Handling Temporal Granularity in Situation-Based Services

Stefan Pfennigschmidt* and Agnès Voisard *†

TR-09-005

July 2009

Abstract

In many application fields such as disaster management or personal planning, efficient mobile services should capture users' context in order to provide them with the most appropriate information at a given time. Situation-based services aim toward this direction. They go beyond location-based services as they take a user's current situation into account. Situations include user profiles together with dimensions such as their location, current activity, or type of connectivity. They can be derived from direct observations or from electronic calendars and are defined as valid during a time interval. Depending on the application but also within the same application the time interval of interest can be of various granularities. It is crucial to handle time granularity efficiently and in a well-defined manner. In this paper, we investigate the handling of various granularities for situations and situation sequences through, in particular, coarsening methods. We propose an algorithm that we tested in concrete projects and we illustrate its use in a tourism application.

* Fraunhofer Institute for Software and Systems Engineering (ISST)
{stefan.pfennigschmidt, agnes.voisard} @isst.fraunhofer.de
† Computer Science Institute, Freie Universität, Berlin, Germany

I. Introduction

Situation-based services have emerged as services that capture (mobile) users' dynamic profiles in order to send them appropriate information. Situations are defined over various dimensions such as location, current activity, or type of connectivity. They can be derived from direct observations (by sensing context), from electronic calendars, or estimated looking at either individual or group behavior. They are defined as valid during a time interval. Depending on the application but also within the same application the time interval of interest can be of various granularities.

Handling temporal granularity in situation-based services is of crucial importance in order to deliver appropriate information to users and match their information needs as accurately as possible. The need to handle granularity generally arises when different sources of situational information are being used and need to be combined and possibly compared. Most current situation-based services — e.g., navigation services, user schedule assistants, or weather forecasts — integrate sources that are based on different temporal granularities. Combining this information is sometimes necessary, for instance to get an integrated view on the overall situation of a user or a user group. In the latter case, we can take the example of a disaster management system that needs to alert all the users who are in a particular situation (location and activity). Furthermore, many services such as Personal Information Management (PIM) services or transportation services rely on a functionality to compare precise schedules with roughly defined plans. This applies, e.g., when one wants to know whether an actual observation matches an intended course of events, or whether certain predictions meet the expectations of a user (see for instance [MPVW05] for more details). Last but not least, in order to keep a user up-to-date with information of different levels of detail (with respect to both content and temporal precision), one needs to switch between different temporal granularities.

The situation model that we take as a reference is defined on many dimensions and each dimension is defined on a hierarchy (for instance, if a user is in the Plaka district in Athens, Greece, she will be in "the Plaka-district/Athens-downtown/Greece/Southern Europe/…" and so on). This is useful to target users at different levels by rolling up in the hierarchy (e.g., all inhabitants of Athens in a disaster management application). With the time being one dimension, one could assume that the same mechanism would apply to the temporal characteristics. However, time plays a particular role as it is a parameter of the situation concept and needs special processing when situations are to be combined and merged. In other words, tackling temporal granularity in this context is different from tackling other types of granularity such as the spatial granularity (which can be handled by a simple model that handles hierarchies, such as an object-oriented model). In this paper, we investigate the problem of handling various temporal granularities for situations and situation sequences. In our approach, we take as a basis existing concepts from the temporal database community and we extend them in order to use them in the context of situation-based services.

In early work, the temporal database community has focused on defining and reasoning on time intervals [All84]. Temporal granularity was introduced in the early 90's [WJL91] and has received notable attention from the temporal database community who mostly described temporal granularity handling in a formal manner (see for instance [Euz95], [MGB+00], [DELS00], [CC02], [CFP04], [GLz+04]). Context handling in mobile environments started receiving attention in the late 90's [ADB+99] and many conceptual approaches have been proposed lately in the context of mobile applications (see for instance [BD07] for a comprehensive survey), although they usually focus on spatial, location-based aspects. Granular context was introduced in various areas such as collaborative work (see e.g., [DSD06], which concentrates on fuzzy intervals) and mobile applications as an inherent concept [Sch05]. The situation model introduced in [MPVW04] is simple yet powerful to capture users' context at some granularity level but does not consider different granularity combinations coming from different sources. To the best of our knowledge, the problem tackled in this paper has not been investigated so far.

This paper is organized as follows. Section II gives the concepts necessary to tackle this problem. Section III establishes the relationship between temporal granularity and situations and presents two coarsening procedures, a naive one and a more elaborate one. Section V presents the architecture of our system and illustrates our approach taking examples from the area of tourist information services. Finally, Section VI draws our conclusions.

II. Main Concepts

This section introduces the main concepts used in this paper, namely our situation model with an extended list of operators, and temporal concepts such as granularity and granule. Part of the basic terminology can be found in [BDE+98].

A. Situations and Situation Sequences

The notion of user context is usually understood as a snapshot of his or her environment at a given time based on raw values (e.g., temperature or location). The situation model proposed in [MPVW04] abstracts from such values and defines a situation as a collection of characteristic values (similar to attribute values) valid during a time interval.

Definition 2.1 (situation): A situation is a tuple: $s = (t_{\mathrm{a}}, t_{\mathrm{b}}, SC)$, with t_{a}: beginning of the validity period, t_{b}: end of the validity period, $SC = \{C_1.c_1, \ldots, C_n.c_n\}$, with $C_1, \ldots, C_n$ characteristics and $C_1.c_1, \ldots, C_n.c_n$ characteristic values.

To capture situations at various levels of abstraction, characteristics are defined along dimensions and are instances of ontologies. Operators on characteristics and characteristic values are introduced as technical tools for reasoning on situations and are: $\mathrm{ancestor}: (c_i, c_j) \rightarrow c_n$, which returns the smallest common ancestor of two characteristics values,

$\text{most-specific-set}: (cs_i, cs_j) \rightarrow cs_l$ which finds the most specific set of characteristic values common to two sets of characteristic values, and $\text{compare-set}: (c_{si}, c_{sj}) \rightarrow [0, 1]$, which gives the similarity between two sets of characteristics (using a similarity metric on the subsume paths of the dimensions). In the following, to make it shorter we sometimes denote *feature* a characteristic value and *pattern* a list of characteristic values. We also use the term *dimension* as a short version of a *dimension structure*, which represents the hierarchy of attributes along one dimension.

