Internal distribution code:

(A) [-] Publication in OJ
(B) [-] To Chairmen and Members
(C) [-] To Chairmen
(D) [X] No distribution

Datasheet for the decision of 17 November 2014

Case Number: T 1730/11 - 3.5.06

Application Number: 03770720.5

Publication Number: 1559069

IPC: G06N5/00

Language of the proceedings: EN

Title of invention:
STARTUP AND CONTROL OF GRAPH-BASED COMPUTATION

Applicant:
Ab Initio Technology LLC

Headword:
Graph-based Computation/AB INITIO

Relevant legal provisions:
EPC 1973 Art. 56

Keyword:
Inventive step - after amendment (yes)

Decisions cited:

Catchword:

Europäisches Patentamt
European Patent Office
Office européen des brevets

Beschwerdekammern
Boards of Appeal
Chambres de recours

European Patent Office
D-80298 MUNICH
GERMANY
Tel. +49 (0) 89 2399-0
Fax +49 (0) 89 2399-4465

Case Number: T 1730/11 - 3.5.06

D E C I S I O N
of Technical Board of Appeal 3.5.06
of 17 November 2014

Appellant: (Applicant)	Ab Initio Technology LLC 201 Spring Street Lexington, MA 02421 (US)
Representative:	Lloyd, Patrick Alexander Desmond Reddie & Grose LLP 16 Theobalds Road London WC1X 8PL (GB)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 23 March 2011 refusing European patent application No. 03770720.5 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman	W. Sekretaruk
Members:	M. Müller
	A. Teale

Summary of Facts and Submissions

I. The appeal lies against the decision of the examining division, with reasons dispatched on 23 March 2011, to refuse European patent application No. 03770720.5. The decision made reference *inter alia* to the documents

D1: Babaoglu O. *et al.*, "Mapping parallel computations onto distributed systems in Paralex", Proc. 5th Annual European Computer Conference, pp. 123-130, IEEE Press, 1991,
D2: Gamma E. *et al.*, "Design Patterns: Elements of Reusable Object-oriented Software", pp. 117-126 and 325-330, Addison Wesley, 1999, and
D3: US 6 314 114 B1,

and came to the conclusion that claim 1 according to the then main and first and second auxiliary requests did not involve an inventive step over D1, in view of D2 and D3, Article 56 EPC, and that independent claims of the first auxiliary request did not comply with Article 123 (2) EPC.

II. A notice of appeal was filed on 2 June 2011, the appeal fee being paid on the same day. A statement of grounds of appeal was received on 2 August 2011, together with amended sets of claims 1 to 25 according to a main and four auxiliary requests. The appellant requested that the decision under appeal be set aside and that a patent be granted based on one of the amended sets of claims. The board understands the remaining application documents to be as follows:

description pages:
1-3 as filed with the grounds of appeal
4-16 as published.

drawings, sheets:
1/7-7/7 as published.

The appellant further requested oral proceedings in case the board were minded to issue a decision adverse to the appellant.

III. Claim 1 of the main request reads as follows.

"A method of executing, on a computer system, graphs expressing computations including:

(a) providing at least two graph templates (310) each associated with a different computation graph (100), each computation graph (100) including a number of graph elements each associated with a corresponding computation;

(b) forming at least two pools of processes, each associated with a different type of processing; and

(c) processing multiple data streams concurrently, each associated with a different instance (300) of a corresponding computation graph, including for each of the data streams,

forming a graph instance (300) from the graph template (310) for the corresponding computation graph (100), including allocating memory for a runtime data structure for that graph instance and copying the graph template (310) into the allocated memory, wherein each runtime data structure includes a copy of the graph template (310), a buffer section (350) which holds work elements as work elements are passed between the graph elements and queued prior to processing, and input

counts for each graph element initialized to the number of inputs for that graph element,

