5
A knowledge-based framework for multitemporal image interpretation

One of the specific objectives of this research is the development of a knowledge-based framework for the interpretation of remote sensing multitemporal data produced by a single or by different sensors, having potentially distinct spectral, spatial and radiometric characteristics.

As the InterIMAGE framework (Costa et al., 2008), already provides for the integration of images from different sensors in a single interpretation problem, the research focused on the design and implementation of a multitemporal extension of InterIMAGE, incorporating the necessary mechanisms to represent and process explicit temporal knowledge.

InterIMAGE is an open-source, multi-platform framework, written in C++, counting currently with implementations for LINUX and Windows operational systems. As it will be made clear in the next sections, InterIMAGE provides support for the integration of image processing operators in the interpretation process and, as such operators are treated as external programs by its control mechanism, they can be coded in any computer language, and can even be proprietary programs. The InterIMAGE framework offers, nonetheless, a repository of operators (http://www.dpi.inpe.br/terraaida), assembled with the software classes and functions supplied by the TerraLib library (Câmara et al., 2000). InterIMAGE's graphical user interface is also based on TerraLib classes.

InterIMAGE is founded on GeoAIDA (Bükner et al., 2001), developed in the TNT Institute (Institut für Informationsverarbeitung) at the Leibniz Hannover University, Germany, and it inherited from that system its basic functional design, knowledge structures and control mechanisms, and that's why the discussion that follows will be anchored in the description of GeoAIDA.

A number of knowledge-based image interpretation systems have been previously proposed, and the most relevant examples were mentioned in Chapter 3. Only one of those provided for explicit temporal knowledge representation and

processing – the AIDA system (Liedtke et al., 1997) – also developed at TNT. AIDA has been superseded by GeoAIDA, which fits better into the definition of a framework, in computer science terms.

By definition, a *framework* is a set of code that provides functionality common to a specific class of applications and the means to seamlessly extend its behavior – a framework fundamentally helps solving recurring problems.. A *framework is concrete*, meaning that it has physical components, e.g. data repositories or executable files; a *framework is incomplete*, in the sense that it is not usable on its own; and a *framework drives solution*, as it dictates overall architecture of complete, specific solutions (Gachet, 2003).

GeoAIDA's core basically provides functionality for the definition of semantic network-based knowledge models (Section 2.1.2) and for the processing of such models though a particular interpretation control strategy (Pahl, 2003). Throughout the interpretation process, GeoAIDA calls external programs, which represent *implicit knowledge* (see Section 2.1.1), and which are responsible for processing the set of remote sensing images or GIS data involved in the interpretation problem. Upon calling an external image processing program, GeoAIDA prepares and passes on to the program only the image subset(s) necessary at that particular point in the interpretation process.

While in AIDA, image data processing was implemented by code written in the *TCL* script language, in GeoAIDA image processing can be performed by more efficient compiled programs, written in any programming language, including commercial software packages whose fonts are not available. GeoAIDA provides, therefore, the means for integration of different image classification or object extraction techniques to solve a particular interpretation problem.

GeoAIDA also provides the basic functionality for the evaluation of the image processing programs' results, but such process can, as well, be carried out outside the system's core (i.e. by external programs), which will in turn deal only with the evaluation results. In AIDA this evaluation was performed internally, so that the introduction of particular relations between nodes of a semantic network, e.g. "adjacent to", demanded reprogramming the system. In GeoAIDA, as the evaluation task is implemented externally, by programs that process object hypotheses descriptions passed on by the core, potentially any possible spatial

relationship among the those hypotheses can take part in their judgment, without the need of reprogramming the core.

In terms of temporal knowledge processing AIDA implemented a specific, sequential strategy (see Section 5.2.2). GeoAIDA, on the other hand, in spite of all its flexibility provides no support for explicit temporal knowledge processing and representation. Therefore, what is envisaged in this work is the extension of GeoAIDA's general problem solving strategy so as to enable the design and processing of knowledge models for multitemporal interpretation.

The general idea is to introduce a *temporal dimension* to the interpretation process, besides what can be called the *structural dimension*, already taken care of. In Figure 30 the individual semantic networks represent the structural dimension – for each date interpretation is guided by the semantic description of the interpretation problem at that particular date, taking into consideration the available data for that point in time. The temporal dimension is transversal to the structural one, so that at some particular points in the local (temporally speaking) interpretation process, *temporal operators* would be able to create or evaluate object hypotheses considering the hypotheses generated at the structural dimension at distinct points in time.

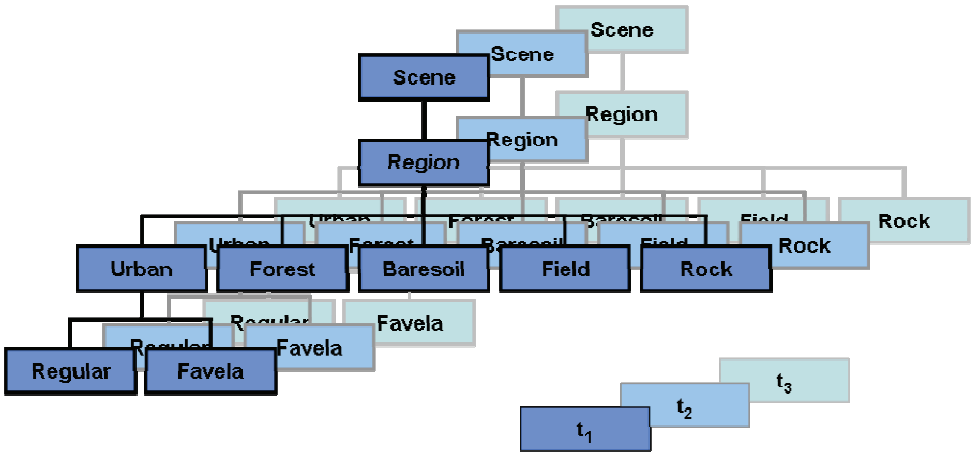


Figure 30. Structural and temporal dimensions of a multitemporal interpretation problem.

