

# Markers – toward general purpose information representation

Antti Tikanmäki, Juha Röning, University of Oulu, Finland

**Abstract—** In this paper, a general purpose information representation for robot’s software development is described. The fundamental idea is to have only one structure for information that is used software system-wide, to represent environment model, knowledge database, robot’s tasks, output of data processing, and other information related to the robot’s operation. Using only one data structure provides several advantages, including high reuse on software core parts, such as searching, comparison, interpolation, coordinate transformation and feature extraction.

## I. INTRODUCTION

Creating a software for the robot is a demanding task. How to connect all the parts, how to express past, present and future in a way that a robot can truly operate in the environment? How to define the task for the robot? How the information learnt could be used on another robot? And most of all, how to make it so that it can be continuously expanded with new information, new algorithms and new functionalities without making the system too complex to handle? All these questions can be answered, when we can define a general purpose unit for information.

Traditionally, research projects around the world have shown too many cases of “reinventing the wheel”. Especially on how data types of algorithms are defined, and in what form the data is delivered. As the main focus has been on creating algorithms, the representation of the information has been defined from mathematical point of view without thinking how it could be a part of a larger system. Without getting known on publically available libraries, parts are implemented from the scratch over-and-over again. A good example for this is how the position is implemented over and over again on each library. This reinvention usually causes focusing on primitive parts of the robot’s software, and in the worst case, will lead to the situation where nothing new is done within the project’s period, and progress of robotics in general will not proceed. Therefore, a standard way of doing things is needed. Choosing a right representation for knowledge (and everything else) is a key issue.

There are lots of examples of how things work in similar ways in different systems. Functions related to comparing positions, like distance, are similar to calculation of correlation in a different space, and similarity of a feature. Nodes and graph are used in many places for analyzing networks, relations, structure and kinematic models. Still, in practice, these are implemented separately on own libraries, and the code is not reused. In this work, these types of

similarities are detected, and a library is implemented to optimize as much as possible under the representation.

Fortunately, the problem has got increasing attention in several places lately. A new interest of how to represent information in a robotic system and how to combine information and algorithms has rose. Common awareness of the fact that “no single algorithm or method solves the problem” has spread among robotics researchers; instead, the robot’s software must be a dynamically adapting hybrid. Large projects like RoboEarth[1], Proteus[2], GeRT[3], Rosetta[4], BRICS[5], KnowRob[6] and many more, focus on how the knowledge should be defined in general and how push robotic research close to operation in real world. Similarly many robotic libraries like ROS, Orocos, Player, are focusing on combing the technologies together.

In this work, the issue has been studied purely from implementation point of view, and a more straightforward way by surveying basic structures from a large number of technologies, detecting their similarities and combining similarities to a building block for representation, and minimizing the size of the core.

## II. RELATED WORK

The main unit in this representation is called a Marker. A marker is a computational component responsible for storing and updating values related to a set of environmental features’ prior knowledge and task information. As the main purpose of the Marker is to act as a generalized structure; the number of technologies related to this work is large, and not reasonable to cover here.

The term marker itself is not new. In earlier work, at least Brill [7,8] used the term as a representation for task related objects in simulated robots. Later Rieki [9] used markers for a soccer robot environment model and Tikanmäki [10] expanded the representation to real robots and for integration of vision and sonar sensors output. This work has been expanded further through many experiments to the form of the marker described in this paper.

As the representation has always been part of the robot’s software, algorithms, planning, similar technologies with a different name exists quite a few. The same structures are repeated over and over again in simulators, 3D modeling tools, game physical engines, etc, where the unit is called, for example, “object”, “entity”, ”target”, “point”, or “nodes”.

In this work, the major difference to other systems is how various technologies are combined to this generalized representation, and how this is used on all parts of the robot’s software architecture. Through several use cases, markers have been used to create various applications with

maximizing code reuse and minimizing the amount of reimplementation.

### III. A MARKER

#### A. Key features

A marker consists of three parts: *properties*, *connections* links to define connections to other markers, and children providing a way to create *hierarchical* structures. Following shortly explains some aspects of these three main features.

##### 1) Properties,

Property consisting of name-value pair is one of the fundamental ways of describing a dynamically expandable structure of information. Properties are used in many systems for storing configuration parameters, attributes, and other information. A typical implementation of this is a hash table, where a key is used for calculating a hash index pointing to a value. For example, many parts on Python rely on a dictionary, where, for example, attributes and functions are stored on the dictionary and can dynamically be expanded during execution.