wherein each graph element of the graph instance (300) is associated with a corresponding one of the pools of processes, based on the type of processing associated with each pool of processes, wherein a first graph element is associated with a corresponding first pool of processes and a second graph element is associated with a corresponding second pool of processes,

for each graph element of the graph instance (300), assigning processes from the corresponding one of the pools of processes when at least some part of all of the inputs for the graph element are available according to the initialized input counts, wherein the processes read and write work elements from and to the buffer section (350) of the runtime data structure for the graph instance (300) during processing of the data stream, and

processing the data stream with the graph instance (300), including performing the computations corresponding to the graph elements of such graph instance (300) using the assigned processes;

wherein steps (a) and (b) are performed prior to step (c)."

Claim 25 of the main request reads as follows:

"A system for executing, on a computer system, graphs expressing computations including:

at least two graph templates (310) stored in data storage each associated swith a different type of graph-based computation, each template (310) comprising a number of graph elements each associated with a corresponding computation;

means for forming at least two pools of processes, each associated with a different type of processing; and

means for processing multiple data streams concurrently, each associated with a different instance (300) of a corresponding graph-based computation, including for each of the data streams,

forming a graph instance (300) from the graph template (310) associated with the corresponding type of graph-based computation, said graph instance (300) having graph elements corresponding to the graph elements of the graph template (310), including allocating memory for a runtime data structure for that graph instance and copying the graph template (310) into the allocated memory, wherein each runtime data structure includes a copy of the graph template (310), a buffer section (350) which holds work elements as work elements are passed between the graph elements and queued prior to processing, and input counts for each graph element initialized to the number of inputs for that graph element,

wherein each graph element of the graph instance (300) is associated with a corresponding one of the pools of processes, based on the type of processing associated with each pool of processes,

wherein a first graph element is associated with a corresponding first pool of processes and a second graph element is associated with a corresponding second pool of processes,

for each graph element of the graph instance (300), assigning processes from the corresponding one of the pools of processes when at least some part of all of the inputs for the graph element are available according to the initialized input counts, wherein the processes read and write work elements from and to the buffer section (350) of the runtime data structure for the graph instance (300) during processing of the data stream, and

processing the data stream with the graph instance (300), including performing computations corresponding to the graph elements of such graph instance using the assigned processes;

wherein the system is configured to form the at least two pools of processes prior to processing the multiple data streams concurrently, each associated with a different instance (300) of a corresponding graph-based computation."

IV. The wording of the independent claims of the auxiliary requests is immaterial to the present decision.

Reasons for the Decision

1. The invention

1.1 The application relates to the efficient execution of computations expressed as data flow graphs (see p. 4,

par. 38). The vertices (or nodes) in these graphs (illustrated in fig. 1) represent computational tasks and the links indicate the paths along which the data flows in chunks referred to as "work elements". A computation at a vertex can start as soon as (but also no earlier than when) a work element is available at each input link. When the computation has terminated, the result is sent as a new working element along the output link.

1.2 It is disclosed that there are several "types of graphs" representing different types of "work flow" or transactions that might be needed. For example, in a banking context, there may be a different such graph for each necessary financial transaction (see par. 87).

1.3 The work is performed by processes which may be tailored to particular vertexes of particular types of graphs (see par. 63).

1.4 To "run" a work flow, a suitable graph data structure is created in a shared memory segment through which the processes can communicate, and each pool is associated with some of the vertices of the graph; this involves "an initialization procedure [on the process] which includes mapping the shared memory segment for the graph instances into the address space of the process" (see par. 58).

1.5 For each type of graph there is a "template" from which instances of graphs are created. It is possible to create and run several such instances concurrently (see par. 41). To create a graph data structure, the template corresponding to the required type of graph is copied into the shared memory segment. In addition, buffer space is allocated to hold the work elements

"queuing" at individual vertices (see fig. 3 and esp. pars. 55 and 67).

