06361 Abstracts Collection
Computing Media and Languages for Space-Oriented Computation
— Dagstuhl Seminar —

André DeHon[1], Jean-Louis Giavitto[2] and Fréderic Gruau[3]

[1] CalTech - Pasadena, US
andre@acm.org
[2] Univ. of Evry, FR
giavitto@ibisc.univ-evry.fr
[3] Univ. Paris Sud, FR
frederic.gruau@lri.fr

Abstract. From 03.09.06 to 08.09.06, the Dagstuhl Seminar 06361 "Computing Media and Languages for Space-Oriented Computation" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Hardware architecture, computing medium, space-oriented computation, nonconventional programming models

06361 Executive Report – Computing Media and Languages for Space-Oriented Computation

With the cheap availability of high capacity spatial computing substrates, an emerging understanding of natural systems, and the possibility of computationally engineered matter, the importance of spatial aspects of computation is growing. These different manifestations of spatial computing have clear intersections where they can share common theory, tools, and insights. A solid mastery of spatial computation will allow us to transform our engineering capabilities, our understanding of the natural world, and ultimately the world in which we live.

Keywords: Hardware architecture, computing medium, space-oriented computation, nonconventional programming models

Joint work of: DeHon, André; Giavitto, Jean-Louis; Gruau, Fréderic

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2007/1025

Proposed Definition of Spatial Computing

A general purpose computing system where position matters for performance or functionality and is embedded in a space whose geometry is defined by communication cost.

Such computers are spatially distributed collections of communicating processing elements of nearly comparable power for which the cost (time/energy) to communicate between any two elements that are far apart is at least proportional the length of the shortest path between them. Path length is defined in the graph that has an edge between two elements if they can communicate directly.

Precise communication cost: $\Omega(d)$

Joint work of: Seth, John, Kati, Daniel[2], Jake, Guy, Christian, Frederic

Proposed Definition of Spatial Computing

Processors are laid out quasi-homogeneously on a manifold in 2D/3D space in which the induced topology and communication bandwidth relates linearly to distance, when average in space.

Joint work of: Gruau, Fréderic

Application Discussion Outbrief

What applications should we consider as drivers and motivators for future spatially-oriented computing systems? This list was collected from all participants. We further identify three characteristics of applications that make them spatial in nature and categorize the applications based on these characteristics.

Keywords: Spatial computing, computationally intense, represnet space, embedded in space

Accommodating Defects in Spatial Computing Outbrief

Defects will become highly prevelant when we can assemble computers with a very large number (e.g., 10^6) of processing elements. The mechanisms for coping with this at the hardware, software, and algorithm levels are not clear. What is clear is that without sufficient coping mechanisms, defects will be a crucial limiting factor to scaling a system to such a large number of elements.

Keywords: Defects, faults, tolerance, robust, language mechansims

Local / Global Discussion Outbrief

Large spatial computing systems will be comprised of a huge number of processing elements (e.g., 10^6). What is the right way to program these computers: specify the local program behaviour to accomplish a predictable emergent global outcome, or specify the global program behaviour and compile this into a number of local programs? Either way appears to be challenging and with pitfalls.

Keywords: Synthesis, local programming, global programming, emerging behaviour

Joint work of: Lemieux, Guy; Coore, Daniel

Programming Manifolds

Jacob Beal (MIT - Cambridge, USA)

Many programming domains involve the manipulation of values distributed through a manifold-examples include sensor networks, smart materials, and biofilms. This paper describes a programming semantics for manifolds based on the amorphous medium abstraction, which places a computational device at every point in the manifold. This abstraction enables the creation of programs that automatically scale to networks of different size and device density. This semantics is currently implemented in our language Proto and compiles for execution on Mica2 Motes.

Keywords: Amorphous computing, spatial computing, Proto

Joint work of: Beal, Jacob; Bachrach, Jonathan

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2007/1023

How Should People Program For Spatial Computers?

Katherine Compton (University of Wisconsin - Madison, USA)

Problem:

- Programmers accustomed to sequential will need to program for spatial computers

Two Questions (for now):

- What should the descriptions/IDEs look like?
- What requirements will we impose on the programmers?