In markers, keys will be based on a commonly used term in physics and robotics, and will follow term standards in robotics. The units for values will follow SI base (or derived) units. Some keys are already defined, for example, "position" (including also orientation), "mass", or "probability" of existence. Properties are features extracted from the physical target, and may contain information like "alias" (what people are calling the object), or storing physical properties of the target like "material", "boundingbox", "shape", and "color".

It is very typical, for example, on machine vision systems, to extract N features of a segmented target and compare it to a learned set of "known targets". In markers, each feature is stored to its own property, and using marker comparison, the similarity of two markers can be compared.

##### 2) Connections

Graphs, a relational structure of nodes and links between them, are used in a number of places. Navigation graphs, search graphs, social relations, and communication networks just to name a few, are all based on graphs, and many algorithms for search strategies, optimizations, complexity estimation or loop detection exist, by means of which the structure of graphs can be analyzed. Further, several traditional representations can be expressed using graphs, for example grids using spatially fixed positions and links to neighbor markers, or topological representations using markers without actual position. In artificial intelligence, commonly used technologies like neural-networks, state-machines, Markov models, etc. can also be thought to consist of nodes and links. Also, in widely used vector-graphics (like SVG), a shape consists of nodes and edges providing many tools for manipulating these, like Bezier-curve interpolations, shape generations that robot's software may reuse in operation.

In this representation, nodes and links are both described using a marker. Each of link-marker may have several targets, and their properties are used to store information about the link itself. For example, a navigation graph contains node-markers of crossings and link-markers telling how to travel from node to node. On properties of link-marker, the properties like traversability of path, suitable driving speed, etc. can be stored. Further, a graph feature is used for describing kinematic models, networks structures, modeling the interaction, or describing information relations in a knowledge database. Again, the fundamental principle of "implement once, use everywhere" is followed.

##### 3) Hierarchy

Hierarchical relations of parent and children is also used in many systems; for coordinate relations, level of details, 3D models, directory structures and search trees, for example. One way to use this hierarchy is by defining coordinate frames, like a local coordinate system. Each frame is a marker, containing at least a position of origin in the parent frame. Figure 1 shows an example of frames used for an outdoor/indoor robot. The world coordinate frame is an Earth bounded frame, containing for example a GPS based metric frame (like UTM or ETRS89 [11] base) coordinates. The regional coordinate frame is, for example, city part, campus, or similar size of region that makes the navigation level of details more reasonable in size. On building level coordinate frame, there might be own initial position and origin that has a known global position and positions of all markers below this level are represented relatively to this. The Robot coordinate frame is tied to the robot's base (for example on the floor touching center of rotation) and all sensors of the robot are located on that coordinate frame. And finally, on the sensor coordinate system, the targets that the sensor and data processing can detect are located in the sensor coordinate system.

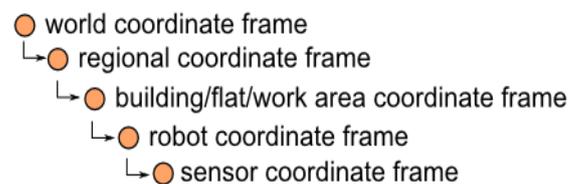


Fig. 1. An example of Hierarchical coordinate system

Hierarchical representation is used in many places for fastening the processing. By dividing the area in smaller regions, computer games optimize their world model. In computer hard drive, we organize files to directories and categories instead of putting all the files in one directory. Similarly a robot's environment model can optimize its performance by reorganizing markers on a hierarchy for faster operation.

#### 4) Implementation

In practice, a marker is implemented with a class that inherits a hash table, like a map in C++, Properties-class in Java or dictionary in Python. Further, the following methods for processing marker and its ancestors are implemented as class methods.

##### B. Methods for manipulating markers

This section will shortly explain some key features of how the marker structure can be used, and what methods the marker has.

#### 1) Coordinate system transformation

Hierarchical coordinate system frames require a transformation between parent and children coordinate systems, and therefore, one essential method for Markers is the way coordinates are transformed to other coordinate system. For 3D markers and dynamic models, 6D spatial vectors called Plücker basis vectors [12] are used, which combines translations and rotations to one 1x6 matrix. If fewer dimensions is used or known, like no rotational information, fields will have zero value.