One of the key notions of the situation model is that of *situation sequences*. A situation sequence describes a sequence of states over time. It can be used to describe the process of writing a paper, or the sequence of transportation means I used on a particular day in order to get from home to work. Situation sequences are defined as follows.

Definition 2.2 (situation sequence): A *situation sequence* $(S, \prec:)$ is a well-ordered set of not-overlapping situations.

The concept of situation sequences can be used by application designers to anticipate upcoming situations and to compare the system knowledge against the user's current situation to find possible discrepancy.

Situation sets. In contrast to situation sequences, simple *sets of situations* are not restricted w.r.t. the compatibility of the situations contained. Situations from a set can overlap, they can express contradictory knowledge, they can express knowledge, which is partly shared by other situations, and so forth; whereas situations belonging to the same situation sequence can not.

Every situation sequence can be represented as a set of situations. And if a set is *consistent*, which means that all situations contained are pairwise compatible, this set can be used to produce a situation sequence out of it by unifying its elements (see *union* operator below). Except for the empty sequence there always is an infinite number of different consistent sets that can produce the same sequence. However, there always is one smallest set of one-dimensional situations that corresponds to every sequence. This set is called *minimal producing set* of a situation sequence. Because both a situation sequence and its minimal producing set are equivalent w.r.t. their situational knowledge, often the set can be used instead of the sequence itself when manipulating situational information.

Operators on situations within situation sequences.

- Temporal operations (based on time-related operators from Allen's logics): *meets* (neighboring situations), *predecessor* (use of *before*), *successor* (use of *after*), *starts*, *finishes*, *synchronous* (share the same time interval).
- Relationship predicates (based on relationships of the intervals and characteristic values): *compatible* (characteristics do not contradict), *independent*, *subsituation*, *part-of*.

Operators on situation sequences.

- Set operators: *union*, *intersection*, *difference*.
- Relational operators: *selection*, *projection*.
- Additional operators: *generalization*, *extraction*.

Since the algorithm presented later relies on the set-based operations we give their definitions hereafter (see Fig. 1 for an illustration).

Definition 2.3 (Difference: $\setminus: \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S}$): The difference of two situation sequences S_1 and S_2 is the situation sequence describing the situational knowledge contained in S_1 that do not appear in S_2. We write $S_1 \setminus S_2$.

Definition 2.4 (Intersection: $\cap: \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S}$): The intersection of two situation sequences S_1 and S_2 is the situation sequence describing the situational knowledge contained in S_1 as well as in S_2. Thus an intersection describes the knowledge shared by two sequences. We write $S_1 \cap S_2$.

Definition 2.5 (Union: $\cup: \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S}$): The union of two compatible situation sequences $S_1 \cup S_2$ is the situation sequence describing the combined situational knowledge of both sequences S_1 and S_2. We write $S_1 \cup S_2$.

B. Temporal Issues

Temporal statements can be made using different precisions. Differences in the accuracy could stem, for instance, from inaccuracy of clocks. They result from the size and exact keeping of measurement intervals. This accuracy or these inaccuracies play a major role when it comes to modeling and interpreting situational information. Time points, used to specify situations, cannot be considered exact in an absolute sense. They can only be considered exact on a certain scale (see Fig. 2 in a PIM application). This scale can be defined by tolerances which define the granularity of the time points.

Temporal indeterminacy. Temporal indeterminacy can have different reasons, which can be grouped into two categories: (i) vagueness due to imprecise measurement and (ii) inaccuracy of data due to the loss of information resulting from a conversion from a higher to a lower granularity. Many approaches to model time concepts using different granularities or units of time have been proposed (see related work in the introduction). The granularity-related concepts and their definitions that are widely accepted and commonly used in the temporal database and temporal reasoning communities (see for instance [BDE+98] or [DELS00]) are given below.

Time domain. A *time domain* is defined as a pair $(T; \leq)$ where T is a non-empty set of time instants and $\leq$ is a total order on T. The time domain represents the primitive atomic values used to describe all other time related concepts. One can distinguish between dense and discrete time domains [BDE+98]. In a dense time domain for any two time instants $t, t' \in T$ with $t < t'$ exists a time instant $t'' \in T$ with $t < t'' < t'$ like in the area of real numbers.

Granularity. Granularities are build by aggregating portions of the time domain into granules. A *granule* is a non-empty subset of the time domain. A set of granules is called a granularity, whereby certain restrictions apply; first, granules in a granularity do not overlap, and second, the granules are ordered according to the temporal order of their time instants. To achieve that, granularities are commonly defined to be a mapping G from integers, called the *index set*, to subsets of the time domain, such that: (1) if $i < j$ and $G(i)$ and $G(j)$ are not

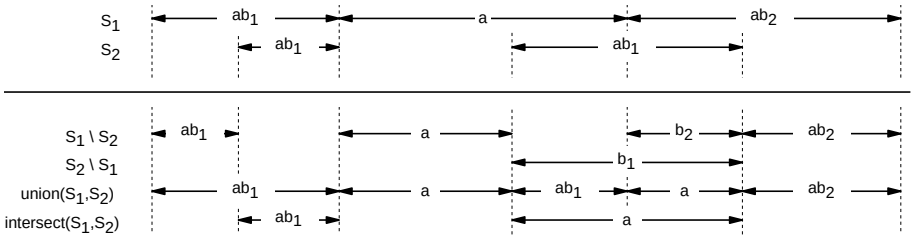

Fig. 1. Intended results of the set-based operators: The sequences S_1 and S_2 contain different (and partially contradicting) knowledge about the situations of a person. The labels represent the characteristics and should be interpreted as follows: a = in Athens, b_1 = using a cab, and b_2 = using the metro. Additionally the knowledge ($b_1 \rightarrow \neg b_2$ and vice versa) is supplied by a transportation taxonomy. The gaps mean that there is no situational knowledge available.

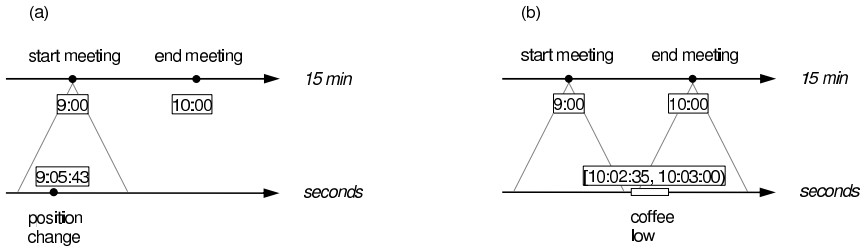

Fig. 2. The occurrence time of events can be given at different granularities and precision (adapted from [CC02]).

empty, then each element of G(i) is less than all elements of $G(j)$, and (2) if $i < k < j$ and $G(i)$ and $G(j)$ are not empty, then $G(k)$ is not-empty [BDE+98]. The second restriction ensures that all integers of the index set actually represent a granule.