2. Prior art

2.1 Document D1 discloses a programming environment, called Paralex, for the development and execution of data flow programs "on distributed systems as if the latter were uniform parallel multiprocessor computers" (abstract). Paralex is based on data flow graphs which express the same sort of "coarse-grain" data flow as the application (see sec. 2.2, esp. 1st two pars.). A graph is also referred to as a "program", *i.e.* the graph can be executed. Before a Paralex program can be executed, the nodes of the graph must be associated with suitable hosts (sec. 2.4); it is said that the "computation graph" is embedded in the "system graph" (see sentence bridging pp. 125 and 126). For each node there are sets of hosts OHi and PHi defining those hosts on which node i *can* respectively *should preferably be* executed (see sec. 4, last par., and sec. 4.2). The requirements and preferences expressed by these sets are taken into account when the nodes are mapped to the hosts of a given network. A set of nodes which have to be executed sequentially - referred to as a "chain" - are mapped to the same host so as to minimize non-local data communication (sec. 4.1). At the hosts, each node will be executed as a Unix process (see sec. 2.4). Nodes at the same host communicate with each other via finite buffers (see sec. 4.6) so as to decouple computations proceeding at varying speed. D1 discloses that different graph instances can be executed in parallel on different hosts, either to achieve tolerance against node failure (sec. 4.4) or have parallelism between different iterations in case of pipelined operation (see sec. 4.6).

2.2 The book excerpt D2 discloses the idea of "prototype patterns" to create instances of classes by "cloning" - *i.e.* deep copying - a given "prototype" and initialising it properly. Prototypes are introduced to avoid the need to replicate class hierarchies.

2.3 Document D3 discloses a distributed system of computing systems referred to as nodes. Each node which offers services to execute on request by (and on behalf of) other nodes, provides a "dedicated process pool" of varying size for each possible requesting node (see col. 1, lines 40-46; and col. 7, lines 4-9).

3. Article 123(2) EPC

3.1 The decision under appeal did not raise any objection under Article 123(2) EPC against the then main request, nor does the board have occasion to raise an objection of its own. Furthermore, the board is satisfied that the amendments made to the claims of the main request are based on the application as originally filed as indicated by the appellant in its statement of grounds of appeal (see p. 1, 3rd par. - p. 2, 4th par.).

4. Article 56 EPC 1973

4.1 The appellant challenges the decision under appeal mainly on two grounds: firstly, it argues that the "sets of nodes" in D1 are so different from the "pools of processes" of the present invention that the skilled person would not have considered replacing the set of hosts with pools of processes (see grounds of appeal, esp. p. 6, penult. para.). Secondly, it points out with respect to the runtime graph data structure that "D1 is concerned with processing computations over a distributed system, and shared memory is not an obvious

or compatible choice for the communications between hosts or work stations on a network" (see p. 7, last para. and p. 8, penult. para.).

4.2 The board agrees on the second point. D1 does not disclose the use of physically shared memory. The board notes that the concept of shared *distributed* memory also exists but that this does not, according to conventional understanding, imply physically shared memory but only a shared address space which makes access to physically distributed memory transparent to applications. Since the nodes of the computation graph are spread across the distributed system, so will be any data structure representing the computation graph. The board concludes in agreement with the appellant that the idea of copying a graph template data structure into the suitably allocated memory is not compatible with – and thus not obvious in view of – D1 as it stands.

4.3 This copying would be compatible, however, with a single multiprocessor computer with shared memory. Although the abstract of D1 introduces Paralex as a "programming environment that allows parallel programs to be developed and executed on distributed system as if the latter were uniform parallel multiprocessor computers" (emphasis by the board) and is thus specifically targeted at distributed systems as distinct from multiprocessor computers, the board deems it nonetheless to be a realistic problem for the skilled person to contemplate which modifications of the system of D1 might be required – and which simplifications achievable – if the system of D1 were adapted to a multiprocessor system. For example, the skilled person might have a suitable microprocessor system at hand and want to assess

the possible speed-up from adapting the system of D1 to it.

