Representing and Integrating Multiple Calendars *

Sarit Kraus† **Yehoshua Sagiv**‡ **V.S. Subrahmanian** §

Abstract

Whenever humans refer to time, they do so with respect to a specific underlying calendar. So do most software applications. However, most theoretical models of time refer to time with respect to the integers (or reals). Thus, there is a mismatch between the theory and the application of temporal reasoning.

To lessen this gap, we propose a formal, theoretical definition of a calendar and show how one may specify dates, time points, time intervals, as well as sets of time points, in terms of constraints with respect to a given calendar. Furthermore, when multiple applications using different calendars wish to work together, there is a need to integrate those calendars together into a single, unified calendar. We show how this can be done.

---

*This work was supported by the Army Research Office under Grants DAAH-04-95-10174 and DAAH-04-96-10297, by ARPA/Rome Labs contract F30602-93-C-0241 (ARPA Order Nr. A716), by an NSF Young Investigator award IRI-93-57756, NSF Grant No. IRI-9423967 and by the Army Research Laboratory under Cooperative Agreement No. DAAL01-96-2-0002.

†Dept. of Mathematics and Computer Science, Bar-Ilan University, Ramat Gan, 52900 Israel, and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 E-Mail: sarit@cs.biu.ac.il

‡Dept. of Computer Science, Ross Building Hebrew University, Givat Ram, Jerusalem, Israel. E-Mail: sagiv@cs.huji.ac.il

§Dept. of Computer Science, Institute for Advanced Computer Studies, Institute for Systems Research, University of Maryland, College Park, MD 20742, USA. E-Mail: vs@cs.umd.edu

Contents

1 Introduction

The need to *represent* and *reason* with time is critical to the success of several real life problems, both in the artificial intelligence arena, as well as in the database arena. For example, AI planning and scheduling systems often need to check whether the state of the world satisfies some condition at *time t* and may just as often assert that some action must be performed *at time t*. Such a planning system may, for instance, be required to execute the action `notify_fire_dept` within 5 minutes of a fire alarm going off provided that a cursory 2-minute check by its sensors reveals no system malfunction. Similarly, a personalized intelligent agent may be provided with the following instruction: Remind Mary on the first Sunday of each month that she needs to pay the phone bill.

To date, most formal work in this area has assumed that a time point is either a natural number or a real number. However, it is extremely cumbersome and unnatural to represent time as a single point on some time-scale. Most human beings would have a hard time expressing the 15th of March, 1994, as the time point 2,777,802,116 or something equally confusing. One reason why human beings have no problem reasoning with time is because they always refer to a particular *calendar* when discussing time. A formal framework for expressing and manipulating calendars would allow AI/DB systems to succinctly and intelligibly express the results of their temporal computations to a human being. At the same time, the human being could represent his/her needs to the system using a calendar that he/she is familiar with.

Now, consider another scenario where two intelligent agents, one a planning agent, and another, a database agent, are cooperating in the solution of some problem. The database agent may provide a facility whereby any other agent may ask queries using a single query language (e.g. SQL). When answering a query, the database agent may not access just one database, but rather access a variety of relational, object oriented, geographic, terrain, and other databases. The planning agent, on the other hand, may have (as usual) a set of planning operators—these could be in the form of STRIPS or HTN operators. At various points in time, the planning agent tries to see if certain conditions are true in the battlefield. Depending on which of those conditions is true, the planning agent infers a goal to be accomplished and constructs a plan to accomplish it, if possible.

Suppose the planning agent asks the database agent a query of the form: *Find all tuples in the relation* `troop_movements` *showing movements of Iraqi troops towards the Kurdish lines during the interval 1000 hours on August 5, 1996, and 1800 hours, August 9, 1996.* If we ignore the non-temporal components of this query, we notice that the notion of time used by the planning agent in its query refers to the concepts of `hours` (of a day), `days`, `months` and `years`. For the database agent to successfully answer the query, it must be able to comprehend the temporal part of the planning agent's query.

However, this problem may be compounded if the database agent uses a different granularity to represent time. For example, the database agent may specify time in terms of `hours` (of a month), `months` and `years`. For the database agent to understand the query, it must be able to map the temporal component of the query (expressed in the time granularity used by the planning agent) to the representation of time that it uses internally.

In each of the above cases, two salient features emerge:

1. Each agent involved is specifying time according to a different *calendar*. Thus, techniques are needed to represent multiple calendars. This paper provides a framework for representing widely varying types of calendars.

    - We show how constraints may be used to represented calendars (these include not only the standard Gregorian Calendar, but others, such as the Hebrew calendar, as well),
    - We provide a formal mathematical definition of what it means for two calendars to be *compatible,* and
    - We provide algorithms to *manipulate* calendars by manipulating the constraints that *represent them.*

2. When two agents need to communicate with each other about time, they need to somehow ensure that time specified according to the first agent's calendar "makes sense" according to the second agent's calendar. Instead of using ad-hoc means of converting time-points in one calendar to time-points in another calendar, we will show how two calendars (represented in our framework) may be neatly combined using a very general and robust technique. Each of the agents involved in the communication may then understand what the other is saying by examining this "combined calendar". Furthermore, the construction of this combined calendar, as well as techniques for agent A_1 to "understand" what agent A_2 is saying, are also automated.

2 Preliminaries

Definition 2.1 (time unit) A *time unit* consists of a *name* and a *time-value set.* The time-value set has a linear order, denoted $<_T$, where T is the name of the time unit. The time unit is either *finite* or *infinite,* depending on whether the time-value set is finite or infinite; an infinite time-value set is assumed to be countable. For simplicity, we assume that a time unit is uniquely identified by its name; that is, there are no distinct time units that have the same name, but different time-value sets. □

Day, month, and year are all examples of time units. Formally, we use the names *day*, *month* and *year* for these time units. The time-value set of *day* consists of the integers 1 through 31. Similarly, the time-value set of *month* consists of the 12 months of the year, i.e., *January*, *February*, etc. For *year*, the time-value set is infinite and consists of all integers.

Definition 2.2 (linear hierarchy of time units) A *linear hierarchy of time units* is a finite collection of distinct time units with a linear order $\sqsubset$ among those time units. The greatest time unit according to $\sqsubset$ is called the *top* of the hierarchy and it must be an infinite time unit, while all other time units in the hierarchy must be finite. $\square$

If we define the order $day \sqsubset month \sqsubset year$ on the time units *day*, *month* and *year*, we get a linear hierarchy of these time units.

Definition 2.3 (time point of a linear hierarchy) Suppose $T_1 \sqsubset \cdots \sqsubset T_n$ is a linear hierarchy H of time units. A *time point* is an n-tuple $(t_1, \ldots, t_n)$, such that t_i is a value of the time-value set of T_i $(1 \leq i \leq n)$. Time points are ordered according to the lexicographic ordering $<_H$, which is defined in the usual way; that is, $(t_1, \ldots, t_n) <_H (s_1, \ldots, s_n)$ if there is an i $(1 \leq i \leq n)$, such that $t_i <_{T_i} s_i$ and $t_j = s_j$ for $j = i+1, \ldots, n$. $\square$

In the above definition, note that $<_{T_i}$ denotes the ordering on the time-value set of the time unit T_i, while $<_H$ denotes the lexicographic ordering on time points of the hierarchy H. If T_i or H is clear from the context, we may simply write $<$.

Given the linear hierarchy $day \sqsubset month \sqsubset year$, both $(30, \mathit{February}, 1996)$ and $(13, \mathit{March}, 1996)$ are time points, and $(30, \mathit{February}, 1996) < (13, \mathit{March}, 1996)$. We call this hierarchy, the *Gregorian linear hierarchy.* However, if this linear hierarchy is supposed to represent the Gregorian Calendar, then $(30, \mathit{February}, 1996)$ is not a valid time point. Therefore, we define a linear calendar in the following way.

Definition 2.4 (linear calendar) A *linear calendar* consists of a linear hierarchy H of time units and a *validity predicate,* which is usually denoted as $valid_H$ (or just as *valid* if H is clear from context). The validity predicate specifies the valid time points; that is, $valid_H(t)$ is true if t is a valid time point. $\square$

The Gregorian Calendar can be represented by the linear hierarchy $day \sqsubset month \sqsubset year$ and a suitable validity predicate that states, for example, that $(13, \mathit{March}, 1996)$ is a valid time point, while $(30, \mathit{February}, 1996)$ is not. An important question is how to represent the validity predicate. Our approach, as will be discussed later (Definition 4.1), is to represent the validity predicate in a suitable constraint language.

Instead of representing a time point by specifying the day of the month, the month and the year, as in the Gregorian Calendar, we may choose to represent it by specifying the day of the week, the week of the month, the month and the year. Formally, the *alternative Gregorian linear hierarchy* is $dow \sqsubset week \sqsubset month \sqsubset year$ (where *dow* means "day of the week") and the time-value sets are as follows.

- The time-value set of *dow* consists of the seven days of the week, i.e., *Sunday*, *Monday*, etc.
- The time-value set of *week* is $\{0, 1, 2, 3, 4, 5\}$. This requires some explanation. A month may have 3 or 4 weeks that are fully contained in that month, and 1 or 2 weeks that are partially contained in that month. Week 1 of the month is always the first week that is fully contained in the month. Week 0 of the month exists if the first day of the month is not a Sunday; in this case, the month starts in the middle of a week, which is referred to as week 0 of the month.
- The time-value set *month* consists of the twelve months of the year.
- The time-value set of *year* is the set of all integers.

The *Alternative Gregorian Calendar (AGC)* consists of the alternative Gregorian linear hierarchy and a suitable validity predicate. For example, both $(\textit{Thursday}, 0, \textit{December}, 1995)$ and $(\textit{Friday}, 0, \textit{December}, 1995)$ are time points of the alternative linear hierarchy, but only the latter is a valid time point (since Friday is the first day of December, 1995). Similarly, $(\textit{Sunday}, 1, \textit{September}, 1996)$ is a valid time point that denotes the first Sunday in September, 1996, while $(\textit{Sunday}, 0, \textit{September}, 1996)$ is not a valid time point (and, in fact, $(\textit{Sunday}, 0, \textit{September}, X)$ is not a valid time point, regardless of what X is).

Remark 2.1 (notation) If C is a calendar with a linear hierarchy H, then we will use C and H interchangeably as subscripts of the orderings $<$ and $\sqsubset$. In other words, both $<_C$ and $<_H$ denote the lexicographic ordering on time points, while both $\sqsubset_C$ and $\sqsubset_H$ denote the ordering on the time units of the calendar. □

Since both the Gregorian Calendar and the AGC are intended to represent the same set of valid time points, these two calendars are compatible. The concept of composability is formally introduced in the next two definitions.

Definition 2.5 (compatible linear hierarchies) Two linear hierarchies, H_1 and H_2, are compatible if both have the same ordering among their common time units. In other words, if time units T_1 and T_2 appear in both H_1 and H_2, then $T_1 \sqsubset_{H_1} T_2$ holds if and only if $T_1 \sqsubset_{H_2} T_2$ holds. □

Proposition 2.1: *If H_1 and H_2 are compatible linear hierarchies that share a time unit T and T is the top of H_1, then T is also the top of H_2.*

Proof: Follows from the fact the top of a hierarchy is the only infinite time unit of the hierarchy. □

Definition 2.6 (compatible linear calendars) Suppose C_1 is a calendar with a linear hierarchy $T_1^1 \sqsubset_{C_1} \cdots \sqsubset_{C_1} T_m^1$, and C_2 is a calendar with a linear hierarchy $T_1^2 \sqsubset_{C_2} \cdots \sqsubset_{C_2} T_n^2$. The linear calendars C_1 and C_2 are *compatible* if their linear hierarchies are compatible and there is a bijection f from the *valid* time points of C_1 to the *valid* time points of C_2, such that

- If $(t_1^1, \ldots, t_m^1)$ and $(\bar{t}_1^1, \ldots, \bar{t}_m^1)$ are valid time points of C_1, such that $(t_1^1, \ldots, t_m^1) <_{C_1} (\bar{t}_1^1, \ldots, \bar{t}_m^1)$, then $f((t_1^1, \ldots, t_m^1)) <_{C_2} f((\bar{t}_1^1, \ldots, \bar{t}_m^1))$, and
- If T_i^1 is the same time unit as T_j^2 and $(t_1^1, \ldots, t_m^1)$ is a valid time point of C_1, such that $f((t_1^1, \ldots, t_m^1)) = (t_1^2, \ldots, t_n^2)$, then $t_i^1 = t_j^2$.

Intuitively, the first condition means that f is order preserving, and the second condition means that f is the identity on the time units that are common to both calendars. □

Proposition 2.2: If f is a bijection satisfying the conditions of the above definition, and $(t_1^1, \ldots, t_m^1)$ and $(\bar{t}_1^1, \ldots, \bar{t}_m^1)$ are valid time points of C_1, then the following holds.

$$(t_1^1, \ldots, t_m^1) <_{C_1} (\bar{t}_1^1, \ldots, \bar{t}_m^1) \Longleftrightarrow f((t_1^1, \ldots, t_m^1)) <_{C_2} f((\bar{t}_1^1, \ldots, \bar{t}_m^1))$$

Proof: One direction follows from the definition. To prove the other direction suppose $(t_1^1, \ldots, t_m^1)$ and $(\bar{t}_1^1, \ldots, \bar{t}_m^1)$ are valid time points of C_1, such that $f((t_1^1, \ldots, t_m^1)) <_{C_2} f((\bar{t}_1^1, \ldots, \bar{t}_m^1))$. Since f is bijective, $(t_1^1, \ldots, t_m^1)$ and $(\bar{t}_1^1, \ldots, \bar{t}_m^1)$ are distinct and, hence, either $(t_1^1, \ldots, t_m^1) <_{C_1} (\bar{t}_1^1, \ldots, \bar{t}_m^1)$ or $(\bar{t}_1^1, \ldots, \bar{t}_m^1) <_{C_1} (t_1^1, \ldots, t_m^1)$ must hold. However, if $(\bar{t}_1^1, \ldots, \bar{t}_m^1) <_{C_1} (t_1^1, \ldots, t_m^1)$, then the above definition implies that $f((\bar{t}_1^1, \ldots, \bar{t}_m^1)) <_{C_2} f((t_1^1, \ldots, t_m^1))$, contradicting the assumption that $f((t_1^1, \ldots, t_m^1)) <_{C_2} f((\bar{t}_1^1, \ldots, \bar{t}_m^1))$. Therefore, $(t_1^1, \ldots, t_m^1) <_{C_1} (\bar{t}_1^1, \ldots, \bar{t}_m^1)$. □

Not every calendar can be represented as a linear calendar. For example, we may want to add the time units *dow* and *week* to the Gregorian Calendar, and there is no natural linear hierarchy of the 5 time units *day*, *month*, *year*, *dow* and *week*. Therefore, we define a hierarchy of time units as follows.

Definition 2.7 (hierarchy of time units) A collection of distinct time units $T_1, \ldots, T_n$ with a partial order $\sqsubset$ (on the time units) forms a *hierarchy* if there is a tree $\mathcal{T}$, such that

- The nodes of $\mathcal{T}$ correspond to the time units $T_1, \ldots, T_n$,
- The root of the tree, called the *top* of the hierarchy, is an infinite time unit, while all other nodes are finite time units, and
- T_j is an ancestor of T_i in tree $\mathcal{T}$ if and only if $T_i \sqsubset T_j$. □

Figure 1 shows two hierarchies. A time point of a hierarchy is defined similarly to a time point of a linear hierarchy.

Definition 2.8 (time point of a hierarchy) Suppose $T_1, \ldots, T_n$ are the time units of a hierarchy H. A *time point* is an n-tuple $(t_1, \ldots, t_n)$, such that t_i is a value from the time-value set of T_i $(1 \leq i \leq n)$. The *projection* of the time point $(t_1, \ldots, t_n)$ on the time units $T_{i_1}, \ldots, T_{i_k}$ is the k-tuple $(t_{i_1}, \ldots, t_{i_k})$. □

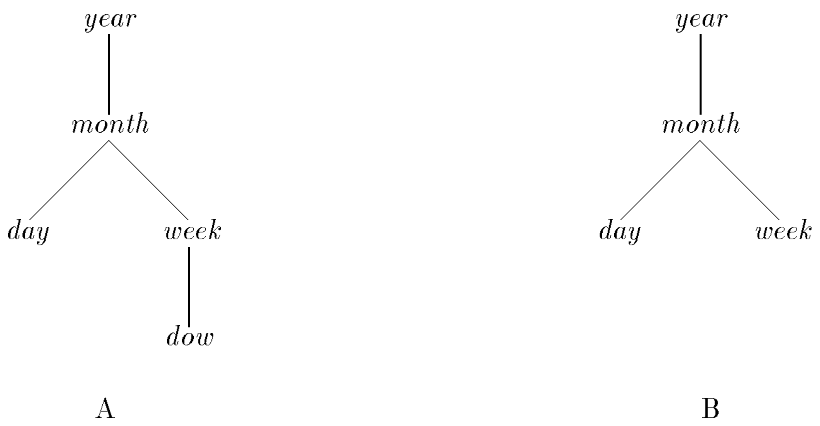


Figure 1: Two hierarchies.