[CFP04] introduces the following terms to describe the properties of granularities with respect to their coverage of the time domain.

- *externally continuous*, if there are no gaps between the granules;
- *internally continuous*, if there are no gaps inside the granules;
- *continuous*, if the granularity is both externally and internally continuous;
- *total*, if the image of the granularity covers the whole time domain;
- *uniform*, if all non-empty granules have the same cardinality.

We extend this nomenclature with the definition of the property *right-total*, which is important when transforming situational information according to different granularities.

Definition 2.6 (Right-total): A granularity G is called *right-total* if and only if G is continuous and every granule $g \in G$ has a successor.

C. Granularity Relationships

Granularities that use the same time domain as their basis can have different relationships. When using granularities to describe situations, their relationships, and operations, the notions presented below are the ones we use.

Subgranularity. A granularity G is called a *subgranularity* of a granularity H, denoted $G \sqsubseteq H$, if the set of granules of G is a subset of the set of granules of H.

Finer than. A granularity G is called *finer than* a granularity H, denoted $G \preceq H$, if for every granule $G(i)$ of G there exists a granule $H(j)$ in H such that $G(i)$ represents a subset of $H(j)$.

Groups into. A granularity G is said to *group into* a granularity H, written $G \trianglelefteq H$, if all granules of H can be represented by the union of a set of granules of G.

Partitions. A granularity G partitions a granularity H if $G \trianglelefteq H$ and $G \preceq H$. In this case, both granularities cover exactly the same portion of the time domain image, i. e., the images of G and H are identical.

Covered By. A granularity G is covered by a granularity H if the image of G is a subset of the image of H.

D. Conversion Operations

Comparing or integrating situational information that use different granularity requires homogenization, that is, the information needs to be converted to a common granularity before the operation is performed. This requires converting time points between granularities at the first place. The two operations *scale* and *cast* have been proposed in order to accomplish this conversion [DELS00]. Because the scale operation tries to maintain as much temporal information as possible it always produces an indeterminate value (i. e., an interval) when used to move to a finer granularity. Since situations are modeled using determinate values (time instants) to describe their start and end, scale cannot be used. In contrast, the *cast* operation always produces a determinate result (when applied to a time point), regardless of whether we switch to a finer or a coarser granularity. That is, by using *cast* the situation model structure is kept. In [DELS00], different semantics (left-operand, right-operand, finer, and coarser) are discussed that differ in how the two operands (in our case situations or situation sequences)

are homogenized. The latter two are used in our approach. Using the *finer* semantics the time values of both operands are converted to the granularity that is finer than the other; or, if both granularities are incomparable, to a granularity that is finer than both is chosen. One should be aware that an interpretation of a situation sequence resulting from a so-called refinement requires caution, because it "pretends" to have information that is more precise than it actually is. Using the *coarser* semantics conversion is done to the coarser granularity, or in case of incomparable granularities to a finest granularity that is coarser then the granularities of both operands.

III. Temporal Granularity and Situations

This section starts with a description of the problem before proposing an advanced granular situation concept. We then present the procedure to coarsen situations, starting with a naive approach (which may introduce information loss) followed by a more efficient approach based on *rasterization.*

A. Interpreting and Combining Situational Knowledge

The aspects of temporal granularity described above strongly influence how situational information is to be interpreted. The sequence of the patterns, for instance, may change with the level of granularity chosen. The question whether detailed or more generic features can be used to describe situations — or whether one can recognize the internal structure of an event or not — depends also on the granularity with which the situations are represented. That is, in order to analyze the knowledge contained in situation sequences we not only have to be able to move between different levels of abstraction (depending on the structure of the dimensions) but we also need (i) a technique that facilitates the representation of situational knowledge at different levels of temporal granularity and (ii) procedures to switch level (similar to querying a database using different granularities). The temporal granularity level also influences the level of abstraction w.r.t the characteristics, hence the underlying database schema that is used.

However, not only the interpretation of single situation sequences, but also the combination or comparison of two ore more sequences depend on these granularities and on the issues of temporal compatibility connected to them. To that end, the temporal information needs to be homogenized with respect to the granularity used. The *coarser* semantics conversion where both arguments of an operation are moved to the coarser granularity of both is best suited to most application cases. In the following, we propose a similar procedure, where situation sequences are first homogenized with respect to their granularities. Homogenizing means that the coarser granularity is chosen and the sequence with the finer granularity is transformed or *coarsened* (a processus similar to generalization), such that both sequences are of the same granularity afterward and can be combined or compared by applying the respective operator without further processing. The comparison or combination operation itself is achieved by applying the set-based operators, namely, *difference* ($\setminus$), *union* ($\cup$), and *intersection* ($\cap$), introduced in Section II.

In contrast, moving to a finer granularity ("refinement") is simple. For example, transforming to the granularity of seconds just requires that all time points be mapped from there own granule to the respective starting second using the *cast* operation.

B. Granular Situation Model

The basic situation model proposed in [MPVW05] relies on the assumption that all time points belong to one single granularity, which is a limitation when situational information is coming from of different sources. That means, for some real world applications the issue of temporal compatibility needs to be addressed when combining, transforming, or comparing situational information.

We therefore extend this simple model with the notions of temporal granularities.

A *granular time point* is given by a tuple (g, G) where G is a granularity and g identifies the granule within G. Following common notation conventions we write the granularity as a subscript to the label of the granule, e.g., $14{:}30_{\mathrm{min}}$. Using granular time points would enable us to model situations and situation sequences where all time points may be of different granularity. However, in most application cases it is (i) sufficient to have one granularity per situation sequence, and (ii) easier to communicate such situational information to the user. Users specify items of their own agenda with all the same temporal precision, first, in order to keep information comprehensible, and second, existing calendar applications do not provide the required functionality. Also, data acquired from sensors mostly depend on some fixed measurement interval. Although a model based on granular time points would allow for the specification of time points of different granularity within a situation or situation sequence, we propose simplified mono-granularity definitions.

