P A N E L

Parallel Processing in Computational Linguistics

Helmut Schnelle
Ruhr-Universität Bochum
Sprachwissenschaftliches Institut
Postfach 102148, D-4630 Bochum 1

P a n e l i s t s:

Garry COTTRELL
University of California, Dept. of Computer Science, San Diego, Mail Code C-014, La Jolla, CA 92093
U.S.A.

Paradip DEY
The University of Alabama at Birmingham
Dept. of Computer & Information Science
Birmingham, AL 35294
U.S.A.

Joachim DIEDERICH
International Computer Science Institute
1947 Center Street, Berkeley, CA 94704
U.S.A.

Peter A. REICH
Dept. of Linguistics, University of Toronto, Toronto, Ontario M5S 1A1
CANADA

Lokendra SHASTRI
University of Pennsylvania
School of Engineering and Applied Science, 200 South 33rd Street
Philadelphia, PA 19104-6389
U.S.A.

Akinori YONEZAWA
Tokyo Institute of Technology
Dept. of Information Science
Ookayama, Meguru-ku, Tokyo 152
JAPAN

Introduction

The topic to be discussed by the panel is new and at present very much under debate. Parallelism is developed in a large variety of approaches. The panel will make an attempt to clarify the underlying concepts, the differences of approach, the perspectives and general tendencies, and the difficulties to be expected. Some differences of approach will be illustrated with examples from the work of the panelists.

The common context of our approaches is the following:Standard computational linguistics tries to solve its problems by programming a von Neumann computer. The execution of the programs is inherently sequential. This is implied by the fact that there is only one central processing unit (CPU) executing the program. In contrast to this, parallel processing defines the solution of problems in terms of sets of computational units which operate concurrently and interactively, unless sequentialized for simulation purposes.

Various approaches to parallelism differ in the computational power they assume for the concurrently active units. The differences may be outlined as follows:

Massively parallel systems are usually systems whose units are, intuitively speaking, purely reactive units, i.e. mathematically defined by a specific function relating the state and output of a unit to its inputs. They could also be called connectionist systems in the wide sense; connectionist systems in the narrow sense are those whose functions are based on weighted sums of input activities.
In contrast to these systems, the units may be themselves complicated systems which compute their states and outputs depending on the messages and control signals which they receive. The units cooperate in solving the problem. In typical cases, each unit may be a central processor unit or even a complete computer. Systems with cooperative processors (computing "agents") are usually considered to be non-massively parallel.
These distinctions suggest different metaphors used in informal talk about the systems: the neural net metaphor on the one hand and the society of minds (demons) metaphor on the other.

Given this context the panelists have answered the following questions:

I. How is the dynamics of your system defined?
- I.(A) 1. What is the computational power of a single unit in your approach
- I.(A) 2. How is the interaction or the interdependency between concurrent units defined?
- I.(B) How do you implement your system?
- I (C) Which methods are used for programming?

II. What is the representational status of your system?
- II.(A) Which parts of grammar or dictionary do you model with your system?
- II.(B) Which parts of grammar or dictionary do you model by a concurrent unit of your system?
- II.(C) Is there a general method such that a grammar determines uniquely a parallel implementation or is this implementation an art?

The answers given seem to be particularly appropriate as an introduction to the topic and will thus be presented in the subsequent passages.(The answers of the different panelists and the organizer are prefixed by their initials)

I. How is the dynamics of your system defined?

I.(A) 1. What is the computational power of a single unit in your approach (a Boolean function, a specific numerical function, a mapping of vectors, a mapping of strings or files, a mapping of trees or configurations of other types, or the power of a CPU or a complete computer)?
G.C.:There are no formal limitations on the power of a unit in my system; the power is a matter of taste, and is expected to be restricted to simple functions. For example a numerical approximation to Boolean functions of the inputs, where the inputs are further broken down into functions of input sites. My implemented system has several hundred units.
P.D.:Each unit has the power of a VAX-11/750.The units share their memories. I'm thus currently working in a shared memory multiprocessing environment. Specifically, my algorithms run on a 30 processor (=unit) Sequent Balance 21000 machine. This is large grain parallelism. I prefer the environment of large grained shared memory multiprocessors, because they are the most popular general purpose parallel computers available today. Earlier, I developed some algorithms for a medium grained tree machine, namely the DADO parallel machine.
J.D.: Each unit implements a simple numerical function, sometimes a simple combination of several functions computing input from several sites of incoming activation.
P.A.R.: Each unit has the power of a finite state device (under 32 different states).There are 16 different types of units which differ in their finite state definition. They implement (sometimes only slightly) different logical functions over their input activations. The more important ones are: concatenation (logical followed by), conjunction (logical and), disjunction (exclusive or), precedence disjunction (if both possibilities are realizable, one takes precedence over the other), random disjunction (pick a choice at random),interjunction (inclusive or), intercatenation (inclusive or; if both: concatenation), zero (network dead ends producing nothing), bottom edge (network outputs something), top edge, feedback barrier. Units operate independently of one another and asynchronously.
H.S.: There are two descriptive levels: large grained and small grained. On the former each unit is a (special purpose) Turing machine (not a universal one). On the small grained level, each unit implements either a simple Boolean function or a simple numerical (additive, fixed length) function. The large grained net is a partitioning of the small grained net; its Turing machines are similar to von Neumann's growing cellular automata.
L.S.: Each unit implements a numerical function - the most complicated ones have the form: If input $a_1 > 0$, then take the product of inputs $b_1, b_2, \ldots b_m$, else take the product of inputs $c_1, c_2, \ldots c_m$.
A.Y.: Each unit is a single CPU with memory. My approach involves thousands of units.