The aim of this talk is to foster discussion on the topic of programming methodologies for spatial computers.

An Introduction to Amorphous Computing

Daniel Coore (The University of the West Indies - Mona, SAM)

We present the Amorphous Computing model: a massively distributed system where processing elements are irregularly positioned, communicate locally and are identically programmed. We identify some of the results so far at controlling these systems, and we discuss the lessons learned for designing programming languages that are especially suited to this type of distributed system. Specifically, we recommend a layered approach to language design, where at the very bottom there is a programming language that is used to program the individual elements, and refers only to activities that can be described in terms of local interactions. Above this layer, we suggest at least one other layer, in which objectives for emergent behaviour are specified in ways that can be systematically translated to the first layer. This approach has already proved to be successful for pattern formation in Amorphous Computing, and we expect that it can be used successfully for many other applications in amorphous computing. On that basis, it appears worthy of pursuit as a general approach to controlling emergence on even complex systems that do not conform exactly to the amorphous computing model.

Keywords: Amorphous computing, swarm computing, self-organizing patterns

Challenges and Opportunities for Spatial Computing

André DeHon (University of Pennsylvania, USA)

With today's integrated circuit (IC) capacity, we can build very large and capable computing systems. While traditional, temporal computing structures cannot fully exploit the capacity now available on modern ICs, spatial computing structures can harness this capability; the importance of spatial computing only grows as IC scaling continues to offer even greater capacities. Further, today's spatial computing structures can be highly programmable, creating opportunities to adapt the computation to the problem, data set, and environment. Nonetheless, physical issues create challenges for all large-scale computing structures. Interconnect delay, energy, and size can be dominant effects demanding computations be organized to exploit spatial locality. Traditional programming approaches developed for ASICs (e.g. VHDL) or sequential computing (e.g. C) do not provide appropriate abstractions to encourge good, scalable spatial computing solutions and do not allow aggressive optimization of spatial designs. Consequently, we need new models, abstractions, and algorithms to fully exploit spatial computing.

As important steps towards programming these spatial computing systems, we introduce system architectures that guide designers to useful organizational strategies to exploit these spatial computing ICs in a manner consistent with the parallelism and control in the application, and design patterns that teach developers how to solve recurring problems in scalable, spatial computing design.

Keywords: Spatial computing, interconnect, system architecture, design patterns

Organizations in Space

Peter Dittrich (Universität Jena, D)

In this talk we will investigate space at different scales.

As an example, we study a chemical computing system or a reaction system where molecules are spreaded over a two-dimensional grid. In order to look at this grid at different spatial scales, we need a means to aggregate information, that is, a way to change resolution of our microscope looking at that system. We take chemical organization theory for this task. Molecules are aggregated by mapping them to the organization they generate.

We will see that different organizations appear at different scale, providing different kinds of information. Techniques for selecting the right scale are suggested.

Keywords: Chemical computing, spatial reaction systems, chemical organization theory, organic computing

Joint work of: Speroni di Fenizio, Pietro; Dittrich, Peter

N-synchronous Kahn Networks

Christine Eisenbeis (INRIA Futurs - Orsay, F)

The design of high-performance stream-processing systems is a fast growing domain, driven by markets such like high-end TV, gaming, 3D animation and medical imaging. It is also a surprisingly demanding task, with respect to the algorithmic and conceptual simplicity of streaming applications.

In search for improved productivity, we propose a programming model and language dedicated to high-performance stream processing. This language builds on the synchronous programming model and on domain knowledge — the periodic evolution of streams — to allow correct-by-construction properties to be proven by the compiler, including resource requirements and delays between input and output streams. Automating this task avoids tedious and error-prone engineering, due to the combinatorics of the composition of filters with multiple data rates and formats. Our language is thus provided with a relaxed notion of

synchronous composition, called *n-synchrony*: two processes are n-synchronous if they can communicate in the ordinary (0-)synchronous model with a FIFO buffer of size n.