We extend the model and define a *granular situation* as a tuple (s, G), where s is a situation $(t_{\mathrm{a}}, t_{\mathrm{b}}, SC)$ and $t_{\mathrm{a}}, t_{\mathrm{b}} \in G$. A *granular situation sequence* is defined as (Seq, G) where $Seq = (S, \prec{:})$ is a sequence of not-overlapping situations and G denotes the granularity common to all time points used.

In the granular situation model, every situation and situation sequence has its own granularity; the granularity of a situation or a sequence is the set of granules used to specify the time points they include. For example: situation $s = (14{:}30_{\mathrm{min}}, 15{:}15_{\mathrm{min}}, \{\text{'meeting'}\})$, the granularity of this situation is given by two granules covering the intervals $[14{:}30{:}00, 14{:}31{:}00)$ and $[15{:}15{:}00, 15{:}16{:}00)$ of the underlying time domain. That such set of granules form a granularity is guaranteed by the definition of a situation and a situation sequence, which both require a total temporal order of the time points used.

Definition 3.1 (Temporal compatibility): Two situation sequences S_1 and S_2 are temporally compatible if there exists a granularity G of which the granularities of both sequences $\mathrm{G}(S_1)$ and $\mathrm{G}(S_2)$ are subgranularities.

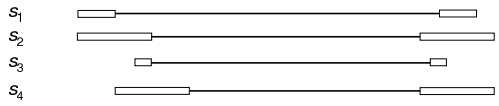

Fig. 3. Synchrony of situations considering temporal granularity: situation s_2 is synchronous to s_1 and to s_3, because there respective begin and end time points match; *but*, s_1 and s_3 are not synchronous, (synchrony is no longer a transitive relation); whether s_2 and s_4 are synchronous is, because of the overlapping begin time, not decidable.

Symbolically:

$$\begin{aligned} &\text{temporally-compatible}(S_1, S_2) \quad \Longleftrightarrow \\ &\exists G \,.\, \mathrm{G}(S_1) \sqsubseteq G \wedge \mathrm{G}(S_2) \sqsubseteq G \end{aligned} \tag{1}$$

Synchrony. The set-based operators (union, intersection, difference), which play a major role in comparing and integrating situational knowledge, are based on the notion of *synchronized situations* and situation sequences. Two situations are called *synchronous* if they share the same time interval. When different temporal granularities are involved this relationship is not always decidable. In order to apply the operators we need *actual synchrony*, which — in contrast to *potential synchrony* — requires that both situations use the same granularity (referred to as *mono-granularity* below).

Definition 3.2 (Mono-granularity synchrony): Two situations s_1, s_2 are *synchronous* if and only if they share the same time interval. Two situations sequences S_1, S_2 are *synchronous* if and only if for all situations $s_1 \in S_1$ there exists a situation $s_2 \in S_2$, which is synchronous to s_1 and vice versa.

Symbolically:

$$\mathrm{Sync}(s_1, s_2) \iff \mathrm{Starts}(s_1, s_2) \wedge \mathrm{Finishes}(s_1, s_2) \tag{2}$$

$$\begin{aligned} \mathrm{Sync}(S_1, S_2) \quad \Longleftrightarrow \\ (\forall\,(s_1 \in S_1)\,.\,\exists\,(s_2 \in S_2)\,.\,\mathrm{Sync}(s_1, s_2)) \wedge \\ (\forall\,(s_2 \in S_2)\,.\,\exists\,(s_1 \in S_1)\,.\,\mathrm{Sync}(s_2, s_1)) \end{aligned} \tag{3}$$

C. Coarsening: Naive Algorithm

In order to simplify the presentation of our coarsening approach, we make the following assumptions about the granularities used:

- All granularities are *right-total*; i. e., every granule has a successor;
- All granularities are *continuous*; i. e., neither a single granule, nor the granularities themselves can have gaps;
- Granularities used build a partitioning hierarchy; i. e., there are no overlaps between the granules;

Problems and requirements. The first naive approach to the problem of coarsening a situation sequence uses a simple algorithm. It just iterates over all single situations of a sequence to coarsen them individually. That is, the begin and end time points of the situation are mapped onto the temporal granularity chosen. As a result a situation can collapse to an event in case both time points — *begin* as well as *end* — are mapped to the same granule. COARSEN-SITUATION (see Alg. 1) is invoked with the situation s to be coarsened and the temporal precision p which is used to define the granularity on the time axis.[1]

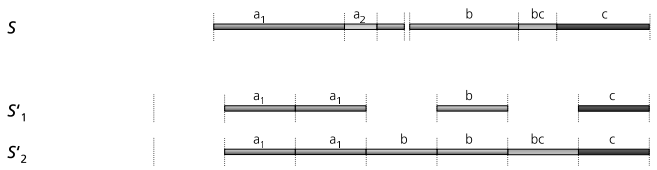

Fig. 4. Different coarsening algorithms produce different situation sequences. Applying the naive version (Alg. 1) results in sequence $S_1^{'}$, which has unexpected gaps; $S_2^{'}$, however, is a coarsened version of S that is closest to the input, and thus represents the intended result.

Alg. 1 Coarsening a situation.

COARSEN-SITUATION(situation s, precision p)
1 $start \leftarrow$ begin time of situation s
2 $end \leftarrow$ end time of situation s

3 $start \leftarrow \textsc{Cast}(start, p)$
4 $end \leftarrow \textsc{Cast}(end, p)$
5 **if** $start = end$
6 **then return** NIL $\triangleright$ s collapsed

7 $pattern \leftarrow$ pattern of situation s
8 **return** $(start, end, pattern)$

Although this algorithm ensures that every time point afterward is specified using the granularity defined by p, it, in many cases, produces unwanted or non-intuitive results because of the information loss implied (see the resulting sequence $S_1^{'}$ in Fig. 4). The major reason is that the *sequence* of situations is not taken into account. First, some characteristic values that were additionally observed create situations too short for the defined grid. They collapse and are therefore no longer visible. Other situations are just left or even broadened (although being much shorter than those that collapsed) just because their begin and end are mapped to different granules. To circumvent this problem, all situations that have a duration that is less then half (or just half) the size of a granule could (for fairness reasons) be removed as well. The first disadvantage could be met by coarsening not the situations from the sequences as they are but rather producing the *minimal producing set* from the sequence, coarsening the situations obtained, and unifying the resulting situations back to get a coarsened version of the input sequence. The result does look better, however, some knowledge is still lost (although it should not be). Some problems arise if two (or more) neighboring situations are themselves too short to be visible, but contain a common higher level information (in case of hierarchically-defined dimensions). For instance, if a sequence of subsequent short working tasks each with a duration of about 10 minutes to 20

[1]The precision p has to be coarser than the precision of the granularity used by the input sequence.

minutes, is coarsened to a granularity of, let us say, 1 hour, every single task will collapse because of its shortness. The information that the user was 'working' all the time is lost as well.

