Journal of Universal Computer Science, vol. 9, no. 9 (2003), 1123-1136
submitted: 10/2/03, accepted: 9/6/03, appeared: 28/9/03 © J.UCS

Direct Granularity Conversions among Temporal Constraints

Claudio Bettini
(University of Milan, Italy
bettini@dico.unimi.it)

Simone Ruffini
(University of Milan, Italy
s.ruffini@infinito.it)

Abstract: This paper considers temporal constraints that can impose a minimum and maximum time distance between the occurrences of two events by specifying the minimum and maximum values in terms of a time granularity. When several constraints using different time granularities are part of the specification of a single problem, a reasonable question is how to convert the constraints in terms of a single granularity in order to apply standard temporal constraint algorithms. This paper investigates the problem of converting a distance constraint expressed in terms of a granularity into another one in terms of a different time granularity. An expressive formal model for time granularities is assumed including common granularities like hours and days as well as user-defined granularities like business days and academic semesters.

Keywords: temporal constraints, time granularity, temporal abstraction.

Category: I.2.4.

1 Introduction

There is a wide spectrum of applications that have to deal with temporal constraints, either for simply specifying a partial order of events, or for specifying more complex qualitative or quantitative time relations. Reactive systems, scheduling, planning, temporal data management, and workflows are just a few examples of the research areas where such applications can be found. Integrated e-commerce applications, in particular, are interesting examples where an advanced and efficient management of temporal aspects may be a significant advantage over the competition. In these complex systems the temporal relationships between events and activities involving different components are often described in terms of different time units. For example, the time between the receipt of an online order and the forwarding of item orders to the warehouses is described in minutes, while the time between the shipment of the order and the delivery is described in business days.

This paper considers a formalization of the notion of temporal constraint with granularity originally introduced in [Bettini et Al. 98a]. These constraints can be used to impose a minimum and maximum time distance between the occurrences of two events by specifying the minimum and maximum values in terms of a time granularity. For example, a constraint of the form $[1, 3]$ `b-day` between variables X and Y (representing event occurrence times) imposes that any assigment t_x and t_y for X and Y, respectively, is such that the distance between t_x and t_y is at least 1 business day and at most 3 business days. Note that t_x and t_y may actually denote a specific hour, minute or arbitrary finer granularity used when detecting the event occurrences, but the constraint satisfaction is checked by first identifying the business days containing t_x and t_y respectively and then checking their distance. The paper investigates the problem of converting this kind of constraints in terms of different granularities. This may be desirable for at least two reasons: a) there are well-known algorithms to process temporal constraint networks where all the constraints are in terms of the same granularity [Dechter et Al. 91], and b) it may be useful for the sake of application process monitoring to view all the constraints in terms of a particular granularity.

It will be immediately clear to the reader that the conversion problem is not a trivial one, even if we only consider common granularities like hours and days. For example, by saying that one event should occur the next day with respect to another one, we do not mean that the event should occur 24 hours after the other. Nor it is satisfactory to allow a range between 1 and 47 hours, as it may be suggested. Indeed 1 and 47 hours are actually the minimum and maximum distance between the occurrence of two events if they occur in consecutive days. However, a distance of 47 hours may also be the distance between two events whose occurrence is 2 days apart. This example actually says that an exact conversion does not exist, i.e., the constraint we obtain is not equivalent to the one we take as input.

Hence, our goal is, given a constraint in terms of a source granularity G, and given a target granularity H, to identify the constraint in terms of H which is the tightest among those implied by the input constraint or a good approximation of it. Being the tightest means that the set of distances allowed by the constraint is contained by any other logically implied constraint. If $[m, n]G$ represents the constraint imposing a distance of at least m and at most n time units of G, then in the above example $[1, 47]$`hour` is the tightest among the constraints in terms of `hour` implied by $[1, 1]$`day`. Indeed, other constraints, as for example, $[0, 48]$`hour` are logically implied by $[1, 1]$`day`, but none of them can exclude distances between 1 and 47.

The concept of granularity as an abstraction tool has been deeply investigated in the AI and DB literature probably starting from [Hobbs 85]. The formalization of time granularity as adopted in this paper has been defined in [Bettini et Al. 98a] and used in several papers on the subject. Interesting recent work on this topic can also be found in [Montanari 99, Dyreson et Al. 00, Goralwalla et Al. 01, Bettini et Al. 02a]. Symbolic formalisms to represent time granularities have been proposed in [Leban et Al. 86] and [Chandra et Al. 94] among others. The complex mathematical relations among granularities and calendars have been studied also in [Dershowitz and Reingold 90]. There are few approaches considering temporal constraints in terms of different granularities (e.g., [Dean 89, Goralwalla et Al. 01]), but in most cases they are restricted to common granularities. Our goal is to admit powerful constraints and very general user-defined granularities both to accurately model the new application domains and to select appropriate abstraction levels. A first comprehensive solution to this problem has been given in [Bettini et Al. 98a] proposing two alternative algorithms for constraint conversion. The main contribution of this paper with respect to [Bettini et Al. 98a] is the construction of a set of formulas based on granularity relationships enabling an efficient implementation of the direct conversion algorithm for a significant set of periodical granularities. The conversion algorithm presented here has a slightly different, theoretically equivalent, formulation regarding the treatment of minimal distances equal to zero, which was introduced to simplfy the implementation. Moreover, the implementation of the conversion algorithm proposed in this paper is supported by extensive experimentation with a prototype system developed at the University of Milan.