I.(A) 2. How is the interaction or the interdependency between concurrent units defined? Is it strictly connectionist and thus also defined by a function? Or is it cooperative and thus defined by the messages sent, encoded and decoded by the units? Is there a distinction between data messages and control-signal messages or is it a data-flow system?
G.C.: Units pass values in a strictly connectionist way.
P.D.:The system is a shared memory system. All units have in principle access to the same information. Actual interaction is defined by shared variables. That is, processes communicate with each other through shared variables.
J.D.: The system is strictly connectionist, i.e. there are no symbolic messages. Each unit computes the weighted sum of inputs.
P.A.R.:Each unit is connected to at most three other units in the network. The connections are active or not. According to their function, three different signals may be distinguished: production signal and positive feedback, negative feedback, and anticipatory.
H.S.: On the small grained level the system is connectionist, but not strictly, since not only weighted sums of inputs are allowed but also other simple functions.
L.S.: The system is strictly connectionist.
A.Y.: The interaction between units involves message passing. Messages carry either control information or data or both.

I.(B) How do you implement your system? By simulation on a von Neumann computer or by programming on a universal parallel machine (like the connection machine) or by designing hardware (e.g. a special-purpose information processing network)? If the first, do you plan to implement it eventually by a parallel system?
G.C.: The system is simulated on a VAX.
P.D.:The system is being implemented on a 30-processor Sequent Balance 21000 machine. It is currently being implemented in parallel-C running under Unix. When a parallel LISP becomes available, it will be implemented in parallel LISP.
J.D.: We use the Rochester Connectionist Simulator on a SUN-3 with Graphics Interface. Implementations on a Sequent (Parallel Unix Machine) are planned.
P.A.R.:Simulation on a personal computer using standard programming language.
H.S.: The connectionist net is defined on a spread-sheet such as LOTUS 1-2-3. Some cells of the spread-sheet are identified with the units of the net to be programmed. In each of these cells a formula for a function is entered; it determines the reactivity of this cell to the states of those neighbouring cells whose addresses are arguments of the function. Thus, the addresses of the formulas on the spread-sheet implement the connectivities between the formulas. We run the spreadsheet in the computation-mode: iterative,columnwise, which defines the sequential simulation. By definition the different cells of the spread-sheet could operate concurrently in each iterative step; their operation is sequentialized (and thus adapted to the simulation on PC) only through columnwise computation.
L.S.:By simulation on a von Neumann computer.
A.Y.:By simulation of a von Neumann computer, and also parallel computers

I (C) Which methods are used for programming? Parallelizing of existing non-parallel programs or independent programming? Methods of hardware design?
G.C.: A network is constructed from a high-level specification such as a grammar. This is given to a network construction routine that specifies the model based on the grammar.
P.D.:The computational model is MIMD (multiple instruction multiple data stream). Parallel programs are developed primarily by data partitioning, although function partitioning is also used.
J.D.: Independent programming. Networks are constructed by writing a C program and use of library function of the simulator.
P.A.R.: The system is programmed by constructing the grammar in network form. There is an algorithm for representing the network in terms of algebraic formulas. Nodes are defined by a series of state transition rules. The grammar is tested by inserting initial input signals and running the simulation.
H.S.: There is a compiler which produces automatically for any given CF-grammar a corresponding network. The processes on the network correspond to the processes defined by an Earley chart parser but, in contrast to the latter, all processes are executed concurrently whenever this is possible. In particular, all parsing paths are followed up in parallel. Hardware design of networks is planned.
L.S.: A "compiler" is provided that translates a high level specification of a conceptual structure (semantic network) into a connectionist network. It is proved, that the network generated by the compiler solves an interseting lass of inheritance and recognition problems extremely fast - in time proportional to the depth of the conceptual hierarchy.
A.Y.:We designed an object-oriented concurrent language called ABCL/1 and program parsers in this language.

