Executive Report : Dagstuhl 06361
Computing Media and Languages for Space-Oriented Computation
September 3–8, 2006

Fréderic, Jean-Louis, André

May 18, 2007

Traditional models of computation have abstracted out physical locations in space (*e.g.* the Internet, superscalar processors, unit delay wires, uniform memory delay) and implementations predominantly perform computations in time (*i.e.* sequentially). Most of our common data structures are spatially agnostic (*e.g.* arrays). But:

1. As scaling continues (both as primitive elements shrink to the atomic scale, and the number of elements composed scales up), computations must be distributed in space and location in space impacts the performance and feasibility of the computation.
2. As we couple and embed computing in the physical world (*e.g.* smart building, reactive surfaces, programmable matter, distributed robotics), position and shape are primary, serving as both the input to computation and a key part of the desired result of the computation.
3. As we understand natural computing systems (*e.g.* cells, ant colonies, system's biology) location and topology define the computation.

Consequently, it is important to make **space** not an issue to abstract away, but a first-order effect that we optimize. The distinguishing feature of *spatial computing* then is that computation is performed distributed in space and position and distance metrics matter to the computation.

During the workshop, three thematic areas have been identified: *intensive computing* where space is used as a mean and as a resource, *computation embeded in space* where location is important for the problem and *space computation* where space is fundamental to the problem and is a result of a computation. Figure 1 summarises some application in these three categories.

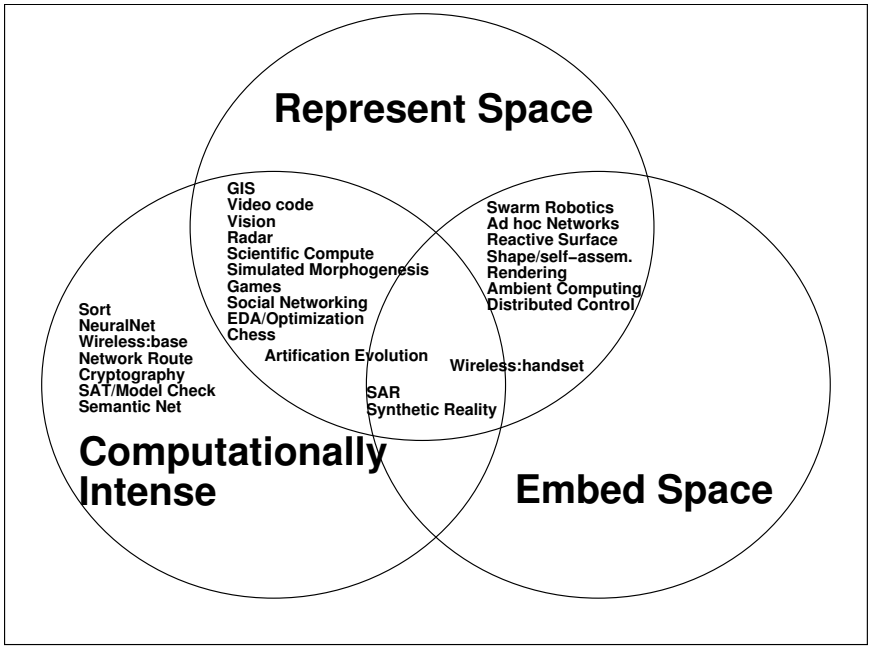

Figure 1: Example Spatial Computing Applications and their Relation to Three Categories

1 Coping with Space (Computationally Intense)

When we build anything more than the most trivial, sequential processor, the computation is distributed across space. The computation may be composed spatially as a pipeline or systolic array running on a Field-Programmable Gate Array (FPGA), or the computation may be a set of cooperating sequential threads running on a multicore processor. Now that we can place hundreds of processors or tens of thousands of spatial datapath elements on an economical silicon die, even inexpensive systems must cope with the distribution of computations across space. At today's aggressive clock rates (*e.g.* Gigahertz) it takes multiple clock cycles to cross even a single die, making delay, and hence performance, a strong function of the distance between communicating operators. Physical locality matters. VLSI complexity theory has provided some approaches and tools to model these spatial aspects of computing; however, these spatial effects take on more universal importance and impact as today's single-chip, programmable components have the capacity for highly parallel computations where the placement of an element of a computation has a strong impact on feasibility and performance of the computation. The continual march of Moore's Law places more components on a die and shrinks the fraction of a die one can cross in a single clock cycle; these trends increase the importance of spatial computation over time.