The conversion algorithm can be applied in several application areas where multi-granularity temporal constraints have been shown to be useful: data mining [Bettini et Al. 98b], workflow management [Bettini et Al. 02b], and clinical data management [Combi and Chittaro 99] among others.

The rest of the paper is organized as follows. In the next section we introduce the formal notions of granularity and temporal constraints with granularities. In Section 3 we illustrate a general algorithm for granularity conversion and we prove its correctness, while in Section 4 we show how the algorithm can be implemented exploiting the periodicity of granularities. In Section 5 we give an example of constraint network conversion, and Section 6 concludes the paper.

2 Temporal Constraint Networks with Granularities

We first define a granularity system, called $\mathcal{GGR}$ (General Granularities on Reals) in [Bettini et Al. 98a], as the set of granularities satisfying the following definition.

Definition 1. A *granularity* is a mapping G from the set of the integers to $2^{\mathcal{R}}$ (i.e., all subsets of reals) such that $G(i) = \emptyset$ for each non-positive integer i, and for all positive integers i and j with $i < j$, the following two conditions are satisfied:

- $G(i) \neq \emptyset$ and $G(j) \neq \emptyset$ imply that each real number in $G(i)$ is less than all real numbers in $G(j)$, and

– $G(i) = \emptyset$ implies $G(j) = \emptyset$.

This is a very general notion of granularity modeling standard ones like `hour`, `day`, `week` and `month` as well as more specific ones like `academic semester`, `b-day` (business day), or `b-week` (business week), and arbitrarily user-defined granularities. For example, `b-week` may be defined by the mapping of each positive integer (index) to a set of elements of the temporal domain denoting the period from a Monday through the next Friday every week. Each set of these sets of elements is called a *granule* of the granularity. If we decide to model time starting from Monday 2001/1/1, then the index 1 is mapped to the instants denoting the period from 2001/1/1 through 2001/1/5, forming the first granule `b-week`(1), the index 2 is mapped to `b-week`(2), denoting 2001/1/8 through 2001/1/12, and so on.

In order to have finite representations of the granularities suitable to be automatically manipulated, we further restrict the granularities to those whose granules can be defined as a periodical pattern with respect to the granules of a fixed bottom granularity. For this purpose we first need to introduce a granularity relationship.

Definition 2. A granularity G groups periodically into a granularity H if

1. for each non-empty granule $H(i)$, there exists a set of positive integers $\{j_0, \ldots, j_k\}$ such that $H(i) = \bigcup_{r=0}^{k} G(j_r)$, and
2. there exist $R, P \in \mathbb{Z}^+$, where R is less than the number of granules of H, such that for all $i \in \mathbb{Z}^+$, if $H(i) = \bigcup_{r=0}^{k} G(j_r)$ and $H(i+R) \neq \emptyset$, then $H(i+R) = \bigcup_{r=0}^{k} G(j_r + P)$.

Condition 1 says that any granule $H(i)$ is the union of some granules of G; for instance, assume it is the union of the granules $G(j_0), G(j_1), \ldots, G(j_k)$, where $j_0, \ldots, j_k$ are not necessarily contiguous positive integers. The periodicity property (condition 2) ensures that if the R^{th} granule *after* $H(i)$ exists (i.e., $H(i+R) \neq \emptyset$), then it is the union of $G(j_0 + P), G(j_1 + P), \ldots, G(j_k + P)$. This results in a periodic "pattern" of the composition of R granules of H in terms of granules of G. The pattern repeats along the time domain by "shifting" each granule of H by P granules of G. The integer P is called the *period*. Many common granularities are in this kind of relationship, for example, both days and months group periodically into years. In general, this relationship guarantees that granularity H can be finitely described providing the specification of granules of H in terms of granules of G in an arbitrary period and the period value. For example, `b-week` can be described by the five business days in the first week of the time domain, and by the value 7 for the period. A granularity G_0 is called *bottom* granularity if G_0 groups periodically into each other granularity in the system.

We can now define a temporal constraint with granularity.

Definition 3. Let $m, n \in \mathbb{Z} \cup \{-\infty, +\infty\}$ with $m \leq n$ and G a granularity. Then $[m, n]\, G$, called a *temporal constraint with granularity* (*TCG*), is the binary relation on positive integers defined as follows: For positive integers t_1 and t_2, (t_1, t_2) *satisfies* $[m, n]\, G$ if and only if (1) $\lceil t_1 \rceil^G$ and $\lceil t_2 \rceil^G$ are both defined, and (2) $m \leq (\lceil t_2 \rceil^G - \lceil t_1 \rceil^G) \leq n$.