In what is proposed here, the semantic descriptions of the individual times do not need to be the same, and the temporal operators can be also implemented by external programs. Generic mechanisms for the evaluation of hypotheses from different times are also provided.

In the next sections some aspects of GeoAIDA's architecture will be reviewed, a detailed description of the system's architecture and implementation can be found in (Pahl, 2003, 2008). In what follows, only the aspects necessary for the understanding of the interpretation process already implemented, which will be subjected to the extensions proposed in Section 5.2, are presented.

5.1. GeoAIDA overview

In Figure 31 the components of the interpretation process in GeoAIDA are depicted. The system implements a specific interpretation control strategy that is guided by a knowledge model structured into a semantic network. *Interpretation control* is performed by system's core, which takes as input a set of image data, GIS layers, digital elevation data, or other types of georegistered data.

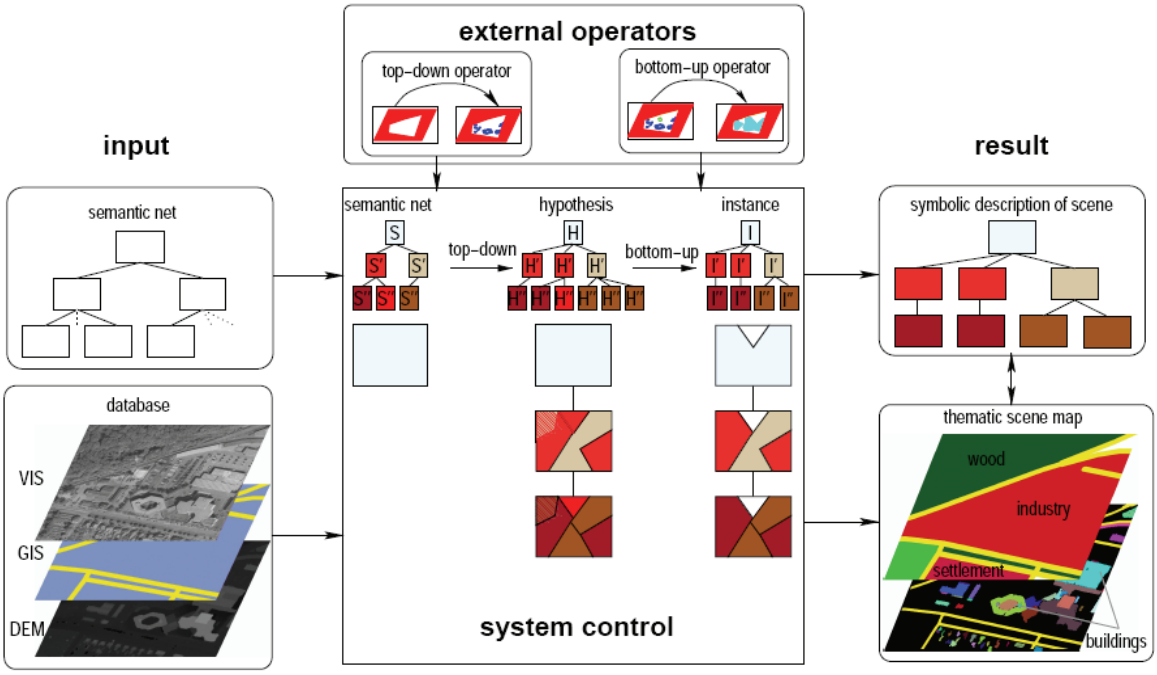


Figure 31. Components of the analysis process. Adapted from (Pahl, 2003).

Throughout scene interpretation the input data are processed with the aid of external programs called *top-down* and *bottom-up* operators. *Top-down operators* are responsible for partitioning the scene into sub-regions regarded as *object hypotheses*. *Bottom-up operators* evaluate the hypotheses, discarding or

validating them, and resolving eventual spatial conflicts among them. At the end of the interpretation process the validated hypotheses become *object instances*.

The output of the interpretation process is a symbolic description of the scene, consisting of an *instance network* and the *label images* that correspond to the regions associated to the object instances. From those label images the system is able to create different *thematic maps* representing the different levels of concepts in the semantic network.

5.1.1. Knowledge representation

A knowledge model in GeoAIDA contains problem specific information used by the control process for the interpretation of a scene. It is represented through a semantic network (Figure 32) wherein the organization of the nodes is hierarchical, each node can be associated to only one superior (father) node and to one or mode subordinate (child) nodes.

Each node in the semantic network[11] corresponds to a class of objects expected to be found in the scene. The nodes have attributes, such as the associated top-down and bottom-up operators, as well as generic and operator specific parameters.

The links of the network have no explicit semantic meaning, representing either *is-a* or *part-of* relationships, depending on the operations performed by the bottom-up operator attached to the higher level node connected to the link.

The objects found in the scene by the top-down operators are initially regarded as *object hypotheses*. It is the task of a bottom-up operator to evaluate the hypotheses associated to the children of the node it is attached to, creating *object instances* from the hypotheses validated by the operator.