#### 2) Level of details

Markers provide a way to easily create level-of-details structure. Each marker represents a certain level of details on environment and children contain more detailed information of the target. For example, in an outdoor navigation map, a marker represents a house with a simple square from house corners as a shape and a center point as a location. On children levels, details like interiors, rooms, furniture, etc. are stored. Depending on the task and position of the robot, it can select the level that is used on navigation. Similarly, a representation for a human is defined using a hierarchical way, and depending on the robot's current task, the robot uses the level of details from a bare location of the human to detailed poses of fingers and facial expressions. The major advantage of using hierarchical representation for the level of details is that it saves processing power.

#### 3) Flattening

Using Hierarchy flattening, the tree can be converted to a flat list of markers on root coordinate system. As each child's coordinate are expressed in the parent's coordinate system, a flattening is a way to get a list of leaves in a certain level coordinate system.

#### 4) Combining markers to structure

Several markers can be combined and stored as children of a new marker by grouping them together. A typical example of this is by detecting targets from camera image or laser scan points. When algorithm for example detects the similarities of shade, color, and distances of each point less than a threshold, a new marker is created to represent this connected object. Each single measurement is stored as child of this marker.

#### 5) Property comparison

The marker's properties can be compared to other markers. Comparison searches through properties, and using a key-tied callback function, a defined distance between each property can be determined. Callback is used for defining how a certain property is compared. Additionally, the method has a parameter for giving a list of keys that are used for comparison, as well as weights for each property (or feature) to be used for calculating a final similarity output value. By using comparison, sensed markers can be compared to previous sensing on each step of the control cycle (for tracking) or compared to a knowledge database template to recognize the target and find more details to a sensed object. The same comparison can be used for template matching, defining the difference, error from reference, derivate in time, progress, to name some.

#### 6) Marker serialization.

The software of the robot is rarely a single program that runs in a single computer. To be able to transfer information between processes and machines, a formal representation is needed. In this architecture, only a list of markers are used. A number of ways to serialize data exist, like XML, S-expression, JSON, just to name few. In this work, S-expression is selected, mainly because it is more compact than XML, simple to parse, and literal (ascii or unicode) use as it is readable by a human and easier to transmit using most of the protocols. S-expression is also used in a mature AI language, probably most widely known in Lisp, and has shown its flexibility as expression. Relations in graph are represented using a reference to markers' ID-property.

Additionally, JSON format is used on web based visualization as it is widely supported by browser based visualization libraries like Protovis. In addition to using serialization for communication between processes, it is used to store and restore representation on file(s) to be used later.

#### 7) Marker generators

One of the key functionality on the system is how markers are generated. Generators are used for creating new markers based on simple rules. For example, a trajectory curve for the robot's movement is defined with control points (represented using markers). The generator interpolates and creates new markers along the path. Another example is a generator that creates markers from operating area definitions, and creates a grid of markers (places) where the robot should visit while executing the task. Later the navigation system gets this list of places and while visiting the places during execution, removes markers from the list as the place is reached. One analogy for generators is particle systems in computer graphics and games, where the generator is used for particle effects like fire, liquids, or smoke.

#### 8) Network interface

One unique feature of architecture is the possibility to start a network interface on any marker on the systems. Using a

selected protocol (like for example http), the properties of the marker can be read and adjusted through a network interface using *SET* and *GET* commands. For example, a marker representing the robot in marker hierarchy is started as service and properties, like position, can be requested by clients. On request, the properties are serialized and served using services protocol.

The network interface provides also functional properties. The most simple one is named with "\*" character. This property will provide the current *state*, complete content dump of the marker in serialized format. Similarly, more complex queries can be made by using template-markers as a parameter. For example, querying all markers from a certain area, a template marker with position and size matching the area is included on request. Knowledge storages and environment models are created by starting the root marker as a service. For limiting the write access to certain properties, a list of restricted access can be defined. The query request starts with "?" character followed by the string containing parameters.

### C. Usage in software architecture

Markers create a base for all communication between system (Figure 2) parts, using serialization parts can be located on several processes and even several computers on the robot's network. Data processing parts can be run parallel and their parameters can be adjusted during operation. In this section, markers' role is further explained.