The $\lceil x \rceil^G$ function returns the index of the granule of G that includes $G_0(x)$, where G_0 is the bottom granularity. Intuitively, to check if a pair of instants (t_1, t_2) satisfies the TCG $[m, n]\, G$, we derive the indexes of the granules of G containing t_1 and t_2, respectively, and then we take the difference. If it is at least m and at most n, then the pair of instants is said to satisfy the constraint. For example, the pair (t_1, t_2) satisfies $[0, 0]$ `day` if t_1 and t_2 are within the same day. Similarly, (t_1, t_2) satisfies $[-1, 1]$ `hour` if t_1 and t_2 are at most one hour apart (and their order is immaterial). Finally, (t_1, t_2) satisfies $[1, 1]$ `month` if t_2 is in the next month with respect to t_1.

Definition 4. A *constraint network (with granularities)* is a directed graph denoted by (W, A, Γ, Dom), where W is a finite set of variables, $A \subseteq W \times W$ a set of arcs, Γ is a mapping from A to the finite sets of temporal constraints with granularities, and Dom is a mapping from W to a possibly bounded periodical set of positive integers.

A set of positive integers S is said to be *periodical* if there exists a granularity G such that $S = \{i \mid \lceil i \rceil^G \text{ is defined}\}$. The set is bounded if an integer U is given such that each value in the set must be less than or equal to U.

Intuitively, a constraint network specifies a complex temporal relationship where each variable in W represents a specific instant (for example the occurrence time of an event) in terms of the bottom granularity. The domain of each variable is essentially a set of granules of the bottom granularity, represented through the positive integers corresponding to the granule indexes. The set of TCGs assigned to an edge is taken as conjunction. That is, for each TCG in the set assigned to the edge (X, Y), the instants assigned to X and Y must satisfy the TCG. Fig. 1 shows an example of a constraint network with granularities with no explicit constraint on domains ($Dom(X) = [1, \infty)$ for each variable X).

3 Conversion of Constraints in Different Granularities

As we have pointed out in the introduction, in general, given a TCG, it does not exists an equivalent one in terms of a different granularity. However, we are interested in converting a given TCG_1 in terms of G_1 into a logically implied TCG_2 in terms of G_2. Note that a TCG A *logically implies* a TCG B if any

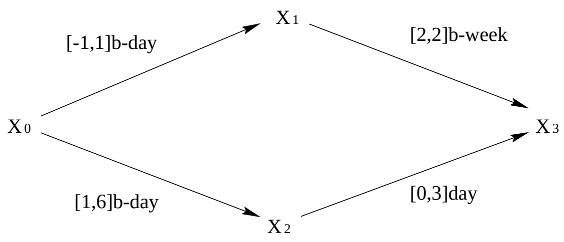

Figure 1: A constraint network with granularities

pair of time instants satisfying A also satisfy B. Among all the logically implied TCGs we prefer the tightest or a good approximation of it. For example, both $[1, 47]$ `hour` and $[0, 50]$ `hour` are logically implied by $[1, 1]$ `day`, with the first being preferable since, in this case it is the tightest, and, indeed, it provides a more precise conversion of the original one. If we only have a total order of granularities with uniform granules, like e.g., `minute`, `hour`, and `day`, then the conversion algorithm is trivial since fixed conversion factors can be used. However, if incomparable granularities like `week` and `month`, or granularities with non-contiguous granules like `b-day` and `b-week` are considered, the conversion becomes more complex.

Moreover, given an arbitrary TCG_1, and a granularity G, it is not always possible to find a logically implied TCG_2 in terms of G. For example, $[0, 0]$ `day` does not logically imply $[m, n]$ `b-day` no matter what m and n are. The reason is that $[0, 0]$`day` is satisfied by any two events that happen during the same day, whether the day is a business day or a weekend day.

3.1 Allowed conversions

In our framework, we allow the conversion of a TCG of a constraint network $\mathcal{N}$ into another TCG if the resulting constraint is implied by the set of all the TCGs in $\mathcal{N}$. More specifically, a TCG $[m, n]\, G$ on arc (X, Y) in a network $\mathcal{N}$ is *allowed* to be converted into $[m', n']\, H$ as long as $[m', n']\, H$ is *implied* by $\mathcal{N}$, i.e., for any pair of values t_X and t_Y assigned to X and Y respectively, if (t_X, t_Y) satisfies $[m, n]\, G$ and t_X and t_Y belong to a solution of $\mathcal{N}$, then (t_X, t_Y) also satisfies $[m', n']\, H$. To guarantee that only allowed conversions are performed, we assign to each variable X of a constraint network, a periodical set G_X which is the intersection of the domain of X and all the periodical sets defined by the granularities appearing in TCGs involving X. Then, a TCG on variables X and Y can be converted in terms of a target granularity H if each element in $G_X \cup G_Y$ is contained in a granule of H.