As they have no children, it makes no sense to attach bottom-up operators to the leaf nodes of the semantic network. On the other hand, as it will be clarified in the next section, there must be a *holistic top-down operator* attached to each leaf node.

[11] The term *semantic network* will be used hereafter to denote a *conceptual network*, see Section 2.1.2.

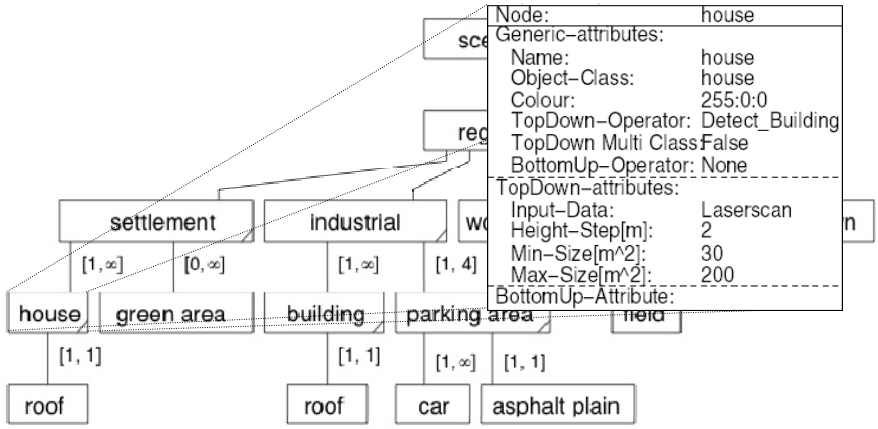


Figure 32. Attributes of a node of the semantic network in GeoAIDA.

5.1.2. Top-down operators

The basic task of a top-down operator is to generate hypotheses of objects for the scene under analysis. As those hypotheses are defined over geographic regions, this task is about determining regions in the scene where objects of a certain class may occur.

Top-down operators are executable programs, called by the system's core during the interpretation process. They can in principle process not only images, but also any type of georegistered information, including vector data from a GIS database, digital elevation models or other types of raster data.

When the core calls a top-down operator, it passes on to the operator information about the limits of the geographical region to be processed. Note that this region of interest (ROI) is defined by another top-down operator attached to an ancestor node. Information about the ROI consists of its bounding box and a binary raster mask identifying precisely the region limits. Furthermore, the core will pass to the operator just a subset (covering only the ROI's bounding boxes) of the image data in the database it has been specified to process.

A *holistic top-down operator* implements a specialized method for the detection of a particular object class, over input data with specific characteristics. Holistic operators embody implicit knowledge about a particular concept (object class) represented by a node of the semantic network. The output of such operators consists of a list of regions and a corresponding label image in which the exact extent of the regions are defined. Recall that those regions are regarded

as object hypotheses in the interpretation process. Additionally, these operators can assign confidence values to the regions and state those values in the output region list, those confidence values can be later used for the evaluation of the corresponding hypotheses (by a bottom-up operator).

Certain nodes of the semantic network can be associated to object classes that can only be detected through their structural components, i.e., objects of those classes can only be defined through their component objects. A *dummy top-down operator* should be attached to such nodes. The dummy operator will output a single region that is equal to the ROI defined at its father node.

As mentioned in the previous section, holistic top-down operators must be attached to the leaf nodes of the semantic network, as those nodes correspond to primitive object classes.

A special type of top-down operator is a multi-class operator. Those are holistic operators capable of detecting objects of different classes. A multi-class top-down operator should be attached to nodes in the same hierarchical level, to be precise: they should be attached to two or more child nodes of the same parent node.

5.1.3. Bottom-up operators

The evaluation of object hypotheses is the task of the bottom-up operators. Bottom-up operators are also executable programs, called by the system's core during the interpretation process. The input of such operators is a list of regions, each region associated to one object hypothesis.

A bottom-up operator evaluates hypotheses associated to the child nodes of the semantic network node the operator is attached to. Prior to the execution of the operator, the system's core creates a list containing the descriptions of all object hypotheses associated to the child nodes. When the operator is called, this list is passed on as an input parameter to the respective executable program. Upon execution, the bottom-up operator judges the input hypotheses, discarding or validating them. This judgment can be made by rules hard coded into the operator or explicitly defined as attributes of the semantic network node.

Properties of the object hypotheses, such as confidence values or feature values, can be used in the judgment process. The validated hypotheses will from then on be regarded as *object instances*.

It is important to note that the object instances generated by the operator, may, at some later point in the interpretation process, be discarded. That will happen if a higher level object hypothesis, to which the instances are related is rejected by another bottom-up operator.

The object instances generated by the bottom-up operator must be spatially disjoined, so it is also a task of such operators to resolve eventual spatial conflicts among their original regions, reshaping them in order to remove region overlaps.

The operator should also group the object instances, associating to each group a region that is equivalent to the union of its components' regions. The groups will originate new object hypotheses for the semantic network node at which operator was called, substituting the original hypothesis, as it will be clarified in the next section.

The output of a bottom-up operator consists of a description of the new object hypotheses and object instances, in addition to two label images, which describe the spatial extent of the regions that correspond respectively to the new hypotheses (associated to the node where the operator was called), and to the instances (associated to the children nodes).

5.1.4. Interpretation control

The fundamental task of the system's core is the control of the interpretation process, which consists of two complementary steps: the top-down and the bottom-up steps.

In the top-down step the control process traverses the nodes of the semantic network, from the root to the leaf nodes, calling the top-down operators attached to each node. Top-down processing will occur in parallel with respect to the semantic network branches. Object hypotheses associated to the network nodes are created during this process and organized into a *hypothesis network*.