Writing the time units of a hierarchy as a sequence $T_1, \ldots, T_n$ is convenient, since it shows how to associate the values of a time point $(t_1, \ldots, t_n)$ with the time units. However, the sequence $T_1, \ldots, T_n$ does *not* define a linear order among the time units; in a hierarchy, there is only a partial order among the time units and we will usually show this partial order as a tree (e.g., Figure 1). In order to make explicit the association between values and time units, we may sometimes write a time point as $(T_1 : t_1, \ldots, T_n : t_n)$, e.g., $(\mathit{day} : 1, \mathit{dow} : \mathit{Thursday}, \mathit{week} : 0, \mathit{month} : \mathit{February}, \mathit{year} : 1996)$ is a time point of the hierarchy shown in Part A of Figure 1.

In the next definition, we define a calendar as a combination of one or more linear calendars and some additional time units (which are not necessarily part of a linear calendar). The definition uses the following terminology. Given a tree $\mathcal{T}$ (representing a hierarchy), a *complete* path is any path from the root to some leaf.

Definition 2.9 (calendar) A *calendar* consists of a hierarchy $\mathcal{T}$ of time units and one or more *embedded calendars,* where an embedded calendar is a linear calendar over a linear hierarchy that coincides with some complete path of the hierarchy $\mathcal{T}$. Furthermore, if there is more than one embedded calendar, then any two embedded calendars are compatible.

In addition, a calendar has a *validity predicate* that satisfies the following conditions.

- If t is a valid time point of the calendar, then the projection of t on any embedded calendar E is a valid time point of E,
- If t_e is a valid time point of an embedded calendar E, then there is exactly one valid time point t of C, such that the projection of t on E is t_e, and
- If t is a valid time point of the calendar, such that t_1 and t_2 are projections of t on two embedded calendars E_1 and E_2, respectively, then $f(t_1) = t_2$, where f is the bijection from E_1 to E_2.

□

Proposition 2.3: Let E be an embedded calendar of some calendar C. If t_1 and t_2 are two valid time points of C that are equal on all the time units of E, then t_1 and t_2 are identical.

Proof: Follows from the second bullet item of Definition 2.9. □

We can combine the Gregorian Calendar and the AGC into one calendar that has the hierarchy shown in Part A of Figure 1. The combined calendar has both the Gregorian Calendar and the Alternative Calendar as embedded calendars. The time point $(day : 1, dow : Thursday, week : 0, month : February, year : 1996)$ is a valid time point of the combined calendar, since $(1, February, 1996)$ is a valid time point of the Gregorian Calendar, $(Thursday, 0, February, 1996)$ is a valid time point of the Alternative Calendar, and these two time points denote the same date.

As another example, consider the calendar that has the hierarchy shown in Part B of Figure 1 and has the Gregorian Calendar as the only embedded calendar. If *day*, *month* and *year* are assigned values that form a valid time point of the Gregorian Calendar, then there is a unique value for *week*, such that all four values form a valid time point of the whole calendar. However, values for *week*, *month* and *year* do not determine a unique value for *day*.

Definition 2.10 (projection/de-projection) Let C be a calendar having a hierarchy H and an embedded calendar E. For a valid time point t_c of C, the projection of t_c on the time units of E is denoted as $\mathsf{pr}(t_c, E)$. For a valid time point t_e of the embedded calendar E, the *de-projection* of t_e w.r.t. the calendar C, denoted $\mathsf{dp}_C(t_e)$, is the unique valid time point t_c of C, such that $\mathsf{pr}(t_c, E) = t_e$. If C is clear from the context, we may write $\mathsf{dp}(t_e)$ instead of $\mathsf{dp}_C(t_e)$. Also, note that $\mathsf{dp}_C(t')$ is well defined whenever t' is a time point over a subset S of the time units of C, such that S contains all time units of E. □

The reason projections and de-projections are needed is to ensure a smooth transition between time points in a calendar and time points in its embedded calendar.

Proposition 2.4: If t_c is a valid time point of C and E is an embedded calendar, then $\mathsf{dp}_C(\mathsf{pr}(t_c, E)) = t_c$.

Proof: Follows from Definition 2.9. □

Definition 2.11 (linear order on valid time points) Suppose C is a calendar and E is an embedded calendar of C. Given two valid time points t_1 and t_2 of C, we write $t_1 <_C t_2$ if $\mathsf{pr}(t_1, E) <_E \mathsf{pr}(t_2, E)$. □

Note that the linear order is well defined, since any two (linear) embedded calendars are compatible. Next, we define a linear order on all time points of C that extends the one defined above on the valid time points.

Definition 2.12 (linear order on all time points) Let E be an embedded calendar of C with a linear hierarchy H. A pseudo-linear hierarchy P of C is a linear hierarchy $T_1 \sqsubset_P \cdots \sqsubset_P T_n$ of all the time units of C, such that for some i, the linear hierarchy H coincides with $T_i \sqsubset_P \cdots \sqsubset_P T_n$. Given time points t_1 and t_2 of C, we define $t <_C t'$ if $t_1 <_P t_2$. □

Note that the order on all time points may depend on the choice of H and the choice of the pseudo-linear hierarchy for the given H. However, the order on all time points extends the one defined previously on the valid time points.

The definition of a linear order on valid time points induces a natural notion of the successor of a valid time point.

Definition 2.13 (successor of a time-value) Consider a time unit T. Given a time value t_1 from the time-value set of T, we say that t_2 is the *successor* of t_1 if either one of the following holds.

- $t_1 <_T t_2$ and there is no t' in the time-value set of T, such that $t_1 <_T t' <_T t_2$, or
- t_1 is the maximal value of the time-value set of T (i.e., T must be finite) and t_2 is the minimal value of the time-value set of T.

The ith successor of t_1 in T is denoted as $t_1 +_T i$. □

The successor of the value 4 in the time unit *day* of the Gregorian Calendar is 5 and the successor of 31 is 1.

Definition 2.14 (successor of a valid time point) Let C be a calendar with a hierarchy H. Given a valid time point t_1 of H, we say that t_2 is the *successor* of t_1 if $t_1 <_H t_2$ and there is no valid time point t' of H, such that $t_1 <_H t' <_H t_2$. The ith successor of t_1 in H is denoted as $t_1 \oplus_H i$ (or simply as $t_1 \oplus i$ if H is clear from the context). □

In the Gregorian Calendar, the successor of $(13, \mathit{March}, 1996)$ is $(14, \mathit{March}, 1996)$ and the successor of $(29, \mathit{February}, 1996)$ is $(1, \mathit{March}, 1996)$. As we shall see below (Proposition 2.5), it turns out that the concept of successor can be propagated from a calendar to its embedded calendars in a natural way.

Proposition 2.5: *Let C be a calendar with an embedded calendar E. Let $\oplus_C$ and $\oplus_E$ denote the successor operators in C and E, respectively. For all valid time points t of C, the following equalities hold.*

$$\begin{aligned} t \oplus_C 1 &= \mathsf{dp}(\mathsf{pr}(t, E) \oplus_E 1) \\ \mathsf{pr}(t, E) \oplus_E 1 &= \mathsf{pr}(t \oplus_C 1, E) \end{aligned}$$

Proof: Suppose $t \oplus_C 1 = t'$. Since E is an embedded calendar of C, Definition 2.11 implies that $\mathsf{pr}(t, E) <_E \mathsf{pr}(t', E)$. If there were a valid time point t^2 of E, such that $\mathsf{pr}(t, E) <_E t^2 <_E \mathsf{pr}(t', E)$, then Definition 2.11 would imply that $\mathsf{dp}(\mathsf{pr}(t, E)) <_C \mathsf{dp}(t^2) <_C \mathsf{dp}(\mathsf{pr}(t', E))$, or equivalently, $t <_C \mathsf{dp}(t^2) <_C t'$ (since, by Proposition 2.4, $\mathsf{dp}(\mathsf{pr}(t, E)) = t$ and $\mathsf{dp}(\mathsf{pr}(t', E)) = t'$). But $t <_C \mathsf{dp}(t^2) <_C t'$ contradicts the assumption that $t \oplus_C 1 = t'$. Therefore, we can conclude that $\mathsf{pr}(t, E) \oplus_E 1 = \mathsf{pr}(t', E)$ and, hence, $\mathsf{dp}(\mathsf{pr}(t, E) \oplus_E 1) = \mathsf{dp}(\mathsf{pr}(t', E)) = t' = t \oplus_C 1$.

The second equation can be proved in a similar way. □

We are now in a position to define what it means for two calendars to be compatible. Intuitively, compatibility of two calendars means that there is a smooth way of converting time points in one calendar to time points in the other calendar, without any loss of temporal information.

Definition 2.15 (compatible calendars) Two calendars C_1 and C_2 are *compatible* if some embedded calendar of C_1 is compatible with some embedded calendar of C_2. □

Recall that embedded calendars are linear, and that compatibility of linear calendars was defined earlier in Definition refclc-def. For example, the Gregorian Calendar and the AGC are compatible.

Before describing how we specify time w.r.t. a calendar, we prove some results concerning the successors of compatible calendars. First we will consider compatible linear calendars.

Proposition 2.6: Suppose C_1 and C_2 are linear calendars that are compatible via the bijection ψ from C_1 to C_2. Let $\oplus_1$ and $\oplus_2$ denote the successor operator in C_1 and C_2, respectively. For all valid time points t of C_1,

$$\psi(t \oplus_1 1) = \psi(t) \oplus_2 1.$$

Proof: Note that since C_1 and C_2 are linear calendars, each serves as its own embedded calendar, respectively.

Suppose $t' = t \oplus_1 1$. By Definition 2.6, $\psi(t) <_{C_2} \psi(t')$. If there were a valid time point t^2 of C_2, such that $\psi(t) <_{C_2} t^2 <_{C_2} \psi(t')$, then Definition 2.6 would imply $t <_{C_1} \psi^{-1}(t^2) <_{C_1} t'$, contradicting the assumption that $t' = t \oplus_1 1$. Therefore, there is no such t^2 and, hence, $\psi(t) \oplus_2 1 = \psi(t') = \psi(t \oplus_1 1)$. □

The above result says that the notion of successor is preserved by any bijection that establishes compatibility between linear calendars. We now extend the above result to general calendars, removing the restriction to linearity.

Proposition 2.7: Suppose C_1 and C_2 are compatible calendars. Let E_1 and E_2 be embedded calendars of C_1 and C_2, respectively, and let ψ be the bijection from E_1 to E_2 that shows the compatibility. Let $\oplus_{C_1}$ and $\oplus_{C_2}$ denote the successor operators in C_1 and C_2, respectively. For all valid time points t of C_1,

$$\mathsf{dp}_{C_2}(\psi(\mathsf{pr}(t \oplus_{C_1} 1, E_1))) = \mathsf{dp}_{C_2}(\psi(\mathsf{pr}(t, E_1))) \oplus_{C_2} 1.$$

Proof: Suppose $\oplus_{E_1}$ and $\oplus_{E_2}$ denote the successor operators of E_1 and E_2, respectively. Proposition 2.6 implies that $\psi(\mathsf{pr}(t, E_1) \oplus_{E_1} 1) = \psi(\mathsf{pr}(t, E_1)) \oplus_{E_2} 1$ and since E_2 is an embedded calendar of C_2, we get the following equation.

$$\mathsf{dp}_{C_2}(\psi(\mathsf{pr}(t, E_1) \oplus_{E_1} 1)) \quad = \quad \mathsf{dp}_{C_2}(\psi(\mathsf{pr}(t, E_1)) \oplus_{E_2} 1) \tag{1}$$

The second equation of Proposition 2.5 implies that

$$\mathsf{pr}(t, E_1) \oplus_{E_1} 1 \quad = \quad \mathsf{pr}(t \oplus_{C_1} 1, E_1). \tag{2}$$

Let $t^2 = \psi(\mathsf{pr}(t, E_1))$; note that t^2 is a valid time point of E_2. Let $t' = \mathsf{dp}_{C_2}(\psi(\mathsf{pr}(t, E_1)))$. Note that $t^2 = \mathsf{pr}(t', E_2)$. Since $t' = \mathsf{dp}_{C_2}(\mathsf{pr}(t', E_2))$, the first equation of Proposition 2.5 implies that

$$\mathsf{dp}_{C_2}(\mathsf{pr}(t', E_2) \oplus_{E_2} 1) \quad = \quad \mathsf{dp}_{C_2}(\mathsf{pr}(t', E_2)) \oplus_{C_2} 1. \tag{3}$$

Since $\mathsf{pr}(t', E_2)) = t^2 = \psi(\mathsf{pr}(t, E_1))$, we get

$$\mathsf{dp}_{C_2}(\psi(\mathsf{pr}(t, E_1)) \oplus_{E_2} 1) \quad = \quad \mathsf{dp}_{C_2}(\psi(\mathsf{pr}(t, E_1))) \oplus_{C_2} 1. \tag{4}$$

We can now substitute in Equation (1) as follows. In the left-hand side according to Equation (2) and in the right-hand side according to Equation (4). The result is

$$\mathsf{dp}_{C_2}(\psi(\mathsf{pr}(t \oplus_{C_1} 1, E_1))) \quad = \quad \mathsf{dp}_{C_2}(\psi(\mathsf{pr}(t, E_1))) \oplus_{C_2} 1. \tag{5}$$

□

The Gregorian calendar is a good example of a calendar that is periodic – a concept that we define formally below. Intuitively, a periodic calendar is whose structure "repeats" over time.

Definition 2.16 (periodic calendar) Let C be a calendar with a hierarchy H consisting of time units $T_1, \ldots, T_n$. We say that C is *periodic* if there is an integer $p > 0$, such that for all time points $(t_1, \ldots, t_n)$ of the hierarchy H, the time point $(t_1, \ldots, t_n)$ is valid for C if and only if $(t_1, \ldots, t_n +_{T_n} p)$ is valid for C (note that T_n is the top time unit). The smallest p satisfying this definition is called the *period of the top time unit.* □

Definition 2.17 (the period of a periodic calendar) Suppose C is a periodic calendar with a hierarchy H consisting of time units $T_1, \ldots, T_n$. Let p be the period of the top time unit. Let $(t_1, \ldots, t_n)$ be a time point of C. The *period of* C is defined to be the number of valid time points t, such that $(t_1, \ldots, t_n) \leq_C t <_C (t_1, \ldots, t_n +_{T_n} p)$. □

For example, the period of the Gregorian Calendar is $(365 \times 4 + 1)$ days. The period of the calendar that combines the Gregorian Calendar and the Alternative Gregorian Calendar (its hierarchy is given in Part B of Figure 1) is $(365 \times 28 + 7)$ days.

Proposition 2.8: The period of a periodic calendar C is well-defined, i.e., the period does not depend on the choice of $(t_1, \ldots, t_n)$ in Definition 2.17.

Proof: Suppose $T_1 \sqsubset_P \cdots \sqsubset_P T_n$ is the pseudo-linear hierarchy of C. Let t denote a time point $(t_1, \ldots, t_n)$ of C. We write $t +_{T_n} k$ as an abbreviation for $(t_1, \ldots, t_n +_{T_n} k)$. For an integer $k \geq 1$, let $S(t, k)$ denote the set of all time points s of C, such that $t \leq_C s <_C t +_{T_n} k$.

Let p be the period of the top time unit of C and consider an integer $h \geq 1$. Consider the bijective mapping ϕ_h on time points of C that is defined as follows.

$$\phi_h((s_1, \ldots, s_n)) = (s_1, \ldots, s_n +_{T_n} hp)$$

Clearly, the image of $S(t, k)$ under ϕ_h is $S(t +_{T_n} hp, k)$. An easy induction on h shows that ϕ_h is validity preserving; that is, t is a valid time point of C if and only if $\phi(t)$ is a valid time point of C. Therefore, we get the following fact.

Fact 2.1: Let p be the period of the top time unit of C and consider an integer $h \geq 1$. The sets $S(t, k)$ and $S(t +_{T_n} hp, k)$ have the same number of valid time points.

Next we will show that the sets $S(t, p)$ and $S(t +_{T_n} d, p)$, where $1 \leq d < p$, have the same number of valid time points. Consider the following three disjoint subsets.

1. $S(t, d)$.
2. $S(t +_{T_n} d, p - d)$.
3. $S(t +_{T_n} p, d)$.

Note that the union of the first two is equal to $S(t, p)$, and the union of the last two is equal to $S(t +_{T_n} d, p)$. Moreover, ϕ_1 maps the first subset to the third one. Therefore, we have shown the following.

Fact 2.2: Given $1 \leq d < p$, where p is the period of the top time unit of C, the sets $S(t, p)$ and $S(t +_{T_n} d, p)$ have the same number of valid time points.