Technically, we extend a core synchronous data-flow language with a notion of periodic clocks, and design a relaxed clock calculus (a type system for clocks) to allow non strictly synchronous processes to be composed or correlated. This relaxation is associated with two sub-typing rules in the clock calculus. Delay, buffer insertion and control code for these buffers are automatically inferred from the clock types through a systematic transformation into a standard synchronous program.

Keywords: Synchrounous languages, stream processing, correct-by-construction, resource constraints, subtyping

Joint work of: Cohen, Albert; Duranton, Marc; Eisenbeis, Christine; Pagetti, Claire; Plateau, Florence; Pouzet, Marc

See also: Albert Cohen, Marc Duranton, Christine Eisenbeis, Claire Pagetti, F. Plateau, and Marc Pouzet. N-Sychronous Kahn Networks. In 33th ACM Symp. on Principles of Programming Languages (PoPL'06), Charleston, South Carolina, pages 180-193, January 2006.

Data structure as spaces : computing in space and space in computation

Jean-Louis Giavitto (University of Evry, F)

The dynamics of natural systems are often described by specifying the basic interactions of subsystems. The specification of the elementary interacting subsystems leads to a topological structure. This point of view is easely extended to more artefactual systems like programs and their execution. In this point of view, a data structure is a space where some computations occur.

This idea underlies an experimental declarative programming language called MGS. MGS introduces the notion of topological collection: a set of values organized by a neighborhood relationship. The basic computation step in MGS relies on the notion of path : a path C is substituted for a path B in a topological collection A. This step is called a transformation and several features are proposed to control the transformation applications.

By changing the topological structure of the collection, the underlying computational model is changed. The topological point of view enables a unified view on several computational mechanisms. Some of them are initially inspired by biological or chemical processes (Gamma and the CHAM, Lindenmayer systems, Paun systems and cellular automata).

The numerous examples developped to validate the MGS idea, we drawn three lessons:

– the multiset topology (underlying the chemical programming paradigm) is a universal one (i.e. all other topology can be recovered from this one);
– optimization problems that follows the Bellman principle (dynamic programming) are especially well suited to the chemical programming paradigm because no synchronization constraint are necessary;
– if usual algorithmic recquires sophisticated interactions, the simulation of natural systems (e.g. in physics) focuses on a very simple and local pattern.

Keywords: Topological collection, declarative and rule-based programming language, rewriting, Paun system, Lindenmayer system, cellular automata, Cayley graphs, combinatorial algebraic topology.

Full Paper:
http://mgs.ibisc.univ-evry.fr/

Proposed spatial computing definition

Seth C. Goldstein (CMU - Pittsburgh, USA)

A general purpose computing system where position matters for performance or functionality and is embedded in a space whose geometry/topology is defined by communication distance.

Programming Matter

Seth C. Goldstein (CMU - Pittsburgh, USA)

draft

In this talk we describe methods of proramming ensembles of spatially distributed computing elements. We ground the talk using claytronics, an instance of programmable matter. Claytronics is an ensemble of computing systems which contain a means of computing, communicating, actuating, sensing, and adhering to other units.

Keywords: Programmable Matter, Claytronics, 3D rendering

Formal Verfication: Issues and Challenges

Daniel Große (Universität Bremen, D)

Due to increasing design complexity and intensive reuse of components, verifying the correctness of circuits and systems becomes a more and more important factor.

In the meantime it has been observed that verification becomes the major bottleneck, i.e. up to 80% of the design costs are caused by verification. Due to this and the fact that pure simulation cannot guarantee suffcient coverage formal verification methods have been proposed.

In this talk after a brief motivation of the overall topic the verification scenarios are discussed. Then the underlying proof techniques are described.

In the main part Bounded Model Checking as one formal verification method is explained. In a case study We focus on a HW/SW Co-Verification approach for embedded systems. Finally research challenges and directions for future work are given.

Tutorial on blob computing

Frédric Gruau (Université Paris Sud, F)

General view of the project " Blob Computing ": the objective is to propose a model of parallelism combining genericity and scalability. A blob machine is a low level parallel virtual machine. It is generic, in the sense that it does not limit a priori the domain of the possible applications. Its primitive are conceived simple enough to be installed on a range of scalable hardware platform hardware , both fine grain or coarse grain, and powerful enough to be the target of a compiler allowing to program via a high-level object oriented language called the blob language. The tutorial will present the blob machine, its properties, and the blob langage, with example of programms, illustrating the concept of parallel data structures.