As processing reaches the leaf nodes, the bottom-up step initiates. The control process starts visiting nodes in the opposite direction, calling the

respective bottom-up operators recursively until the root node has been reached and an *instance network* has been created. In this process the object hypothesis are discarded or turned into object instances.

A more formal description of the interpretation process, derived from (Pahl, 2003, 2008), is given below.

If N is a generic node, N' is a subordinate (child) node with respect to N, N'' is a child of N', and so on. T_N denotes a hierarchically structured network, i.e., a tree, and N'_m e N'_n stand for different nodes in a same hierarchical level of T_N.

Let S be a node of the semantic network T_S, and let H and I be nodes of the networks T_H and T_I, created by the interpretation process. H represents a node associated to an object hypothesis and I represents a node associated to an object instance.

Let R represent a region in the scene to which the H or I nodes are associated. R' is, therefore, associated to either H' or I'. Considering the hypothesis network T_H, all R' associated to nodes H' are subsets of R; considering the instance network T_I, all pairs R'_m e R'_n associated to nodes I'_m e I'_n are disjoint.

The objective of the interpretation process is to create an instance network T_I and a corresponding network of regions T_R, by applying the knowledge represented through the semantic network T_S in the interpretation of region R. Initially a network T_H of hypotheses is created, gradually the hypothesis nodes H are substituted by instance nodes I, so that at the end of the process the network T_I is completed.

Top-down processing is depicted on the top of Figure 33. It starts from the point where the hypothesis node H, associated to the region R and to the semantic network node S, has already been already created. From this point on, control passes recursively to the S' nodes.

Hypothesis nodes H'_{mi} are generated through the execution of a holistic top-down operator attached to the semantic node S'_m over the region R. The H'_{mi} nodes are associated to the partial regions R'_{mi}. If there is no holistic operator attached to S'_m, a single hypothesis node, H'_{m1}, will be associated to the whole region defined for its superior (father) node H. R'_{m1} is, in this case, is equal to R. If S'_m is a leaf node (has no child nodes), a holistic top-down operator must be attached to it.

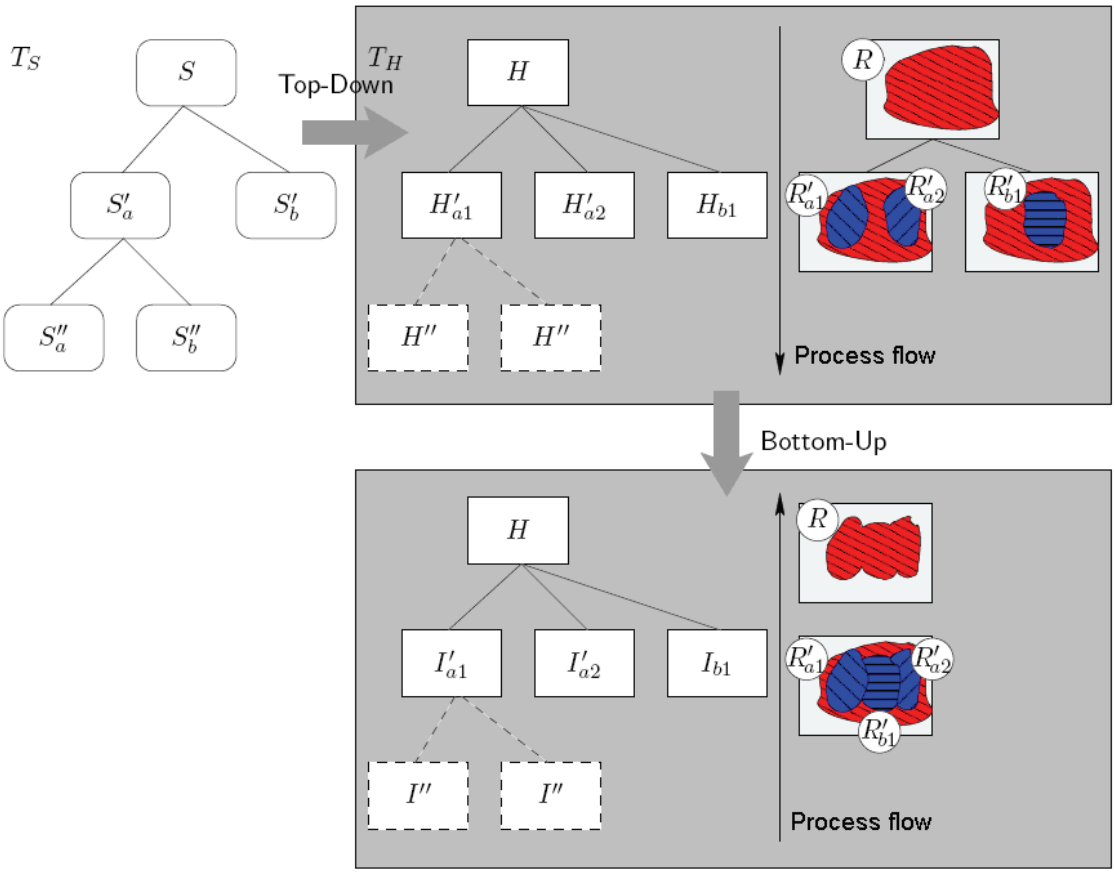


Figure 33. Interpretation process flow. Adapted from (Pahl, 2003).

