https://orcid.org/0000-0002-4168-7190

UDC 623.764

P.S. SAPATY*

COMPREHENDING DISTRIBUTED WORLDS WITH THE SPATIAL GRASP PARADIGM

*Institute of Mathematical Machines and Systems Problems of the National Academy of Sciences of Ukraine, Kyiv, Ukraine

Анотація. *У статті розглядаються можливості застосування розробленої Парадигми просторового захоплення (ПЗ) для вирішення складних проблем цілісним і повністю розподіленим шляхом. У роботі представлені результати подальшого розвитку парадигми у двох взаємопов'язаних напрямах: філософсько-концептуальному та технологічно-реалізаційному. В рамках першого напряму обговорюються деталі розвитку ПЗ у розподілених просторах у вигляді хвиль або навіть вірусів і її здатності одночасно схоплювати рішення просторових проблем. Зокрема, увага приділяється принциповим відмінностям між розробленою парадигмою і традиційними уявленнями про системи та їх рішення як елементами, що обмінюються повідомленням. Філософія ПЗ також нагадує такі високі поняття, як сприйняття, усвідомлення, свідомість і навіть душа. Що стосується другого напряму, у роботі коротко подаються отримані деталі щодо Технології просторового захоплення, повідомляється, де її інтерпретатори Мови просторового захоплення (МПЗ) можуть бути об'єднані в мережу як просторові комп'ютери, які охоплюють будь-які наземні і небесні середовища. У роботі детально обговорюються розподілені механізми інтерпретації основних складників МПЗ, за допомогою яких можна реалізувати просторову функціональність без централізованих ресурсів. У статті також розглядаються приклади повністю розподілених рішень МПЗ для спостереження та оцінки дуже масштабних явищ, таких як урагани, лісові пожежі і навіть галактики, а також із метою виявлення зображень у розподілених мережах, які можна сприймати як єдине ціле відповідно до концептуальної орієнтації ПЗ. Розроблена парадигма надає можливість безпосереднього вираження вищої семантики і цілісних методів вирішення складних проблем, динамічного створення і побудови необхідного середовища для реалізації, забезпечуючи таким чином найсуворіший шлях від виявлення проблеми до її практичного вирішення.*

Ключові слова: *система, розподілена система, паралельні та розподілені обчислення, хвилі, захоплення, сприйняття, усвідомлення, свідомість, душа, Технологія просторового захоплення, Мова просторового захоплення, розподілена інтерпретація, просторове бачення, розподілена мережа, відповідність патерну.*

Abstract. *This paper relates to the developed Spatial Grasp (SG) Paradigm for solving complex problems in a holistic and fully distributed way. It presents the results of its further development in two interlinked directions: philosophical-conceptual and technological-implementational. In the first direction, there are discussed the details of how SG develops in distributed spaces as waves or even viruses and how grasps at the same time solutions of spatial problems, also how it fundamentally differs from traditional representations of systems and their solutions as parts exchanging messages. The SG philosophy also resembles higher concepts like perception, awareness, consciousness, and even soul. In the other direction, the resultant Spatial Grasp Technology details are briefed where its Spatial Grasp Language (SGL) interpreters can be networked as spatial computers covering any terrestrial and celestial environments. Distributed interpretation mechanisms of basic SGL constructs are discussed in detail, allowing for the implementation of spatial functionality without centralized resources. The paper also provides examples of fully distributed SGL solutions for observing and evaluating very large phenomena, such as hurricanes, forest fires, even galaxies, as well as discovering images in distributed networks, which can be perceived as a whole in line with the conceptual orientation of SG. The developed paradigm allows direct expression of top semantics and holistic methods for solving complex problems, dynamic creation and composition of the needed implementation environment, thus providing the strictest way from problem definition to a practical solution.*

Keywords: *system, distributed system, parallel and distributed computing, waves, grasps, perception, understanding, consciousness, soul, Spatial Grasp Technology, Spatial Grasp Language, distributed interpretation, spatial vision, distributed networking, pattern matching.*

DOI: 10.34121/1028-9763-2022-1-12-30

1. Introduction

The aim of this paper is to further deepen and develop the definition and description of the Spatial Grasp (SG) Paradigm in the two interlinked directions: philosophical-conceptual and technological-implementational. Having been ideologically born more than half a century ago and called WAVE in its childhood, it represents a fundamentally different approach to solving problems in large distributed systems and networks, where instead of placement of activities inside them to communicate with each other, ahead of problem-solving, it uses a high-level holistic approach. This approach, being above and over these systems, investigates, processes and controls them by dynamic covering and matching their physical or virtual bodies with active recursive code which spreads like waves and viruses but unlike them can also provide any solutions of both local and global problems in fully distributed and parallel mode. SG can also explicitly implement, simulate, or mimic any other existing models and approaches, including the creation of any distributed systems with any structures and topologies, putting any needed communicating activities into their nodes, however, considering this as conceptually lower levels solutions which often appear just automatically on the SG distributed implementation levels. SG has been matured and strengthened over decades being prototyped in different countries and tested on numerous applications, including graph and network theory, computer networking, collective robotics, simulation of battlefields, psychology, sociology, security, disaster management, and recently even celestial orbital systems. The obtained results and experience have formed the necessity to further deepen and extend the approach and make more investigations of its power and potential applicability in extended and important applications. The rest of this paper is organized as follows.

Section 2 presents and discusses the main ideas of the developed Spatial Grasp (SG) Paradigm. It starts with how usual systems and their solutions are organized, with all being represented as consisting of parts exchanging messages. Then the main principles of SG are explained, which provides the world coverage with parallel active code having a symbiotic combination of wave-like (even virus-like) and grasp-like spatial activity. It is also shown how this may relate to much higher mental and philosophical concepts like perception, awareness, consciousness and soul, with the previous SG version WAVE compaired with well known book Society of Mind using traditional system principles.

Section 3 shows how the discussed SG paradigm has resulted in a real technology prototyped and tested on different applications, with the main features of its latest version summarized. It equally operates with different worlds including physical, virtual, executive, and any of their combinations. Recursive Spatial Grasp Language (SGL) is briefed with its rules, different types of spatial variables, and elementary programming examples. The basics of SGL interpretation are also mentioned where networked interpreter copies, potentially from millions to billions, can operate as powerful spatial machines serving SGL scenarios in parallel. Full descrition of SGL is provided too.

Section 4 reveals some important technical details of the SGL interpretation that were not discussed in previous publications. These include parallel advancement in a distributed space which may be asynchronous or synchronized, repetitive asynchronous or synchronous propagation, as well as the use of the dynamically created track system for performing different operations on remotely accessed values by effectively converting it into a hierarchical spatial computer. The latter can also be used in a combination of forward, backward, and forward again manners for further space exploration.

Section 5 provides some examples of how to see, understand, and evaluate as a whole very large distributed phenomena, unobservable from any single point, by their physical or virtual coverage under the SG paradigm. The high-level solutions in SGL contain self-spreading and coverage of the regions of interest, like hurricanes and forest fires, for registering their external borders, with similar techniques potentially applicable for much larger, like celestial, cases. It also shows how to work in SGL for finding certain images in arbitrary distributed networks using the spatial graph-pattern matching technique. Different possibilities of implementation of high-level semantic SGL programs are also discussed.

Section 6 concludes the paper by mentioning the implementation availability of the latest SGT version which similarly to the previous versions can be performed even within standard university environments.

2. Main ideas of the Spatial Grasp (SG) Paradigm

2.1. Traditional system representations

The following words and meanings are often used when speaking about systems and namely distributed systems. A system is a group of interacting or interrelated elements that act according to a set of rules to form a unified whole [1]. A distributed system is a system whose components, which are located on networked computers, communicate and coordinate their actions by passing messages, as shown in Fig. 1 (rectangles symbolize basic system components, circles – solutions in them). The components interact with each other in order to achieve a common goal [2]. A distributed control system (DCS) is a computerized control system for a process, or a plant usually with many control loops where autonomous controllers are distributed throughout the system but there is no central operator supervisory control. This is in contrast to systems that use centralized controllers, being either discrete ones located at a central control room, or within a central computer [3]. The following concepts usually relate to distributed systems too: message-based communications, distributed agents, and distributed objects [4], also languages for parallel and distributed computing which provide synchronization constructs whose behavior is defined by a parallel execution model. A concurrent programming language is defined as one which uses the concept of simultaneously executing processes, or threads of execution as the means of structuring a program [5].