I.(D) Is your system fixed or does it learn ? If the latter, which learning functions or learning algorithms are used?
J.D.:Learning is the most important topic. Natural language descriptions of structured objects are learned. These objects are also present in a restricted visual environment. The interaction between language and vision in learning is investigated. Various forms of weight changes are used: Hebbian learning with slow weight change, fast weight change for temporary binding, modified Hebbian learning with restriction on the increase of weights.
P.A.R.: A substantial number of learning rules have been developed but not yet implemented on computer. Learning involves "ingestion" and "digestion". Ingestion consists of co-occurrence rules. If two signals previously unconnected co-occur, they are connected together. Digestion makes use of equivalence relationships to simplify the network. Equivalence relationships include: associativity, commutativity, distributivity, and a number of other relationships which have no name in standard algebra. Ingestion and digestion operate more or less alternately. First a piece of new information is connected to the network, then equivalence relations are tried in a search for simplification.
L.S.: Structure is fixed but weights on links can be learned using a Hebbian weight change rule.
G.C.,P.D.,H.S.,A.Y.: Our systems do not learn.

II. What is the representational status of your system?

II.(A) Which parts of grammar or dictionary do you model with your system?
G.C.:I have separate systems designed to work together to handle lexical access, case-grammar semantics, and fixed-length context free grammar.
P.D.:Lexicon, grammar and semantics. The lexicon has words with their categories, subcategories, and lexical meaning.
J.D.:Fixed-length context-free grammar.
P.A.R.:In theory the entire system from a representation of general cognitive information through language specific "deep" or "functional" structure, through a syntax-morphology structure, and then through a phonological structure. In actuality, the syntax-morphology and phonology sections have been worked out in greatest detail, and the functional structure in bits and pieces.
H.S.:Syntax and phonology as a part of a lexical access system.
L.S.: Domain knowledge in terms of a hierarchy of concepts/frames - where each concept is a collection of attribute-value (or slot/filler) pairs. Such information structures are variably referred to as frame-based languages, semantic networks, inheritance hierarchies, etc.
A.Y.:Syntax and some semantics.

II.(B) Which parts of grammar or dictionary do you model by a concurrent unit of your system?
G.C.: I use a localist approach: One unit stands for a word, a meaning, a syntactic class, and a binding between meanings and roles, syntactic and semantic.
P.D.: Parts of syntax; lexical search is also parallel
J.D.: Localist representation, i.e. one syntactic category - one unit
P.A.R.:Each category (such as noun phrase) is distributively represented by many units.
H.S.:(Localist; on small grained level:) Each occurrence of a category-in-rule-context (a dotted rule in Earley's parser definition) is represented by a unit.(On the large grain level:) The set of possible small grain units of each category corresponds to a Turing machine, such that one of its units represents the current state of the "head" of the TM and the others its "tape".
L.S.:(Localist:) A unit may "represent" a concept, an attribute, a value, a binder between <concept,attribute,value> triples, or control nodes that mediate and control the spreading of activation among these units.
A.Y.:(Localist:) Each grammatical category is represented as a unit, actually each occurrence of each category in a grammar description is a unit.

II.(C) Is there a general method such that a grammar determines uniquely a parallel implementation or is this implementation an art?

G.C.:Given a grammar, I have an algortihm to generate the network for that grammar.

P.D.:Parsing algorithms are developed for Tree Adjoining Grammars.

J.D.: Implementation is still an art.

P.A.R.: To a certain extent it is an art, at this point, but the comprehension-acquisition rules, if successfully implemented, should provide the general method.

H.S.: Writing grammars as high-level specifications is an art. From there on there is a general method (same answer as L.S.)

L.S.:The networks are constructed from a high-level specification of the conceptual knowledge to be encoded. The mapping between the knowledge level and the network level is precisely specified. This mapping is performed automatically by a network compiler.

A.Y.: Given a grammar, we have an algorithm to make a network of units.

III.A short list of papers related to your research?