We propose a modelof parallelism emancipated from the sequential dogma, having the potentiality to combine genericity and scalability. It allows to program a scalable computing medium , that is shaped just like space: homogeneous, isotrope, with only local connections 2D or 3D between processors. The model uses two levels:

The first level, " the system level ", implements the virtual blob machine capable of self developing, « building itself » by using three types of building blocks: mealy automata, links and blobs. Automata are treated as punctual, inseparable objects, while blobs and links are conceptually comparable to physical objects of variable size. Blob behaves as membranes and allows to group together a set of sub blobs. Blobs are thus encapsulated. A link behaves as a thread connecting two blobs at every extremity, and allows to install a communication channel between them. Every blob (resp. every link) possesses a finite state mealy automaton which actions can trigger the division (resp. duplication) or the fusion (resp. destruction) of the blobs (resp. of the link) under their control. The configuration of the machine at a given moment is a network of automata with connections between two automata if 1-they both control a blob, and one is contained in the other one. 2-one controls a link and the other one control a blob connected to that link; the key of the blob approach is to manipulated very simple basic building block: blob, links as well as system calls defined above (to

copy or delete those object); This simplicity allows to make the system itself responsible for dynamically mapping those building block onto the harware. To obtain a good parallel performance, just the system has to guarantee that the mapping homogenizes the density of automata while maintaining close by pair of automata which are neighbor.

In the second level, one programs the machine virtual blob. A program compiled as a mealy automaton is loaded. This automaton controls an ancestor blob, which creates links and divides iteratively, so developing a network of automatas, communicating towards the outside via a set of fixed blob playing the role of ports. It is certainly very difficult to program with such a self developping machine, however: 1-The automaton is compiled from a high-level language: the blob language. 2-the programming makes no supposition on hardware, for example its granularity can vary from the very fine : cellular automata, or FPGA; to a 2D grid of classic processors, 3- the program can run in parallel on whatever of these hardware devices, as soon as the virtual machine is installed on it 4- The blob machine possesses a theoretical property of universality in connection with the parallelism which demonstrates its expressiveness. It was also used in learning, since 1992 under the name of " cellular encoding ".

The tutorial will present the virtual blob machine, its theoretical underpinning, and the highlevel blob langage. Exemple of programs will be given to illustrate the concept of parallel data structure, where each data is stored together with methods, as in object oriented programming, with the additionnal feature of being able to run in parallel. The complexity of the presented programms is presented

Keywords: Blob computing, spatial computing, massive parallelism, cellular automata

Programming self developing blob machines for spatial computing.

Frédric Gruau (Université Paris Sud, F)

This is a position paper introducing blob computing: A Blob is a generic primitive used to structure a uniform computing substrate into an easier-to-program parallel virtual machine. We find inherent limitations in the main trend of today's parallel computing, and propose an alternative unifiying model trying to combine both scalability and programmability. We seek to program a uniform computing medium such as fine grain 2D cellular automata, or more generally coarse grain 2D grids of Processing Elements, using two levels:

In the first "system level", a local rule or run time system is implemented on the computing medium. It can maintain global connected regions called blobs. Blobs can be encapsulated. A blob is similar to a deformable elastic membrane filled with a gas of atoms. (elementary empty blobs). Blobs are interconnected using channels, which act as a spring to bring connected blobs closer to each

other. The system implements in a distributed way: movement, duplication and deletion of blobs and channels. It can also propagate waves to communicate signals intra-blob, or inter-blob.

In the second "programmable level", each blob and channel contains a finite state automaton, with output instruction triggering duplication or deletion . Execution starts with a single ancestor blob that duplicates and creates channels repeatedly, thus generating a network of automata. It installs a higher level virtual machine on top of a low level uniform computing medium. This "blob machine" is an example of "self developing automata network".

We present in detail, the blob machine, and how to program it using a higher level language called blob ml. We illustrate the execution of many examples of small programs They all exhibits optimal complexity results, under some reasonable hypothesis concerning the -not yet finished to implement - system level, and considering the model of VLSI complexity.