To complete the proof of the proposition, consider the sets $S(t,p)$ and $S(t +_{T_n} b, p)$, and let $d = b$ **mod** p and $h = \lfloor \frac{b}{p} \rfloor$; that is, the second set can be written as $S(t +_{T_n} (d+hp), p)$. If d is zero or h is zero, then Fact 2.1 or Fact 2.2, respectively, implies that $S(t,p)$ and $S(t +_{T_n} (d+hp), p)$ have the same number of valid time points. If both d and h are nonzero, then Fact 2.1 implies that $S(t,p)$ and $S(t +_{T_n} hp, p)$ have the same number of valid time points, and Fact 2.2 implies that $S(t +_{T_n} hp, p)$ and $S(t +_{T_n} (d+hp), p)$ have the same number of valid time points. $\Box$

Proposition 2.9: Suppose C is a periodic calendar and p is the period of the top time unit. For a time point $(t_1, \ldots, t_n)$, let m denote the number of time points t, such that $(t_1, \ldots, t_n) \leq_C t <_C (t_1, \ldots, t_n +_{T_n} p)$. If C has some valid time points, then the length of any contiguous sequence (according to $<_C$) of invalid time points is bounded by m.

Proof: First note that m depends only on p, but not on $(t_1, \ldots, t_n)$, since $<_C$ is a lexicographic order according to the pseudo-linear hierarchy of C. Second, if there is contiguous sequence of m or more invalid time points, then Definition 2.17 (together with Proposition 2.8) implies that C has no valid time points. $\Box$

3 Specifying Time w.r.t. a Calendar

Let us consider any application that refers to an arbitrary, but fixed calendar C. The application developer must be able to specify time points, intervals and sets of intervals w.r.t. calendar C. For instance, when we consider the example of Mary from the Introduction, we need to specify a set of time points, viz. the first Sunday of every month. Constraints provide a convenient and succinct representation of such sets of time points. For example, the constraint

$$(\texttt{t.week} = \texttt{1} \,\&\, \texttt{t.dow} = \texttt{Sunday})$$

captures all time points `t` that denote the first Sunday of any month. The variables in the above constraint are `t.week` and `t.dow`, and they range over values from the time-value sets of `week` and `dow`, respectively. As another example, to say that time point t_1 occurs at least 3 days before time point t_2, we might write the constraint

$$\texttt{t}_1 \oplus \texttt{3} \leq \texttt{t}_2.$$

In this constraint, the variables involved are $\texttt{t}_1$ and $\texttt{t}_2$ and they range over time points. Thus, different kinds of constraints have different types of variables that potentially could range either over members of time-value sets or over time points. In general, writing constraints over time points is often syntactically much more convenient and intuitive for the user. In particular, a human user will find it very cumbersome (the reader is invited to try it!) to specify the constraint $\texttt{t}_1 \oplus \texttt{3} \leq \texttt{t}_2$ in terms of variables that range over values of time-value sets.

Consequently, we will define two constraint languages. The first is TUCL (Time Unit Constraint Language) which allows us to express constraints using variables that range over values of time-value sets. The second language is TPCL (Time Point Constraint Language) and it is for expressing constraints using variables that range over time points.

Definition 3.1 (TPCL—time point constraint language) Let C be a calendar with a hierarchy H of time units $T_1, \ldots, T_n$.

- TPCL **Variables:** There is an infinite set of variable symbols $t, t', t^1, t^2, \ldots$ that range over time points of C.
- TPCL **Constants:** Every time point is a constant symbol.
- TPCL **Atoms:** $x \Re y$, where $\Re \in \{<_C, =\}$, is a TPCL atom if at least one of x and y is a TPCL variable, say t^i, and the other is either (i) a TPCL constant symbol, (ii) a TPCL variable t^j, or (iii) of the form $t^j \oplus_C n$ for some natural number $n > 0$.
- TPCL **Constraints:** A TPCL constraint is any boolean combination of TPCL atoms constructed using the logical connectives $\wedge$, $\vee$ and $\neg$ (sometimes we use & instead of $\wedge$). $\Box$

For example, $t^1 < (1, \mathit{January}, 1995)$, $t^1 = t^2 \oplus 1$ and $t^2 = (3, \mathit{February}, 1996)$ are TPCL atoms w.r.t. the Gregorian Calendar. The conjunction $t^1 = t^2 \oplus 1 \wedge t^2 < (1, \mathit{January}, 1995)$ is a TPCL constraint that intuitively refers to any two consecutive days before 1995.

Definition 3.2 (TUCL—time unit constraint language) Let C be a calendar with a hierarchy H of time units $T_1, \ldots, T_n$. Let V_i denote the time-value set associated with T_i.

- TUCL **Variables:** If t is any TPCL variable, then $t.T_j$ is a TUCL variable of *type* T_j that ranges over values of V_j.
- **Constants:** Every member of $V_1 \cup \cdots \cup V_n$ is a TUCL constant symbol.
- TUCL **Atoms:** $x \Re y$, where $\Re \in \{<_{T_k}, =\}$, is a TUCL atom of *type* T_k if at least one of x and y is of the form $t^i.T_k$ and the other is either (i) a TUCL constant symbol from the set V_k, (ii) a TUCL variable of the form $t^j.T_k$, or (iii) of the form $t^j.T_k +_{T_k} n$ for some integer $n > 0$.
- TUCL **Constraint:** A TUCL constraint is any boolean combination of TUCL atoms constructed using the connectives $\wedge$, $\vee$ and $\neg$ (sometimes we use & instead of $\wedge$).

  A TUCL constraint Ψ is said to be *associated with* the TPCL variable t^i if all TUCL variables appearing in Ψ are of the form $t^i.T_j$ $(1 \leq j \leq n)$. $\Box$

For example, $\texttt{t.week} = \texttt{1}$ is a TUCL atom that has one TUCL variable, namely $\texttt{t.week}$. An example of a TUCL constraint in the Gregorian Calendar is $t^1.day = 1 \wedge t^1.month = March$, which is associated with t^1 and intuitively refers to the first of March (of all years). Similarly, the TUCL constraint $(t^1.month = May \vee t^1.month = June) \wedge t^1.year <_{year} 1990$ intuitively refers to the May–June period of any year before 1990.

The reader should note that variables of the form $\texttt{t.week}$ (or more generally $t.T_i$) may only appear in TUCL atoms, but not in TPCL atoms. Furthermore, the symbol $\oplus_C$ may occur in TPCL atoms but not in TUCL atoms. In contrast, the symbol $+_{T_i}$ may occur in TUCL atoms, but not in TPCL atoms.

Definition 3.3 (solution of TPCL constraints) A solution, σ, to a TPCL constraint Ξ is an assignment of a time point to each TPCL variable in Ξ, such that when each occurrence of each TPCL variable V in Ξ is replaced with $\sigma(V)$, the result is true. We say that two TPCL constraints are *equivalent* if they have the same set of solutions. □

For example, $t^1 = (2, February, 1990)$ and $t^1 = (5, March, 1980)$ are solutions to the constraint $t^1 < (1, January, 1995)$.

Definition 3.4 (solution of TUCL constraints) A solution, σ, to a TUCL constraint Ξ is an assignment of a time point to each TPCL variable $\texttt{t}$ occurring in Ξ, such that when each occurrence of each TUCL variable $\texttt{t.T}_\texttt{i}$ in Ξ is replaced with $\sigma(t.T_i)$, the result is true. We say that two TUCL constraints are *equivalent* if they have the same set of solutions. □

For example, $\{t^1.month = January, t^1.day = 1, t^1.year = 1995\}$ is a solution to the constraint $t^1.month <_{month} February \wedge t^1.day <_{day} 3$. Note that $\{t^1.month = January, t^1.day = 2\}$ is not a solution, since $t^1.year$ is not assigned a value.

Example 3.1: Suppose we return to the example concerning Mary. In this example, the action "Remind Mary to send out her phone bill" is fired for each solution of the constraint

$$(\texttt{t.week} = \texttt{1} \,\&\, \texttt{t.dow} = \texttt{Sunday}).$$

This constraint has many solutions. For example,

$$\sigma_1 = \{\texttt{t.}dow = \texttt{Sunday}, \texttt{t.}week = \texttt{1}, \texttt{t.}month = \texttt{January}, \texttt{t.}year = \texttt{1997}\}$$

is a solution.

Notation: Suppose Ψ is a conjunction of constraints involving a temporal (i.e., TPCL or TUCL) variable t and further suppose that t' is either a constant or a temporal variable. We use the notation $\Psi[t/t']$ to denote the expanded constraint $\Psi \wedge \{t = t'\}$.

In addition to the TUCL and TPCL constraint languages, we define below a hybrid constraint language that combines both of the above. This hybrid language provides the flexibility of expressing temporal constraints using both time-unit based atoms and time-point based atoms.

Definition 3.5 (hybrid constraint language) *An* HCL *constraint is any boolean combination of* TPCL *and* TUCL *atoms constructed using the connectives* $\wedge$, $\vee$ *and* $\neg$ *(sometimes we use* & *instead of* $\wedge$*).* $\square$

For example, $t^1 = t^2 \oplus 1 \wedge (t^1.month = June \vee t^1.month = January)$ is an HCL constraint that intuitively refers to any two consecutive days, where the first day is either in June or January of any year.

4 Using Constraints to Define Calendars

Suppose C is a calendar and Ψ is a constraint w.r.t. C. Solutions to this constraint merely assign time points to the temporal variables appearing in the constraint. However, the resulting time point may or may not be valid. Thus far in the paper, we have merely assumed the existence of a validity predicate that tells us which time points are valid. In this section, we will show that validity predicates may be elegantly and succinctly expressed in terms of constraints. Furthermore, it turns out that solving constraints that only contain variables with respect to some time units, e.g., TUCL constraints, can be done using well known techniques. Therefore, we consider validity predicates that can be expressed using such constraints.

4.1 Realizing Validity Predicates Through Constraints

Recall that in any calendar, only the top time unit is infinite. Therefore, it is possible to represent the validity predicate of a calendar by a set of pairs of constraints (Ψ, Φ), such that:

> *If the values $t_1 \ldots, t_{n-1}$ of the finite time units $T_1, \ldots, T_{n-1}$ of the calendar satisfy the constraint Ψ and if the value t_n of the top time unit of the calendar is a solution of the constraint Φ, then $(t_1 \ldots, t_{n-1}, t_n)$ is a valid time point.*

Thus, Ψ only involves the variables $t.T_1, \ldots, t.T_{n-1}$, while Φ only involves the variable $t.T_n$. We leave open the possibility that the language for expressing the constraint Ψ on the finite time units is different from the language for expressing the constraint Φ on the top time value. The *top constraint* is any TUCL constraint whose only variable is of the form $t.T_n$, where t is a TPCL variable and T_n is the top time unit. For example, in the case of the Gregorian Calendar, $t.year \bmod 4 = 0$ is a top constraint. Note that in this example, the language in which Φ is expressed includes the "mod" operator. Other calendars may or may not require this operator and, therefore, we leave open the precise choice of the language for expressing the top constraint. The constraint Ψ on the finite time units will be expressed as a TUCL constraint that has a single TPCL variable.

Definition 4.1 (realization of a validity predicate via constraints) Let C be a calendar with a hierarchy H of time units $T_1, \ldots, T_n$, where T_n is the top time unit. A set $\{(\Psi_1, \Phi_1), \ldots, (\Psi_m, \Phi_m)\}$ of pairs of constraints is said to *realize a validity predicate* if

- Ψ_i is a TUCL constraint that has a single TPCL variable t and only the TUCL variables $t.T_1, \ldots, t.T_{n-1}$ (but not $t.T_n$) may appear in Ψ_i,
- Φ_i is a top constraint involving only the TUCL variable $t.T_n$,
- For all $1 \leq i < j \leq n$, the TUCL constraints Ψ_i and Ψ_j have no common solutions, and
- $(t'_1, \ldots, t'_n)$ is a valid time point of C if and only if there is a pair (Ψ_k, Φ_k), such that $t.T_1 = t'_1, \ldots, t.T_{n-1} = t'_{n-1}$ is a solution of Ψ_k and $t.T_n = t'_n$ is a solution of Φ_k (in other words, $(t'_1, \ldots, t'_n)$ is a solution of the conjunction $\Psi_k \wedge \Phi_k$). $\square$

The Gregorian Calendar can be realized by the constraints $\{(\Psi_1, \Phi_1), (\Psi_2, \Phi_2)\}$, where

$$\begin{aligned}
\Psi_1 &= (t.day <_{day} 29) \vee (t.day <_{day} 31 \,\&\, t.month \neq February) \vee \\
&\quad (t.day = 31 \,\&\, (t.month = January \vee t.month = March \vee t.month = May \vee t.month = July \vee \\
&\quad t.month = August \vee t.month = October \vee t.month = December)) \\
\Phi_1 &= 0 \leq t.year \\
\Psi_2 &= (t.day = 29 \,\&\, t.month = February) \\
\Phi_2 &= t.year \bmod 4 = 0
\end{aligned}$$

For example, the pair (Ψ_2, Φ_2) says that if the *day* field is 29 and the *month* field is *February*, then the *year* field must be a multiple of 4. Notice, in particular, that the language of Φ_2 is somewhat more complex than that used for Φ_1—it involves the use of some arithmetic through the `mod` operator.

4.2 Realizing Successors Through Constraints

Recall that TPCL constraints are allowed to contain the successor operator $\oplus$. In this section, we show how we may specify the successor of a time point t through the use of constraints. The reason we need to specify successors using constraints is that adding 1 to the lowest time unit and handling the carries appropriately (which may seem adequate, prima facie) is not adequate in general. This is because not all combinations of time values are valid time points. For example, adding 1 to Feb. 28, 1993, leads (naively) to the syntactically correct, but invalid date of Feb. 29, 1993. As the validity predicate itself is realized through the use of constraints, we continue to use constraints to specify the notion of a successor.

Definition 4.2 (realization of a successor via constraints) Let C be a calendar with a hierarchy H of time units $T_1, \ldots, T_n$, where T_n is the top time unit. A set $\{(\Theta_1, \Upsilon_1), \ldots, (\Theta_m, \Upsilon_m)\}$ of pairs of constraints is said to *realize a successor* if

- Θ_i is a TUCL constraint that has a single TPCL variable t, and the TUCL variables $t.T_1, \ldots, t.T_n$ may appear in Θ_i,
- Υ_i is a TUCL constraint that has exactly two TPCL variable t and t^+, and the TUCL variables $t.T_1, \ldots, t.T_n$ and $t^+.T_1, \ldots, t^+.T_n$ may appear in Υ_i,
- For each valid time point t', there is exactly one Θ_i $(1 \leq i \leq n)$, such that $t = t'$ is a solution of Θ_i, and
- If $t = t'$ is a solution of Θ_i, then $\Upsilon_i[t/t']$ must have exactly one solution $t^+ = t''$ and t'' is the successor of t' in C. □

Intuitively, the successor specification allows us to find the successor of a given time p point as follows. First, determine if the time point p is valid. If the time point p is not valid, then it does not have a successor. Otherwise, determine which of the Θ_i has the solution $t = p$. Next, find the solution $t^+ = p'$ of $\Upsilon_i[t/p]$, and p' is the desired successor. For example, the set $\{(\Theta_1, \Upsilon_1), (\Theta_2, \Upsilon_2), (\Theta_3, \Upsilon_3)\}$ realizes the successor in the Gregorian Calendar, where

$$\begin{aligned}
\Theta_1 &= (t.day <_{day} 28) \vee (t.day <_{day} 29 \,\&\, t.month \neq February) \vee \\
&\quad (t.day = 30 \,\&\, (t.month = January \vee t.month = March \vee t.month = May \vee t.month = July \vee \\
&\quad t.month = August \vee t.month = October \vee t.month = December)) \vee \\
&\quad (t.day = 28 \,\&\, t_1.month = February \,\&\, t.year \bmod 4 = 0) \\
\Upsilon_1 &= t^+.month = t.month \,\&\, t^+.year = t.year \,\&\, t^+.day = t.day +_{day} 1 \\
\Theta_2 &= (t.day = 31 \,\&\, (t.month = January \vee t.month = March \vee t.month = May \vee t.month = July \vee \\
&\quad t.month = August \vee t.month = October)) \vee \\
&\quad (t.day = 30 \,\&\, (t.month = April \vee t.month = June \vee t.month = September \vee t.month = November)) \vee \\
&\quad (t.day = 28 \& t.month = February \& t.year \bmod 4 \neq 0) \\
\Upsilon_2 &= t^+.month = t.month +_{month} 1 \,\&\, t^+.year = t.year \,\&\, t^+.day = 1 \\
\Theta_3 &= t.month = December \,\&\, t.day = 31. \\
\Upsilon_3 &= t^+.month = 1 \,\&\, t^+.day = 1 \,\&\, t^+.year = t.year +_{year} 1
\end{aligned}$$

5 Solving Temporal Constraints

In general, given an HCL constraint, we would like to check if there exists a *valid* solution of that constraint. We emphasize the point that viewing HCL constraints as ordinary constraints over the integers is not enough, because the special nature of the validity predicate may cause some solutions (over the integers) to represent invalid time points. In this section, we describe the following three aspects of determining existence of valid solutions of temporal constraints:

1. First, we show how we can convert a temporal constraint Ξ, into a new constraint Ξ' such that Ξ' has a solution (over the time points) iff Ξ has a valid solution over the time points.
2. Second, we show how we can eliminate all occurrences of $\oplus$ from Ξ'.
3. Finally, we show that as a consequence all TPCL-atoms can be eliminated as well, while preserving satisfiability.