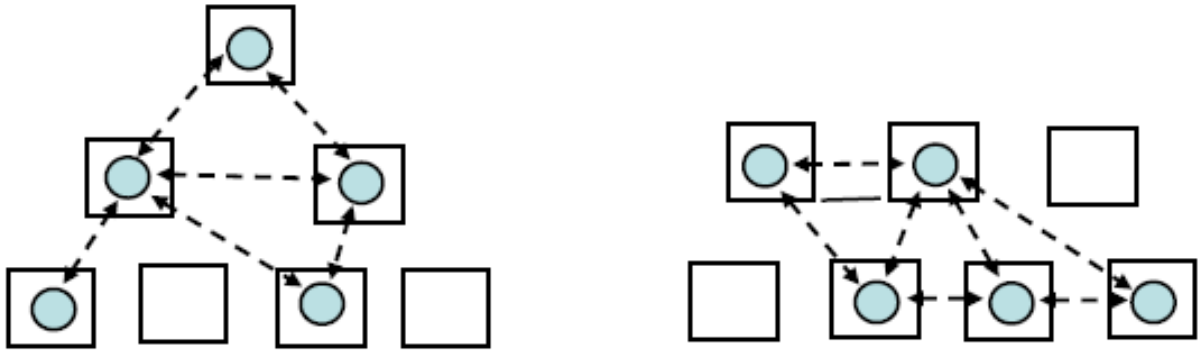


Figure 1 – Traditional system representations and their solutions:
(a) hierarchical, (b) distributed

2.2. Spreading, covering, matching and grasping paradigm

The discussed here concept is based on a quite different philosophy of dealing with large distributed systems. Instead of representing systems and their solutions in the form of communicating parts or agents, the developed Spatial Grasp Paradigm organizes everything by integral, holistic

and parallel self-spreading, self-covering and self-matching distributed spaces that may be physical, virtual, or combined, as symbolically shown in Fig. 2.

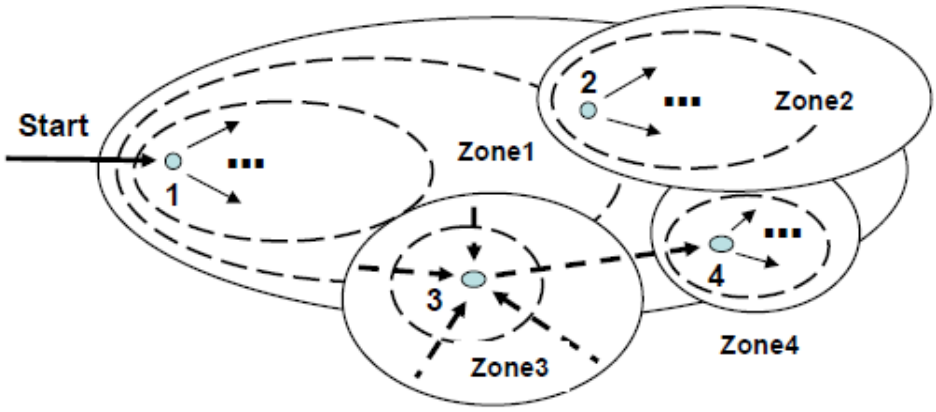
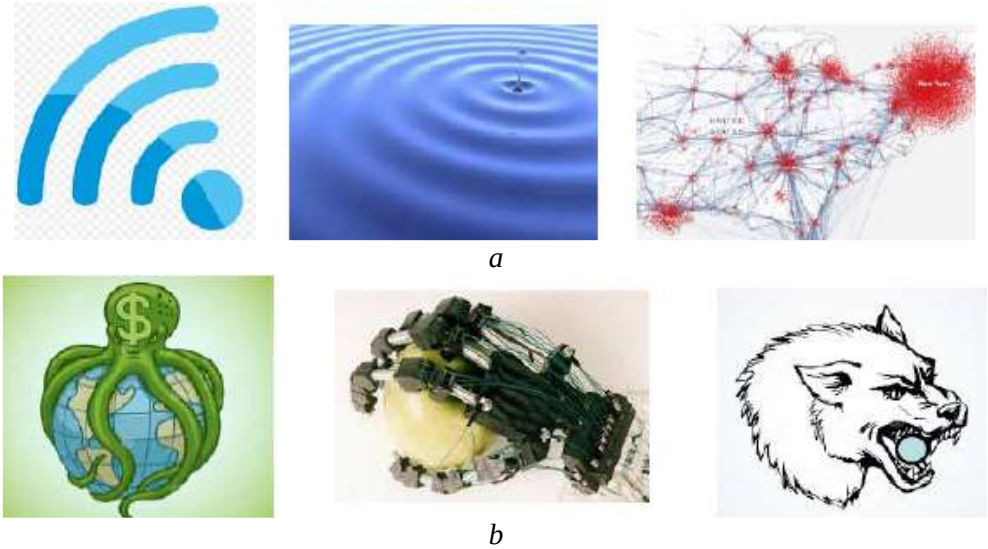


Figure 2 – Holistic and parallel world coverage by an active recursive code

Originating in some world point 1 and spreading as Zone 1, it may trigger another space coverage starting in secondary centers (like 2, 3, and 4). In different areas (like Zone 3) complex solutions may be found in both forward and feedback manner, on the results of which another spreading and solution areas may appear (like Zone 4), and so on. Combinations of activation, spreading, and forward-backward problem solving may be arbitrary and by any numbers, depths, hierarchies, and space coverage. Some physical analogies of this paradigm are shown in Fig. 3 which may include spreading waves or even viruses (Fig 3 *a*), as well as different grasping capabilities (Fig. 3 *b*) where objects or areas seized or grasped could be arbitrarily large and of any nature.



a

b

Figure 3 – Physical analogies: (a) spreading waves and viruses, (b) grasping capabilities

2.3. Relation of SG to higher mental and philosophical concepts

The paradigm discussed in this paper is philosophically based on much higher and more general concepts than the ones used for the system descriptions mentioned in Subsection 2.1, with some of them following. Understanding is a psychological process related to an abstract or physical object, such as a person, situation, or message whereby one is able to use concepts to model an object. Understanding is a relation between the knower and an object of understanding [6]. Also,

understanding the problem is often the main part of its solution [7, 8]. Perception is the organization, identification, and interpretation of sensory information in order to represent and understand the presented information or environment [9]. Self-Awareness and Mental Perception go even higher [10]. At the highest level, there is the concept of Consciousness [11], with many theories and fantasies of what it actually means. It can exist outside the head [12] or even does consciousness pervade the Universe [13], also a relation of consciousness to space [14, 15]. The soul within many religious, philosophical, and mythological traditions is the incorporeal essence of a living being. It is to comprise the mental abilities of a living being: reason, character, feeling, consciousness, memory, perception, thinking, etc. Depending on the philosophical systems, a soul can be either mortal or immortal [16], and such guesses as its possible separation from the body [17], when does it enter the human body, or where does it reside [18] are even fantasized too.

2.4. WAVE concept against The Society of Mind

In conclusion to this section, it is worth mentioning a well-known book of Prof. Marvin Misky The Society of Mind [19], and its comparison with the previous author's concept called WAVE [20–27] based on the discussed above ideas. The book portrayed the human mind as a "society" of tiny components, or agents, themselves mindless, together like a mosaic, but their frequent interactions were somehow forming something that was called "intellect". This view reminded representation of traditional systems discussed in Subsection 2.1, whereas WAVE was conceptually much closer to such notions as "awareness", "consciouness", and even "soul", which was discussed later in the related publications [28, 29].