IV. Efficient Coarsening Minimizing Situational Knowledge Loss

Because the algorithms discussed so far are not efficient in the sense that they lose too much knowledge and therefore do not produce the results that one would expect, we now present an algorithm that is able to coarsen a situation sequence while preserving as much situational knowledge as possible.

Task definition. We revise the coarsening task in order to find a coarse representation of a situation sequence that approximates the input sequence as near as possible, such that only a minimum of situational knowledge is lost; in other words, our goal is *to find a representation S_{G} of S in granularity G such that* $\mathrm{distance}(S, S_{\mathrm{G}})$ *is minimal.*

Information loss. A first question arising from the task definition is what does "losing knowledge" precisely mean and how to measure the loss. We obviously lose knowledge if a characteristic value holding at a certain time point or over a certain interval is no longer present in the coarsened version of the input sequence. However, also adding facts that are not present in the input sequence introduce errors and can be regarded as lost information. In the following, we assume that losing as well as "gaining" information are equally wrong and have to be considered when computing the overall error.

The coarsening procedure should minimize the error caused by retracting or adding information. In order to understand this minimum criterion, we have to quantify the error and define a distance measure. There are of course many different distance metrics or procedures or algorithms that could be applied. One possible approach would be, e. g., using a shortest-editing-path algorithm. Here we would chop the sequences that are to be compared into single segments according to the temporal granularity chosen. Using the algorithm we could analyze and compare the resulting pattern sequences, which would result in information where to insert or where to remove a certain segment. However, this kind of algorithm emphasize the retaining of the sequence and retaining of the transitions and weights these much more than the actual position of a segment. This means that movements of situations on the time axis are considered only insufficiently, when applying this procedures to situation sequences. We therefore follow another approach called *rasterization*, known, for instance, from digital imaging. The idea is to split the time axis into granules (in digital imaging defined by the *resolution*) and to analyze each granule separately.

A. Measuring Distances

The main objective of the development of a distance metrics is its ability to estimate different coarsening algorithms with respect to their capability to preserve situational information. Because we only need it to decide whether one algorithm is better than another one, only a relative distance is needed and a simple metrics is sufficient.

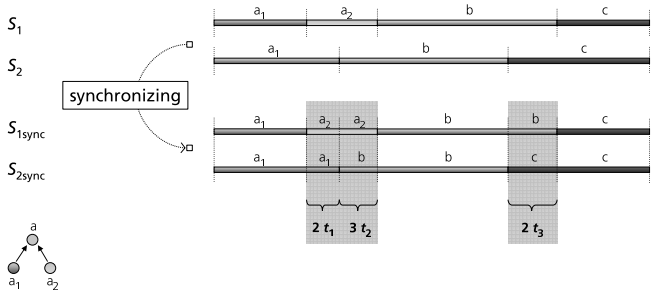

Fig. 5. Measuring the distance by first synchronizing both sequences and computing the distance element-wise.

The measuring procedure proposed in the following comprises the following basic steps:

1) Cast the situation sequences to a common temporal granularity (using the *finer semantics*).
2) Synchronize the situation sequences to be compared.
3) Measure the distance element-wise by iterating through the built interval pairs.
4) Add up the intermediate results.

By synchronizing the sequences S_1 and S_2 both are split into the same number of intervals (see Fig. 5), where the i-th elements of either sequence are of equal length. This enables us to iterate through the sequences and compute the distance element-wise. Because there cannot be a change of patterns within these elements, we can compute the distance by comparing the patterns and multiplying the result with the length or duration of the interval. Adding up the partial distances yields the overall distance between the sequences. To compute the distance between two patterns, we use a simple approach by counting the number of values that are exclusively in either pattern. Because we only need a relative measure, this approach is sufficient (we will see later that we can also use more sophisticated distance measures by introducing the notion of *information content* or *cost* for every value). Let t_i denote the length of interval i, P_{1i} and P_{2i} the patterns of S_1 and S_2 in interval i, and $\mathrm{T}(P)$ the theory of a pattern P, i. e., the set of features derivable from this pattern, the following formulas describe this procedure.

$$d_i^+ = \mathrm{card}(\mathrm{T}(P_{1i}) \setminus \mathrm{T}(P_{2i})) \quad (4)$$

$$d_i^- = \mathrm{card}(\mathrm{T}(P_{2i}) \setminus \mathrm{T}(P_{1i})) \quad (5)$$

$$\mathrm{distance}(S_1, S_2) \stackrel{\mathrm{def}}{=} \sum_{i=1}^{n} t_i (d_i^+ + d_i^-) \quad (6)$$

Formula 4 describes the number of features exclusively in P_{1i}, where Formula 5 describes the number of features only in P_{2i}. The sum of d_i^+ and d_i^- represents the distance between both patterns.

In order to define a more sophisticated distance metrics, we could associate a real number with any node in a hierarchical dimension representing the information value of the respective proposition. Note that such values can also differ for a node

in case the same structure is used in different dimensions. That is, the cost value does not only depend on the concept from the dimension but on the feature. We introduce a cost function $\mathrm{v}\colon \mathcal{F} \to \mathbf{R}$ that gives the cost or information value of a single feature $f \in \mathcal{F}$. This value represents the amount of information content that is directly associated with this feature. It does not include the cost or value of the features derivable from f. In order to make this clear, we define the *cumulative value* of a feature f to be the sum of the values of the theory $\mathrm{T}(f)$ of f:

$$\mathrm{v}_{\mathrm{cm}}(f) \stackrel{\mathrm{def}}{=} \sum_{j=1}^{m} \mathrm{v}(f_j) \mid f_j \in \mathrm{T}(f)\,. \tag{7}$$

Applying the cost function $\mathrm{v}(f)$ to our distance measure formulas 4 and 5 look slightly different:

$$d_i^+ = \sum_{j=1}^{m} \mathrm{v}(f_j) \mid f_j \in (\mathrm{T}(P_{1i}) \setminus \mathrm{T}(P_{2i})) \tag{8}$$

$$d_i^- = \sum_{j=1}^{m} \mathrm{v}(f_j) \mid f_j \in (\mathrm{T}(P_{2i}) \setminus \mathrm{T}(P_{1i})) \tag{9}$$

In the special case that $\mathrm{v}(f)$ equally returns 1 for all features, adding the values simply means to count the features. That is, in that case, the formulas 8 and 9 are equivalent to the previous versions.