The above transformations allow us to determine satisfiability of the resulting constraint by using standard constraint solving techniques over finite domains and the over the integers [12]. In fact, if we consider only TPCL-constraints over periodic calendars, then it turns out that $\oplus$ and subtraction of time points can be solved in polynomial time.

5.1 Incorporating Validity

Suppose C is a calendar and Ξ is a constraint. We wish to create a constraint Ξ' such that: σ is a solution of Ξ' iff σ is a valid solution of Ξ. This is because an HCL constraint Ξ may have solutions that are not valid. However, as we are only interested in *valid* solutions of Ξ, we create a new constraint, Ξ', by adding some validity criteria to the constraint Ξ. The following definition shows how Ξ' may be defined for this purpose.

Definition 5.1 (extension of a constraint to handle validity) Consider a calendar C whose validity predicate is realized via a set of pairs $\{(\Psi_1, \Phi_1), \ldots, (\Psi_m, \Phi_m)\}$ with the variable t. Let $valid(t)$ denote the following formula:

$$(\Psi_1(t) \wedge \Phi_1(t)) \vee \cdots \vee (\Psi_m(t) \wedge \Phi_m(t)).$$

Suppose Ξ is an HCL constraint with variables $t^1, \ldots, t^k$ (note that t does not occur in Ξ). The *extension* of Ξ with respect to the validity predicate of C is the following formula.

$$\Xi \wedge \left(\bigwedge_{1 \leq i \leq k} valid[t/t^i] \right)$$

□

Proposition 5.1: σ is a solution of $\Xi \wedge \left(\bigwedge_{1\leq i\leq k} valid[t/t^i]\right)$ if and only if σ is both a solution of Ξ and a valid time point of C.

Proof: Follows from Definition 4.1. □

5.2 Eliminating Successor

In this section, we show how to eliminate all occurrences of $\oplus$ from a constraint Ξ by iteratively adding conjuncts to Ξ based on the constraints that realize the successor.

Algorithm 1 (Elimination of $\oplus$) Let C be a calendar whose successor is realized via the set of pairs $\{(\Theta_1, \Upsilon_1), \ldots, (\Theta_m, \Upsilon_m)\}$, where t and t^+ are the variables occurring in these constraints. We use $succ(t, t^+)$ to denote the following formula.

$$(\Theta_1(t) \wedge \Upsilon_1(t, t^+)) \vee \cdots \vee (\Theta_m(t) \wedge \Upsilon_m(t, t^+))$$

Suppose Ξ is an HCL constraint, such that neither t not t^+ occurs in Ξ. The elimination of $\oplus$ is achieved by rewriting Ξ according to the following two rules until no rewriting is possible.

1. If $v \oplus 1$ occurs in Ξ, then do the following. First, replace all occurrences of $v \oplus 1$ in Ξ with a new variable v^+ that does not occur in Ξ. Second, replace Ξ with

$$\Xi \wedge succ[t/v, t^+/v^+]$$

2. If $v \oplus b$ occurs in Ξ, where $b > 1$ is an integer, then do the following. First, replace all occurrences of $v \oplus b$ in Ξ with $v^c + (b-1)$, where v^c is a new variable that does not occur in Ξ. Second, replace Ξ with

$$\Xi \wedge v^c = v \oplus 1$$

Proposition 5.2: Let Ξ' be the result of rewriting Ξ according to the above rules. σ is a solution of Ξ' if and only if the restriction of σ to the variables of Ξ is a solution of Ξ.

Proof: Follows from Definition 4.2 by an induction on the number of applications of the rewriting rules. □

Even if Ξ has only one occurrence of a term of the form $v \oplus b$, the rewriting needed to eliminate $\oplus$ is exponential in the binary representation of b. However, most (if not all) calendars in daily use are periodic (in the sense of Definition 2.16) and the following lemma holds for such calendars.

Lemma 5.1: Suppose C is a periodic calendar, and let p be the period of the top time unit and c be the period of the calendar. Then for all valid time points $(t_1, \ldots, t_n)$ and all integers $c \geq 1$, the following is true, where $d = b \textbf{ mod } c$ and $h = \lfloor \frac{b}{c} \rfloor$.

$$(t_1, \ldots, t_n) \oplus b = (t_1, \ldots, t_n +_{T_n} hp) \oplus d$$

Proof: Follows from Definition 2.16, Definition 2.17 and Proposition 2.8. □

When the above lemma holds, we may replace a term of the form $v \oplus b$ by the term $v' \oplus d$ and add the following conjunction

$$v'.T_1 = v.T_1 \wedge \cdots \wedge v'.T_{n-1} = v.T_{n-1} \wedge v'.T_n = v.T_n + hp$$

where v' is a new TPCL variable. Thus, the process of eliminating $v \oplus b$ requires at most $O(p)$ applications of the rewriting rules (since $d \leq p$). Since p is fixed (i.e., it depends only on the given calendar, but not on Ξ), the elimination of $v \oplus b$ can be done in linear time in the size of Ξ.

5.3 Subtracting Time Points of Periodic Calendars

In order to solve constraints efficiently, we may need to subtract one time point from another. Subtraction of time points can be done efficiently if calendars are periodic. Suppose C is a periodic calendar, such that p is the period of the top time unit and c is the period of the calendar. Let $t \leq_C s$ be two valid time points of C, where $t = (t_1, \ldots, t_n)$ and $s = (s_1, \ldots, s_n)$. Computing the non-negative integer a, such that $s = t \oplus a$, can be done as follows. First, let $b = s_n - t_n$, $d = b$ **mod** p and $h = \lfloor \frac{b}{p} \rfloor$. By the definitions of p and c, it follows that $a = a' + hc$, where a' satisfies

$$(t_1, \ldots, t_n) \oplus a' = (s_1, \ldots, t_n +_{T_n} d).$$

Now, let $d' = t_n$ **mod** p and $h' = \lfloor \frac{t_n}{p} \rfloor$. By Definition 2.17 and Proposition 2.8, a' also satisfies

$$(t_1, \ldots, d') \oplus a' = (s_1, \ldots, d' +_{T_n} d).$$

Since d' and d are nonnegative and bounded by c, and c is a constant that depends only on the calendar, we may compute a' in polynomial time.

5.4 Eliminating TPCL Atoms

In this section, we show how we may iteratively eliminate TPCL atoms from a constraint Ξ.

Algorithm 2 (Elimination of TPCL Atoms) *Let $E_1, \ldots, E_k$ be the embedded calendars of C where $T_1^j \sqsubset \ldots \sqsubset T_{n_j}^j$ is the linear hierarchy of the embedded calendar E_j, $1 \leq j \leq k$. For each constraint atom* **A** *in Ξ do:*

1. **If A** *is of the form $t = (v_1, \ldots, v_n)$* **then** *replace it by:*

$$t.T_1 = v_1 \;\wedge\; t.T_2 = v_2 \;\wedge \ldots \wedge\; t.T_n = v_n$$

2. **If A** *is of the form $t^1 = t^2$* **then** *replace it by:*

$$t^1.T_1 = t^2.T_1 \;\wedge\; t^1.T_2 = t^2.T_2 \;\wedge \ldots \wedge\; t^1.T_n = t^2.T_n$$

3. **If A** *is of the form* $t^1 <_H t^2$ **then** *replace it by* $t^1 <_{E_1} t^2 \wedge \ldots. \wedge t^1 <_{E_k} t^2$.

4. **If A** *is of the form* $t <_H (v_1, \ldots, v_n)$ **then** *replace it by* $t <_{E_1} (v_1^1, \ldots, v_{n_1}^1) \wedge \ldots \wedge t <_{E_k} (v_1^k, \ldots, v_{n_k}^k)$ *where* $(v_1^j, \ldots, v_{n_j}^k)$ *is the projection of* $(v_1, \ldots, v_n)$ *on* E_j.

5. If **A** *is of the form* $t <_{E_j} (v_1^j, \ldots, v_{n_j}^j)$ *replace it by*

$$\begin{array}{rrrrrl}
(t.T_1^j > v_1^j \wedge & t.T_2^j = v_2^j \wedge & t.T_3^j = v_3^j \wedge & \ldots & \wedge t.T_{l_j}^j = v_{n_j}^j & \vee \\
& (t.T_2^j > v_2^j \wedge & t.T_3^j = v_3^j \wedge & \ldots. & \wedge\, t.T_{n_j}^j = v_{n_j}^j) & \vee \\
& & & \ldots & & \vee \\
& & & & (t.T_{n_j}^j > v_{n_j}^j) &
\end{array}$$

6. If **A** *is of the form* $t^1 <_{E_j} t^2$ *replace it by*

$$\begin{array}{rrrrrl}
(t^1.T_1^j > t^2.T_1^j \wedge & t^1.T_2^j = t^2.T_2^j \wedge & t^1.T_3^j = t^2.T_3^j \wedge & \ldots & \wedge t^1.T_{l_j}^j = t^2.T_{n_j}^j & \vee \\
& (t^1.T_2^j > t^2.T_2^j \wedge & t^1.T_3^j = t^2.T_3^j \wedge & \ldots & \wedge\, t^1.T_{n_j}^j = t^2.T_{n_j}^j) & \vee \\
& & & \ldots & & \vee \\
& & & & (t^1.T_{n_j}^j > t^2.T_{n_j}^j) &
\end{array}$$

Based on the techniques described thus far, it is easy to define both a concept of a temporal normal form, and procedure that can be used to convert any constraint Ξ into a temporal normal form constraint Ξ' such that Ξ has a valid solution iff Ξ' has a solution. (The technical report version of this paper [15] provides an appendix consisting of such a normal form, and a transformation procedure, but is omitted here to avoid lengthening the paper.) Furthermore, there is a close correspondence between solutions of Ξ and Ξ' – if $VALIDSOL(\Xi)$ denotes the set of *valid* solutions of Ξ, then:

$$VALIDSOL(\Xi) \quad = \quad \{\sigma \mid \sigma' \text{ is a solution of } \Xi' \text{ and } \sigma \text{ is the restriction of } \sigma' \text{ to the variables of } \Xi\}.$$

6 Combining Multiple Calendars

Earlier in the paper, we stated that we can combine the Gregorian Calendar and the AGC into one calendar that has the hierarchy shown in Part A of Figure 1. However, this statement was an informal one—we stated what it means for two calendars to be compatible with each other, but we did not formally state what it means to combine two arbitrary calendars.

Multiple calendars may arise naturally in a vast number of settings. For instance, consider the case where an agent must access two data sources, one of which uses a calendar with the hierarchy

$$\texttt{dow} \sqsubset \texttt{week} \sqsubset \texttt{month} \sqsubset \texttt{year}$$

while the other uses calendar with the hierarchy

$$\texttt{day} \sqsubset \texttt{month} \sqsubset \texttt{year}.$$

In this case, the agent must be able to reason with both calendars by constructing a new calendar that combines the two. The purpose of this section is to define what constitutes such a valid combination, and to develop techniques to compute these combinations.

6.1 What Does It Mean to Combine Two Calendars?

Any notion of a combination of two calendars must satisfy certain intuitive requirements. In this section, we will describe a number of examples and, in each case, we will define what intuitively constitutes a valid combination. This will enable us to specify various criteria that should be satisfied by a combined calendar.

Suppose C_1 and C_2 are two calendars that we wish to combine. In other words, we want to find a calendar C that "captures" calendars C_1 and C_2. There are three parts involved in any method for creating C:

1. **Hierarchy Construction:** Given the hierarchies associated with C_1 and C_2, we should be able to define a hierarchy that will be associated with the calendar C. In other words, we should be able to merge the trees associated with C_1 and C_2 into a new tree $\mathcal{T}$ that preserves some of the ancestor relationships of the original trees.
2. **Selecting Embedded Calendars:** Once we have computed the tree $\mathcal{T}$ of the hierarchy associated with the combined calendar, we need to identify (at least) one complete path in this tree and designate it as an embedded calendar.
3. **Merging Validity Specifications:** Finally, we must merge the validity specifications associated with C_1 and C_2 into a new validity specification that guarantees that the embedded calendars selected in the preceding step satisfy the conditions in Definition 2.9.

We now present some examples illustrating the above steps.

Example 6.1: Suppose we wish to combine the Gregorian calendar and the AGC. We need to go through the following steps.

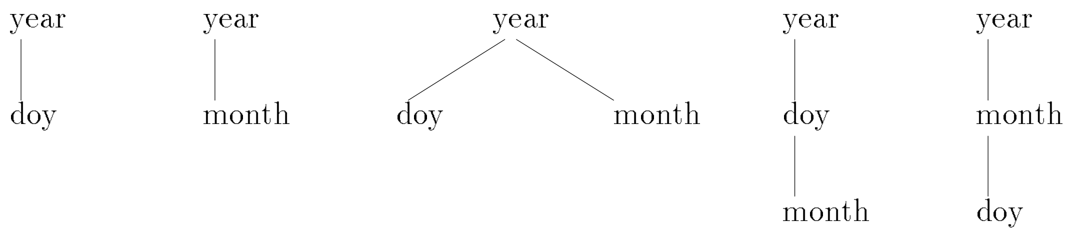
1. Hierarchy Construction: In this step, we need to construct a hierarchy that preserves the ancestor relationships of both the Gregorian Calendar and the AGC. Figure 1(A) shows one such hierarchy that preserves the desired ancestor relationships. Figure 2 also preserves the ancestor relationships. However, the hierarchy of Figure 2 includes an ordering constraint, viz. `day` $\sqsubset$ `dow`, that is not required by either the Gregorian calendar or the AGC. Therefore, we should prefer the more conservative hierarchy shown in Figure 1(A).
2. Select Embedded Calendars: We now need to examine all the complete paths in Figure 1(A) and select some of these paths as embedded calendars. There are two possibilities, in this

Figure 2: A Non-conservative hierarchy construction.

simple example, corresponding to the Gregorian calendar and the AGC, respectively. Since these calendars were compatible, we can safely designate both as embedded calendars of the combination.

3. Merging Validity Specifications: We now need to construct a validity predicate, keeping in mind the choice of the hierarchy made above (i.e., Figure 1(A)) and the choice of the embedded calendars. In particular, we want to ensure that if t is a valid time point of the combined calendar (that we are constructing), then the projections of t on both the Gregorian Calendar and the AGC are valid w.r.t. the validity predicates of the Gregorian Calendar and the AGC, respectively. In addition, we need to specify the "link" between the AGC and the Gregorian Calendar. In effect, this means that we specify, given values for *year*, *month*, *week* and *dow*, how to obtain the corresponding value for *day*, and vice-versa. In order to realize the validity predicate of the combined calendar, we need to manipulate the constraints that realize the validity predicates of the Gregorian calendar and the AGC.

Example 6.2: Now consider a slightly more complicated example, where we have two very simple linear calendars C_1 and C_2 whose hierarchies are shown in Figure 3 Part (a) and Part (b), respectively. There are exactly three ways of constructing a hierarchy that merges them and these are depicted in Figure 3 Parts (c), (d) and (e). Of these, only Figure 3(c) is conservative and does not introduce any new ancestor relationships. There are two complete paths in this case that are candidates for being embedded calendars. However, in contrast to the preceding example, the two original calendars C_1 and C_2 are not compatible. Intuitively, we know that values for `year` and `doy` uniquely determine the `month`. The converse does not apply, since given the `year` and the `month`, we cannot uniquely specify a `doy`. Therefore, it is preferable to select the complete path `year`–`doy` as the embedded calendar of the combination. The complete path `year`–`month` will then be a secondary calendar of the combination.



(a) (b) (c) (d) (e)

Figure 3: Possible hierarchy constructions.

Finally, to realize the validity predicate of the combined calendar, we need to explicitly specify the link between the `year` and the `doy`, on one hand, and the `month`, on the other hand; i.e., we need to state how the `month` may be computed from the `year` and the `doy`.

Example 6.3: Consider the calendars $\mathcal{C}_a$ and $\mathcal{C}_b$, where

$$\mathcal{C}_a: \texttt{ day} \sqsubset \texttt{month} \sqsubset \texttt{year}$$

$$\mathcal{C}_b: \texttt{ hour} \sqsubset \texttt{day} \sqsubset \texttt{month} \sqsubset \texttt{year}.$$

In this case, the hierarchy construction leads to the same hierarchy as that of $\mathcal{C}_b$. Consequently, $\mathcal{C}_b$ can be the only embedded calendar, and the validity predicate must coincide with that of $\mathcal{C}_b$.

Finally, we consider a case where the calendars cannot be combined.

Example 6.4: Suppose $\mathcal{C}_a$ and $\mathcal{C}_b$ are as given below.

$$\mathcal{C}_a: \texttt{ month} \sqsubset \texttt{year}.$$

$$\mathcal{C}_b: \texttt{ woy} \sqsubset \texttt{year}.$$

Here, `woy` stands for "week of the year." There is one hierarchy that most conservatively merges the hierarchies of $\mathcal{C}_a$ and $\mathcal{C}_b$. The two complete paths of this combined hierarchy correspond to $\mathcal{C}_a$ and $\mathcal{C}_b$, respectively. In the preceding example, we showed how we could use the time units of one embedded calendar to determine uniquely the values of the other time units. However, in this case, neither of the two can be designated as the embedded calendar, because neither one of them uniquely determines the other.