This condition guarantees the existence of an *allowed* conversion; Indeed, if a value t_X is assigned to X in a solution of the network, then it must be in G_X.

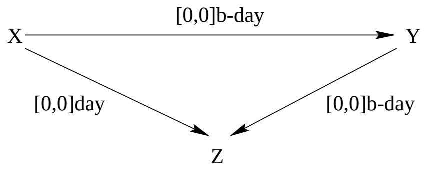

Figure 2: A network to illustrate conversion conditions

Similarly, if t_Y is assigned to Y, $t_Y \in G_Y$. Then, both t_X and t_Y are in $G_X \cup G_Y$, and hence each one of them is contained in a (possibly different) granule of H. Thus, their distance in terms of H can be evaluated.

Example 1. Consider the constraint network in Fig. 2. Both G_X and G_Z are `b-day`. Hence, the constraint $[0, 0]$ `day` can be converted in terms of `b-day` since the target granularity `b-day` covers a span of time equal to that covered by G_X and G_Z (i.e., $G_{XZ} \subseteq \mathtt{b-day}'$, where $\mathtt{b-day}'$ is the periodical set corresponding to the granularity $\mathtt{b-day}$). Similarly, consider the network in Fig. 1. The TCG in terms of `day` cannot be converted in terms of `b-day` or `b-week` without considering the other constraints in the network. However, both G_{X_2} and G_{X_3}, i.e., the periodical sets obtained as the intersection of the domain of X_2 (resp. X_3) and all the periodical sets defined by the granularities appearing in TCGs involving X_2 (resp. X_3), can be obtained from `day` by dropping all granules that are not a business day. Since both `b-day` and `b-week` cover the same span of time as this granularity, the constraint $[0, 3]$ `day` can be converted in terms of `b-day` and `b-week`, obtaining $[0, 3]$ `b-day` and $[0, 1]$ `b-week`, respectively.

3.2 The conversion algorithm

In Fig. 3 we propose a general conversion method. It is based on the functions $mindist()$ and $maxdist()$. Intuitively, $mindist(G_1, m, G_2)$ denotes the minimal distance (in terms of the number of granules of G_2) between all pairs of instants in a granule of G_1 and in the mth granule after it, respectively. For example, $mindist(\mathtt{b-week}, 1, \mathtt{day}) = 3$, i.e., the minimum distance in terms of days between two events that occur in two different business weeks is 3 (one instant on Friday and the other on Monday). The definition restricts the considered pairs of instants to those in which the first instant is included in G_X and the second in G_Y. Indeed, only these pairs are candidate solutions for the constraint on the arc (X, Y), and this restriction improves the precision of the conversion as well as it detects some inconsistencies. The value $maxdist(G_1, n, G_2)$ is the corresponding maximum distance. For example, $maxdist(\mathtt{b-week}, 1, \mathtt{day}) = 11$, and $maxdist$(`b-day`, 1,`day`)= 3.

INPUT: a network $\mathcal{N}$ with a TCG $[m,n]\,G_1$, with $m \in \mathbb{Z} \cup \{-\infty\}$ and $n \in \mathbb{Z}^+ \cup \{+\infty\}$, associated with an arc (X,Y); a target granularity G_2, s.t. $\forall t\ (t \in G_X \cup G_Y \Rightarrow \exists j\ \lceil t \rceil^{G_2} = j)$.

OUTPUT: a logically implied TCG $[\overline{m},\overline{n}]G_2$ for (X,Y) or **undefined**.

METHOD:

Step 1 **if** $m = -\infty$ **then** $\overline{m} = -\infty$
else if $m > 0$ **then** $\overline{m} = mindist(G_1, m, G_2)$
else $\overline{m} = -maxdist(G_1, |m|, G_2)$

Step 2 **if** $n = +\infty$ **then** $\overline{n} = +\infty$
else $\overline{n} = maxdist(G_1, n, G_2)$

Step 3 **if** either $\overline{m}$ or $\overline{n}$ is **undefined**, **then return undefined**
else return $[\overline{m},\overline{n}]\,G_2$

where
$mindist(G_1, m, G_2) = min(S)$ if $S \neq \emptyset$, **undefined** otherwise, where
$S = \{\lceil t_2 \rceil^{G_2} - \lceil t_1 \rceil^{G_2} \mid t_1 \in G_X, t_2 \in G_Y, \text{ and } \lceil t_2 \rceil^{G_1} - \lceil t_1 \rceil^{G_1} \geq m\}$;
$maxdist(G_1, n, G_2) = max(R)$ if $R \neq \emptyset$, **undefined** otherwise, where
$R = \{\lceil t_2 \rceil^{G_2} - \lceil t_1 \rceil^{G_2} \mid t_1 \in G_X, t_2 \in G_Y, \text{ and } \lceil t_2 \rceil^{G_1} - \lceil t_1 \rceil^{G_1} \leq n\}$;
$G_X = \bigcap(Dom(X), H_1, \ldots, H_k)$ if $H_1, \ldots, H_k$ are the periodical sets corresponding to the granularities in the TCGs of $\mathcal{N}$ involving node X. (Similarly for G_Y.)