If S'_m has subordinate (child) nodes, the procedure described in the previous paragraph will be repeated for each H'_{mi} until the semantic network leaf nodes are reached. At that point bottom-up processing starts (bottom part of Figure 33), beginning at the parents of the leaf nodes. The bottom-up operator attached to S'_m will be executed for H'_m once all the leaf hypothesis nodes H''_{ni} have been created – the n index identifies the direct descendants of H'_m. The operator will then evaluate the H''_{ni} hypothesis nodes and decide if they will be turned into instance nodes I''_n or deleted from the network. Additionally, the bottom-up operator will make regions R''_{ni} associated to the I''_{ni} nodes disjoined.

The bottom-up operator will in turn group the nodes I''_n and generate new hypothesis nodes H'_g to which the instance nodes will be linked. The H'_g will then be placed in the hypothesis network and linked to the node H, and the original node H'_m deleted. Control is then passed to node H, for the evaluation and grouping of the H' nodes. Bottom-up processing proceeds until the root node of

the hypothesis network is processed. At this point the instance network T_I is completed.

5.1.5. Implementation of the control mechanism

In GeoAIDA the nodes of the semantic network T_S are implemented by a class[12] named `SNode` and the nodes of the hypothesis, T_H, and instance, T_I, networks are both implemented by a class named `INode`. The networks T_H and T_I are thus considered in the implementation as a single network, and the `INode` objects[12] can either refer to nodes of types *H* or *I*. `SNode` and `INode` are both specializations of an ancestor class called `GNode`. Additionally, each `INode` object is related to a `SNode` object.

Interpretation control is a recursive process, implemented through a finite state machine (FSM) associated to the `INode` class. As all actions in the FSM are independent of external events, such as the execution of an operator, or the generation of lower level `INode` instances, interpretation occurs with a high degree of parallelism, with few synchronization points.

The basic FSM states are: `TD_START`, `TD_EVALUATE`, `START_CHILDREN`, `BU_START` e `BU_EVALUATE`. Usually the `INode` objects go through those states in that order. There are also error states that will not be mentioned.

Initially an `INode`[13] related to the highest level `SNode`[13] (the *Scene* node of the semantic network) is created, the state of this node is set as `START_CHILDREN`. In this state one child `INode` will be created for each child of the related `SNode`. The state of these new `INode` objects will be set as `TD_START` and, after its children have been created, the state of the father `INode` is set to `BU_START`. A counter associated to father `INode` will be initialized with the number of children, and decremented every time a child node reaches its final state (after `BU_EVALUATE`), only then the actions associated to the state `BU_START` are executed for the father `INode`.

[12] In this section the terms *class* and *instance* refer to programming language constructs, in the context of object-oriented programming.

[13] From this point on, the terms `INode` and `SNode`, when not followed by the word *class*, will refer to `INode` and `SNode` instances.

At the TD_START state the top-down operator attached to the corresponding SNode is called. The actual execution of the operator is commanded by the *task manager*, a scheduler that controls the number of processes being executed simultaneously. After the execution of the operator, the state of the INode is set to TD_EVALUATE.

When an INode enters the TD_EVALUATE state, the output of the top-down operator is processed. For each hypothesis found by the operator a new INode will be created, with its state set to START_CHILDREN, and linked to the father of the original INode. The original INode will be deleted, and the children counter of the father INode will be decremented. If the SNode related to the original INode (and to the newly created INode objects) has children, the children counter of the father INode will be incremented by the number of INode objects created, and the actions defined for the START_CHILDREN state will be executed for each new INode. On the other hand, if the SNode has no children, no FSM actions will be executed for the newly created INode objects, and the BU_START actions for the father INode will be executed.

The BU_START actions have to do with the calling of the bottom-up operator attached to the corresponding SNode. The execution of the operator is also commanded by the task manager. After the execution of the operator, the state of the INode is set to BU_EVALUATE.

In the BU_EVALUATE state, the output of the bottom-up operator is processed. New INodes, which represent the grouping of the child INode objects, are created, linked to the father of the original INode, and the original INode deleted. The children counter of the father INode is then decremented, eventually leading to the start of the BU_START actions for that node (when all its children have been processed). When the highest level INode reaches the BU_EVALUATE state, the interpretation is completed.

The pseudo code of the above mentioned FSM states is shown in Appendix A. Actually, the code in the appendix incorporates the modifications made in the control mechanism of GeoAIDA/InterIMAGE the lines in bold letters represent the implemented changes.

5.2. Multitemporal extension of InterIMAGE

In the following sections a description of the proposed multitemporal extension of the InterIMAGE framework will be presented. In short, the idea is to enhance the knowledge representation capacity of the system and to alter the interpretation control process, introducing temporal processing steps in addition to the existing top-down and bottom-up steps.

Two different *multitemporal interpretation strategies* have been considered in the multitemporal extension of the control process. In the first strategy, which can be related to the general mutitemporal model proposed in Chapter 3, the classification of image objects in all dates is performed simultaneously. In the second strategy, initial hypotheses of image objects at a particular date are created based on the interpretation of the prior date, which can be either the true classification or the outcome of automatic interpretation.

5.2.1. Temporal knowledge representation

A knowledge model in InterIMAGE is represented through a semantic network, whereat the nodes represent concepts and the links have no explicit semantic meaning, as described in Section 5.1.1. The main issue tackled in this section is how to extend the current knowledge representation scheme, making it capable to represent temporal relations and the semantic concepts involved in those relations.

Initially let's consider the fact that there might be data (images or GIS layers), which would aid creation or judgment of hypotheses, available for a particular date, say $t+1$, and not available for another date t.