3. Spatial Grasp Technology Basics

Here is shown how the discussed above spatial paradigm has been patented [30] and converted into a real technology prototyped in different countries and tested on numerous applications [31–37]. Within Spatial Grasp Technology (SGT), a high-level scenario for any task to be performed in a distributed world is represented as an active self-evolving pattern rather than a traditional program, sequential or parallel one. This pattern, written in a high-level Spatial Grasp Language (SGL) and expressing top semantics of the problem to be solved, can start from any point of the world. Then it spatially propagates, replicates, modifies, covers and matches the distributed world in a parallel wave-like mode, while echoing the reached control states and data found or obtained for making decisions at higher levels and further space navigation.

3.1. The worlds SGT operates with

SGT allows direct operation with different world representations: Physical World (PW), considered as continuous and infinite where each point can be identified and accessed by physical coordinates; Virtual World (VW) which is discrete and consists of nodes and semantic links between them; and Executive world (EW) consisting of active "doers" with communication possibilities between them. Different kinds of combinations of these worlds can also be possible within the same formalism, like Virtual-Physical World (VPW), Virtual-Execution World (VEW), Execution-Physical World (EPW), and Virtual-Execution-Physical World (VEPW) combining all features of the previous cases.

3.2. Spatial Grasp Language syntax

SGL top-level syntax can be defined as follows:

grasp → *constant* | *variable* | *rule* [({ ***grasp***,})]
constant → *information* | *matter* | *custom* | *special* | ***grasp***
variable → *global* | *heritable* | *frontal* | *nodal* | *environmental*

rule → *type* | *usage* | *movement* | *creation* | *echoing* | *verification* | *assignment* | *advancement* | *branching* | *transference* | *exchange* | *timing* | *qualifying* | ***grasp***

An SGL scenario, called grasp, applied in some point of the distributed space, can just be a constant directly providing the result to be associated with this point. It can be a variable whose content, assigned to it previously when staying in this or (remotely) in another space point (as variables may have non-local meaning and coverage), provides the result in the application point as well. It can also be a rule (expressing certain action, control, description, or context) optionally accompanied with operands separated by comma (if multiple) and embraced in parentheses. These operands can be of any nature and complexity (including arbitrary scenarios themselves) and defined recursively as grasp, i.e. can be constants, variables, or any rules with operands (i.e., as grasps again), and so on. The full syntax of the latest SGL version is provided in Subsection 3.7.

3.3. SGL Rules

Rules, starting in some world points, can organize navigation of the world sequentially, in parallel, or any combinations thereof. They can result in staying in the same application point or can cause movement to other world points with the obtained results to be left there, as in the final points of the rule. Such results can also be collected, processed, and returned to the starting point of the rule, the latter serving as the final one on this rule. The final world points reached after the rule invocation can themselves become starting ones for other rules. The rules, due to recursive language organization, can form arbitrary operational and control infrastructures expressing any sequential, parallel, hierarchical, centralized, localized, mixed and up to fully decentralized and distributed algorithms.

3.4. SGL Variables

SGL Variables include Global variables (the most expensive and rarely used ones) which can serve any SGL scenarios and be shared by them, also by their different branches; Heritable variables appearing within a scenario step and serving all subsequent, descendent steps; Frontal variables serving and accompanying the scenario evolution, being transferred between subsequent steps; Environmental variables allowing us to access, analyze, and possibly change different features of physical, virtual and executive worlds during their navigation; and finally, Nodal variables as a property of the world positions reached by scenarios and shared with other scenarios in the same positions.

3.5. Elementary SGL programming examples

• `add(7,8)` – finds the sum of two values when staying in some world point and leaves the result there.

• `assign(Result, add(7,8))` – the found sum of two values is assigned to a variable Result which will be staying in the same point.

• `move(x_y)` – from the current world point there is made a physical move to another physical point with the given coordinates.

• `create(John)` – there is created an isolated virtual node John.

• `hop(John); create(+father, Peter)` – there appears the extension of the existing single-node virtual network with a new node and relation to it, where John will be treated as the father of Peter.

• `move(x_y); repeat(shift(dx_dy); TEMPERATURE > 0)` – starting from the world point with proper coordinates, in the chosen direction there is organized a repetitive movement unless the temperature in the reached physical locations remains above zero.

• `if(hop(Peter), create(Lilia, Olga, Ann))` – in case of the existence of virtual node Peter, there will be created three new isolated virtual nodes with proper names.

3.6. SGL spatial interpretation

Communicating Interpreters of SGL can be in an arbitrary number of copies, up to millions and billions, which can be effectively integrated with any existing systems and communications, and their dynamic networks can represent powerful spatial engines capable of solving any problems in terrestrial and celestial environments. Such collective engines can simultaneously execute many cooperative or competitive tasks-scenarios without any central resources or control, as symbolically depicted in Fig. 4 (SGL interpreters are named U as universal computational and management nodes which may be stationary or requested and runtime located in proper space points on the demand of SGL scenarios).

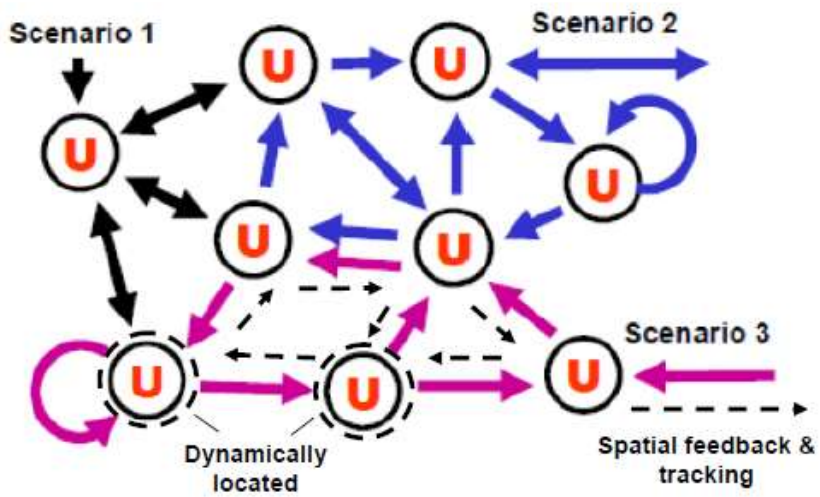


Figure 4 – Spatial interpretation of SGL scenarios

As both the backbone and nerve system of the distributed interpreter, its self-optimizing spatial track system provides hierarchical command and control as well as remote data and code access. It also supports spatial variables and merges distributed control states for decisions at different organizational levels in a feedback mode. The track infrastructure is automatically distributed between active components (humans, robots, computers, smartphones, satellites, etc.) during the self-spreading of the scenario in distributed environments.

3.7. Full SGL Language

In the language descrtiption given below, syntactic categories are in italics, vertical bar separates alternatives, parts in braces indicate zero or more repetitions with a delimiter at the right, and constructs in brackets are optional. The remaining characters and words are the language symbols (including boldfaced braces).