In order to estimate the quality of a coarsening algorithm we do not really need to actually quantify the distance between two situation sequences. Only a relative distance, determining whether the distances between an input sequence and two coarsening results are equal ore which of them is greater than the other, is necessary. Because the relation $(<, >, =)$ for any two terms $\sum v_1(f_j)$ and $\sum v_2(f_j)$, where v_1 and v_2 denote different cost functions, will be the same, provided the costs returned by v_1 as well as v_2 are greater than zero, we can use any cost function even a constant. However, we should note that we additionally need the restriction of a constant cost functions over time.

Shortcomings. The metrics proposed above is not able to handle non-convex granules or non-continuous granularities. Gaps, i. e., portions of the time line that are not covered by a granularity are ignored. Problems arise from this fact, when situation sequences are compared, the granularities of which have different images, i. e., the gaps in both granularities have different locations or extents. If both granularities, however, cover the same portion of the time domain — hence sharing the same image, as is the case, for instance, for *business-minutes* and *business-hours* — then only the quantitative result will be distorted; the computed distance will be greater than it should be. The relative distance, however, and with it the qualitative properties $(<, >, =)$ are preserved.

B. Algorithm

We now present a coarsening algorithm that is able to produce a coarsened representation of a situation sequence that approximates the input sequence as close as possible

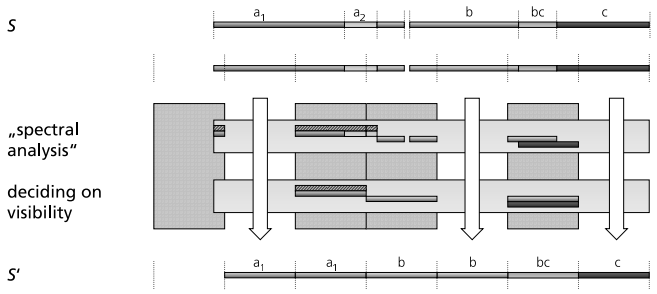

Fig. 6. The general coarsening algorithm. Note that the rules $a_1 \to a$ and $a_2 \to a$ are given by a dimension structure.

with respect to this distance metrics. The algorithm uses the rasterization approach mentioned before. That is, in contrast to the simple algorithms already presented, it does not treat situations and their patterns as a whole, but analyzes the features of the situations based on granules. The general procedure comprises the following steps:

1) *Slicing.* The input sequence is chopped according to the granularity chosen, at the start of every granule. This first step can be regarded as synchronizing the sequence with the sequence of granules. The next steps are done element-wise for every granule.
2) *Analyzing.* The features of all situation patterns present within a single granule are being extracted.
3) *Deciding on visibility.* A single feature is visible if its overall presence within a granule is greater than half the size of a granule, thus a feature either vanishes or fills a complete granule. This ensures that the distance between the original and the resulting situation sequence and thus the error introduced is minimal.

After having processed all granules, the resulting sequence is coalesced. Fig. 6 illustrates this approach.

In order to simplify the procedure, we here assume that the granularity is built of equal sized granules of a certain precision p. The *cast* operation maps an arbitrary time point to (the start of) a granule. Any subsequent granule can then be identified by adding a multiple of p.

The simple algorithm shown in Fig. 6 is not really efficient, as it iterates over all granules covered by the situation sequence, and should be regarded as a more or less declarative specification of the coarsening procedure. To optimize it, we have only to consider the granules in which the input sequence has at least one event. Between such granules, there will be no changes and the pattern can be transmitted unchanged into the resulting sequence. The final procedure is presented in Alg. 2.

The algorithm iterates through all events of the input sequence. That is, for every time point where some change occurs it first computes the temporal granule this change belongs to. It computes the lower bound t_{lb} and the upper bound t_{ub} (lines 12–14). Using the interval $[t_{\mathrm{lb}}, t_{\mathrm{last}})$ (where t_{last} represents the upper bound of the last step) we add a situation to the result using the pattern of the input sequence that holds during this interval (lines 15–17). Because there is no transition in that particular interval, we can just take

the pattern at some point between t_{last} and t_{lb} (line 17). Following this step we process the interval $[t_{\text{lb}}, t_{\text{ub}})$. We therefore first extract all features from the situation pattern of the input sequence (line 25). Then, for all features extracted we compute the overall time of presence of a feature in the interval (line 28), check whether this duration is longer than half the resolution (line 29), and in case this is true, we add the feature to the resulting pattern P' (line 32). If there was at least one feature added to this pattern, P' will not be empty, and we add a new situation to the resulting sequence. (line 36). The algorithm ends when all events of the input sequence have been processed.

V. Application

This section illustrates the practical use of the algorithm presented above. It starts with a description of a general architecture for systems relying on our situation model. It then shows two examples borrowed from a tourist application.

A. System Architecture

The general system architecture of a situation-based system is shown on Fig. 7 using an "intelligent" tourist guide as application example. The situation provider component provides situation information and services (including granularity handling) to the other components. It connects to various sources of situational information, including personal sources (e.g., positioning) and external sources (e.g., weather forecasts). Situation information is stored in a situation repository using application specific dimensions and different categories (e.g., user expectations, observations, predications). In the example, user applications access situational information (historic, current, and anticipated) using the situation provider's interface and use it to retrieve appropriate content from a situation-enabled content repository.

B. Illustration Examples

Example 5.1: Let us assume that a tour operator prepares a coach tour to Athens, Greece. They do their internal planning (take off, arrival at the ferry, at the hotel, and so on) as precise as possible. However, information about the tour is given out to (potential) members of the coach party — as an external view — using days or half-days as granularity. This external information forms the expectations of the party members about the journey. Due to external conditions, the internal plans may change (e.g., change of coach contractor, use of a different ferry, severe weather announced). In order to keep the party up-to-date the tour operator compares the updated internal plan with the information given to the party. A change would be considered relevant when comparing the new plan (viewed at the granularity of the external information) yields a delta to the information that was originally being given out. In that case the tour members should be informed (because they expect a different course of events).