6.2 Combining Hierarchies

In this section, we first specify declaratively what it means to construct a hierarchy that combines hierarchies from two different calendars.

Definition 6.1 (hierarchy combination) Suppose C_1 and C_2 are two calendars with hierarchies $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. The hierarchy $\mathcal{T}$ is said to be a *hierarchy combination* of $\mathcal{T}_1$ and $\mathcal{T}_2$ if

1. The set of time units of $\mathcal{T}$ is the union of the sets of time units of $\mathcal{T}_1$ and $\mathcal{T}_2$,
2. If $T_i \sqsubset_{\mathcal{T}_k} T_j$, where $k \in \{1, 2\}$, then $T_1 \sqsubset_{\mathcal{T}} T_2$, and
3. If $T_i \sqsubset_{\mathcal{T}} T_j$, then either $T_i \sqsubset_{\mathcal{T}_1} T_j$ or $T_i \sqsubset_{\mathcal{T}_2} T_j$.

□

Intuitively, a hierarchy combination merely ensures that the orderings on the time units, from the two calendars being combined, are preserved in the most conservative possible way.

Proposition 6.1 (uniqueness of hierarchy combination) Suppose C_1 and C_2 are two calendars with the hierarchies $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. There exists at most one hierarchy $\mathcal{T}$ that is a hierarchy combination of $\mathcal{T}_1$ and $\mathcal{T}_2$.

Proof: Suppose there are two distinct trees $\mathcal{T}$ and $\mathcal{T}'$, such that each one is a hierarchy combination of $\mathcal{T}_1$ and $\mathcal{T}_2$. First, note that $T_i \sqsubset_{\mathcal{T}} T_j$ if and only if $T_i \sqsubset_{\mathcal{T}'} T_j$. In proof, if $T_i \sqsubset_{\mathcal{T}} T_j$, then by the third part of Definition 6.1, either $T_i \sqsubset_{\mathcal{T}_1} T_j$ or $T_i \sqsubset_{\mathcal{T}_2} T_j$; in either case, $T_i \sqsubset_{\mathcal{T}'} T_j$ follows from the second part of Definition 6.1. The other direction is proved similarly.

Since $\mathcal{T}$ and $\mathcal{T}'$ are distinct, there is a pair of time units T_i and T_j, such that in one of these two trees, say in $\mathcal{T}$, time unit T_j is the parent of T_i (implying $T_i \sqsubset_{\mathcal{T}} T_j$), but T_j is not the parent of T_i in the other tree, $\mathcal{T}'$. However, since $T_i \sqsubset_{\mathcal{T}} T_j$, we must also have $T_i \sqsubset_{\mathcal{T}'} T_j$; that is, T_j is an ancestor (but not the parent) of T_i in $\mathcal{T}'$. Let T_k be the parent of T_i in $\mathcal{T}'$. Therefore, $T_i \sqsubset_{\mathcal{T}'} T_k \sqsubset_{\mathcal{T}'} T_j$ and, so, we must also have $T_i \sqsubset_{\mathcal{T}} T_k \sqsubset_{\mathcal{T}} T_j$, contradicting the assumption that T_j is the parent of T_i in $\mathcal{T}$. □

Definition 6.2 (combinable hierarchies) Suppose $\mathcal{T}_1$ and $\mathcal{T}_2$ are two hierarchies of time units. We say that $\mathcal{T}_1$ and $\mathcal{T}_2$ are *combinable* if

1. $\mathcal{T}_1$ and $\mathcal{T}_2$ share the same top time unit,
2. If time units T_i and T_j appear in both $\mathcal{T}_1$ and $\mathcal{T}_2$, then $T_i \sqsubset_{\mathcal{T}_1} T_j$ holds if and only if $T_i \sqsubset_{\mathcal{T}_2} T_j$ holds,
3. If $T_i \sqsubset_{\mathcal{T}_m} T_j$ and $T_j \sqsubset_{\mathcal{T}_p} T_k$, where $m, p \in \{1, 2\}$, then either $T_i \sqsubset_{\mathcal{T}_1} T_k$ or $T_i \sqsubset_{\mathcal{T}_2} T_k$.
4. If $T_i \sqsubset_{\mathcal{T}_m} T_j$ and $T_i \sqsubset_{\mathcal{T}_p} T_k$, where $m, p \in \{1, 2\}$, then either $T_j \sqsubset_{\mathcal{T}_q} T_k$ or $T_k \sqsubset_{\mathcal{T}_q} T_j$ for some $q \in \{1, 2\}$.

All the hierarchies of the calendars in the examples of Section 6.1 are combinable. The result below states that if the hierarchies of the calendars C_1 and C_2 are combinable hierarchies, then they uniquely determine a hierarchy combination.

Theorem 6.1 (existence of hierarchy combination) Suppose C_1 and C_2 are two calendars with hierarchies $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. If $\mathcal{T}_1$ and $\mathcal{T}_2$ are combinable hierarchies, then there exists exactly one $\mathcal{T}$ that is a hierarchy combination of $\mathcal{T}_1$ and $\mathcal{T}_2$.

Proof: Let $\mathcal{T}$ consists of the time units of $\mathcal{T}_1$ and $\mathcal{T}_2$, and let $\sqsubset_{\mathcal{T}}$ be defined as the transitive closure of $\sqsubset_{\mathcal{T}_1} \cup \sqsubset_{\mathcal{T}_2}$. We claim that if $T_i \sqsubset_{\mathcal{T}} T_j$ holds, then either $T_i \sqsubset_{\mathcal{T}_1} T_j$ or $T_i \sqsubset_{\mathcal{T}_2} T_j$ must also hold. In proof, by the definition of $\sqsubset_{\mathcal{T}}$, if $T_i \sqsubset_{\mathcal{T}} T_j$, then there is a sequence $T_{i_1}, \ldots, T_{i_m}$ of time units, such that $T_{i_1} = T_i$, $T_{i_m} = T_j$ and for all $1 < j \leq m$, either $T_{i_{j-1}} \sqsubset_{\mathcal{T}_1} T_{i_j}$ or $T_{i_{j-1}} \sqsubset_{\mathcal{T}_2} T_{i_j}$. Therefore, an easy induction, using the third part of Definition 6.2, shows that for all $1 < j \leq m$, either $T_{i_1} \sqsubset_{\mathcal{T}_1} T_{i_j}$ or $T_{i_1} \sqsubset_{\mathcal{T}_2} T_{i_j}$ must hold.

It thus follows that $\sqsubset_{\mathcal{T}}$ satisfies all the conditions of a hierarchy combination. It also follows that $\sqsubset_{\mathcal{T}}$ is indeed a partial order. In other words, there is no time unit T_i, such that $T_i \sqsubset_{\mathcal{T}} T_i$ (since if there were such a T_i, then either $T_i \sqsubset_{\mathcal{T}_1} T_i$ or $T_i \sqsubset_{\mathcal{T}_2} T_i$ would hold, contradicting the fact that both $\sqsubset_{\mathcal{T}_1}$ and $\sqsubset_{\mathcal{T}_2}$ are partial orders).

It remains to be shown that the partial order $\sqsubset_{\mathcal{T}}$ coincides with a tree, as required by the third bullet item in Definition 2.7. We will now show how to construct the tree $\mathcal{T}$. The root of $\mathcal{T}$ is the time unit that is the top of both $\mathcal{T}_1$ and $\mathcal{T}_2$ (by the first part of Definition 6.2, $\mathcal{T}_1$ and $\mathcal{T}_2$ share the same top time unit). For every other time unit T_i of $\mathcal{T}$, we define the parent of T_i as follows. First, we define the set $A(T_i) = \{T_j \mid T_i \sqsubset_{\mathcal{T}} T_j\}$; note that this set is not empty, since T_i is smaller than the top time unit. According to the fourth part of Definition 6.2, every two elements of the above set are comparable according to $\sqsubset_{\mathcal{T}}$ and, therefore, that set has a least element. We define the least element of $A(T_i)$ to be the parent of T_i.

We will now show that the tree constructed above satisfies the third bullet item in Definition 2.7; that is, T_j is an ancestor of T_i if and only if $T_i \sqsubset_{\mathcal{T}} T_j$. The "only if" direction is true, since by definition, T_j is a parent of T_i only if $T_i \sqsubset T_j$. The "if" direction follows by an easy induction, once we show that $A(T_i) - \{T_j\} = A(T_j)$, where T_j is the least element of $A(T_i)$. To show that, note that if $T_i \sqsubset_{\mathcal{T}} T_k$ ($k \neq j$), then $T_j \sqsubset_{\mathcal{T}} T_k$ (since T_j is the least element of $A(T_i)$) and, therefore, $A(T_i) - \{T_j\} \subseteq A(T_j)$. To prove the other direction of the containment, note that if $T_j \sqsubset_{\mathcal{T}} T_k$ then $T_i \sqsubset_{\mathcal{T}} T_k$ (since $T_i \sqsubset_{\mathcal{T}} T_j$) and, hence, $A(T_i) - \{T_j\} \supseteq A(T_j)$.

Finally, the uniqueness of $\mathcal{T}$ follows from Proposition 6.1. □

Example 6.5: Consider the calendars having the following hierarchies:

$$\mathcal{C}_a : \; T_3 \sqsubset T_2 \sqsubset T_1$$

$$\mathcal{C}_b: \; T_2 \sqsubset T_3 \sqsubset T_1.$$

These two hierarchies cannot be combined, because $T_3 \sqsubset T_2$ holds in $\mathcal{C}_a$, while $T_2 \sqsubset T_3$ holds in the second.

Proposition 6.2: Suppose C_1 and C_2 are two calendars with hierarchies $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. If there is a hierarchy combination $\mathcal{T}$ of $\mathcal{T}_1$ and $\mathcal{T}_2$, then Conditions 1, 3 and 4 of Definition 6.2 must hold.

Proof: Suppose hierarchy $\mathcal{T}$ is a hierarchy combination of $\mathcal{T}_1$ and $\mathcal{T}_2$. Condition 1 of Definition 6.2 follows from the fact that a hierarchy has exactly one infinite time unit.

To show that Condition 3 of Definition 6.2 is true, suppose that $T_i \sqsubset_{\mathcal{T}_m} T_j$ and $T_j \sqsubset_{\mathcal{T}_p} T_k$, where $m, p \in \{1, 2\}$. By Condition 2 of Definition 6.1, $T_i \sqsubset_{\mathcal{T}} T_j$ and $T_j \sqsubset_{\mathcal{T}} T_k$ must hold and, by the transitivity of $\sqsubset_{\mathcal{T}}$, it follows that $T_i \sqsubset_{\mathcal{T}} T_k$. By Condition 3 of Definition 6.1, either $T_i \sqsubset_{\mathcal{T}_1} T_k$ or $T_i \sqsubset_{\mathcal{T}_2} T_k$ must also hold.

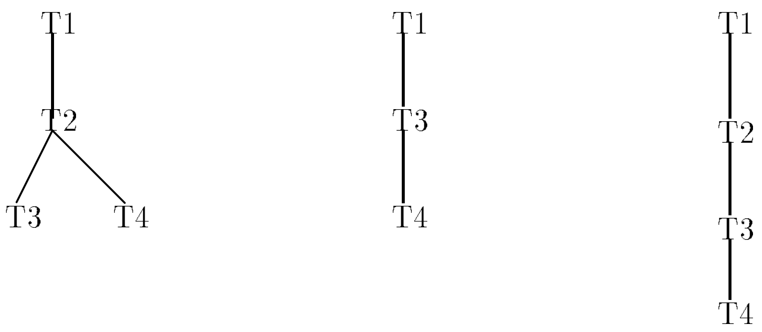
To show that Condition 4 of Definition 6.2 is true, suppose $T_i \sqsubset_{\mathcal{T}_m} T_j$ and $T_i \sqsubset_{\mathcal{T}_p} T_k$, where $m, p \in \{1, 2\}$. By Condition 2 of Definition 6.1, $T_i \sqsubset_{\mathcal{T}} T_j$ and $T_i \sqsubset_{\mathcal{T}} T_k$ must also hold and, hence, both T_j and T_k are ancestors of T_i in $\mathcal{T}$. Therefore, either $T_j \sqsubset_{\mathcal{T}} T_k$ or $T_k \sqsubset_{\mathcal{T}} T_j$ must hold and, so, Condition 3 of Definition 6.1 implies that either $T_j \sqsubset_{\mathcal{T}_q} T_k$ or $T_k \sqsubset_{\mathcal{T}_q} T_j$ must hold for some $q \in \{1, 2\}$. $\square$

Parts (a) and (b) of Figure 4 show two hierarchies that satisfy Conditions 1, 3 and 4 of combinable hierarchies (Definition 6.2). The hierarchy combination of these two hierarchies is shown in Part (c) of Figure 4. The one complete path in the hierarchy of Part (c) appears neither in Part (a) nor in Part (b). Therefore, none of the embedded calendars of the two original hierarchies is an embedded calendar of the combined hierarchy and, therefore, the combined hierarchy cannot be compatible with either one of the two original hierarchies. The next proposition shows that if all four conditions of Definition 6.2 are satisfied, then the situation shown in Figure 4 cannot occur.

Proposition 6.3: Suppose $\mathcal{T}_1$ and $\mathcal{T}_2$ are combinable hierarchies, and $\mathcal{T}$ is a hierarchy combination of $\mathcal{T}_1$ and $\mathcal{T}_2$. Let $\wp$ be any complete path in $\mathcal{T}$. Then $\wp$ is a complete path in either $\mathcal{T}_1$ or $\mathcal{T}_2$.

Proof: Consider the complete path $\wp$ of $\mathcal{T}$. We claim that either $\mathcal{T}_1$ or $\mathcal{T}_2$ contains all time units appearing on $\wp$. Suppose not. Therefore, there is a time unit T_i on $\wp$ that is not in $\mathcal{T}_1$ and, similarly, there is a time unit T_j on $\wp$ that is not in $\mathcal{T}_2$. Since both T_i and T_j are on the same path of $\mathcal{T}$, one of the two, say T_j, is an ancestor of the other, T_i. Thus, $T_i \sqsubset_{\mathcal{T}} T_j$. By Condition 3 of Definition 6.1, either $T_i \sqsubset_{\mathcal{T}_1} T_j$ or $T_i \sqsubset_{\mathcal{T}_2} T_j$. But this is impossible, since neither $\mathcal{T}_1$ nor $\mathcal{T}_2$ contains both T_i and T_j. Thus, we have shown that one of the two original hierarchies, say $\mathcal{T}_1$, contains all the time units appearing on the complete path $\wp$.

We will now show that $\wp$ is also a complete path of $\mathcal{T}_1$. So, consider a pair of time units T_i and T_j, such that T_j is the parent of T_i in $\wp$ and, hence, $T_i \sqsubset_{\mathcal{T}} T_j$. By Condition 3 of Definition 6.1,



(a) (b) (c)

Figure 4: Hierarchies that their combination include a new embedded path.

either $T_i \sqsubset_{\mathcal{T}_1} T_j$ or $T_i \sqsubset_{\mathcal{T}_2} T_j$ and, by Condition 2 of Definition 6.2, $T_i \sqsubset_{\mathcal{T}_1} T_j$. So, T_j is an ancestor of T_i in $\mathcal{T}_1$. Suppose the parent of T_i in $\mathcal{T}_1$ is T_k, where $k \neq j$. Therefore, $T_i \sqsubset_{\mathcal{T}_1} T_k \sqsubset_{\mathcal{T}_1} T_j$ and, by Condition 2 of Definition 6.1, $T_i \sqsubset_{\mathcal{T}} T_k \sqsubset_{\mathcal{T}} T_j$, contradicting the assumption that T_j is the parent of T_i in $\wp$. Therefore, we have shown that if T_j is the parent of T_i in $\wp$, then T_j is also the parent of T_i in $\mathcal{T}_1$.

It remains to be shown that the leaf, T_m, of the complete path $\wp$ is also a leaf of $\mathcal{T}_1$. Suppose not. Therefore, there is a node T_p that is a child of T_m in $\mathcal{T}_1$ and, hence, $T_p \sqsubset_{\mathcal{T}_1} T_m$. By Condition 2 of Definition 6.1, $T_p \sqsubset_{\mathcal{T}} T_m$, contradicting the fact that T_m is a leaf of $\mathcal{T}$. □

Proposition 6.4: Suppose $\mathcal{T}_1$ and $\mathcal{T}_2$ are combinable hierarchies, and $\mathcal{T}$ is a hierarchy combination of $\mathcal{T}_1$ and $\mathcal{T}_2$. If a complete path $\wp$ of $\mathcal{T}_1$ is not a complete path of $\mathcal{T}$, then it must be a complete path of $\mathcal{T}_2$.