Figure 3: A general method for the conversion of constraints

The values of these functions cannot be automatically obtained for general (infinite) granularities, however, they can be computed efficiently when the involved granularities are periodic, as in most practical cases. The computation of $mindist()$ and $maxdist()$ is more involved when we want to check the condition, appearing in their specification, i.e., $t_1 \in G_X$ and $t_2 \in G_Y$. Note that omitting this check leads to a still sound but less precise conversion. The other condition involving G_X and G_Y appears in the input description. This cannot be ignored since it guarantees an allowed conversion.

The method imposes $n \in \mathbb{Z}^+$ if $n \neq +\infty$ for the input constraint. This is not a limitation since any constraint $[-n,-m]\,G$ on (X,Y) with $m,n > 0$ can be expressed as $[m,n]\,G$ on (Y,X). When only the lower bound is negative, it is sufficient to consider its absolute value and treat it exactly as the upper bound except for reversing the sign of the result. For example, for $[-1,1]$ `week` we derive the upper bound 13 for `day` according to the method. Since the absolute value for -1 is 1, we derive $[-13, 13]$ `day` as the implied constraint. The case of $m = 0$ must be treated exactly as for negative values.

When $m = -\infty$ and/or $n = +\infty$ the conversion method simply keeps these constants also for the TCG in the target granularity. It is easily seen that this always leads to an implied TCG. However, when G_1 is bounded, the ∞ constants may be converted more precisely into finite values. For example, $[1, +\infty]$ `month-b2000`, where `month-b2000` denotes the months before year 2000, is converted into $[0, +\infty]$ `year`, while $+\infty$ in this last TCG may be safely substituted with the number of years from the beginning of the time line and year 2000. This refinement is left as an optimization which may be useful in implementing the method.

Theorem 5. *The general conversion method is correct: Any constraint obtained as output is implied by the original network.*

Proof. Suppose, by contradiction, that the derived TCG $[\overline{m}, \overline{n}]G_2$ on (X, Y) is not implied by the network $\mathcal{N}$. It follows that there exist two values x and y that, assigned to X and Y, respectively, belong to a solution of the given network, satisfying the constraint $[m, n]\, G_1$ between X and Y, but not satisfying the constraint $[\overline{m}, \overline{n}]\, G_2$ between the same variables. By definition, if the constraint is not satisfied, one of the following must hold: (a) $\lceil x \rceil^{G_2}$ or $\lceil y \rceil^{G_2}$ is not defined, (b) $\lceil y \rceil^{G_2} - \lceil x \rceil^{G_2} > \overline{n}$, (c) $\lceil y \rceil^{G_2} - \lceil x \rceil^{G_2} < \overline{m}$.
Suppose (a) holds and $\lceil x \rceil^{G_2}$ is not defined. From the definition of $\lceil \rceil$, this means that there does not exist j such that $x \in G_2(j)$. However, since x is part of a solution of the given network, $x \in G_X(k)$ for some positive integer k. Then, the condition $\forall i, t\ (t \in G_X(i) \cup G_Y(i) \Rightarrow \exists j\ t \in G_2(j))$ imposed by the algorithm on its input, guarantees $\exists j\ x \in G_2(j)$, leading to a contradiction. The same argument applies to $\lceil y \rceil^{G_2}$.
Suppose (b) holds. Let $\overline{n}' = \lceil y \rceil^{G_2} - \lceil x \rceil^{G_2}$. Hence, $\overline{n}' > \overline{n}$. From the computation of $\overline{n}$ by the algorithm, we have either $\overline{n} = +\infty$, which would immediately contradict (b), or $\overline{n} = max(R)$, where $R = \{r \mid \exists t_1, t_2,\ \lceil t_1 \rceil^{G_X}$ and $\lceil t_2 \rceil^{G_Y}$ are both defined, $\lceil t_2 \rceil^{G_1} - \lceil t_1 \rceil^{G_1} \leq n$ and $\lceil t_2 \rceil^{G_2} - \lceil t_1 \rceil^{G_2} = r\}\}$. Let $t_1 = x$ and $t_2 = y$. Clearly, $\lceil x \rceil^{G_X}$ and $\lceil y \rceil^{G_Y}$ are both defined since x and y are part of a solution. $\lceil y \rceil^{G_1} - \lceil x \rceil^{G_1} \leq n$ since (x, y) satisfies the TCG $[m, n]G_1$, and $\lceil y \rceil^{G_2} - \lceil x \rceil^{G_2} = \overline{n}'$ by hypothesis. Then, $\overline{n}' \in R$ and hence, $\overline{n}' \leq \overline{n} = max(R)$. This is a contradiction since we assumed $\overline{n}' > \overline{n}$.
Finally, suppose (c) holds. Let $\overline{m}' = \lceil y \rceil^{G_2} - \lceil x \rceil^{G_2}$. Hence, $\overline{m}' < \overline{m}$. We first consider the case when $m \geq 0$. From the computation of $\overline{m}$ by the algorithm, we have $\overline{m}' < min(S)$, where $S = \{s \mid \exists t_1, t_2,\ \lceil t_1 \rceil^{G_X}$ and $\lceil t_2 \rceil^{G_Y}$ are both defined, $\lceil t_2 \rceil^{G_1} - \lceil t_1 \rceil^{G_1} \geq m$ and $s = \lceil t_2 \rceil^{G_2} - \lceil t_1 \rceil^{G_2}\}\}$. Let $t_1 = x$ and $t_2 = y$. Clearly, $\lceil x \rceil^{G_X}$ and $\lceil y \rceil^{G_Y}$ are both defined since x and y are part