G.C.:-Cottrell, G., Small, S.: Viewing Parsing as a Word Sense Discrimination: A Connectionist Approach. In B.Bara, G,Guida (eds.), Computational Models of Natural Language Processing, Amsterdam: North Holland 1984

-Cottrell, G. : A Connectionist Approach to Word Sense Disambiguation. (Techn. Rep. 154) Rochester: The University of Rochester, Dept of Computer Science. Revised version to be published by Pitman in the Research Notes in Artificial Intelligence Series

P.D.:-Dey, P., Iyengar, S.S., Byoun,J.S. : Parallel processing of Tree Adjoining Grammars. Dept. of Computer Science, University of Alabama at Birmingham, Report 1987

-Joshi, A.K., Levy, L.S., Takahashi, M.: Tree Adjoining Grammars. Journal of the Computer and System Sciences, Vol. 10, pp. 136 - 163, March 1975

Vijay-Shankar, K., Joshi, A.K. : Some Computational Properties of Tree Adjoining Grammars.Proc.23rd Ann. Meeting Ass.Comp.Ling., pp. 82-93, 1985

J.D.:-Cottrell, G.W. Parallelism in Inheritance Hierachies with Exceptions. IJCAI-85, 194-202, Los Angeles, 1985.

-Fanty, M. Context-Free Parsing in Connectionist Networks. TR 174, University of Rochester, Department of Computer Science, November 1985.

-Fanty, M. Learning in Structured Connectionist Networks. Ph.D. Thesis, CS Department, Univ. of Rochester,1988.

-Feldman, J.A., Fanty, M.A., & Goddard, N. Computing with Structured Neural Networks. IEEE Computer 1988; in press.

-Shastri, L. & Feldman, J.A. Semantic Networks and Neural Nets. TR 131, University of Rochester, Department of Computer Science, June 1984.

-Shastri, L. Evidential reasoning in semantic networks: a formal theory and its parallel implementation. Ph.D. Thesis and TR 166, Comp. Sci. Dept., Univ. of Rochester, September 1985.

P.A.R.:Literature from systemic linguistics and parallel distributed processing.

H.S.:-Schnelle, H.: Elements of theoretical net-linguistics, Part 1: Syntactical and morphological nets - Neuro-linguistic interpretations. Theoretical Linguistics. Berlin: Walter de Gruyter & Co., 8, 1981, pp. 67-100.

-Schnelle, H., Job, D.M.: Elements of theoretical net-linguistics, Part 2: Phonological nets. Theoretical Linguistics, 10, 1983, pp. 179-203.

-Schnelle, H.: Array logic for syntactic production processors. In Mey, J.(ed.), Language and Discourse: Test and Protest (Sgall-Festschrift) Amsterdam: John Benjamins B.V., 1986, pp. 477-511.

-McClelland, J.L., Elman, J.L.: Interactive procession speech perception: The TRACE model, pp. 58-121 in: McClelland, J.L., Rumelhart, D.E. and the PDP-Group, Parallel Distributed Processing - Exploration in the Microstructure of Cognition, Vol. 2, 1986.

-Aho, A.V., Ullman, J.D.: Principles of Compiler Design - Reading Mass., § 4.22 The Parsing Method of Erley: Addison-Wesley, 1979.

L.S.:-Fahlman, S.E. NETL: A System for Representing and Using Real-World Knowledge, The MIT Press, Cambridge, MA, 1979.

-Hinton, G.E. Implementing Semantic Networks in Parallel Hardware. In Parallel Models of Associative Memory. pp. 161- 187 in: G.E.Hinton and J.A. Anderson (Eds.),. Lawrence Erlbaum Associates, Hillsdale, N.J., 1981.

-Derthik, M. A Connectionist Architecture for Representing and Reasoning about Structured Knowledge. Proceedings of the ninth annual conference of the Cognitive Science Society. Seattle, July, 1987. Lawrence Erlbaum Associates, Hillsdale N.J.

-Shastri,L.: A Connectionist Approach to Knowledge Representation and limited inference. To appear in Cognitive Science:12,3 (1988)

-Shastri, L.: Semantic Nets: An Evidential Formalization and its Connectionist Realization. Los Altos: Morgan Kauffman, London: Pitman Publ.Comp.

A.Y.:-Kaplan R.: A Multi-Processor Approach to Natural Language, Proc. National Computer Conference, 1973, pp. 435-440.

-Small S., Rieger C.: Parsing and Comprehending with Word Experts, in Strategies for Natural Language Processing (EDs. M.D. Ringle and W. Lenher) Lawrence Erlbaum Associates, 1988.

-Matsumoto Y.: A Parallel Parsing System for Natural Language, Springer Lecture Notes in Computer Science, No. 225, 1986, pp. 396-409.

-Yonezawa A, Ohsawa I.: A New Approach to Parallel Parsing for Context-Free Grammars, Research Report on Information Sciences C-87, Dept. of Inf. Sci. Tokyo Institute of Technology, 1987.