4.4 Doing this, the skilled person would realize that on a multiprocessor computer there would be no need for the sets of nodes or the chains of D1. D1 distinguishes between sets of hosts OHi and PHi which can or preferably should execute a certain node i. Since a single multiprocessor would be a single "host", this distinction would become void. Further, D1 groups nodes which lie along a "chain of data flow edges" and maps entire such chains to hosts so as to keep "all of the data communication along a chain local" (see sec. 4.2, 1st para.; sec. 4.3) and thus to achieve the overall goal of "maximizing parallel execution and minimizing remote communication" (see sec. 4, 1st para.).

4.5 The board considers that on a single multiprocessor computer with shared memory it would be obvious for the Unix processes (see sec. 2.4) executing the individual nodes to communicate with each other via buffers in shared memory. As the data flow is determined by each computation graph, the necessary buffers are effectively implied by the computation graphs. Under these circumstances, the board considers that the claimed creation of a computation graph data structure by copying a suitable template representing the computation graph and extending it with suitable buffer space (see also fig. 3) would be obvious for the skilled person, be it from first principles or from the prototypes according to D2.

4.6 The board further considers that the dynamic allocation of processes from a pool of processes to individual nodes of the computation graphs being executed would be obvious as a matter common programming practice.

4.7 However, the system so obtained would still differ from the claimed invention in not having the features of "forming at least two pools of processes, each associated with a different type of processing" (*e.g.* step b of method claim 1), "each graph element [being] associated with a corresponding one of the pools of processes, based on the type of processing associated with each type of processes" (claim 1, step c, 3rd para.), prior to instantiating and processing a computation graph (claim 1, last line).

4.8 The decision denied (reasons 2.2.1) that the "forming of the pool and assigning of processes from the pools" had a technical effect because "[t]he technical effect of pools is usually obtained if" – and 'only if', as the board understands the argument – "the pooled resource is reused". While the independent claims indeed do not literally specify that the pooled resource is "returned" to the pool and may be "reused", the board notes that claims 1 and 25 disclose the dynamic allocation of processes ("when ... inputs ... are available"). The board agrees with the appellant that already the provision of pools of processes has a technical effect, namely at least that of enabling the technical effect of "pooled resources" which the examining division referred to. Hence, it can be left open whether, as the board tends to think, the skilled person would even consider it implicit from the term "pool" in the given context that processes are returned to the pool and possibly reused once their task is finished and the used inputs are no longer available.

4.9 In summary, the board agrees with the appellant that the pre-computation of tools of processes dedicated to different types of processing and vertexes is not void

of any technical effect but that it rather contributes to the efficient execution of computation graphs.

4.10 As argued above, the sets of hosts in D1 neither imply nor suggest these difference features, since they serve, as the appellant correctly argues, a different purpose which, moreover, is irrelevant on a single shared memory machine. The board also considers that neither the other documents on file nor the common knowledge in the art discloses or suggests these features in the given context.

4.11 Therefore, the board comes to the conclusion that the independent claims 1 and 25 of the main request involve the required inventive step in the sense of Article 56 EPC 1973 over the prior art on file.

Further comments

5. The board has no occasion to raise any objection to the dependent claims 2-24.

6. The board notes however that the description does not conform with the amended claims according to the main request. For example, the very feature causing the board to acknowledge an inventive step (see points 4.7 and 4.9) above, is disclosed as optional in the present description: see paragraph 63, which discloses that different types of pools only "may be made of processes ... tailored to a particular vertex".

7. Since this decision allows the appellant's main request, its conditional request for oral proceedings does not come into play.

Order

For these reasons it is decided that:

1. The decision under appeal is set aside.
2. The case is remitted to the department of first instance with the order to grant a patent based on claims 1-25 according to the main request as filed with the grounds of appeal with a description to be adapted.

The Registrar: The Chairman:

B. Atienza Vivancos W. Sekretaruk

Decision electronically authenticated