Proof: By Proposition 6.5, all the nodes of $\wp$ lie on some path $\wp'$ of $\mathcal{T}$. If $\wp'$ and $\wp$ are not identical, then $\wp'$ cannot appear in $\mathcal{T}_1$ and, by Proposition 6.3, $\wp'$ must be a complete path of $\mathcal{T}_2$. □

We end this section with a description of how to construct a hierarchy combination $\mathcal{T}$ of the combinable hierarchies $\mathcal{T}_1$ and $\mathcal{T}_2$. Essentially, the construction was given in the proof of Theorem 6.1, where it was shown that the root of $\mathcal{T}$ is the infinite time unit that is shared by $\mathcal{T}_1$ and $\mathcal{T}_2$, and for every other time unit T_i, the parent of T_i in $\mathcal{T}$ is the least element of the set $A(T_i) = \{T_j \mid T_i \sqsubset_{\mathcal{T}} T_j\}$. The following proposition shows what that least element is.

Proposition 6.5: Suppose $\mathcal{T}_1$ and $\mathcal{T}_2$ are combinable hierarchies, and $\mathcal{T}$ is a hierarchy combination of $\mathcal{T}_1$ and $\mathcal{T}_2$.

- Let T_i be a finite time unit (i.e., T_i is not the top time unit). If T_i appears in both $\mathcal{T}_1$ and $\mathcal{T}_2$, then T_i has the same parent T_j in both.

- The parent of T_i in either $\mathcal{T}_1$ or $\mathcal{T}_2$ is also the parent of T_i in $\mathcal{T}$.

Proof: Let T_i be a node of $\mathcal{T}_1$ and suppose T_j is the parent of T_i in $\mathcal{T}_1$. Therefore, $T_i \sqsubset_{\mathcal{T}_1} T_j$ and, by Condition 2 of Definition 6.1, $T_i \sqsubset_{\mathcal{T}} T_j$. Now suppose that T_k, where $k \neq j$, is the parent of T_i in $\mathcal{T}$. Therefore, $T_i \sqsubset_{\mathcal{T}} T_k \sqsubset_{\mathcal{T}} T_j$. By Condition 3 of Definition 6.1 and Condition 2 of Definition 6.2, $T_i \sqsubset_{\mathcal{T}_1} T_k \sqsubset_{\mathcal{T}_1} T_j$, contradicting the assumption that T_j is the parent of T_i in $\mathcal{T}_1$. Therefore, we have shown that if T_i appears in $\mathcal{T}_1$, then it has the same parent in $\mathcal{T}_1$ and in $\mathcal{T}$. It can be shown similarly that if T_i appears in $\mathcal{T}_2$, then it has the same parent in $\mathcal{T}_2$ and in $\mathcal{T}$. □

If Condition 2 of Definition 6.2 is not satisfied (as in the case of Figure 4), then T_i may have distinct parents in $\mathcal{T}_1$ and $\mathcal{T}_2$ and, in this case, the parent of T_i in $\mathcal{T}$ is the least one (according to $\sqsubset_{\mathcal{T}}$) among its two original parents. More specifically, if there are two distinct parents in the original hierarchies, then one of the two hierarchies must impose an order among those parents.

6.3 Combining Calendars

In this section, we will describe how to combine two calendars. In addition to constructing the hierarchy combination of the two calendars, as discussed in the previous section, we also need a mapping that links time points of C_1 with time points of C_2, as defined next.

Definition 6.3 (linking function) Let C_1 and C_2 be a pair of calendars with combinable hierarchies $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. A *linking function* from C_1 to C_2 is a mapping f from the valid time points of C_1 to valid time points of C_2, such that

- If t and t' are valid time points of C_1 and $t <_{C_1} t'$, then $f(t) \leq_{C_1} f(t')$, and
- If t is a valid time point of C_1 and the time unit T_j is in both C_1 and C_2, then the time points t and $f(t)$ have the same value for T_j.

Intuitively, the first condition means that f is order preserving, and the second condition means that f is the identity on the time units that are common to both calendars. □

The above definition is similar to the definition of compatible linear calendars (Definition 2.6) as well as to the definition of compatible calendars (Definition 2.15). However, there is also an important difference; that is, f is not required to be a bijection (in particular, it may neither be surjective nor injective).

Proposition 6.6: Let C_1 and C_2 be two calendars with combinable hierarchies $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. Suppose all the time units of some embedded calendar E_2 of C_2 also appear in C_1. If there is a linking function f from C_1 to C_2, then it is unique and satisfies the following condition. Given a valid time point t_1 of C_1, the equality $f(t_1) = t_2$ holds if and only if t_2 is the unique valid time point of C_2, such that t_1 and t_2 are equal on E_2.

Proof: Suppose t_1 is a valid time point of C_1. By Definition 6.3, t_1 and $f(t_1)$ are equal on E_2. Since E_2 is an embedded calendar of C_2, the second bullet item of Definition 2.9 implies that there is exactly one valid time point t_2 of C_2, such that t_1 and t_2 are equal on E_2. Thus, $f(t_1) = t_2$ if and only if t_2 is the unique valid time point of C_2, such that t_1 and t_2 are equal on E_2. □

Definition 6.4 (combination of calendars) Let C_1 and C_2 be calendars with combinable hierarchies $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, and let $\mathcal{T}$ be the hierarchy combination of $\mathcal{T}_1$ and $\mathcal{T}_2$. Suppose f is a linking function from C_1 to C_2 and there is an embedded calendar E of C_1, such that the linear hierarchy of E forms a complete path of $\mathcal{T}$. The *combination* of C_1 and C_2 is a calendar C defined as follows.

1. The hierarchy of C is $\mathcal{T}$.
2. Every embedded calendar E' of C_1, such that the linear hierarchy of E' forms a complete path in $\mathcal{T}$, is an embedded calendars of C. (Note that E is one such embedded calendar).
3. The validity predicate of C, denoted $valid_C$, is defined by

$$valid_C(t) \equiv valid_{C_1}(\mathsf{pr}(t, C_1)) \wedge valid_{C_2}(\mathsf{pr}(t, C_2)) \wedge f(\mathsf{pr}(t, C_1)) = \mathsf{pr}(t, C_2)$$

   where $valid_{C_1}$ and $valid_{C_2}$ are the validity predicates of C_1 and C_2, respectively.

□

Note that the definition assumes the existence of a complete path $\wp$ in $\mathcal{T}$, such that $\wp$ is also an embedded calendar E of C_1 (in general, this assumption does not necessarily hold). The embedded calendar E is an embedded calendar of C.

Proposition 6.7: Definition 6.4 is correct, i.e., C is a well-defined calendar.

Proof: By Theorem 6.1, the hierarchy of C is well defined. We need to show that the validity predicate of C satisfies the three bullet items of Definition 2.9. The first and third items are satisfied, since the embedded calendars of C are also embedded calendars of C_1 and $valid_{C_1}(\mathsf{pr}(t, C_1))$ must be true when $valid_C(t)$ is true.

To show that the second item is satisfied, suppose t_e is a valid time point of an embedded calendar E of C. Since E is also an embedded calendar of C_1, the second item of Definition 2.9 implies that there is exactly one valid time point t_{c_1} of C_1, such that $\mathsf{pr}(t_{c_1}, E_1) = t_e$. Moreover, there is exactly one time point t of C that satisfies both $\mathsf{pr}(t, C_1) = t_{c_1}$ and $\mathsf{pr}(t, C_2) = f(t_{c_1})$; note that t exists, since f is the identity on the time units that are common to C_1 and C_2, and t is unique, since $f(t_{c_1})$ is unique for a given t_{c_1}. By the definition of a linking function (Definition 6.3), $f(t_{c_1})$ is a valid time point of C_2. Therefore, $valid_C(t)$ is true. Thus, we have shown that there is exactly one valid time point t of C, such that $\mathsf{pr}(t, E) = t_e$. □

Proposition 6.8: Suppose C_1 and C_2 are two calendars with combinable hierarchies $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, and validity predicates $valid_{C_1}$ and $valid_{C_2}$, respectively. Let calendar C be the combination of C_1 and C_2. If all the time units of some embedded calendar E_2 of C_2 also appear in C_1, then $valid_C(t)$ is equivalent to the conjunction $valid_{C_1}(\mathsf{pr}(t, C_1)) \wedge valid_{C_2}(\mathsf{pr}(t, C_2))$.

Proof: Follows from Definition 6.4 and Proposition 6.6. □

Proposition 6.9: If calendar C is the combination of calendars C_1 and C_2, then C is compatible with calendar C_1.

Proof: Follows since C and C_1 have the same embedded calendars. □

Proposition 6.10: Suppose calendar C is the combination of calendars C_1 and C_2. If the linking function f is bijective, then C is compatible with calendar C_2.

Proof: Let E_1 be an embedded calendar of C (and, hence, E_1 is also an embedded calendar of C_1), and let E_2 be an embedded calendar of C_2. Consider the mapping f' defined as follows (intuitively, f' is the restriction of f to E_1 and E_2). Given a valid time point t_{e_1} of E_1, we define $f'(t_{e_1}) = t_{e_2}$ if $f(t_{c_1}) = t_{c_2}$, where t_{c_1} is the unique valid time point of C_1, such that $\mathsf{pr}(t_{c_1}, E_1) = t_{e_1}$, and $\mathsf{pr}(t_{c_2}, E_2) = t_{e_2}$.

We will show that f' satisfies the conditions of Definition 2.6. First, f' is a bijection from the valid time points of E_1 to the valid time point of E_2, since f is a bijection and for every valid time point t_{e_i} ($i \in \{1, 2\}$) of E_i, there is exactly one valid time point t_{c_i} of C_i, such that $\mathsf{pr}(t_{c_i}, E_i) = t_{e_i}$.

Second, f' is the identity on the time units that are common to E_1 and E_2, since f is the identity on the time units that are common to C_1 and C_2.

Third, suppose t_{e_1} and t'_{e_1} are valid time points of E_1, such that $t_{e_1} <_{E_1} t'_{e_1}$. Since E_1 is an embedded calendar of C_1, Definition 2.11 implies that $t_{c_1} <_{C_1} t'_{c_1}$, where t_{c_1} and t'_{c_1} are the unique time points of C_1, such that $\mathsf{pr}(t_{c_1}, E_1) = t_{e_1}$ and $\mathsf{pr}(t'_{c_1}, E_1) = t'_{e_1}$, respectively. Since f is a linking function and bijective, $f(t_{c_1}) <_{C_1} f(t'_{c_1})$. Since E_2 is an embedded calendar of C_2, Definition 2.11 implies that $\mathsf{pr}(f(t_{c_1}), E_2) <_{C_2} \mathsf{pr}(f(t'_{c_1}), E_2)$. By definition of f', it follows that $f'(t_{e_1}) <_{C_1} f'(t'_{e_1})$.

Thus, the function f' is a bijection showing that E_1 and E_2 are compatible linear calendars and, hence, C_1 and C_2 are compatible calendars. □

Finally, it turns out that if C is the combination of two periodic calendars C_1, C_2 w.r.t a *bijective* linking function, then C itself is periodic. In other words, when combining two periodic calendars using a bijective linking function, the combination is guaranteed to be periodic as well.

Proposition 6.11: Suppose C_1, C_2 are two periodic calendars that are combinable and have combination C. If the linking function f is bijective, then C is also periodic.

Proof: Suppose $T_1, ..., T_n$ are the time units of C, and p_1, p_2 are the periods of the top time unit of C_1 and C_2 respectively. We denote by p, the least common multiple of p_1 and p_2. We will show that a time point $t = (t_1, ..., t_n)$ is valid for C if and only if $t^p = (t_1, ..., t_n +_{T_n} p)$ is valid in C. Since C is the combination of C_1 and C_2, according to definition 6.4 it enough to show that

$$valid_{C_1}(\mathsf{pr}(t, C_1)) \wedge valid_{C_2}(\mathsf{pr}(t, C_2)) \wedge f(\mathsf{pr}(t, C_1)) = \mathsf{pr}(t, C_2)$$

iff

$$valid_{C_1}(\mathsf{pr}(t^p, C_1)) \wedge valid_{C_2}(\mathsf{pr}(t^p, C_2)) \wedge f(\mathsf{pr}(t^p, C_1)) = \mathsf{pr}(t, C_2)$$

.

Since T_n is the top time value of both C_1 and C_2 and p is a multiple of the periods of the top time unit of C_1 and C_2, it is easy to see that $valid_{C_1}(\mathsf{pr}(t, C_1))$ iff $valid_{C_1}(\mathsf{pr}(t^p, C_1))$ and $valid_{C_2}(\mathsf{pr}(t, C_2))$ iff $valid_{C_2}(\mathsf{pr}(t^p, C_2))$.

It is left to show that $f(\mathsf{pr}((t_1, ..., t_n), C_1)) = \mathsf{pr}((t_1, ..., t_n), C_2)$ iff $f(\mathsf{pr}((t_1, ..., t_n +_{T_n} p), C_1)) = \mathsf{pr}((t_1, ..., t_n +_{T_n} p), C_2)$. Since T_n is a time unit of both C_1 and C_2, the value of T_n in $\mathsf{pr}((t_1, ..., t_n), C_1)$ and $\mathsf{pr}((t_1, ..., t_n), C_2)$ is the same (i.e., t_n), and similarly the value of T_n is the same for both $\mathsf{pr}((t_1, ..., t_n +_{T_n} p), C_1)$ and $\mathsf{pr}((t_1, ..., t_n +_{T_n} p), C_2)$ (i.e., $t_n +_{T_n} p$). Furthermore, since p is a multiple of p_1, the number of valid time points of C_1 with the value t_n of T_n is equal to the number of valid time points of C_1 with the value $t_n +_{T_n} p$. Similarly, for C_2. As f is a linking function, and thus satisfies the first bullet of Definition 6.3 and as f is bijective, we can conclude that $f(\mathsf{pr}(t, C_1)) = \mathsf{pr}(t, C_2)$ iff $f(\mathsf{pr}(t^p, C_1)) = \mathsf{pr}(t, C_2)$. □

6.4 Realizing Linking Functions via Constraints

As already stated earlier, the combination of two calendars depends upon the existence of a linking function specifying the semantic relationship between the two calendars. Usually, a linking function cannot be inferred automatically from the syntax of the original calendars; it must be specified explicitly by the user or agent that wishes to combine the two calendars. It turns out that constraints form a natural way of expressing linking functions. In fact, we will show that if calendar C is the combination of calendars C_1 and C_2, then we can automatically obtain the realization of the validity predicate of C from the realizations (via constraints) of the linking function and the validity predicates of C_1 and C_2.

Definition 6.5 (restriction of a linking function) Suppose f is a linking function from C_1 to C_2, and E_1 and E_2 are embedded calendars of C_1 and C_2, respectively. The *restriction* of f to E_1 and E_2 is a linking function from E_1 to E_2 that is defined as follows: $f'(t_{e_1}) = t_{e_2}$ if $f(t_{c_1}) = t_{c_2}$, where t_{c_1} is the unique valid time point of C_1, such that $\mathsf{pr}(t_{c_1}, E_1) = t_{e_1}$, and $\mathsf{pr}(t_{c_2}, E_2) = t_{e_2}$. □

The following proposition shows that for the purpose of realizing the validity predicate of C, it is sufficient to realize the restriction of f to E_1 and E_2.

Proposition 6.12: Let $valid_C$, $valid_{C_1}$ and $valid_{C_2}$ be the validity predicates of the calendars C, C_1 and C_2, respectively, that are mentioned in the above definition. Then

$$valid_C(t) \iff valid_{C_1}(\mathsf{pr}(t, C_1)) \wedge valid_{C_2}(\mathsf{pr}(t, C_2)) \wedge f'(\mathsf{pr}(t, E_1)) = \mathsf{pr}(t, E_2)$$

Proof: By Definition 6.5 and Proposition 2.3, if $f(\mathsf{pr}(t, C_1)) = \mathsf{pr}(t, C_2)$, then $f'(\mathsf{pr}(t, E_1)) = \mathsf{pr}(t, E_2)$. Thus,

$$valid_C(t) \Longrightarrow valid_{C_1}(\mathsf{pr}(t, C_1)) \wedge valid_{C_2}(\mathsf{pr}(t, C_2)) \wedge f'(\mathsf{pr}(t, E_1)) = \mathsf{pr}(t, E_2)$$

Conversely, if the conjunction

$$valid_{C_1}(\mathsf{pr}(t, C_1)) \wedge valid_{C_2}(\mathsf{pr}(t, C_2)) \wedge f'(\mathsf{pr}(t, E_1)) = \mathsf{pr}(t, E_2)$$

is true, then by Definition 6.5, $f(\mathsf{pr}(t, C_1)) = \mathsf{pr}(t, C_2)$, since $\mathsf{pr}(t, C_i)$ ($i \in \{1, 2\}$) is the unique valid time point of C_i that is equal to $\mathsf{pr}(t, E_i)$ on E_i. □

Definition 6.6 (realizing a linking function via constraints) Suppose calendar C is a combination of calendars C_1 and C_2, according to a linking function f. Let $T_1, \ldots, T_n$ be all the time units that appear in either C_1 and C_2, where T_n is the top time unit that is shared by both C_1 and C_2. A set $\{(\Theta_1, \Upsilon_1), \ldots, (\Theta_m, \Upsilon_m)\}$ of pairs of constraints is said to *realize the linking function f* if

- Θ_i is a TUCL constraint that has a single TPCL variable t and only the TUCL variables $t.T_1, \ldots, t.T_{n-1}$ (but not $t.T_n$) may appear in Θ_i,
- Υ_i is a top constraint involving only the TUCL variable $t.T_n$,
- For all $1 \leq i < j \leq n$, the TUCL constraints Θ_i and Θ_j have no common solutions, and
- $f(\mathsf{pr}(t', C_1)) = \mathsf{pr}(t', C_2)$ if and only if there is a pair (Θ_i, Υ_i) $(1 \leq i \leq m)$, such that the assignment $t = t'$ is a solution of $\Theta_i \wedge \Upsilon_i$. □

Example 6.6: A very simple example showing how to realize a linking function is given below. Consider the two calendars:

$$C_1 : \ \texttt{doy} \sqsubseteq \texttt{year}$$

$$C_2 : \ \texttt{month} \sqsubseteq \texttt{year}.$$

As these calendars are linear, each one is also its own embedded calendar. Intuitively, we would like to construct a linking function from C_1 to C_2, because the day of year uniquely determines the month, but not the other way round. Thus, we can realize this linking function using a set of 23 pairs of constraints $\{(\Theta_1, \Upsilon_1), \ldots, (\Theta_{23}, \Upsilon_{23})\}$. The first three pairs are given below.

$$\begin{aligned}
\Theta_1 &= 1 \leq t.doy \leq 31 \wedge t.month = January. \\
\Upsilon_1 &= t.year \geq 0 \\
\Theta_2 &= (32 \leq t.doy \leq 59) \wedge t.month = February. \\
\Upsilon_2 &= t.year \geq 0 \\
\Theta_3 &= t.doy = 60 \wedge t.month = February \\
\Upsilon_3 &= t.year \bmod 4 = 0.
\end{aligned}$$

The other pairs may be similarly expressed, building upon (Θ_1, Υ_1), (Θ_2, Υ_2) and (Θ_3, Υ_3).