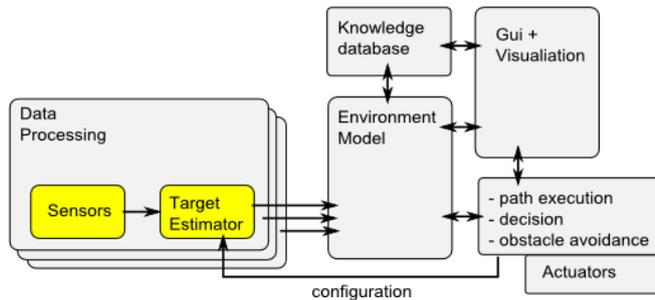


Fig. 2. An overall view of software architecture. Communication between parts is done using marker representation

This section will shortly introduce some of the parts in the robot's software that can be represented using markers.

#### 1) Sensor data processing

To minimize transmission of raw sensor data, the processing is located right next to the software part that communicates with the physical device (like reading the serial port). Therefore, each sensor data processing algorithms gives out the information using a marker and properties. The data fusion part of the software will later combine these markers to represent markers with more properties. For example, machine vision can detect the color of the object and shape, while a laser scanner can provide a more accurate position for the target. The fusion combines these two markers to one marker that has color, shape, and position properties. Further, if algorithm can detect the material of object, it can estimate the mass or other

properties for the target. This information can further be used for task planning, environment state progress prediction, etc.

#### 2) Environment model

An environment model forms a base for the robot's task planning, navigation, obstacle avoidance, and for example human interaction. The model may also contain kinematic relations (like forces and masses), and it can be used for predicting the upcoming state of the model. Further, in technologies, like particle filters, several additional predictions may exist, which are represented by using several parallel models and choosing the most likely. In practice, the environment model is a marker with all the content stored on children. It may also have one configuration related properties, and links to other environment models on the system. One essential part of the environment model is to provide an integrated way of doing simultaneous localization and mapping, SLAM and integrate algorithms like graph slam [13] on it.

#### 3) Knowledge storage

Knowledge storage is a database with a query interface. Information is stored in several groups of markers, representing the spatial knowledge of environment and prior information of objects in operating environment. Using a query interface, markers from certain spatial area boundaries, or markers with certain properties can be requested, and a returned message is a list of markers matching the query. For example, the robot may request all printers on a laboratory floor, or a navigation graph (with several additional routes) for a certain area or a target position (expressed with a marker).

In addition to the spatial model, the knowledge database contains "template"-markers for objects that the robot can already detect. These templates contain properties for features how to detect target, meanings of object, names for objects used by humans, material, mass, or similar properties. By comparing features of detection, software can add additional properties to a marker that represents the object. Knowledge storage can be build from commonly available information, later in the use case section it will be shown in practice how Open Street Map information is converted to a knowledge base for an outdoor robot.

In practice, knowledge storage is a marker started with a network interface. This will be explained in more detail in the next section.

#### 4) Task definition.

The robots' task can be expressed using structures like state machines, parallel scripts, etc. By using markers, a state machine with sub states (children of the marker) can be used for constructing a task description for the robot. This state machine may have state or spatially related properties proving a hybrid approach to the task description.

The environment model can also be expanded by defining the use of virtual markers, sort of "hallucinated walls" or "virtual fences", by means of which the operation area of the

robot can be limited. The path or navigation graph with additional routes is expressed using markers. Task definition is part of the environment model. In multi robot systems, this can also be shared among several robots, which provides an easy way for robot group control and task definition and sharing.

### 5) Visualization

One important part for verifying system operation is to see how well the system is working. For a complex and large amount of information, a visual view is needed to see how the software is performing. Using level-of-details and properties related to the visual appearance are used for drawing the markers on the user interface, for example, color, shape, and obviously position.

## IV. USE CASES

In this chapters, several examples of robot related topics are studied from marker-perspective. Following gives example cases where this representation has been used in the robot's software architecture. Due lack of space, it will not cover completely all aspects of robot's software, but gives an idea of markers flexibility in each case.

