic array modelling (Pat Sockett)

Statechart construction and animation (Rob Carpenter)

Eden-Oracle interface (Son Truong)

Graphical editors (Simon Yung, C K Leung, MacDonald, M Roberts)

4. Foundations

How to express "all"! notions of CS in terms of semantics we are developing for LSD and definitive notations. Examples of concepts we can deal with to some extent:
agent, process, object, observation, experiment, program, execution.
More work needed on synchronisation, data types (specify the stack etc), time, constraints, hypermedia ...

How to represent models? Jaratsri's lines, chcruise etc