Going back to the land cover interpretation problem, let's assume that there are medium resolution images for times t and $t+1$, and holistic (monotemporal) operators specialized in classifying land cover objects into the classes of concern (for example, forest, pasture, savannah and urban). Let's also assume that there is a high resolution image available for time t, through which a specific class of vegetation that only occurs inside savannah areas can be identified. Furthermore, the identification of that type of vegetation, which will be

called “savannah tree”, could be done by a hypothetical holistic operator specialized in finding savannah trees in that specific type of high resolution image.

Considering only monotemporal interpretations, one could think of two different semantic networks, one for time *t* and another for time *t+1*, as shown in Figure 34.

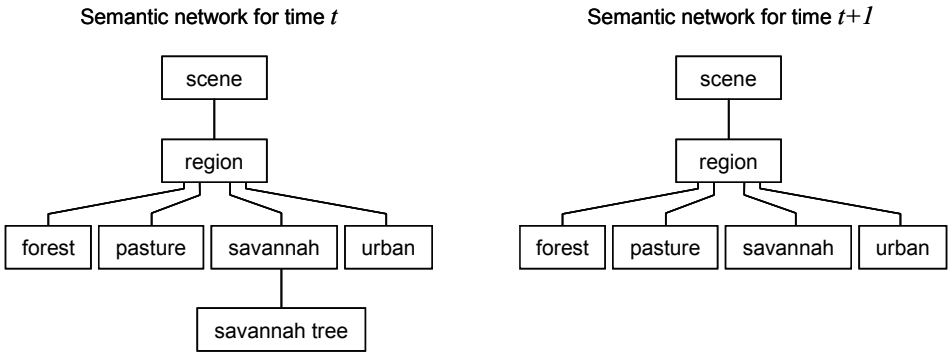


Figure 34. Different semantic networks for different points in time.

Considering now the multitemporal aspect of the interpretation problem, one could think of temporal connections between the two networks, representing the temporal relations among the concepts of interest.

An alternative representation for the knowledge model could be based on a single semantic network, in which some nodes would be explicitly associated to a specific point in time, as shown in Figure 35. The nodes inside the dashed outline box represent temporal connections between monotemporal semantic models, and will be called *temporal nodes*.

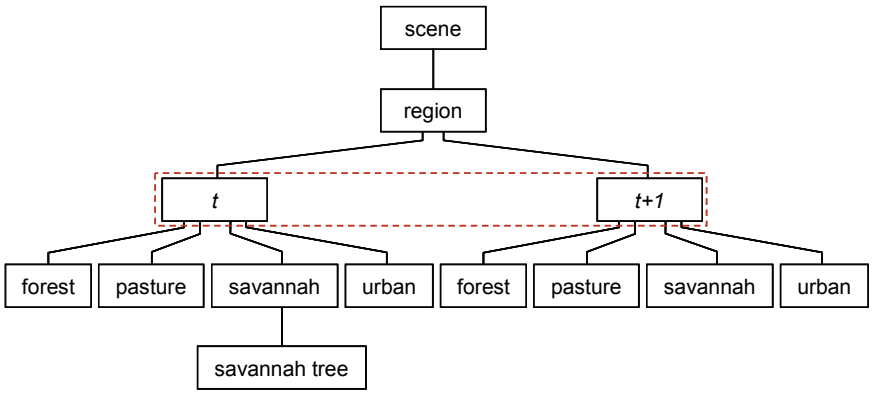


Figure 35. Representing the multitemporal interpretation knowledge model through a single semantic network with temporal nodes.

The temporal nodes do not represent semantic concepts such as a particular object class, they represent points in time for which images or GIS data are available. In the context of the control process, as it will be explained in the next section, the temporal nodes represent synchronization points, where *temporal operations* are preformed.

A basic characteristic of temporal operations is that they do not manipulate images or other types of input data, instead they manipulate object hypotheses or instances (Section 5.1) based on the available temporal knowledge. In principle they can create or delete hypotheses or instances, group them, reshape their associated regions, or alter their properties in any other way.

The descendent nodes of the temporal nodes form sub networks that correspond to the monotemporal classifications. Those sub-networks are regarded as *temporal branches* of the complete semantic network. The children of the temporal nodes correspond to the concepts for which temporal knowledge is available, e.g. a class transition matrix.

In this work, a specific multitemporal classification method, in which temporal relations are modeled by a transition matrix, has been described, but ideally the framework should accept other multitemporal models, which can rely on other forms of temporal knowledge representation.

The specific parameters of the multitemporal model, which express the temporal knowledge, will be made explicit as attributes of *temporal operators* attached to the temporal nodes, such as the operators proposed in Sections 5.3.1 and 5.3.2. Moreover, temporal nodes have a special attribute that defines the order of processing of the temporal branches, used by the interpretation control process as described in the next section.

5.2.2. Control process

Considering the current design of the control process (Section 5.1.4), and the knowledge representation proposed in the previous section, the issue in this section is where and how the temporal operations should be introduced in the control process.

In prior semantic multitemporal interpretation approaches, such as the one presented in (Mota et al., 2007) and (Weis et al., 2005), in which the classification from the past is known, temporal knowledge consisting of a class transition matrix is used to expand the hypotheses network by creating hypotheses for a later time based on the prior classification. In (Pakzad, 2001), class transition knowledge is also used to generate hypotheses based on the outcome of automatic classification of the prior epoch, which is assumed to be the correct classification of the past.