grasp → *constant* | *variable* | [*rule*] [({ *grasp* , })]
constant → *information* | *matter* | *special* | *custom* | *grasp*
information → *string* | *scenario* | *number*
string → ' { *character* } '
scenario → **{** { *character* } **}**
number → [*sign*] { *digit* } [. { *digit* } [e [*sign*] { *digit* }]]

matter → "{*character*}"

special → thru|done|fail|fatal|infinite|nil|any|all|other|allother|current|passed|existing|neighbors|direct|forward|backward|synchronous|asynchronous|virtual|physical|executive|engaged|vacant|firstcome|unique

variable → *global* | *heritable* | *frontal* | *nodal* | *environmental*

global → G{*alphameric*}

heritable → H{*alphameric*}

frontal → F{*alphameric*}

nodal → N{*alphameric*}

environmental → TYPE|NAME|CONTENT|ADDRESS|QUALITIES|WHERE|BACK|PREVIOUS|PREDECESSOR|DOER|RESOURCES|LINK|DIRECTION|WHEN|TIME|STATE|VALUE|IDENTITY|IN|OUT|STATUS

rule → *type* | *usage* | *movement* | *creation* | *echoing* | *verification* | *assignment* | *advancement* | *branching* | *ransference* | *exchange* | *timing* | *qualifying* | *grasp*

type → global|heritable|frontal|nodal|environmental|matter|number|string|scenario|constant|custom

usage → address|coordinate|content|index|time|speed|name|place|center|range|doer|node|link|unit

movement → hop|hopfirst|hopforth|move|shift|follow

creation → create|linkup|delete|unlink

echoing → state|rake|order|unit|unique|sum|count|first|last|min|max|random|average| sortup|sortdown|reverse|element|position|fromto|add|subtract|multiply|divide|degree|separate|unite|attach|append|common|withdraw|increment|decrement|access|invert|apply|location|distance

verification → equal|nonequal|less|lessorequal|more|moreorequal|bigger |smaller|heavier|lighter|longer|shorter|empty|nonempty|belong|notbelong|intersect|notintersect|yes|no

assignment → assign|assignpeers

advancement → advance|slide|repeat|align|fringe

branching → branch|sequence|parallel|if|or|and|or_sequence|or_parallel|and_sequence |and_parallel| choose|quickest|cycle|loop|sling|whirl|split|replicate

transference → run|call

exchange → input|output|send|receive|emit|get

timing → sleep|allowed

qualification → `contain|release|trackless|free|blind|quit|abort|stay|lift|seize|exit`

4. Examples of some mechanisms of distributed SGL interpretation

Generally, such examples can be better explained on the growing spatial trees, but for compactness, they are given below in a simplified way.

4.1. Advancement

Advancement: `advance(g1,g2,g3)` or `advance_sync(g1,g2,g3)`.

Operations g1, g2, g3 are considered to be executed sequentially, one after another, where each new one should be applied in parallel to all positions in space reached by the previous operation. Such a scenario can be executed in an asynchronous or synchronized manner, as follows.

Asynchronous advancement

In an asynchronous solution, each new operation begins to develop immediately from any Pi points reached by the previous operation without waiting for its completion in other Pi points, with the rest of the scenario moving in space while omitting the used operations, as shown in Fig. 5.

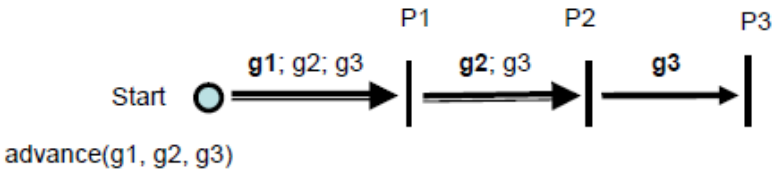


Figure 5 – Non-synchronized advancement in space

Synchronous advancement

In a synchronized case, each new operation can be executed from Pi points reached by the previous operation only after all Pi have been completed, which should be reported to the Start point. After this, the next operation will be launched from all Pi, to which its text should be delivered from the Start, as in Fig. 6. This mode uses the feedback capability via the SGT tracking mechanism of self-spreading scenarios which may have different implementations, not necessarily using the found paths from the Start to Pi, and even may just be in returning to Start the direct address of Pi nodes if the latter is available.

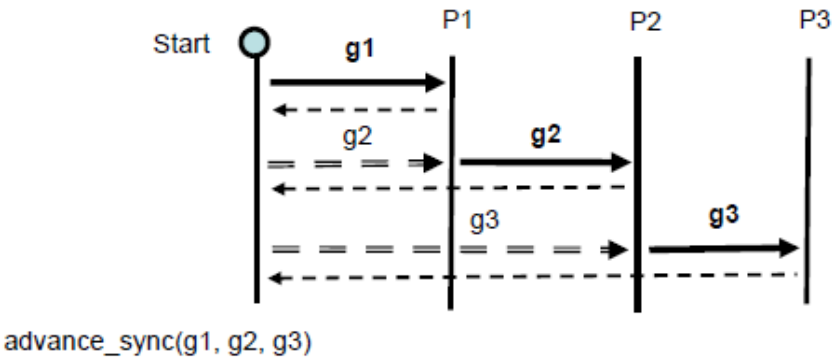


Figure 6 – Synchronized advancement in space

Another solution for the synchronized advancement may be movement in space of the whole scenario text losing each time the utilized operations, similar to the asynchronous case of Fig. 6, as now in Fig. After the feedback from full completion of Pi, the Start node will issue the permission in all Pi to start the next operation, with the rest of the scenario followed by it in space.

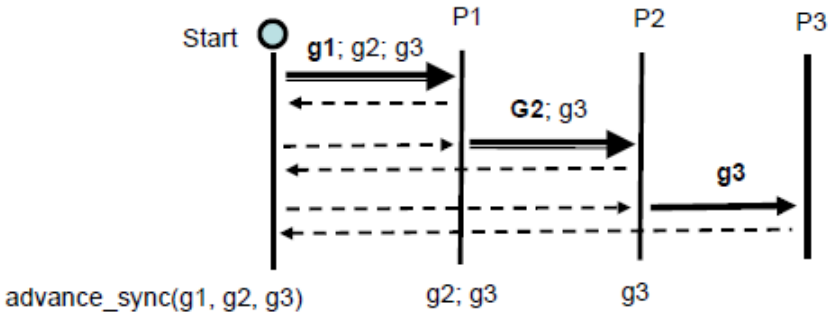


Figure 7 – Another variant of the synchronized advancement

4.2. Repetitive advancement

Repetitive advancement: `repeat(g1);g2` or `repeat_sync(g1);g2`.

Below there is given an example of how the repeated invocation of operation g1 can be managed in the asynchronous and synchronized way, similar to the previous advancement case, after the termination of which the next operation g2 should be invoked from all final points reached by the repetition of g1.

Asynchronous repetition

For the asynchronous way, the same scenario repeated for g1 with g2 following it will be invoked in all points of Pi reached by the latest g1 copy, and in case it is impossible, g2 will be immediately invoked from the same points, as in Fig. 8. In the fully asynchronous way, with independent and parallel development is space which may have different features in different directions, g1 may terminate in different space points after different repetitions, so the applications of g2 may happen to be involved not only from Pn as in Fig. 8 but also potentially from any previous Pi.

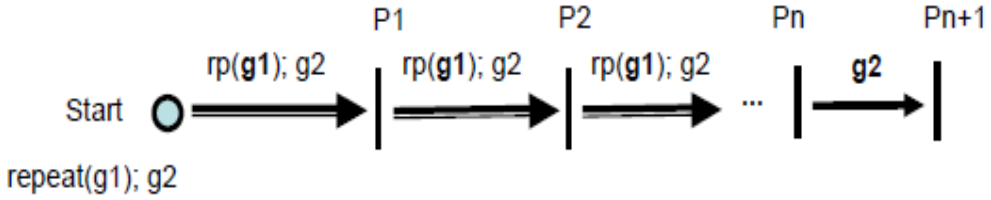


Figure 8 – Non-synchronized repeated advancement in space

Synchronous repetition

For the synchronized repetition in space, each new repetition of g1 can be applied only after full completion in space of all copies of g1, and then to the successfully reached nodes Pi another copy of g1 will be delivered from the Start. In case of its final invocation, to the points Pn from which it failed, operation g2 will be delivered for execution, as in Fig. 9. This synchronous case may happen to operate differently in comparison with the previous asynchronous one, and failure of g1 in previous Pi will not be followed by g2.