of a solution. $\lceil y \rceil^{G_1} - \lceil x \rceil^{G_1} \geq m$ since (x, y) satisfies the TCG $[m, n]G_1$, and $\overline{m}' = \lceil y \rceil^{G_2} - \lceil x \rceil^{G_2}$ by hypothesis. Then, $\overline{m}' \in S$ and hence, $\overline{m}' \geq \overline{m} = min(S)$. This is a contradiction since we assumed $\overline{m}' < \overline{m}$. When $m < 0$, either $\overline{m}' = -\infty$, which would immediately contradict (c), or the fact that $\overline{m}' < \overline{m}$ can be interpreted as the distance between x and y in terms of granules of G_2 being greater (in absolute value) than the bound given by the algorithm. The violation of the bound and the corresponding proof are, in this case, equivalent to point (b) above, since the values of x and y can be exchanged, reversing the sign of the bounds. The conversion algorithm, indeed, treats the negative lower bound as a positive upper bound, reversing the sign of the result.

4 Implementation

In this section, we show how the conversion algorithm can be implemented. Essentially, this can be done by exploiting the periodicity of granularities and variable domains. First of all, we need to ensure that the conversion is allowed, as required by the condition specified on the INPUT in Fig. 3. Secondly, we need to compute the values of the distance functions $mindist()$ and $maxdist()$ for arbitrary parameters.

4.1 Excluding illegal conversions

From Section 3, we know that the conversion of a constraint on the distance from X to Y with target granularity G_2 is allowed if each element in $G_X \cup G_Y$ is in G_2. If we consider G_2' as the periodical set induced by G_2, it is easily seen that this condition holds if $G_X \cap G_2' \equiv G_X$ and $G_Y \cap G_2' \equiv G_Y$. Intersection of periodical sets can be easily implemented by considering their common period (see [Bettini et Al. 02a]), and similarly can be implemented equivalence.

4.2 Computation of distance functions

The computation of $mindist()$ and $maxdist()$ following the formulas in their definition may be computationally very inefficient. Our choice is to devise an efficient computation method which return values equivalent to $min(S)$ and $max(R)$ respectively, assuming that the conditions $t_1 \in G_X$, $t_2 \in G_Y$, appearing in the definitions of the distance functions, are always satisfied. Note that, if this is not the case, we derive a constraint which is looser than the optimal constraint, but still logically implied by the input constraint network. Based on extensive experiments we have seen that the resulting approximation is very good, and hence the trade-off between efficiency and precision favors our solution, except for special critical applications.

Origin	Target	Computation of mindist(), maxdist()
day	hour	mindist(day, k, hour) $= 24 * (k-1) + 1$ maxdist(day, k, hour) $= 24 * (k+1) - 1$
week	hour	mindist(week, k, hour) $= 168 * (k-1) + 1$ maxdist(week, k, hour) $= 168 * (k+1) - 1$
bday	hour	mindist(bday, k, hour) $= 24 * (k-1) + 48 * (k \text{ DIV } 5) + 1$ maxdist(bday, k, hour) $= 24 * (k+1) + 48 * ((k+4) \text{ DIV } 5) - 1$
bweek	hour	mindist(bweek, k, hour) $= 168 * (k-1) + 49$ maxdist(bweek, k, hour) $= 168 * (k+1) - 49$
day	bday	mindist(day, k, bday) = **if** $(k+5)$ MOD $7 \neq 0$ **then** $k - ((k+4) \text{ DIV } 7) * 2$ **else** $k - 1 - ((k+4) \text{ DIV } 7) * 2$ maxdist(day, k, bday) = **if** $(k+2)$ MOD $7 \neq 0$ **then** $k - ((k+2) \text{ DIV } 7) * 2$ **else** $k + 1 - ((k+2) \text{ DIV } 7) * 2$
day	bweek	mindist(day, k, bweek) $= (k+2)$ DIV 7 maxdist(day, k, bweek) $= (k+4)$ DIV 7
week	day	mindist(week, k, day) $= 7 * (k-1) + 1$ maxdist(week, k, day) $= 7 * k + 6$
day	week	mindist(day, k, week) $= k$ DIV 7 maxdist(day, k, week) $= (k+6)$ DIV 7
year	day	mindist(year, k, day) $= 365 * (k-1) + 1 + ((k-1) \text{ DIV } 4)$ maxdist(year, k, day) $= 365 * (k+1) + (k \text{ DIV } 4)$
year	month	mindist(year, k, month) $= 12 * (k-1) + 1$ maxdist(year, k, month) $= 12 * (k+1) - 1$
month	year	mindist(month, k, year) $= k$ DIV 12 maxdist(month, k, year) $= (k+11)$ DIV 12
bweek	bday	mindist(bweek, k, bday) $= 5 * (k-1) + 1$ maxdist(bweek, k, bday) $= 5 * k + 4$
bday	bweek	mindist(bday, k, bweek) $= k$ DIV 5 maxdist(bday, k, bweek) $= (k+4)$ DIV 5
bweek	week	mindist(bweek, k, week) $= k$ maxdist(bweek, k, week) $= k$
bday	day	mindist(bday, k, day) $= k + (2 * (k \text{ DIV } 5))$ maxdist(bday, k, day) $= k + (2 * ((k+4) \text{ DIV } 5))$
bweek	day	mindist(bweek, k, day) $= 7 * (k-1) + 3$ maxdist(bweek, k, day) $= 7 * k + 4$