Example 5.2: When preparing a business trip or a family vacation, users usually do some planning about activities to do, sites to see, and so on. Such information can be modeled as a situation sequence, whereby at first only approximate information is available, that is, a rudimentary plan using a coarse grained time scale is specified.

$$S_{\text{week}} = \left\{ \begin{array}{l} (\text{cw}20, \text{cw}21, \{\text{loc}(\mathit{Athens})\}), \\ (\text{cw}21, \text{cw}22, \{\text{loc}(\mathit{Corinth})\}), \\ (\text{cw}22, \text{cw}23, \{\text{loc}(\mathit{Patras})\}), \end{array} \right\} \quad (10)$$

A detailed planning on that basis refines the situation sequence afterwards.

$$S_{30\ \text{min}} = \left\{ \begin{array}{l} \ldots \\ (11.05\ 07{:}00, 11.05\ 12{:}30, \\ \{\text{loc}(\mathit{Europe}), \\ \text{transp}(\mathit{airplane})\}), \\ (11.05\ 12{:}30, 11.05\ 13{:}30, \\ \{\text{loc}(\mathit{Athens}), \\ \text{transp}(\mathit{metro})\}), \\ (11.05\ 13{:}30, 11.05\ 14{:}30, \\ \{\text{loc}(\mathit{Plaka}), \\ \text{activity}(\mathit{eating})\}), \\ (11.05\ 14{:}30, 11.05\ 16{:}00, \\ \{\text{loc}(\mathit{Acropolis})\}), \\ \ldots \end{array} \right\} \quad (11)$$

Such situational information (patterns, sequences) can be used to propose tasks to the user and in turn select appropriate content (from travel guides or the web) to prepare for that task or to accomplish it (this approach is investigated in the TALOS project [TAL08]). When viewed on different temporal granularity levels the visibility of features of situations change. For instance, viewing the "using the metro" situation from formula (11) on a *minutes* granularity would yield a situation sequence that shows a first short walk to the airport metro station, second, a ride with the Blue line (to Syntagma station), then changing to the Red line (to Acropolis station), and finally a short walk into the old town of Plaka.

Viewing the same sequence on a coarser scale hides this detailed information. That is, which features of the situations are relevant is just specified using a time parameter. On that basis the TALOS travel guide selects and presents different content to the user; when plans are rudimentary only general information on museums around are given; when plans get more concrete (going to the Acropolis by public transport) tasks and information get also more concrete (get the route to the museum, buy tickets for the metro, and so on). The idea is to select travel guide content of different levels of detail just by switching between different temporal granularities of the situation descriptions.

VI. Conclusion

Situation-based Services aim at capturing users' environment in order to provide them with the most appropriate information at a given time. Situations are multidimensional concepts where each dimension is defined according to a hierarchy of concepts. While rolling up along the hierarchies in order to consider aggregated concepts is a well understood

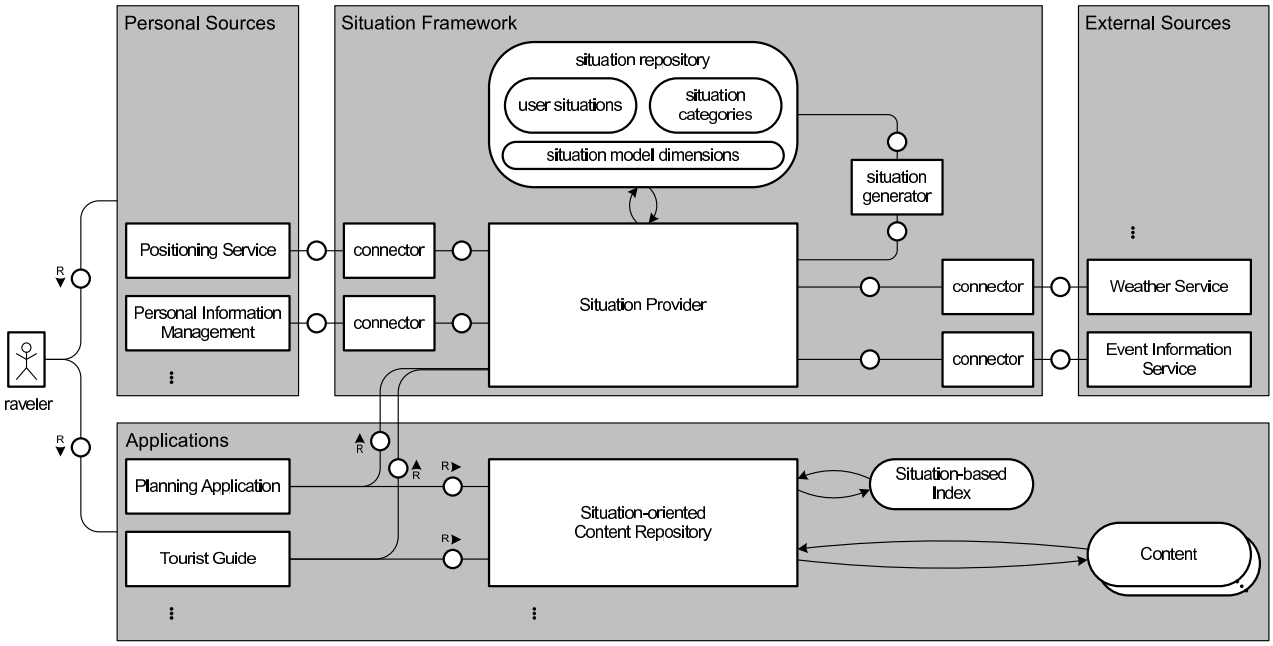

Fig. 7. General System Architecture.

Fig. 8. TALOS Screenshot Examples (with kind permission of Michael Müller Verlag, Erlangen, Germany).

operation, many problems arise when considering the time dimension. This is because time plays a particular role as being a part of the situation definition (through the intervals). When situations are derived from different sources of information, the problem of finding the appropriate temporal granularity and of combining the information arises. However, it is of major importance for targeting individual users (e.g., in a planning application) or groups of users (e.g., in an early warning system), either as a simple notification (like in alerting) or as a notification that the situations they planned differs from the situations that the system knows and are likely to occur. In this paper, we presented an advanced situation model together with operators that handle various granularities and which serves as a basis for the coarsening procedure that we detailed, both in a simple way and in a more efficient form. We finally presented our current architecture and illustrated our approach using examples from the tourism area. This approach was implemented in Java, tested in different projects on JavaSE and JavaME environments, and ported to Objective-C to support the iPhone platform. The situation model and operations has been implemented on a mySQL 5 database, using a star schema to model the multidimensional situation space. The approach presented in this paper uses the *cast* operator, which performs a precise mapping onto an exact starting point of a situation. However, in applications such as personal information management, this is not always realistic (e.g., a meeting around 9 a.m. may mean plus or minus 5 minutes and certainly not 9 o'clock by the second!). Therefore our future work includes investigating the use of the *round* operator together with the specification of a concept that would play an intermediary role between user-defined time (such as the one of a calendar entry) and granules.