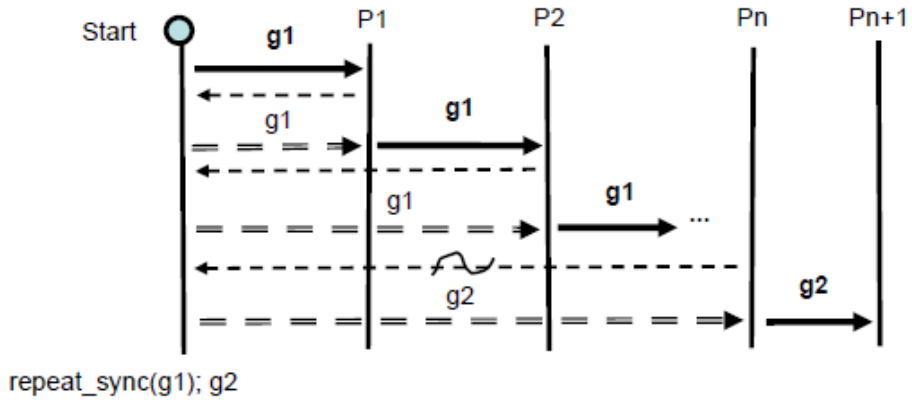


Figure 9 – Synchronized repeated advancement in space

Another solution for the synchronous repetition development may be a combination with techniques of the previous asynchronous one, where both g1 and g2 always propagate in space between Pi, being left in the reached Pi points for the next step. And from the Start node, only permission to develop g1 further of apply g2 will be issued on the received feedback, without each time delivering of their texts from the Start, as in Fig. 10.

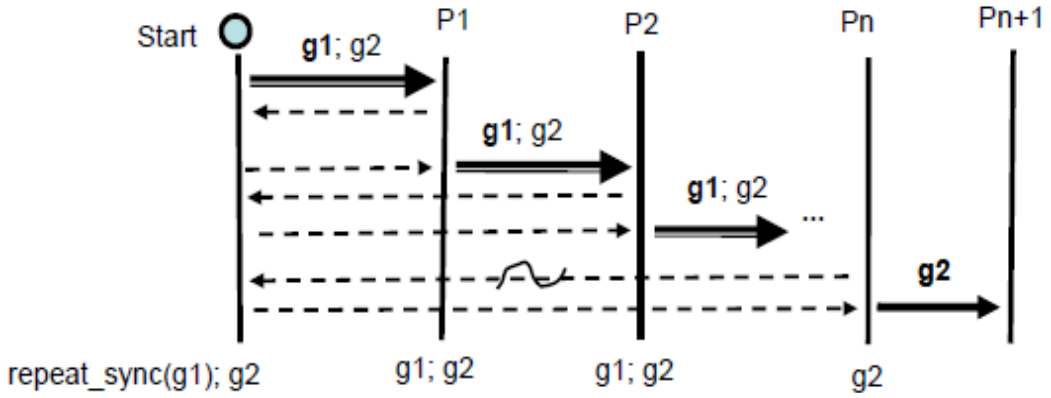


Figure 10 – Another variant of the repeated advancement

4.3. Feedback-based parallel spatial computation

Feedback-based parallel spatial computation: `sum(g1;g2;g3)`.

This scenario propagates in a distributed space stepwise by g1, g2 and g3 operations, similar to Fig. 6, with each new one developing from all space points reached by the previous operation, and then summates all values reached by the final operation g3. Fig. 11 shows how to use the track tree that appeared from the spreading from Start to all final values (which may be remote and in large quantity) to make this summation hierarchical and in a feedback manner while preliminary leaving in all forwardly passed Pi points the summation operation. This will effectively convert the SGT track mechanism into a parallel spatial computer obtaining and returning the final result to the Start position.

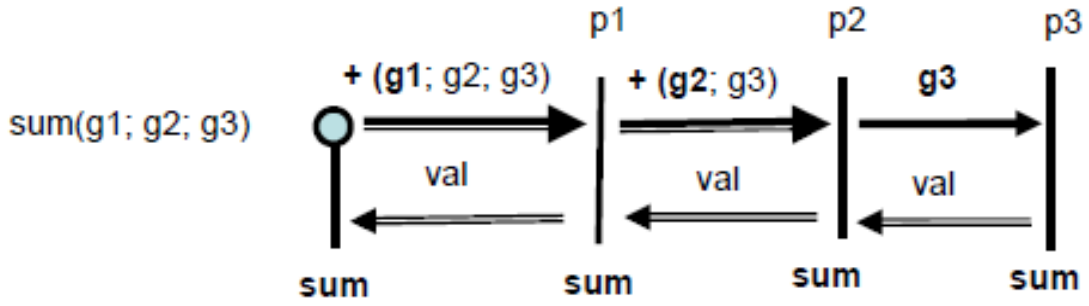


Figure 11 – Parallel hierarchical feedback operation

4.4. Combination of feedback computation with forward advancement

Combination of feedback computation with forward advancement: `maxdest(g1;g2;g3);g4`.

This scenario (see Fig. 12), which is similar to the previous one presented in Fig. 11, propagates in space to the final values obtained by g3 and finds maximum value among them using preliminary left max operation in the Start and passed Pi points. After that it delivers the next operation g4 to the point in P3 having this maximum value, using from the Start the resultant track path to this particular point. Thus this scenario works in a combination of spatial forward, backward, and then forward again mode, and uses the distributed track system as a parallel hierarchical computer.

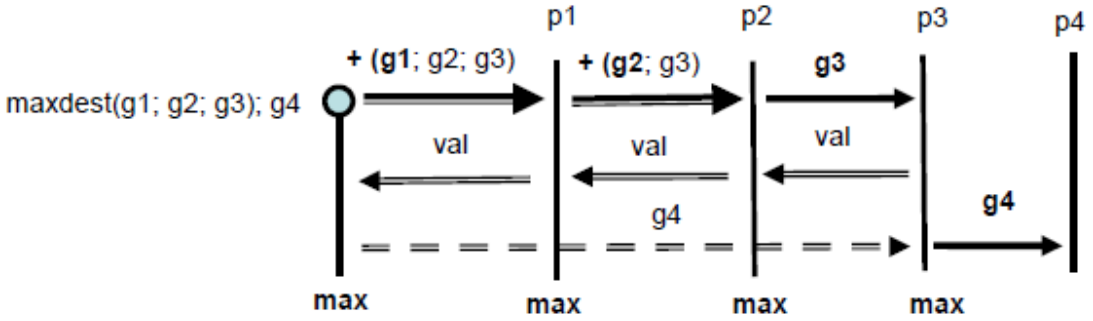


Figure 12 – Combining hierarchical feedback operation with further space navigation

5. Examples of high-level spatial vision and comprehension

The following examples combine and integrate two main functionalities of SGL scenarios: parallel self-spreading in distributed spaces like waves or viruses, some with spatial pattern-matching, and parallel grasping of obtained solutions in a feedback hierarchical mode, reflecting natural world analogies shown in Fig. 3 *a* and *b*, being also in line with higher, or mental, features of the SG paradigm discussed in Subsection 2.3.

5.1. Hierarchical coverage in finding borders of a large spatial image

Self-spreading in space in different directions

Starting from some point that is definitely inside the region of interest, the following scenario organizes stepwise wave-like propagation through the region in different extending directions until the reached points are classified inside the region, as in Fig. 13. And the first points that are reached in this self-spreading spatial process which are not considered as the ones belonging to the region (by a given brightness `Threshold`), are declared as those being on the border of the

region. Their coordinates will be returned to the `Start` position in a parallel echo process while collected and declared there as the external border of the region.

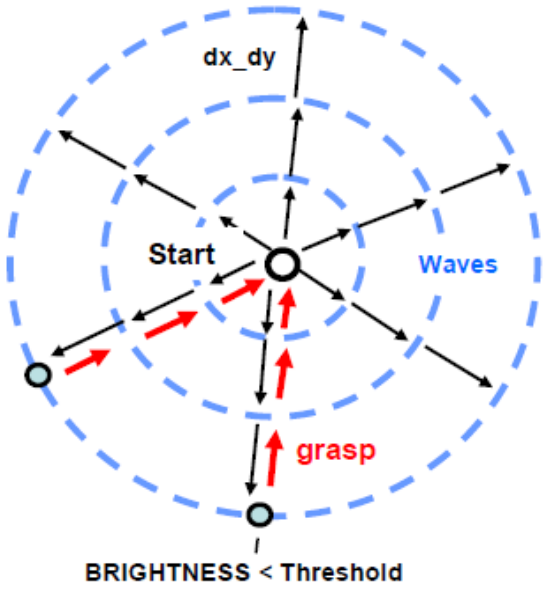


Figure 13 – Combining wave-like spreading with feedback grasping in finding border of a region

```
move(Xstart_Ystart); frontal(Threshold = brightness, Shift);
Border =
  (dx1_dy1, dx2_dy2, ..., dx6_dy6; Shift = VALUE;
   repeat(shift(Shift); if(BRIGHTNESS < Threshold,
done(WHERE))));
output(Border)
```

The scenario first splits into a recommended number of directions with local *dx_dy* shift parameters for each. After that, it repeatedly propagates through the chosen directions with each individual `Shift` until it is possible with the given `Threshold` brightness.