Table 1: Computation of $mindist()$ and $maxdist()$

The method we have implemented is based on a table lookup for each pair of source and target granularities. Each table entry contains a formula for $mindist(k)$ and $maxdist(k)$ having k as the only variable. Intuitively, each formula is based on the relationship between the structures of the two granularities. In some cases the formulas are trivial, but this is not true in general. For example, in order to derive the formula $mindist(month, k, year)$, it is sufficient to consider that each year is made of 12 months; indeed that formula is k DIV 12.

Less intuitive is how to evaluate $mindist(bday, k, hour)$, since we have to take into account noncontiguous granules. In Table 1 we report some of the formulas

Origin	Target	Computation of mindist(), maxdist()
month	day	mindist(month, k, day) $= mindist(month, k \text{ MOD } 12, day)+$ $365 * (k \text{ DIV } 12) + ((k + 10) \text{ DIV } 48)$ maxdist(month, k, day) $= maxdist(month, k \text{ MOD } 12, day)+$ $365 * (k \text{ DIV } 12) + ((k + 37) \text{ DIV } 48)$

Table 2: Computation of $mindist(month, k, day)$ and $maxdist(month, k, day)$

involving a quite rich set of granularities. Note that, according to the algorithm, the values of $mindist()$ need to be computed only for $k \geq 1$; Hence, for example, our formula $mindist(year, k, day)$ does not give a correct value for $k = 0$ (it gives -364 while the value, considering leap years, should be -365), but the algorithm would correctly compute the desired value, since $-maxdist(year, 0, day)$ would be used.

There are pairs of source and target granularities for which is particularly complex to derive a direct formula. This happens in particular when the granules of the source granularity have different sizes in terms of the target granularity, as for example, in the computation of $mindist(month, k, day)$. Our choice in this case is to precompute the value of the distance functions for some values of k (less than the number of granules of the source granularity in the common period of the two granularities). For example, in order to compute $mindist(month, k, day)$ and $maxdist(month, k, day)$, we precompute them for $k = 0, \ldots, 11$ based on their definition, and obtain all other values by the formulas in Table 2.

A library of formulas for common granularities can definitely speed up the conversion process. For cases in which formulas do not exist yet, and are not trivial to derive, we may consider an alternative conversion algorithm based on the relationship of each of the source and target granularities with the bottom granularity. A preliminary version of such an algorithm was illustrated in [Bettini et Al. 98a]. However, that method introduces an additional source of approximation.

5 Application of direct conversions

Given a constraint network, we can derive a logically implied network in terms of a target granularity H by applying the conversion algorithm to each constraint for which a conversion into H is allowed. The following example shows the derivation of an implied network following the steps of the algorithm and the implementation techniques illustrated in the previous section.

Example 2. Suppose we want to derive from the network in Fig. 1 an implied network in terms of business days. This means the constraints on arcs (X_1, X_3)

and (X_2, X_3) must be converted. Considering $[2, 2]$ `b-week`, the lower bound is 2, hence we must compute $mindist(\texttt{b-week}, 2, \texttt{b-day})$. Looking at Table 1, this value is $5*(2-1)+1 = 6$. For the upper bound, we have $maxdist(\texttt{b-week}, 2, \texttt{b-day}) = 5*2+4 = 14$. Accordingly to the algorithm, we obtain the new TCG $[6, 14]$ `b-day`. In Example 1, we have seen that, considering the whole network, the TCG on arc (X_2, X_3) can be converted in terms of `b-day` and `b-week`, despite its granularity is `day`. Its conversion accordingly to the algorithm and Table 1 is $[0, 3]$ `b-day`, and the resulting implied network in terms of `b-day` is shown in Fig. 4.