Acknowledgments.

Part of this work was carried out in the framework of the European project TALOS.

References

[ADB+99] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In H.W. Gellersen, editor, *Handheld and Ubiquitous Computing: First International Symposium, HUC'99*, volume 1707 of *LNCS*, pages 304–307, Berlin/Heidelberg/New York, 1999. Springer Verlag.

[All84] J. F. Allen. Towards a general theory of action and time. *Artifical Intelligence*, 23(2):123–154, 1984.

[BD07] M. Baldauf and S. Dustdar. A survey on context-aware systems. *Intl. Journal of Ad Hoc and Ubiquitous Computing*, 2(4), 2007.

[BDE+98] C. Bettini, C. E. Dyreson, W. S. Evans, R. T. Snodgrass, and X. S. Wang. A glossary of time granularity concepts. In O. Etzion, S. Jajodia, and S. Sripada, editors, *Temporal databases: research and practice*, volume 1399 of *LNCS*, pages 406–413, Berlin/Heidelberg/New York, 1998. Springer Verlag.

[CC02] L. Chittaro and C. Combi. Temporal granularity and indeterminacy in reasoning about actions and change: an approach based on the event calculus. *Annals of Mathematics and Artificial Intelligence*, 36(1-2):81–119, nov 2002.

[CFP04] C. Combi, M. Franceschet, and A. Peron. Representing and reasoning about temporal granularities. *Journal of Logic and Computation*, 14(1):51–77, 2004.

[DELS00] C. E. Dyreson, W.S. Evans, H. Lin, and R. T. Snodgrass. Efficiently supporting temporal granularities. *IEEE Transactions on Knowledge and Data Engineering*, 12(4):568–587, Jul/Aug 2000.

[DSD06] C. Dorn, D. Schall, and S. Dustdar. Granular context in collaborative mobile environments. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *OTM Workshops (2)*, volume 4278 of *LNCS*, pages 1904–1913. Springer, 2006.

[Euz95] J. Euzenat. An algebraic approach to granularity in qualitative time and space representation. In *Proceedings of the 14$^t h$ International Joint Conference on Artificial Intelligence (IJCAI)*, pages 894–900. Morgan Kaufman, 1995.

[GLz+04] I. Goralwalla, Y. Leontiev, T. M. Özsu, D. Szafron, and Carlo C. Combi. Temporal granularity: Completing the puzzle. *Journal of Intelligent Information Systems*, 16(1):41–63, nov 2004.

[MGB+00] I. Merlo, G. Guerrini, E. Bertino, E. Ferrari, and S. Gadia. Querying multiple temporal granularity data. In *Proc. of the 7th Intl. Workshop on Temporal Representation and Reasoning (TIME'00)*, volume 00, page 103, Los Alamitos, CA, USA, 2000. IEEE Computer Society.

[MPVW04] U. Meissen, S. Pfennigschmidt, A. Voisard, and T. Wahnfried. Context- and situation-awareness in information logistics. In *Current Trends in Database Technology – EDBT 2004 Workshops*, volume 3268 of *LNCS*, pages 335–344, Berlin/Heidelberg/New York, 2004. Springer Verlag.

[MPVW05] U. Meissen, S. Pfennigschmidt, A. Voisard, and T. Wahnfried. Resolving knowledge discrepancies in situation-aware systems. *International Journal of Pervasive Computing and Communication JPCC*, 1(4):327–336, December 2005.

[Sch05] H. R. Schmidtke. Granularity as a parameter of context. In A. K. Dey, B. N. Kokinov, D. B. Leake, and R. M. Turner, editors, *Modeling and Using Context, Proc. of the 5th Intl. and Interdisciplinary Conference (CONTEXT 2005)*, volume 3554 of *LNCS*, pages 450–463, Berlin/Heidelberg/New York, 2005. Springer Verlag.

[TAL08] TALOS Consortium. TALOS project – task aware location based services for mobile environments. http://www.talos.cti.gr/, 2008.

[WJL91] G. Wiederhold, S. Jajodia, and W. Litwin. Dealing with granularity of time in temporal databases. In *Proc. of the 3rd Intl. Conf. on Advanced Information Systems Engineering (CAISE91)*, pages 124–140, Berlin/Heidelberg/New York, 1991. Springer Verlag.

Alg. 2 Coarsening a situation sequence.

COARSEN-SITUATION-SEQUENCE(
situation sequence S, precision p)

```
1  ▷ if S is the empty sequence there is nothing to do
2  if S = ∅
3     then
4          return S

5  T ← an array containing all time points of S
6  S_res ← ∅

7  t_last ← NIL
8  for i ← 1 to length[T]
9       do
10         ▷ compute the granule (lower bound t_lb
           and upper bound t_ub) for time t
11         t ← T[i]
12         t_lb ← t/p
13         t_lb ← t_lb · p
14         t_ub ← t_lb + p
15         if (t_last ≠ NIL) ∧ (t_last < t_lb)
16         ▷ the interval between the upper bound of the
           last granule (t_last) and the lower bound of
           the current (t_lb) forms a single situation
17           then P ← pattern of s at time t_last
18                if P ≠ ∅
19                ▷ create a new situation s and add it
                  to the resulting sequence
20                  then s ← (t_last, t_lb, P)
21                       S_res = S_res ∪ {s}

22         if t_last < t_ub
23           then P ← pattern of S at time t_lb
24                P' ← ∅
25                F ← the set of all single features of P
26                for j ← 1 to length[F]
27                     do f ← F[i]
28                        d ← the duration of feature f
                              in [t_lb, t_ub)
29                        if d > p/2
30                          then
31                               ▷ add feature f to the
                                 resulting pattern P'
32                               P' ← P' ∪ {f}
33                if P ≠ ∅
34                ▷ create a new situation s and add it
                  to the resulting sequence
35                  then s ← (t_lb, t_ub, P')
36                       S_res = S_res ∪ {s}
37                t_last ← t_ub

38 return COALESCE(S_res)
```