Self-spreading coverage by growing and splitting tree

A more detailed solution for finding the region's external border can be obtained by the growing tree-like space coverage, where at each step of extending the distance from the `Start` the tree splits into more branches, as shown in Fig. 14. The finally reached and returned points (which may be much more than presented in Fig. 13) will characterize the border of the region with higher details.

```
move(Xstart_Ystart);
frontal(Step = 1, Depth = distance, Branch = number,
        Threshold = brightness, Start = WHERE);
Border =
   repeat(shift(define_next(Start, WHERE, Depth, Step, Branch));
          if(BRIGHTNESS < Threshold, done(WHERE));
          increment(Step));
output(Border)
```

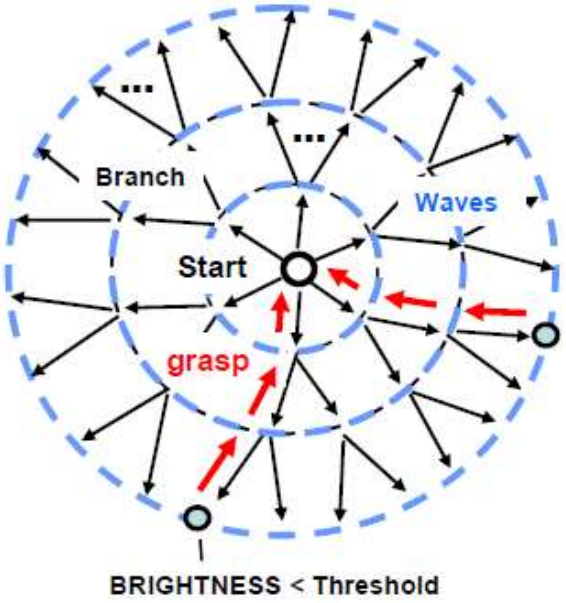
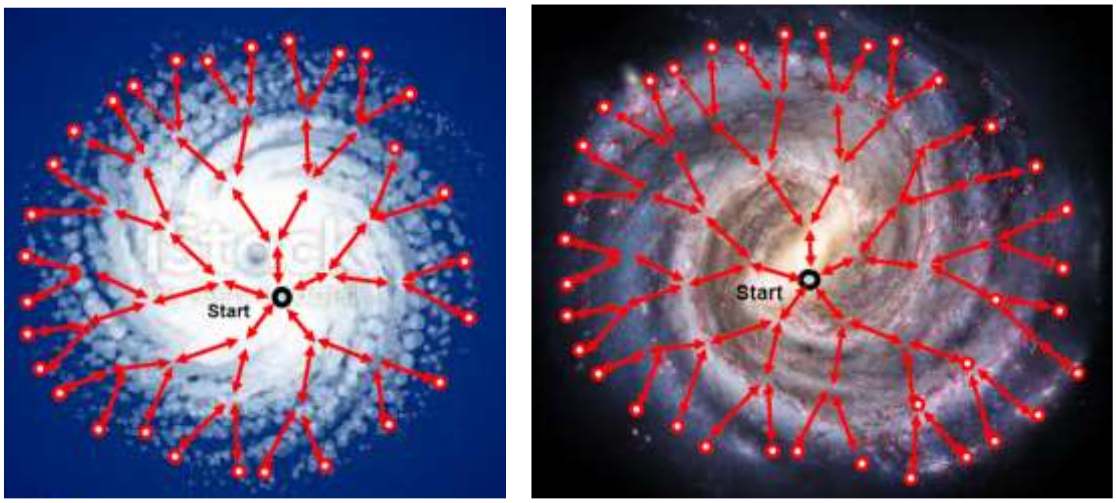


Figure 14 – Combining tree-like spreading with feedback grasping
in finding the border of the region

In each reached point of space the rule `define_next`, which uses proper parameters, defines a set of new physical shifts which are implemented in parallel by the rule `shift`. These parameters include `Start` (propagating scenario starting point coordinates), `WHERE` (current point coordinates), `Depth` (propagating the given distance between the neighboring wavefronts), `Step` (propagating increasing step number), and `Branch` (propagating the recommended number of splits in each point).

Using the stated above methods for the practical finding of borders of large regions

In Fig. 15 *a*, there is shown a possible application of the mentioned above tree-like space coverage used for finding the external border of a hurricane by parallel propagation through its local visibilities with getting the final global image in a spatial grasp mode. A similar solution can also be used in other applications oriented on seeing and analyzing the extension and shape of larger spatial formations like, for example, a galactic, as in Fig. 15 *b*.



a *b*

Figure 15 – Possible application to seeing and outlining the hurricane (a) or galactic (b)

5.2. Virus-like spatial coverage for finding borders of complex regions

But the shown above solutions may not work well for the analysis of very complex regions like, for example, forest fires, as in Fig. 16, which may have any shapes (including distorted and curved) which are not so effectively covered by well-structured top-down trees. The following very simple solution for such cases is based on the idea of global viruses which can self-replicate and self-spread in various directions, including backward (using the resemblance of SGT to viruses, as in Section 2 and Fig. 3*a*). The repeated and free propagation in the solution given below does not leave spatial tracks for the feedback use as in the previous scenarios but rather individually picks up and sends coordinates of the reached border points directly to the `Start` position. The latter finally outputs the collected positions as the border of the region (see Fig. 16), with issuing final solution after some `Expected` time interval, as the `free` propagation and termination of self-spreading waves are not controlled from `Start` (which could otherwise be expensive due to their virus-like behavior).

```
move(Xstart_Ystart);
frontal(Start = ADDRESS, Limit = steps, Zone = color,
        Branches = split, Expected = time);
nodal(Border);
branch(
  free_repeat(Limit)(
    replicate(Branches, shift(random(dx_dy)));
    if(COLOR != Zone,
       done(append((hop(Start); Border), WHERE)))),
  (sleep(Expected); output(Border)))
```