6.5 Realizing the Validity Predicate of a Combined Calendar

Suppose calendar C is a combination of calendars C_1 and C_2. In this section, we show how to construct a realization (according to Definition 4.1) of the validity predicate of C from realizations of the linking function from C_1 to C_2 and the validity predicates of C_1 and C_2. Note that by Proposition 6.12, it is sufficient to use a realization of the restriction of f to some embedded calendars E_1 and E_2 of C_1 and C_2, respectively.

Theorem 6.2 (realizing the validity predicate of a Combined Calendar) Suppose calendar C is a combination of calendars C_1 and C_2, according to a linking function f. Let $T_1, \ldots, T_n$ denote the time units of C, where T_n is the top time unit, and let $valid_C$, $valid_{C_1}$ and $valid_{C_2}$ denote the validity predicates of C, C_1 and C_2, respectively. Suppose f is realized by the set

$$\{(\Theta_1, \Upsilon_1), \ldots, (\Theta_k, \Upsilon_k)\}$$

and $valid_{C_i}$ $(i = 1, 2)$ is realized by the set

$$\{(\Psi_1^i, \Phi_1^i), \ldots, (\Psi_{m_i}^i, \Phi_{m_i}^i)\}.$$

Moreover, suppose that (by an appropriate renaming, if necessary) all of the above realizations use the same TPCL variable t. Then the following set of pairs realizes $valid_C$.

$$\{(\Psi_u^1 \wedge \Psi_v^2 \wedge \Theta_w \,,\, \Phi_u^1 \wedge \Phi_v^2 \wedge \Upsilon_w) \mid 1 \leq u \leq m_1, 1 \leq v \leq m_2, 1 \leq w \leq k\}$$

Proof: We need to show that the four conditions of Definition 4.1 hold.

1. Since Ψ_u^1, Ψ_v^2 and Θ_w are all TUCL constraints involving only the TUCL variables $t.T_1, \ldots, t.T_{n-1}$, so is $(\Psi_u^1 \wedge \Psi_v^2 \wedge \Theta_w)$.

2. Since Φ^1_u, Φ^2_v and Υ_w are all top constraints (involving only the TUCL variable $t.T_n$), so is $(\Phi^1_u \wedge \Phi^2_v \wedge \Upsilon_w)$.

3. We need to show that $(\Psi^1_{u_1} \wedge \Psi^2_{v_1} \wedge \Theta_{w_1})$ and $(\Psi^1_{u_2} \wedge \Psi^2_{v_2} \wedge \Theta_{w_2})$, where either $u_1 \neq u_2$, $v_1 \neq v_2$ or $w_1 \neq w_2$, have no common solutions. It is true, since none of the following pairs can have common solutions.

   - $\Psi^1_{u_1}$ and $\Psi^1_{u_1}$, where $u_1 \neq u_2$.
   - $\Psi^2_{v_1}$ and $\Psi^2_{v_2}$, where $v_1 \neq v_2$.
   - Θ_{w_1} Θ_{w_2}, where $w_1 \neq w_2$.

4. Suppose $t' = (t'_1, \ldots, t'_n)$ is a valid time point of C; that is, $valid_C(t')$ is true. By definition of $valid_C$, it follows that $valid_{C_1}(\mathsf{pr}(t', C_1))$, $valid_{C_2}(\mathsf{pr}(t', C_2))$ and $f(\mathsf{pr}(t', C_1)) = \mathsf{pr}(t', C_2)$ are all true.

   - Since $valid_{C_1}(\mathsf{pr}(t', C_1))$ is true, there is a u, such that $t = \mathsf{pr}(t', C_1)$ is a solution of $\Psi^1_u \wedge \Phi^1_u$.
   - Since $valid_{C_2}(\mathsf{pr}(t', C_2))$ is true, there is a v, such that $t = \mathsf{pr}(t', C_2)$ is a solution of $\Psi^2_v \wedge \Phi^2_v$.
   - Since $f(\mathsf{pr}(t', C_1)) = \mathsf{pr}(t', C_2)$ is true, there is a w, such that $t = t'$ is a solution of $\Theta_w \wedge \Upsilon_w$.

   Therefore, there are u, v and w, such that $t = t'$ is a solution of $\Psi^1_u \wedge \Psi^2_v \wedge \Theta_w \wedge \Phi^1_u \wedge \Phi^2_v \wedge \Upsilon_w$.

   Conversely, suppose $t = t'$ is a solution of $\Psi^1_u \wedge \Psi^2_v \wedge \Theta_w \wedge \Phi^1_u \wedge \Phi^2_v \wedge \Upsilon_w$. Then the following must hold.

   - $t = \mathsf{pr}(t', C_1)$ is a solution of $\Psi^1_u \wedge \Phi^1_u$ and, hence, $valid_{C_1}(\mathsf{pr}(t', C_1))$ is true.
   - $t = \mathsf{pr}(t', C_2)$ is a solution of $\Psi^2_v \wedge \Phi^2_v$ and, hence, $valid_{C_2}(\mathsf{pr}(t', C_2))$ is true.
   - $t = t'$ is a solution of $\Theta_w \wedge \Upsilon_w$ and, hence, $f(\mathsf{pr}(t', C_1)) = \mathsf{pr}(t', C_2)$ is true.

   Therefore, $valid_C(t')$ is true.

$\square$

6.6 Realizing the Successor of a Combined Calendar

In order to complete the realization of calendar C that combines calendars C_1 and C_2, we need to realize the successor of C. As the following propositions shows, the realization of the successor of C can be constructed from the realization of $valid_C$ and the realization of successor of C_1.

The following result will be helpful in realizing successors. Suppose t is a time point in the combined calendar, and suppose (without loss of generality) that the embedded calendar of the combination is originally from C_1. Then we may *project* t onto C_1 and find the successor (in C_1) of this projected time point. We may then *de-project* this time point back to the combined calendar and obtain the successor of t. The following result says that this claim is indeed correct.

Proposition 6.13: Suppose calendar C is the combination of calendars C_1 and C_2. Then the following equality holds.

$$t \oplus_C 1 = \mathsf{dp}(\mathsf{pr}(t, C_1) \oplus_{C_1} 1).$$

Proof: Let E_1 be an embedded calendar of C (and, hence, also of C_1). Consider a valid time point t of C. By the first equality of Proposition 2.5,

$$\mathsf{pr}(t, C_1) \oplus_{C_1} 1 \quad = \quad \mathsf{dp}_{C_1}(\mathsf{pr}(t, E_1) \oplus_{E_1} 1)$$

since $\mathsf{pr}(t, C_1)$ is a valid time point of C_1 and $\mathsf{pr}(\mathsf{pr}(t, C_1), E_1) = \mathsf{pr}(t, E_1)$. By applying de-projection to both sides of the above equality, we get the following:

$$\mathsf{dp}_C(\mathsf{pr}(t, C_1) \oplus_{C_1} 1) \quad = \quad \mathsf{dp}_C(\mathsf{dp}_{C_1}(\mathsf{pr}(t, E_1) \oplus_{E_1} 1)).$$

By the second bullet of Definition 2.9,

$$\mathsf{dp}_C(\mathsf{dp}_{C_1}(\mathsf{pr}(t, E_1) \oplus_{E_1} 1)) \quad = \quad \mathsf{dp}_C(\mathsf{pr}(t, E_1) \oplus_{E_1} 1),$$

since both sides of the equality have the same projection on the embedded calendar E_1. From the last two equations, we get the following:

$$\mathsf{dp}_C(\mathsf{pr}(t, C_1) \oplus_{C_1} 1) \quad = \quad \mathsf{dp}_C(\mathsf{pr}(t, E_1) \oplus_{E_1} 1).$$

By the first equality of Proposition 2.5,

$$t \oplus_C 1 \quad = \quad \mathsf{dp}_C(\mathsf{pr}(t, E_1) \oplus_{E_1} 1),$$

and from the last two equations, we get the following.

$$t \oplus_C 1 \quad = \quad \mathsf{dp}_C(\mathsf{pr}(t, C_1) \oplus_{C_1} 1)$$

□

We are now able to state how the successor function of a combined calendar is realized.

Theorem 6.3 (realization of a successor of a combined calendar) Suppose calendar C is a combination of calendars C_1 and C_2, according to a linking function f from C_1 to C_2, and $T_1, \ldots, T_n$ are the time units of C, where T_n is the top time unit. Let

$$\{(\Theta_1, \Upsilon_1), \ldots, (\Theta_k, \Upsilon_k)\}$$

be the set of pairs that realizes the successor of C_1 (according to Definition 4.2). Similarly, let

$$\{(\Phi_1, \Psi_1), \ldots, (\Phi_m, \Psi_m)\}$$

be the realization of the validity predicate of C. Moreover, suppose that (by an appropriate renaming) the TPCL variable in each pair (Φ_i, Ψ_i) is t^+. Then the successor function of C is realized by the following set:

$$\left\{\left(\Theta_i\,,\ \Upsilon_i \wedge \left(\vee_{j=1}^{m} \Phi_j \wedge \Psi_j\right)\right) \mid 1 \le i \le k\right\}.$$

Proof: Note that both Θ_i and Υ_i only have TUCL variables of the time units that appear in C_1. However, we will consider solutions of Θ_i and Υ_i that are defined on all the time units $T_1, \ldots, T_n$.

We need to show that the above set of pairs satisfies the four conditions of Definition 4.2. It is easy to see that the first two conditions are satisfied. In order to prove that the third condition is satisfied, suppose $t = t'$ is a solution of both Θ_p and Θ_q, where $p \neq q$ and t' is a valid time point of C. Since Θ_p and Θ_q only have time units of C_1, it follows that $t = \mathsf{pr}(t', C_1)$ is also a solution of both Θ_p and Θ_q; moreover, $\mathsf{pr}(t', C_1)$ is a valid time point of C_1, since C and C_1 share some embedded calendar. However, the pairs (Θ_p, Υ_p) and (Θ_q, Υ_q) are in a set realizing the successor of C_1 and, therefore, $t = \mathsf{pr}(t', C_1)$ cannot be a solution of both Θ_p and Θ_q. This contradiction implies that the third condition of Definition 4.2 is satisfied.

In order to prove the fourth condition, suppose $t = t'$ is a solution of Θ_i, where t' is a valid time point of C. Thus, $\mathsf{pr}(t', C_1)$ is a valid time point of C_1 and also a solution of Θ_i.

Since the set $\{(\Phi_1, \Psi_1), \ldots, (\Phi_m, \Psi_m)\}$ realizes the validity predicate of C and the only TPCL variable appearing in this set has been renamed to t^+, it follows that every solution of $\Upsilon_i[t/t'] \wedge \left(\vee_{j=1}^{m} \Phi_j \wedge \Psi_j\right)$ is a valid time point of C.

Suppose $\Upsilon_i[t/t'] \wedge \left(\vee_{j=1}^{m} \Phi_j \wedge \Psi_j\right)$ has two solutions $t^+ = t_1$ and $t^+ = t_2$. Since only time units of C_1 appear in Υ_i, both $\mathsf{pr}(t_1, C_1)$ and $\mathsf{pr}(t_2, C_1)$ are solutions of $\Upsilon_i[t/\mathsf{pr}(t', C_1)]$. Since the pair (Θ_i, Υ_i) is in a set realizing the successor of C_1, it follows from the fourth condition of Definition 4.2 that $\mathsf{pr}(t_1, C_1) = \mathsf{pr}(t_2, C_1)$ and $\mathsf{pr}(t', C_1) \oplus_{C_1} 1 = \mathsf{pr}(t_1, C_1)$. Since C and C_1 share some embedded calendar and both t_1 and t_2 are valid time points of C, it follows from $\mathsf{pr}(t_1, C_1) = \mathsf{pr}(t_2, C_1)$ that $t_1 = t_2$. By Proposition 6.13, $\mathsf{pr}(t', C_1) \oplus_{C_1} 1 = \mathsf{pr}(t_1, C_1)$ implies $t_1 = t' \oplus_C 1$. Thus, we have shown that the fourth condition of Definition 4.2 is satisfied. □

6.7 Combining non Compatible Calendars

Suppose C_1 and C_2 are two calendars that we wish to combine. The preceding sections specify how to combine such calendars, even if these calendars are not compatible with one another. In this section, we briefly describe how our calendar combination techniques vary, depending upon whether C_1 and C_2 are compatible or not.

For C_1 and C_2 to be *compatible*, there must be a *bijection* between time points determined by their respective embedded calendars. This definition of compatibility bears a close resemblance to the definition of a *linking function* between such embedded calendars. However, linking functions may *not* be bijections, while functions establishing compatibility are bijections.

Suppose we return to the calendars in Figure 3 (cf. also Example 6.6), and consider the two incompatible calendars shown in (a) and (b) of that figure, i.e.,

$$C_1 : \ \texttt{doy} \sqsubseteq \texttt{year}$$

$$C_2 : \ \texttt{month} \sqsubseteq \texttt{year}.$$

Their combination is captured by Figure 3(c) where the embedded calendar is $doy \sqsubseteq year$. Notice that there is a linking function between the calendars of Figure 3(a) and (b); however, this function is not a bijection. Intuitively, given a year, and a day in the year, we can uniquely determine the month in which that day falls; however, given a year and a month, we cannot uniquely determine a day; this is what prevents the existence of bijection between the calendars of Figure 3(a) and (b).

Consequently, when C_1 and C_2 are incompatible, but there exists a linking function between their respectively embedded calendars, we are faced with the following problem (which does not occur if the calendars are compatible). A time-point, t, according to C_2 determines a *set*, $\chi(t)$, of time points in the combined calendar. Thus, if a user who is familiar with calendar C_2 wishes to ask whether a particular proposition p is true at time t (in calendar C_2), there are several possible ways of answering this query such as the three shown below:

- Answer "yes" if p is true at all time points in $\chi(t)$ and "no" otherwise;
- Answer "yes" if p is true at some time point in $\chi(t)$ and "no" otherwise;
- Answer "don't know" if p is true at some time point in $\chi(t)$, and p is false at some time point in $\chi(t)$.

For example, returning to the calendars in Figure 3, the user may ask the query "Did John visit Turkey in June 1996." June 1996 is a perfectly good time-point w.r.t. the calendar of Figure 3 (b); however, if the data's temporal aspect is specified by the combined calendar shown in Figure 3(c), then its not clear whether it is appropriate to answer "yes" when John was in Turkey for a few (but not all) days in June 1996.

Another situation that could occur is when two intelligent agents use two different calendars (locally), but communicate with each other through a combined calendar. Thus, for instance, a planning agent may use the calendar of Figure 3(a), while a database agent uses that of Figure 3(b) – they communicate with each other through the calendar of Figure 3(c). Now suppose the planning agent asks the database agent whether query Q holds at time t where t is a time-point according to the combined calendar. Thus, for example, the planning agent may ask: "Were the steel widgets shipped on $doy = 76$, $month = March$, $year = 1996$." However, the database agent in this case only has information represented with calendar C_2. The answer provided by the database agent is (similarly to previous discussion) contingent upon the semantics of the database agent. If it has information that steel widgets were shipped in March 1996, this semantics should tell us whether this refers to a truth about *all days* in March 1996, or just *some* day in March 1996. Thus, in this case, the semantics of the database agent's calendar will determine the response. In contrast, if the database agent merely information that the steel widgets were shipped in June 1996, then it can safely answer "no" to the planning agent's query, independently of the forall/existential semantics.