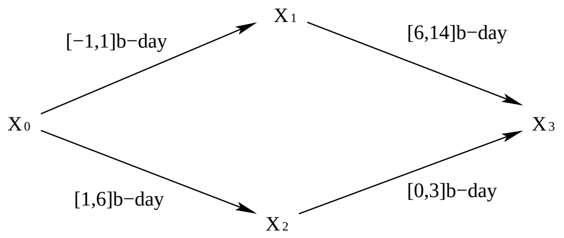

Figure 4: Implied network in terms of `b-day`

Similarly, we can obtain an implied network in terms of `b-week` shown in Fig. 5. In this case, the constraints to be converted are those on arcs (X_0, X_1), (X_0, X_2) and (X_2, X_3). Accordingly to the algorithm, we obtain $[-1, 1]$ `b-week`, $[0, 2]$ `b-week`, and $[0, 1]$ `b-week`, respectively.

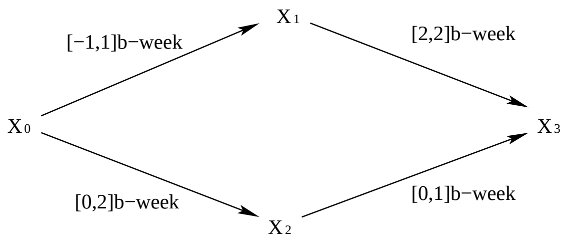

Figure 5: Implied network in terms of `b-week`

6 Conclusion

In this paper we proposed an algorithm and implementation techniques for the conversion of temporal constraints in terms of different granularities. By applying the algorithm, it is possible to derive a new constraint network in terms of a single target granularity which is implied by the original set of constraints. The conversion algorithm can also be integrated with arc and path consistency techniques for constraint propagation [Dechter et Al. 91, Bettini et Al. 02a], leading to the derivation of implicit constraints, and to the refinement of existing ones.

Acknowledgments

This work has been partially supported by Italian MIUR (FIRB "Web-Minds" project).

References

[Bettini et Al. 02a] C. Bettini, X. Wang, S. Jajodia, "Solving Multi-Granularity constraint networks"; AIJ (Artificial Intelligence Journal), 140, 1-2 (2002), 107–152.

[Bettini et Al. 02b] C. Bettini, X. S. Wang, S. Jajodia, "Temporal Reasoning for Supporting Workflow Systems"; DPD (Distributed and Parallel Databases), 11, 3 (2002), 269-306.

[Bettini et Al. 98a] C. Bettini, X. Wang, S. Jajodia, "A general framework for time granularity and its application to temporal reasoning"; AMAI (Annals of Mathematics and Artificial Intelligence), 22, 1-2 (1998), 29–58.

[Bettini et Al. 98b] C. Bettini, X. Wang, J. Lin, S. Jajodia, "Discovering Frequent Event Patterns With Multiple Granularities in Time Sequences"; TKDE (IEEE Transactions on Knowledge and Data Engineering), 10, 2 (1998), 222–237.

[Chandra et Al. 94] R. Chandra, A. Segev, and M. Stonebraker, "Implementing calendars and temporal rules in next generation databases"; Proc. ICDE'94 (International Conference on Data Engineering), (1994), 264–273.

[Combi and Chittaro 99] C. Combi, L. Chittaro, "Abstraction on clinical data sequences: an object-oriented data model and a query language based on the event calculus"; AIM (Artificial Intelligence in Medicine), 17, 3 (1999), 271–301.

[Dean 89] T. Dean, "Using temporal hierarchies to efficiently maintain large temporal databases"; JACM (Journal of the ACM), 36, 4 (1989), 687–718.

[Dechter et Al. 91] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks"; AIJ (Artificial Intelligence Journal), 49, (1991), 61–95.

[Dershowitz and Reingold 90] Nachum Dershowitz, Edward M. Reingold, "Calendrical Calculations"; (Software - Practice and Experience), 20, 9 (1990), 899–928.

[Dyreson et Al. 00] Curtis E. Dyreson, William S. Evans, Hong Lin, Richard T. Snodgrass, "Efficiently Supported Temporal Granularities"; TKDE (IEEE Transactions on Knowledge and Data Engineering), 12, 4 (2000), 568–587.

[Goralwalla et Al. 01] I.A. Goralwalla, Y. Leontiev, M.T. Ozsu, D. Szafron, C. Combi, "Temporal Granularity: Completing the Puzzle"; JIIS (Journal of Intelligent Information Systems), 16, 1 (2001), 41-63.

[Hobbs 85] J.R. Hobbs, "Granularity"; Proc. of IJCAI'85 (International Joint Conference on Artificial Intelligence), Los Angeles (1985), 432–435.

[Leban et Al. 86] B. Leban, D. Mcdonald, and D. Foster, "A representation for collections of temporal intervals"; Proc. of AAAI'86 (National Conference on Artificial Intelligence), (1986), 367–371.

[Montanari 99] A. Montanari, A. Peron, A. Policriti, "Theories of Omega-Layered Metric Temporal Structures: Expressiveness and Decidability"; (The Logic Journal of IGPL), 7, 1 (1999), 79–102.