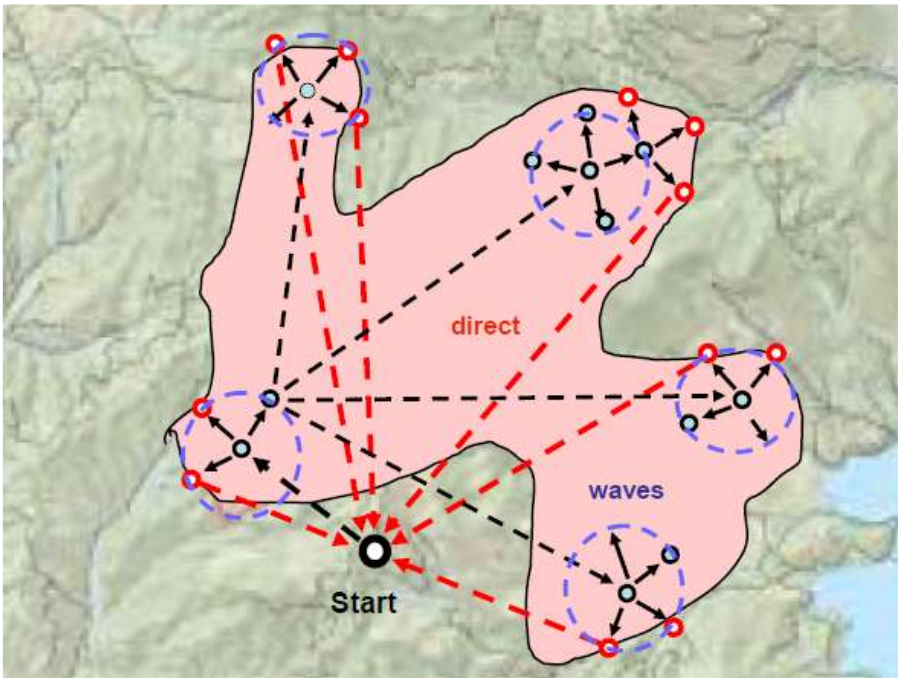


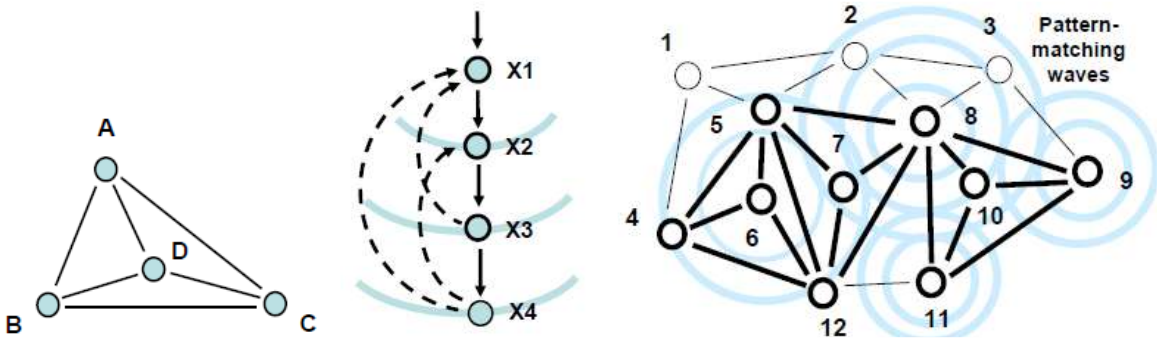
Figure 16 – Virus-like investigation and finding the border of the forest fire

5.3. Discovering images in a distributed network

This example relates to spatial seeing of fully distributed networks and finding proper structures, or images, in them like the one in Fig. 17 *a*, which is represented for the wave-like coverage as a

search template of Fig. 17*b*. Many similar tasks and solutions can be found in existing publications on SGT, including [20–47]. Therefore below there is shown only a solution for this specific case without a detailed explanation which may be for the network of Fig. 17*c*. It should be mentioned here that the structure to be found represents a three-node clique where each node has links with the other two nodes (general solutions for any cliques can be found in [33–36]). Providing that finding solutions is activated in parallel from all nodes, with the parallel wave-like propagation and pattern matching until proper images are found, there are allowed solutions for each clique to grow only in proper sequence of values of their names to exclude solution duplicates (comparing every new `NAME` of the node with its predecessors by `PRED`).

```
hop(all); frontal(X) = NAME;
hop(neighbors); NAME < PRED; append(X, NAME);
hop(neighbors); NAME < PRED; yes(hop(X[1])); append(X, NAME);
hop(neighbors); NAME < PRED; yes(hop_and(X[1,2])); append(X,
NAME);
output_unit(X)
```



a *b* *c*

Figure 17 – Finding certain graph structures in a distributed network: (a) image to be found, (b) matching template, (c) parallel wavelike network matching

The results of the described above scenario will be issued individually by the last nodes reached by the search template of Fig. 17 *b* (i.e., the nodes matched by `X4`). It confirms the foud solution. But it is also possible to collect and output all found solutions in a single position from which the whole scenario starts, using the SGL global grasp capability, as follows:

```
output(
  hop(all); frontal(X) = NAME;
  hop(neighbors); NAME < PRED; append(X, NAME);
  hop(neighbors); NAME < PRED; yes(hop(X[1])); append(X, NAME);
  hop(neighbors); NAME < PRED; yes(hop_and(X[1,2])); append(X,
NAME);
  done_unit(X))
```

The result of this scenario for the network of Fig. 17*c* will be the following three cliques: (12, 6, 5, 4), (12, 8, 7, 5), (11, 10, 9, 8).

5.4. Possible implementation issues

The discussed scenarios describe mainly the top semantics of the mentioned problems and their high-level solution methods, in the most parallel and fully distributed way. They can be practically implemented in a variety of ways, some of which are mentioned below.

• The scenario is processed by the already existing SGL interpretation network where each measurement device with its embedded processor happens to be located just in or nearby every spatial point reached by the scenario.

• The measurement-processing devices with SGL interpreters are requested and placed into the space points reached by the scenario on the scenario demand, whereas the number of such available and moving devices can determine the implementation productivity which can also depend on the time needed for them to reach the proper physical points.

• The scenario works in a simulation mode, with the model of space being distributed between different computers which do not need physical movement, so their existing number will determine real parallelism and execution time.

• The needed information on different space points is runtime taken from existing databases and publications which can be obtained from different sources and by different computers, so this case may be in some sense close to the previous one.

• The description of space, whose scenario processes, is already located or received in a single computer, and the scenario can work similarly to as if they have been written in traditional languages, although many features of SGL may fit better the discussed spatial methods.

• Any possible combinations of the described above cases.

Different possibilities of practical implementation of scenarios in SGL and its previous versions can be found in the existing publications including [20–47].

6. Conclusions

This paper has described some latest results on the further development of Spatial Grasp (SG) Paradigm in two interlinked directions: philosophical-conceptual and technological-implementational, with the following obtained main results. More details have been presented and explained of how SG develops in distributed spaces in the form of waves or even viruses, and how it grasps at the same time complex solutions of spatial problems, providing physical analogies helping to understand these features. The paper has also shown how it fundamentally differs from traditional representations of distributed systems and solutions in them as consisting of parts or agents exchanging messages. The SG philosophy has also been directly linked with some higher intellectual and mental features like “perception”, “awareness”, “consciousness” and even “soul”, with comparison of its earlier version called WAVE with the existing publications treating human intellect as a “society of mind”. The latest Spatial Grasp Technology details have been briefed, implementing main SG features and capabilities, where Spatial Grasp Language (SGL) interpreters can be arbitrarily networked as spatial computers and cover any terrestrial and celestial spaces. Full description of the latest version of SGL language has provided as well. Distributed interpretation mechanisms of main SGL constructs have been revealed and discussed for the first time and in full detail, which relate to the new patent on SGT being prepared. These mechanisms allow implementation of powerful spatial control and processing functionality without any centralized resources and in any environments. The paper also provided examples of fully distributed solutions for seeing and evaluating very large distributed phenomena like hurricanes and forest fires (up to galaxies), as well as large networks which cannot be fully observed from any points but can be clearly seen and analyzed under SGT. This provides a sort of holistic spatial vision and analysis of large distributed worlds as a whole, thus confirming the higher-level conceptual orientation and mental-like capability of the SG paradigm. The latest SGT version can be

effectively implemented and used even in traditional university environments, similar to the previous versions in different countries.