### A. Representing a human model and robot's structures

In Human-robot interaction, the model of human may vary from simple locating of the human to a detailed representation of gesture, pose, and emotional state. Depending on the current task of the robot, level-of-detail on interest varies a lot. If the task is to drive next to a human, only location is relevant, while trying to understand the hand marks, the movements of hands and fingers are focused. The hierarchical representation gives a way to store all detected information of a human on one structure, and use the required level of detail on a certain task (if it is available). Figure 3 shows an example of how human parts are organized on hierarchical representation.

By fitting the kinematic model of human to measured poses, the current movement can be used for predicting near future possible poses of the human. Similar technology is used on Ragdoll-physics[14] in computer game engines.

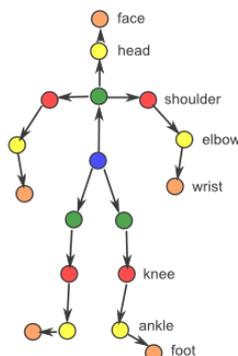


Fig. 3. An example of human coordinate hierarchy, each color represent a certain level of on marker hierarchy

A similar hierarchical model is used for the robot, including

the sensors on child nodes on hierarchy. The same structure of joints is used for legged robots, robot manipulators, or more complex robots. Similarly by using a kinematic model of the robot, possible causes of actions for certain control can be predicted by stepping the current state forward. This online simulation for certain parts is then used for finding the best possible action for the robot's actuators.

### B. Obstacle avoidance

Using a local environment model of obstacles along the robot's path, an obstacle avoidance path can be created. One way of doing it is by calculating the sum of forces defined by the attractive force toward a goal and repulsive forces from each obstacle. The sum of these forces shows the suitable movement. This can be expanded further by calculating several variations of weighting the forces and choosing the fastest one.

### C. Sensing and measuring wireless LAN coverage

In this use case, a mobile robot or hand-held device is continuously scanning available wireless LAN access points and stores measurements to a marker-based structure. The structure is continuously updated, and further used for robot localization. On the base level, a list of markers is created, and each marker represents an access point. If a new unique ESSID is detected on scanning, a new marker is added to the list. Each measurement of the strength of network is stored on the corresponding marker's children, and location of measurement is rounded according the grid size. Figure 4. shows a simplified example of the sensed structure. A and B represent base level markers for estimation of two access points with unique color, and each other circles in the figure represents a measurement with opacity related to signal strength. Markers' A and B locations are calculated based on each measurement after measurement cycle.

Similarly, any other kind of measurement maps can be created, for example temperature map, where each marker stores the spatial temperature that the robot has measured.

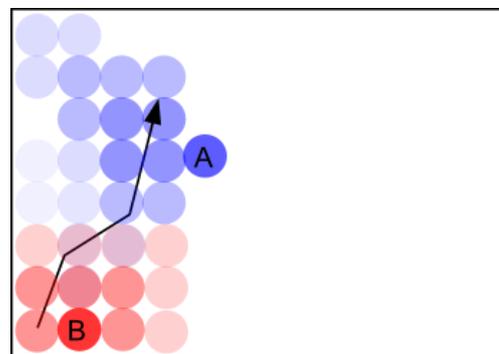


Fig. 4. Visualization of Wi-Fi signal strength sensing

### D. Human-robot interaction

Human-robot interactions have become one major topic of research and development along with the term service robotics. The main requirement for operating among humans is to be able to integrate the information from various

sources, and quickly react on changing situations. Any algorithm has limitations and works only on limited features on environment and therefore, the robot must have several sensors, several parallel running algorithms and a way to adapt algorithm parameters to a current situation on the environment. For example, one edge detection algorithms works better on a certain view, while other algorithm or threshold works better on other view. Running these parallel and integrating the outputs to one richer model is required. Figure 5 shows an example of visualization output of the robot's vision system integrating object detection using two algorithms. The system uses a simple knowledge database for storing features for classification using template markers.

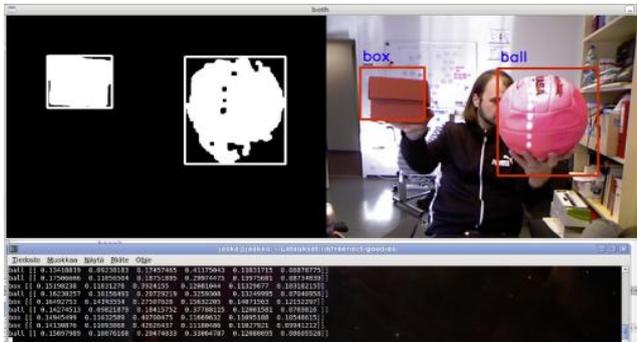


Fig 5. Visualization of vision system detecting objects

The integration of various machine vision algorithms has also done using markers. In an experiment, a Microsoft Kinect was used as the main sensor and the camera's depth information was used for detecting the location of humans and other target objects. OpenNI provided a human pose which was converted to structural representation. By running in parallel a detection of human, its pose, finding a face, detecting the person, facial expression, and gesture movements, a combined representation has build. As each method can detect, the properties are combined on the marker representing the human.