Keywords: Cellular automata, amorphous computing, blob machine, blob computing, massive parallelism, graph rewriting, parallism, parallel langage

Joint work of: Gruau, Frédric; Eisenbeis, Christine

Building Systems from Actors —An approach to addressing the spatial software crisis

Jörn W. Janneck (Xilinx - San José, USA)

The availability of large spatial computing devices (Platform FPGAs) has created a "spatial software crisis"—a huge gap between the potential productivity of spatial programmers, and the productivity they currently enjoy by using HDL-based programming practices.

This talk discusses characteristics of potential remedies, and alludes to one approach currently developed at Xilinx, for the area of DSP.

Keywords: Spatial software crisis, actors, dataflow, CAL, spatial programming, FPGA

Genetic Programming of an Algorithmic Chemistry

Christian Lasarczyk (Kamen, D)

We present Algorithmic Chemistries and how to evolve solutions using Genetic Programming. Special emphasize is put on the integration of variability and robustness, because this has been an issue implementing the recombination operation.

Outline
+Motivation

++Computer Architecture
++Evolvability
++Artificial Chemistry
+Implementation
++Chemical Representation
++Genetic Programming
+Results
++Algorithm and Problem Design
++Evolution Properties
++Solution Properties
++Height, Time and Assembling
+Outlook
++Space vs. Time
++Robustness
++AC-Architecture

Keywords: Algorithmic Chemistry, Genetic Programming, Spatial Computation, Space Oriented Computation

Joint work of: Lasarczyk, Christian; Banzhaf, Wolfgang

Getting Real... FPGA and Silicon Trends for Spatial Computing

Guy Lemieux (University of British Columbia - Vancouver, CA)

Silicon will be the most important medium, in the form of an FPGA, for realizing spatial computers and raising their popularity in the near term. In an FPGA, long wires are prefabricated and can be customized though programming connectivity to implement almost any network and computational element. This provides for a very rich design space where locality/position matters and affects performance/cost.

Keywords: FPGAs, silicon computing, spatial computing, array of processors

Specification and Analysis at the Nanoscale

John Savage (Brown Univ. - Providence, USA)

Self-assembled architectures will be regular structures at the mesoscale but random or uncertain at the nanoscale. Although defects and faults will occur in nanoscale components at very high rates, chips will also contain very reliable lithographic-scale components that can be used to good effect. Discovery algorithms will be used to identify the unpredictable structure at the nanoscale including defects. Nanoscale chips will be programmed rather than designed at

the physical level and converted to layouts, as is the case with VLSI. Chip programming will require smart compilers that utilize both high-level functional specifications as well as discovered architectural information. Modeling will help to understand the inherent limitations of these new technologies. The VLSI model will be extended to include stochastic assembly and wire delays comparable to gate delays.

Keywords: Stochastic assembly, reliable computation, modeling, analysis

Paths and Patches: Declarative Handling of Higher-Dimensional Data-Structures.

Antoine Spicher (University of Evry, F)

Rewriting systems (RS) are a computing tool that is well suited to model and specify complex systems. Classical RS make it possible to handle terms organized into tree-like data structures. Such data structures are not expressive enough to describe complex organizations. The goal of the MGS (Modèle Général de Simulation) project is to develop new rewriting techniques operating on arbitrary data-structures, using a topological point of view. Pursuing this goal, we developed two new concepts:

- Topological collections, which represent data structures in a unified manner based on concepts from topology,
- Transformations, which are case-based functions for specifying rewriting rules on topological collections.

These concepts have been implemented in an experimental functional language.

In this presentation, we introduce a new kind of topological collection based on cellular complexes, and a new kind of transformation that makes it possible to match patterns of collections of any dimension. In this context, we will discuss the shortcomings of a "path pattern" language that led us to the development of a more powerful "patch pattern" language. Elements of implementation and examples of MGS code will be presented. The latter include: Floyd-Fulkerson algorithm, wave propagation, diffusion-limited aggregation, generation of fractals, surface subdivision, and an abstract simulation of neurulation.