REFERENCES

1. System. URL: https://en.wikipedia.org/wiki/System.
2. Distributed systems. URL: https://computersciencewiki.org/index.php/Distributed_systems.
3. Distributed control system. URL: https://en.wikipedia.org/wiki/Distributed_control_system.
4. Buyya R., Selvi T. Parallel and Distributed Computing, in Mastering Cloud Computing. Elsevier Inc. 2013. URL: https://www.sciencedirect.com/topics/computer-science/distributed-programming.
5. List of concurrent and parallel programming languages. URL: https://en.wikipedia.org/wiki/List_of_concurrent_and_parallel_programming_languages.
6. Understanding. URL: https://en.wikipedia.org/wiki/Understanding.
7. First, seek to understand the problem. URL: https://www.emergn.com/insights/first-seek-to-understand-the-problem/.
8. Understanding Your Problem Is Half the Solution (Actually the Most Important Half). URL: https://www.henricodolfing.com/2018/05/understanding-your-problem-is-half.html.
9. Perception. URL: https://en.wikipedia.org/wiki/Perception.
10. Kobayashi H. Self-Awareness and Mental Perception. *Journal of Indian Philosophy*. 2010. Vol. 38. P. 233–245. URL: https://link.springer.com/article/10.1007/s10781-010-9096-6.
11. Consciousness. URL: https://en.wikipedia.org/wiki/Consciousness.
12. Tonneau F. Consciousness outside the Head. *Behavior and Philosophy*. 2004. Vol. 32, N 1. The Study of Behavior: Philosophical, Theoretical and Methodological Challenges. P. 97–123. URL: https://www.jstor.org/stable/27759473.
13. Cook G. Does Consciousness Pervade the Universe? *Scientific American*. 2020. January 14. URL: https://www.scientificamerican.com/article/does-consciousness-pervade-the-universe/.
14. Galland D., Grønning M. Spatial consciousness. The Wiley-Blackwell Encyclopedia of Urban and Regional Studies. Wiley-Blackwell. 2019. URL: https://www.researchgate.net/publication/330753755_Spatial_consciousness.
15. McGinn C. Consciousness and Space. Rutgers University. 1997. URL: https://www.nyu.edu/gsas/dept/philo/courses/consciousness97/papers/ConsciousnessSpace.html.
16. Soul. URL: https://en.wikipedia.org/wiki/Soul.
17. Smith A. Separation of Soul From Body. In: Porphyry's Place in the Neoplatonic Tradition. Springer, Dordrecht. 1974. P. 20–39. URL: https://link.springer.com/chapter/10.1007/978-94 -010-1604-9_2.
18. How And When Does The Soul Enter The Human Body & Where Does It Reside, Daaji, Heartfulness. URL: https://www.youtube.com/watch?v=dhsXyUE8qc8.
19. Minsky M. The Society of Mind. Simon & Schuster, 1988. 336 p.
20. Sapaty P.S. The WAVE-0 language as a framework of navigational structures for knowledge bases using semantic networks. *Proc. USSR Academy of Sciences. Technical Cybernetics*. 1986. N 5 (in Russian).
21. Sapaty P.S. A wave language for parallel processing of semantic networks. *Computers and Artificial Intelligence*. 1986. Vol. 5, N 4. P. 289–314.
22. Sapaty P.S. The WAVE-1: A new ideology and language of distributed processing on graphs and networks. *Computers and Artificial Intelligence*. 1987. N 5. P. 9–10.
23. Sapaty P.S. WAVE-1: A new ideology of parallel processing on graphs and networks. *Future Generations Computer Systems*. North-Holland, 1988. Vol. 4. P. 9–10.
24. Sapaty P.S. The WAVE machine project. Proc. IFIP Workshop on Silicon Architectures for Neural Nets, St. Paul de Vence, France. 1990. 28–30 November.
25. Sapaty P.S. Logic flow in active data. *VLSI for Artificial Intelligence and Neural Networks* / W.R. Moore & J. Delgado-Frias (ed.). Plenum Press, New York and London, 1991.
26. Sapaty P.S. The WAVE paradigm. Proc. JICSLP'92 Post-Conference Joint Workshop on Distributed and Parallel Implementations of Logic Programming Systems, Washington, D.C. 1992. 13–14 November.
27. Sapaty P.S., Borst P.M. An overview of the WAVE language and system for distributed processing in open networks. Technical Report, Dept. Electronic & Electrical Eng, University of Surrey, June 1994. P. 9–10.

28. Sapaty P.S. Simulating distributed and global consciousness under spatial grasp paradigm. Adv. *Mach. Learn. Artif. Intell*. 2020. Vol. 1, N 22. URL: https://www.opastonline.com/wp-content/uploads/ 2020/ 12/simulating-distributed-and-global-consciousness-under-spatial-grasp-paradigm-amlai-20.pdf.
29. Sapaty P. Symbiosis of Real and Simulated Worlds Under Global Awareness and Consciousness, Abstract at The Science of Consciousness Symposium TSC 2020. URL: https://eagle.sbs.arizona.edu/sc/ report_poster_detail.php?abs=3696.
30. Sapaty P.S. A distributed processing system. European Patent N 0389655, Publ. 10.11.93. European Patent Office. 35 p.
31. Sapaty P.S. Mobile Processing in Distributed and Open Environments. New York: John Wiley & Sons, 1999. 410 p.
32. Sapaty P.S. Ruling Distributed Dynamic Worlds. New York: John Wiley & Sons, 2005. 255 p.
33. Sapaty P.S. Managing Distributed Dynamic Systems with Spatial Grasp Technology, Springer, 2017. 284 p.
34. Sapaty P.S. Holistic Analysis and Management of Distributed Social Systems, Springer, 2018. 234 p.
35. Sapaty P.S. Complexity in International Security: A Holistic Spatial Approach, Emerald Publishing, 2019. 160 p.
36. Sapaty P.S. Symbiosis of Real and Simulated Worlds under Spatial Grasp Technology, Springer, 2021. 251 p.
37. Sapaty P.S. Spatial Grasp as a Model for Space-Based Control and Management Systems, Taylor and Francis, 2022. 280 p. (in print). URL: https://www.routledge.com/Spatial-Grasp-as-a-Model-for-Space-based-Control-and-Management-Systems/Sapaty/p/book/9781032136097
38. Sapaty P.S. Global Network Management under Spatial Grasp Paradigm. Global Journal of Researches in Engineering: J General Engineering. 2020. Vol. 20, Issue 5, Version 1.0. P. 58–69. URL: https://engineeringresearch.org/index.php/GJRE/article/view/2082/2013.
39. Sapaty P.S. Advanced terrestrial and celestial missions under Spatial Grasp Technology. *Aeronautics and Aerospace Open Access Journal*. 2020. Vol. 4, Issue 3. P. 81–88. URL: https://medcraveonline.com/AAOAJ/AAOAJ-04-00110.pdf.
40. Sapaty P.S. Managing Multiple Satellite Architectures by Spatial Grasp Technology. *Mathematical machines and systems*. 2021. N 1. P. 3–16. URL: http://www.immsp.kiev.ua/ publications/eng/2021_1/.
41. Sapaty P.S. Spatial Management of Large Constellations of Small Satellites. *Mathematical machines and systems*. 2021. N 2. P. 3–14. URL: http://www.immsp.kiev.ua/ publications/articles/ 2021/2021_2/02_21_Sapaty.pdf.
42. Sapaty P.S. Global Management of Space Debris Removal Under Spatial Grasp Technology. *Acta Scientific Computer Sciences.* 2021. Vol. 3, Issue 7. URL: https://www.actascientific.com/ ASCS/ pdf/ASCS-03-0135.pdf.
43. Sapaty P.S. Global Network Management under Spatial Grasp Paradigm. *International Robotics & Automation Journal*. 2020. Vol. 6, Issue 3. P. 134–148. URL: https://medcraveonline.com/ IRATJ/IRATJ-06-00212.pdf.
44. Sapaty P.S. Space Debris Removal under Spatial Grasp Technology. *Network and Communication Technologies*. 2021. Vol. 6, N 1. URL: https://www.ccsenet.org/journal/index.php/nct/article/view/0/45486.
45. Sapaty P.S. Spatial Grasp Model for Management of Dynamic Distributed Systems. *Acta Scientific Computer Sciences*. 2021. Vol. 3, Issue 9. URL: https://www.actascientific.com/ ASCS/ pdf/ ASCS-03-0170.pdf.
46. Sapaty P.S. Spatial Grasp Model for Dynamic Distributed Systems. *Mathematical machines and systems*. 2021. N 3. P. 3–21. URL: http://www.immsp.kiev.ua/publications/articles/2021/2021_3/03_21_Sapaty.pdf.
47. Sapaty P.S. Development of Space-based Distributed Systems under Spatial Grasp Technology. *Mathematical machines and systems*. 2021. N 4. P. 3–14. URL: http://www.immsp.kiev.ua/publications/ articles/2021/2021_4/04_21_Sapaty.pdf.

Стаття надійшла до редакції 03.01.2022