### E. Swarm of robots

In this use case, markers are used for representing a group of robots and their coordinated operation. Each robot is represented with a marker, and the base marker represents the whole group. Using the base marker and adjusting its properties, for example, the target for movement, the whole group can be controlled. The group can be reorganized by changing the location in a marker tree. Internally, the group operation includes functionalities like task distribution, or coordinated movement. Externally, the groups operation is controlled by adjusting the properties of the group marker.

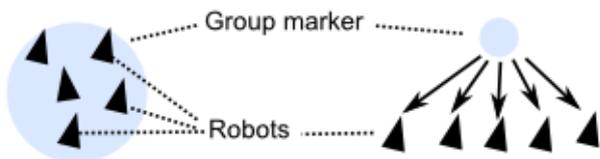


Fig. 6. Left - spatial location of markers, Right - Hierarchical representation of robot swarm

In addition to grouping robots to a tree, the swarm movement can be organized using markers' links, so that each robot is linked to neighbors and following their movements.

### F. Creating a navigation model

In this use case, the prior information for outdoor robot navigation is calculated from two sources. The base is formed from Open Street Map repository, by requesting xml representation of nodes on the operating area of the robot. Graph representation of markers is created and information of road type, and targets like bus stop, are stored on properties of the markers. The robot uses this mode for calculating several additional routes from the current position to the target position, and update sensed details, like a visual estimation of traversability of soil to the model.

## V. CONCLUSION

By detecting the primitive similarities of mostly used technologies, a rather simple, but powerful representation has been created for representing the information in the robot's software architecture. Using the same structure system-wide, algorithms are faster to implement and the robot's software can truly step on the next level.

## VI. FURTHER WORK

The initial structure for markers has been defined, and further work will include a standardization of keys used for certain information. As the core of the system is available, the major focus will be on creating more practical examples and real world applications for robots using this representation.

## REFERENCES

- [1] RoboEarth project web page: <http://www.roboearth.org>
- [2] Proteus project web page: <http://www.anr-proteus.fr/>
- [3] GeRT project web page: <http://www.gert-project.eu/>
- [4] Rosetta project web page: <http://www.fp7rosetta.org/>
- [5] BRICS project web page: <http://www.best-of-robotics.org/>
- [6] KnowROB project web page: <http://ias.cs.tum.edu/research-areas/knowledge-processing/knowledge4cotesys>
- [7] F. Brill , W. Martin , T. Olson, "Markers Elucidated and Applied in Local 3-Space," In Proceedings of the 1995 IEEE International Symposium on Computer Vision, 1995
- [8] F. Brill "Representation of Local Space in Perception/ Action Systems: Behaving Appropriately in Difficult Situations", University of Virginia Charlottesville, VA, USA
- [9] J. Riecki "Reactive Task Execution of a Mobile Robot." Doctoral Thesis, Infotech Oulu and Department of Electrical Engineering, University of Oulu, 1999
- [10] A. Tikanmäki "Mobile robot's environment modeling and visualization." Department of Electrical Engineering, University of Oulu, Oulu, Finland. Diploma thesis, 2002
- [11] European standard for spatial information web page: <http://etrs89.ensg.ign.fr/>
- [12] R. Featherstone "Plücker Basis Vectors", IEEE Int. Conf. Robotics & Automation, Orlando, Florida, May 15-19, pp. 1892-1897, 2006
- [13] E. Olson and J. Leonard and Seth Teller, "Fast Iterative Optimization of Pose Graphs with Poor Initial Estimates", ICRA2006, pp. 2262-2269.
- [14] W. Armstrong and M. Green. *Visual Computer*, "The dynamics of articulated rigid bodies for purposes of animation", pages 231-240. Springer-Verlag, 1985.