Abstracting from the fact that the two works mentioned in the previous paragraph were developed using different computational systems, and different control strategies, further analysis of them brings forth some interesting reflections. In (Mota et al., 2007) and (Weis et al., 2005) the classification from the past is known, so the creation of hypotheses for a future epoch can occur as soon as the interpretation process reaches the nodes associated to concepts for which the true prior classification is available. In the case of (Pakzad, 2001) where no true prior classification is known, new hypotheses based on temporal knowledge will only be generated after the full interpretation of the prior time.

Considering this last method, therefore, it would only be correct to expand the hypotheses network, by creating hypotheses for the future epoch, after the hypotheses from the past epoch have been verified, i.e. turned into object instances. Thinking in terms of the representation proposed in the previous section, the temporal branch of the network associated to a prior time, say *t*, would have to be completely processed (through top-down and bottom-up processing), before start processing the next temporal branch, associated to time *t+1*.

In this interpretation strategy, which will be hereafter denoted as *sequential multitemporal strategy*, when reaching the temporal nodes, for instance nodes *t* and *t+1* in Figure 35, top-down processing stops at node *t+1*, and continues only after all top-down and bottom-up operations have been concluded for time *t*. At that point the verified hypotheses (object instances) associated to children of temporal node *t* can be use for the generation of the hypotheses associated to the children of temporal node *t+1*. This idea can easily be generalized to consider more than two epochs.

Up until now we have discussed multitemporal interpretation approaches in which the classification of the past is used in the creation of hypotheses for a future point in time. In the general multitemporal model proposed in Chapter 3,

however, classification of both times is performed at the same moment, and the hypotheses for the concepts associated to the children of the temporal nodes are created independently. Additionally, the presence or absence of object instances associated to the descendents of the temporal nodes' children; and the measurement of the characteristics of those objects could be used to update the confidence values of the hypotheses associated to the children of the temporal nodes, values that influence their evaluation (validation/falsification).

Therefore, when the multitemporal classification is simultaneous, temporal operations should be performed after the all temporal branches of the network have been completely processed. In this case, top-down processing on the temporal branches runs concurrently, as in the current implementation of the control process (Section 5.1.4). Bottom-up processing also proceeds as in the current implementation until it reaches the temporal nodes, and then stops. After all temporal branches have been completely processed, the instances of the children of all temporal nodes are subjected to a temporal operation, and bottom-up processing can be resumed. The interpretation strategy described in this paragraph will be denoted hereafter as *synchronous multitemporal strategy*.

In order to make the framework support the two above mentioned control strategies, namely *sequential* and *synchronous multitemporal strategies*, two temporal processing steps have been included in the control process. These processing steps are associated to the temporal nodes of the semantic network, what means that they will be executed during the top-down or bottom-up analysis when processing reaches those nodes.

The new processing steps are denoted respectively as *temporal top-down step* and *temporal bottom-up step*. For the sake of clarity, the processing steps described in Section 5.1.4 will hereafter be named as *structural top-down* and *structural bottom-up* steps, and the operators associated to them as *structural top-down* and *structural bottom-up* operators.

The temporal top-down step enables the sequential multitemporal interpretation strategy as it supports the control of the interpretation execution order with respect to the temporal branches. The interpretation process first enters the temporal top-down step when structural top-down processing reaches the temporal nodes, after the dummy (structural) top-down operators (Section 5.1.2) attached to the temporal nodes have been executed.

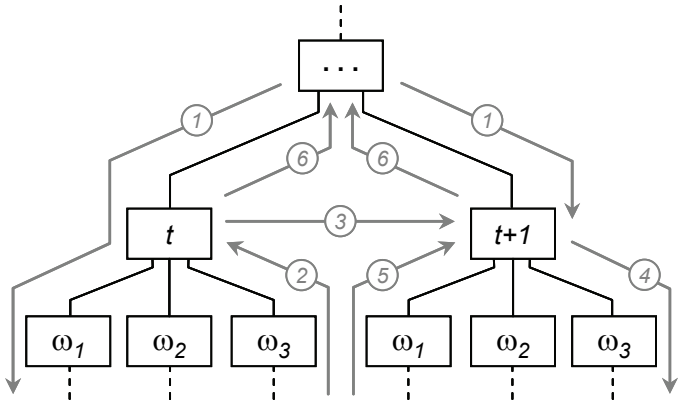
Structural processing of the children of the temporal nodes will then resume in an ordered fashion. Initially the interpretation process will resume for the first temporal branch, leaving the temporal step. When structural (top-down and bottom-up) processing has been concluded for the descendents of the first temporal node, interpretation enters once more the temporal top-down step. At this point a temporal top-down operator (Section 5.2.3) is called, and informed of the temporal branch to be processed. The operator also receives as input the object instances of the children of the temporal node last processed.

The basic task of a temporal top-down operator is to create hypotheses for the children of the temporal nodes, and it does that by using temporal knowledge, e.g. a class transition matrix, which can be hard coded into the operator, automatically defined, or inputted through the system's GUI, according to the particular multitemporal method the operator implements. After the new hypotheses are created, interpretation resumes for the respective temporal branch, and goes back to the temporal top-down step when processing of the branch is concluded (after entering first the temporal-bottom-up step, as it will be made clear in the next paragraphs). The temporal top-down operator is called again and this sequence of steps continues until the last temporal branch has been processed.

The temporal bottom-up step is used in both sequential and synchronous multitemporal interpretation. Processing in this step involves the execution of a temporal bottom-up operator (Section 5.2.4), which is responsible for the validation or falsification of the hypotheses of the children of the temporal nodes.

Depending on the strategy, interpretation will enter the temporal bottom-up step in different ways. In sequential multitemporal interpretation the temporal bottom-up step will be evoked after structural processing of each temporal branch is concluded, before entering the temporal top-down step. The task of the temporal bottom-up operator in this case is to evaluate the hypotheses of the children of the respective temporal node.