2 Embedded in Space

.

The physical world performs computations all the time; springs compute a restorative force, capacitors integrate current, and cells compute responses to stimuli. As we attempt to better understand natural processes (*e.g.* morphogenesis, immune system), we must understand the computations they perform. Further, we can potentially bring the benefits of engineered design to physical objects by controlling the computation they perform:

- smart materials – provide new properties to physical matter by controlling the computation and response
- synthetic biology – engineers the computation inside cells to control molecular synthesis or response systems
- programmable matter – forms physical shapes and objects which interact with the world as the output of a computation

These tasks require sensing, computation, and actuation on large ensembles of elements distributed in space. The position in the ensemble is an important part of sensing (input) and the position of an element in the ensemble is an essential part of the actuation (output); the position may also be essential to the integrity and stability of the ensemble. Continued miniaturization of robotics and control over biological elements (*e.g.* engineered cells), suggests a Moore's-Law-like curve for physically programmable ensembles in space. Controlling, programming, and understanding these spatial ensembles demands new theory and models to answer questions, such as: How do we connect between desired global behavior of the ensembles and the local behavior of the primitive elements?

3 Represent Space

In both computationally intensive, spatially distributed computations and computations embedded in space, we must represent the space as part of the computation. This representation is also necessary for simulation of natural systems (*e.g.* cellular behavior, biological systems, circuit simulation, heat and fluid flow, electromagnetic wave propagation). While we could layer abstractions on top of an existing, space-agnostic programming language, this creates an additional burden to spatial programming—forcing the programmer to reason both about the spatial system and the discipline for modeling the spatial systems on top of the conventional programming language. Alternately, can we create or augment programming languages that naturally capture the spatial aspects and semantics of these spatial programming tasks?

The spatial program must take into account the fact that computation is spatially distributed among potentially heterogeneous local islands of computation (*e.g.* processing elements (*e.g.* gates, microprocessors), cells, devices (*e.g.* transistors, capacitors, diodes), materials with different behavior (*e.g.* insulators, conductors)). Not only must we model spatial connections, many of these computations proceed by computing and implementing changes to their topology, and this must be captured and supported in the computational

model. Ideally, we would like to program at a high level of abstraction so as to lighten the burden exposed to the programmer. At the same time, a good abstraction will facilitate portability so that a program can run on arbitrary spatial computing media, including FPGAs, amorphous computers, multicore processors, and natural platforms such as chemical reaction diffusion computers, DNA self-assembly, and natural or synthetic cellular assemblies.

This leads us to ask: "What are the correct, relevant abstraction for spatial computing?" Researchers have approached this problem both bottom up and top down. Bottom up researchers are developing libraries of generic, robust primitives which can run efficiently on arbitrary computing media; these primitives capture known, spatially-distributed algorithms and capabilities and allow designers to think more abstractly about the computational media. Existing primitives include: establishing a set of coordinates, generating gradients, using gradients to move particles, and computing a Voronoi Tesselation. Top down researchers propose new languages where the declared data structure natively represent a space and the computation which occurs at each location in the space. We expect, a mature spatial programming framework will bridge between the bottom up and top down approaches—allowing high level behavior to be efficiently translated to low-level behaviors.

4 Cross Cutting Issues

Each of these three views of spatial computing have their own motivation and rationale. However, they are clearly intertwined. While various concerns and issues take on different priorities in each domain, a common set of theory and techniques may be applicable across the domains. *e.g.*

- Distributed, self-organizing, and potentially fault-tolerant computations are essential for computations embedded in space. Such algorithms can also ease programming of computationally intensive computations.
- Intensive computations address how to use the spatial parallelism of a large spatial media to accelerate hard computations. Computations embedded in space will ultimately need to perform such computations on their ensemble of components. For example, a moving programmable matter application will ultimately need to both compute (render) the next shape it should form and perform the movement to assume the computed shape.
- Programming languages which model space give us ways to efficiently, compactly, and cleanly program these spatially distributed computations, whether embedded in a space or computationally intense.

5 Summary

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.

6 Acknowledgements

This fruitfull event was made possible thanks to the cordial reception and the perfect support of all the Dagsthul staff. The coordinators of the workshop wish particularly thanks Jutka Gasiorowski, Angelika Mueller and Annette Beyer for their patience and constant help. Our acknowledgements also goes to all the participants that have made this first meeting a success.