Figure 36 illustrates the sequential multitemporal interpretation control strategy. The flow of control is represented by the arrows, the number on the arrows correspond to the order of execution of the processing steps. The same number on different arrows indicates parallel execution.



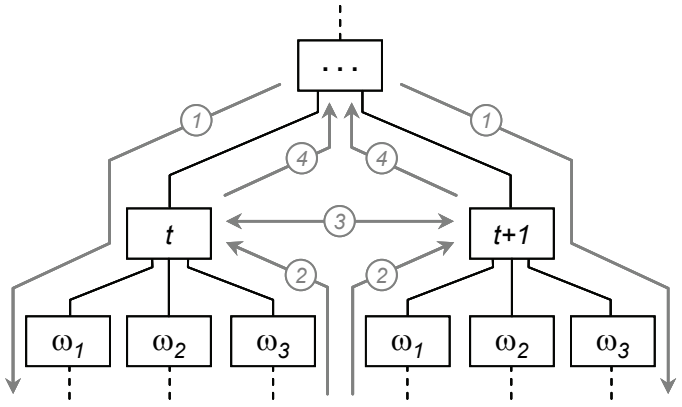
(1) Structural top-down processing: parallel until it reaches the temporal nodes. From there it will continue only for the first temporal branch.
(2) Structural bottom-up processing of the first temporal branch.
(3) Temporal top-down processing: creation of initial hypotheses for the children of the next temporal node based on the instances generated by structural processing of the prior temporal branch and on the available temporal knowledge.
(4) Structural top-down processing of the second temporal branch.
(5) Structural bottom-up processing of the second temporal branch.
(6) Parallel structural bottom-up processing.

Figure 36. Flow of control in sequential multitemporal interpretation.

For the synchronous multitemporal interpretation only the temporal bottom-up step is needed.

In this case structural top-down and bottom-up analysis will be performed concurrently in all temporal branches. When interpretation is completed for all children of a temporal node, processing at that branch will halt until all temporal branches have been processed. When that happens the temporal bottom-up operator is called to evaluate the hypotheses of the children of all temporal nodes simultaneously, and this evaluation will be based on the particular multitemporal method implemented by the operator. As for the temporal top-down operator, input temporal knowledge can be hard coded, automatically defined, or set through the GUI.

Figure 37 illustrates the synchronous multitemporal interpretation control strategy, which is based on the temporal bottom-up processing step.



(1) Parallel structural top-down processing.
(2) Structural bottom-up processing: parallel until it reaches the temporal nodes.
(3) Temporal bottom-up processing: evaluation (validation or falsification) of alternative hypotheses exploring the temporal knowledge.
(4) Parallel structural bottom-up processing.

Figure 37. Flow of control in synchronous multitemporal interpretation.

Although there are only two temporal branches in the semantic networks of figures 36 and 37, in principle there is no limitation, concerning the control strategies, for the number of temporal nodes and branches. That will depend fundamentally on the temporal knowledge underlying the analysis and on the information available (image or GIS data) for the different epochs.

The realization of the temporal processing steps proposed above is achieved by introducing two new states in the finite state machine (FSM) that implements the former control strategy (see Section 5.1.5): `TEMP_TD_EVALUATE` and `TEMP_BU_EVALUATE`. The pseudo code of the extended FSM is presented in Appendix A.

5.2.3. Temporal top-down operator

Temporal operators are, like the structural top-down and bottom-up operators described in sections 5.1.2 and 5.1.3, external programs called by the control process during the interpretation run.

There is, however, an important difference between the functional design of a temporal top-down operator and that of a structural top-down operator. While structural top-down operators directly manipulate image or GIS data as they try to identify regions associated to a particular semantic concept, temporal top-down

operators only manipulate object hypotheses and instances. In addition to temporal knowledge, the inputs of a temporal top-down operator consist of a set of object instances and a variable denoted as *temporal order*, which indicates the temporal branch for which object hypotheses should be created. Its output is a set of object hypotheses.

The execution of a temporal top-down operator will result in a set of object hypotheses associated to the child nodes of a particular temporal node of the semantic network (indicated by the temporal order input variable). To create those hypotheses, the operator applies the temporal knowledge over the object instances (evaluated hypotheses) related to the children of all temporal nodes already processed.

A temporal top-down operator will function in a way that is somewhat similar to a multiclass top-down operator (Section 5.1.2), in that it's output – a set of object hypotheses – is associated to more than one object class, namely the classes associated to the child nodes of a temporal node in the semantic network.

Moreover, different temporal top-down operators can be attached to the temporal nodes involved in a sequential multitemporal interpretation and, depending on the temporal branch to be processed, they can use different components of the temporal knowledge, e.g. different class transition matrixes specific to each pair of dates.

5.2.4. Temporal bottom-up operator

A temporal bottom-up operator is also an external program. Its typical inputs are: a list of all instances associated to the children of the temporal nodes; and some representation of the temporal knowledge needed by the particular multitemporal method that the operator implements.

In the specific case of the multitemporal classification method proposed in Chapter 3, the temporal knowledge consists of a class transition matrix, but for other potential multitemporal methods it could contain other components.

The basic function of such an operator is to select form the object hypotheses associated to the children of the temporal nodes the set best evaluated by the multitemporal classification method it implements, taking into account the

input temporal knowledge. The operator must also resolve eventual spatial conflicts among the instances associated to the children of the temporal nodes, reshaping the related regions in order to remove eventual overlaps.