In this paper, we are primarily interested in the problem of characterizing, representing, and combining calendars. The problem of *processing queries using such (possibly combined) calendars* will build upon these definitions and will be discussed in a future paper.

The above situation, of course, does not arise, if the two calendars are compatible.

7 Combining Calendars with Different Top Time Units

In the preceding sections, we have shown how to combine different calendars under the assumption that the calendars being combined share the same top time unit. However, certain applications may require the combination of two calendars with different top time units. For example, the Gregorian calendar and the Hebrew calendar have different notions of "year" and hence, they have different top time units. Any application which requires combining the Gregorian calendar and the Hebrew calendar is not supported by the definitions of the previous sections. Fortunately, there is a simple extension of the preceding sections that allow us to combine calendars with different top time units.

Definition 7.1 (temporal forest of time units) A collection of distinct time units $T_1, \ldots, T_n$ with a partial order $\sqsubset$ (on the time units) forms a *temporal forest* if there is a set of trees $\{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$, such that

- Each $\mathcal{T}_i$, $1 \leq i \leq m$, is a hierarchy of time units.
- For each $i \neq j$, $\mathcal{T}_i$ and $\mathcal{T}_j$ have disjoint sets of time units.
- Each time-unit T_r, appears in one of the trees $\mathcal{T}_i$.

The Disjoint-ness Assumption. We will require that whenever we wish to combine two calendars having different top time units, that those two calendars have disjoint sets of time-units.

We now show, through examples, why this is a reasonable assumption and furthermore, demonstrate the applicability of the notion of temporal forest.

Example 7.1: Let us consider the combination of the Hebrew calendar and the Gregorian calendar and the Hebrew calendar[1]. The hierarchy associated with the Gregorian calendar is:

$$day \sqsubseteq month \sqsubseteq year$$

while that associated with the Hebrew calendar is

$$hday \sqsubseteq hmonth \sqsubseteq hyear.$$

Notice that the set of time values associated with *year* and *hyear* and the ordering on those time values are identical. However, these two time units must be treated differently for the following reasons:

1. First, the current year in the Gregorian calendar is 1997, while in the Hebrew calendar it is 5757. Thus, the same "physical" point in time is captured in the two calendars with two different time unit values, and thus, some distinction must be drawn between the Hebrew notion of year, and the corresponding Gregorian notion.
2. Second, the number of months (*hmonth*) that constitute a Hebrew year (*hyear*) varies from year to year – a phenomenon that does not occur in the Gregorian calendar. Similarly, the number of days (*hday*) that constitutes a Hebrew month (*hmonth*) is either 29 or 30; in contrast, the values of the time unit *day* in the Gregorian calendar is $\{1, \ldots, 31\}$.
3. Of course, the Gregorian and Hebrew calendars use different time unit values for the time units *month*, *hmonth* and *day*, *hday*.

This explains why the set of time-units associated with the Hebrew and Gregorian calendars are totally disjoint. An analogous observation applies also to the Moslem and Hindu calendars.

Suppose now, that an application requires integrating the Hebrew and Gregorian calendars. For example, in Israel, checks are usually dated with Gregorian dates, but sometimes, they may be dated with Hebrew dates. To reason with this kind of heterogeneity, we must be able to "map" both the Gregorian and the Hebrew dates into a combined calendar. This is done, as before, in three steps.

1. **Forest Construction:** Instead of constructing a hierarchy (as we did previously when combining calendars with the same top time unit), we will merge the hierarchies of two calendars with different time units into a forest.
2. **Selecting Embedded Calendar:** This is the same step as before.

[1]The reader may be interested to know that the Gregorian calendar is based entirely on the sun; the Islamic calendar is based entirely on the moon, while the Hebrew calendar is a hybrid, based largely on the moon, but "adjusted" to take care of some solar phenomena.

3. **Merging Validity Specification:** This too, remains identical to what we had before.

The forest associated with merging the Gregorian and Hebrew calendars just consists of the two trees originally associated with the Gregorian and Hebrew calendars.

The definition of combination of two calendars is almost identical to Definition 6.4 except that it now generates a forest structure, rather than a hierarchy. We provide it below for the sake of completeness.

Definition 7.2 (forest combination of calendars with different top time units) Suppose C_1 and C_2 are disjoint calendars with different top time units. The *forest combination*, FC, of C_1 and C_2 is given by:

1. **Forest of** FC**:** This consists of two trees, T_1, T_2 viz. the hierarchy of C_1 and the hierarchy of C_2.
2. **Embedded Calendar of** FC**:** There exists a complete path $\wp$ in some T_i, $i = 1$ or 2, such that:
   (a) There exists an embedded calendar E of C_i (i is either 1 or 2) whose linear hierarchy is $\wp$ and
   (b) there exists a linking function f between C_i and C_j ($i \neq j$) w.r.t. E and E' where E' is an embedded calendar of C_j. (Note that for this definition to "work", we need to use a modified definition of linking function from that given earlier, so as to account for the fact that C_i and C_j have different top time units. Thus, the metavariable Υ_i in the second bullet of Definition 6.6, may now involve two top time-units, one from each of the calendars being combined.)
   (c) E is the embedded calendar of FC .
3. **Validity Predicate of** FC**:** The validity predicate, $valid_{FC}$ associated with FC is defined as follows:
   (a) If $valid_{FC}(t)$ is true, then $\mathsf{pr}(t, \mathcal{C}_1)$ is valid w.r.t. $\mathcal{C}_1$ and $\mathsf{pr}(t, \mathcal{C}_2)$ is valid w.r.t. $\mathcal{C}_2$.
   (b) If $valid_{FC}(t)$ is true, then $f(\mathsf{pr}(t, E)) = \mathsf{pr}(t, E')$.

This completely defines how to combine two calendars with disjoint hierarchies. Notice that the validity predicate is unchanged from Definition 6.4. When considering the *realization* of forest combinations, we note the following elementary differences.

1. In the second bullet of Definition 6.6, Υ_i, which is used in the realization of the linking function, may now involve two top time-units, one from each of the calendars being combined.

2. The constraint Φ_i used in the realization of the validity predicate (as specified in the second bullet of Definition 4.1) may now involve two top time units. This is because of two reasons: the first of course, is that there are two top time units in the two calendars being combined. The second is that according to Theorem 6.2, the pair that realizes the validity predicate of the combination involves one Φ_u^1 from one calendar, and Φ_v^2 from the other, each having a different top time unit in it.

8 Related Work

There has been extensive work in the area of temporal representation and reasoning in both the AI and database communities. We classify the related work into three parts – work on temporal constraints in AI, work on temporal databases, and work on calendars. In order to place our work in the context of extensive research in these areas, we focus discussion on selected works; the reader interested in detailed surveys of temporal representation and reasoning may consult Chomicki [5] or Gabbay et. al.'s excellent survey book[10].

Temporal Constraints in AI: Allen[1] was one of the first to develop a logical framework for reasoning with interval time for actions. He presented 13 relationships between intervals of time and proposed a logic to reason with these interval relationships. Kautz and Vilain [26] study three types of primitive relations between time points and allow the expression of disjunctive information between time points. They show that under certain conditions, relations between interval can be captured in their framework. Dean and McDermott [8] developed a technique for reasoning about point-based temporal databases. In particular, they develop a framework called Time Map Management (TMM) to answer queries involving temporal inference, as well as to perform maintenance tasks when the temporal database is updated. Dechter et. al. [9] were one of the first to apply general purpose constraint solving techniques (such as the Floyd-Warshall shortest path algorithm) to reason about temporal relationships. Van Beek[2] develops algorithms that finds consistent scenarios (as defined in his paper) based on Allen's interval algebra and Vilain and Kautz's approach. Some other works dealing with events and actions in the AI field include Kowalski and Sergot's event calculus[18], Schwalb et. al.'s [24] work on events, efficient algorithms for temporal constraints by Ladkin and Reinefeld [21], and studies of the complexity of temporal reasoning [11].

In all these frameworks, time points are assumed to be either integers or reals, and intervals are bounded by time points. However, these time points are not specified with reference to an underlying calendar. The aim of this paper is to formalize the notion of calendars, i.e. to provide a frame of reference against which techniques of these authors may be fruitfully applied.

Temporal (Constraint) Databases: Kabanza et. al. [13] presented a framework for infinite temporal data bases. They view time as isomorphic to the integers. They represent time points using linear repeating points (points of the form $c + kn$, lrps for short) and restricted constraints.

Though they showed that the relational algebra may be extended to handle lrps, it still remains a fact that representing time in terms of lrps is highly unnatural for human beings. As we stated in the introduction, a human being may have trouble representing the 15th of March, 1994, as the time point 2,777,802,116 or something equally confusing. In contrast to their work,we do not attempt to map time to the integers. Instead, we represent time according to the notion of a calendar, which we have defined, and allow the user and the system to manipulate (representations of) calendars, without the user having to be aware of the internal calendar representation. Our framework is rich enough to handle a wide variety of calendars. Kabanza et. al.'s results may, however, be used in conjunction with ours as follows: their results on linear repeating points can be used to represent the constraints for realizing calendars, by using their languages to express constraints on the *top* time unit, which is isomorphic to the integers in our system.

In a similar vein, Koubarakis has published an elegant series of papers in which he reasons about definite and indefinite temporal specifications [16, 17] – in particular, he shows that constraints may be used to capture indefinite temporal information (points and interval) and manipulated these constraints to implement operations that extend those in the relational algebra of databases. However, he does not use symbolic time that refers to an underlying calendar, nor does he discuss the integration of multiple calendars.

Calendars: Ladkin [19, 20] uses sequences of integers to represent standard units assuming a linear hierarchy of time units, year, month, day, minutes etc. He uses this simple time system to define intervals. In our framework, intervals can be easily represented by simple constraints such as

$$\langle \text{lower bound} \rangle \leq t \leq \langle \text{upper bound} \rangle.$$

However, as solutions to such constraints are time-points, this representation of intervals ultimately ends up reasoning with time points, and thus does not capture Ladkin's intuitions. We will study how interval reasoning with multiple calendars can be performed in future work.

Leban et. al. [22] define a calendar to be a collection of an infinite sequence of intervals that span the time line. In other words, the space of all integers is "sliced" up into intervals that capture different parts of a calendar. For example, if the unit measure is 1 second, and t_0 denotes $Saturday$, $December$ 31, 1904 then they suggest the following definitions:

1. $Days \equiv << t_0; 86400 >>$ (this intuitively specifies that each day is an interval of 86400 seconds).

2. $Months \equiv << Days; 31; 28; 31; 30 \ldots$
   $31; 28; 31; 30 \ldots$
   $31; 28; 31; 30 \ldots$
   $31; 29; 31; \ldots >>$.

Intuitively, this provides an *explicit* way of defining calendars, while constraints are more compact and depend less upon the selection of the start time t_0. In contrast, our framework is very general, and can use well known techniques (e.g. Bellman Ford [6]) to solve constraints, rather than specialized techniques. In addition, we are able to seamlessly define what it means to combine calendars, which they do not address. On the other hand, they study intervals in much greater detail than we do.

Niezette and Stevenne [23] have made an elegant preliminary attempt to develop a symbolic representation of periodic time. They too use constraints to represent time points. In contrast, we use constraints to represent not only time points, but calendars as well. A consequence of our approach is that we can represent very general calendars as opposed to the periodic calendars that they focus on. They project each time unit (called calendar in their terminology) onto the integers – something that we do not do, thus allowing us to present to the user, an interface that shows only time as represented in the calendar (which is more natural to the user than representing time as an integer).

Jajodia and his colleagues [27] provide a formal framework for reasoning with multiple time granularities, but do not either represent time symbolically or represent calendars. They too convert time expressed in multiple granularities to the integers. Hence, as in the case of Niezette and Stevenne [23], our framework offers the advantage that the user can work with his own calendar representation of time, rather than with answers coming back in terms of the integer representation of time.

Bettini et. al. [3] develop the notion of an event structure with temporal granularities. An event structure is basically a graph whose nodes represent events and whose edges represent transitions between these events. However, the edges are labeled with a set of "timed constraints with granularity" (TCGs). They then show how to mine temporal dependency information (e.g. they may notice that IBM stock experiences a rise in any interval where Microsoft stock experienced a decline) from such event structures. Their work is related to ours because TCGs allow symbolic time where different time units may label a TCG. Their work is complementary to ours in the following sense: they develop techniques to convert one time unit to another, and essentially these conversion functions may be used to implement our linking functions. On the other hand, they have no notion of a calendar (they did not require it for their purposes), nor do they present a technique to integrate calendars.

TSQL2 is a query language which is an extension to SQL2 that supports multiple calendars – in particular, each user can define his own calendar. This is much closer in spirit to what we have in mind. Frameworks such as those of [7] fall within this category. Another effort to define multiple calendars is that of Chandra et. al. [4]. However, as Chomicki [5] pointed out:

> "We are not aware of any comprehensive description, even informal, of semantics of TSQL2 queries. This makes it impossible to establish formal properties of this language."

In contrast, our framework provides a formal foundation within which calendars may be expressed, combined, and reasoned with. Chomicki [5] provides an excellent survey of temporal databases, but only briefly touches upon symbolic representations of time.

9 Conclusions

Though most theoretical models of time (correctly) assume that the integers are adequate to reason about time, the fact of the matter is that most human beings, as well as most applications dealing with temporal data, specify time, not as integers, but rather as "dates" with respect to a symbolic, underlying calendar (e.g. the Gregorian calendar). The first contribution we have made in this paper is to provide a formal definition of a calendar that captures this intuition. Calendars are composed of a hierarchy of time units (e.g. `day`, `month`, `year`, etc.) and each of these time units has an associated set of symbolic values that are totally ordered. It may turn out that these symbolic values are integers, but they could just as well not be integers (e.g. the values could be `Monday, Tuesday,`...). In addition, calendars come equipped with a *syntactic* notion of a time-point w.r.t. the calendar, and an accompanying *semantic* notion called *validity* which specifies which time-points "make sense" (e.g. the syntactically valid date, "Feb 29, 1995" is not semantically valid).

Constraints over the domain of time points provide a unifying framework for expressing all the above concepts.

Furthermore, we consider what happens when multiple applications using different calendars wish to work together. In such a situation, the mismatch between their calendars needs to be resolved. To do this, we define a declarative notion of *combination* of multiple calendars. We show how this declarative notion may be *operationally realized* by manipulating the constraints used to represent the calendars being combined.

References

[1] J. Allen (1984). *Towards a general theory of action and time*, Artificial Intelligence, 23:123-154.

[2] P. van Beek (1992). *Reasoning about qualitative temporal information*, Artificial Intelligence, 58:297-326.

[3] C. Bettini, X. Sean Wang, and S. Jajodia. (1996) *Testing Complex Relationships Involving Multiple Granularities and its Application to Data Mining*, Proc. 15th ACM Symp. on Principles of Database Systems, Montreal, Canada, pps 68–78.

[4] R. Chandra, A. Segev and M. Stonebraker. (1994) *Implementing calendars and temporal rules in next generation databases*, Proc. IEEE Conf. on Data Engineering, 1994.

[5] J. Chomicki. (1994) *Temporal Query Languages: A Survey*, Proceedings of the 1st International Conference on Temporal Logic (eds. D. M. Gabbay and H. J. Ohlbach), pps 506–534, Lecture Notes in AI Vol. 927, Springer.

[6] T. Cormen, C. Leiserson and R. Rivest. (1990) "Introduction to Algorithms." MIT Press and McGraw Hill.

[7] Curtis E. Dyreson and Richard T. Snodgrass (1995) "Temporal Granularity", in The TSQL2 Temporal Query Language. Richard T. Snodgrass, editor. Kluwer Academic Press, 1995. pp. 347-383.

[8] T. Dean and D. McDermott (1987). *Temporal data base management*, Artificial Intelligence, 32(1):1-55.

[9] R. Dechter, I. Meiri and J. Pearl (1991). *Temporal Constraint Networks*, Artificial Intelligence, 49:61-95.

[10] D. Gabbay, C.J. Hogger and J.A. Robinson. (1994) *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 4: Epistemic and Temporal Reasoning*, Clarendon Press, Oxford.

[11] M. C. Golumbic and R. Shamir (1993). *Complexity and algorithms for reasoning about time: A graph-theoretic approach*, Journal of the ACM, 40(5):1108-1133, November 1993.

[12] F. Hillier and G. Lieberman. (1974) *Operations Research*, Holden-Day.

[13] F. Kabanza, J.-M. Stévenne and P. Wolper. (1995) *Handling Infinite Temporal Data*, Journal of computer and systems sciences, 51, 1, pps 3–17.

[14] D. E. Knuth. (1973) *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*, Addison Wesley.

[15] S. Kraus, Y. Sagiv and V.S. Subrahmanian. (1997) *Representing and Integrating Multiple Calendars*, Univ. of Maryland Technical Report, Feb. 1997.

[16] M. Koubarakis. (1994) *Database Models for Infinite and Indefinite Temporal Information*, Information Systems, Vol. 19, 2, pps 141–173.

[17] M. Koubarakis. (1994) *Complexity Results for First Order Theories of Temporal Constraints*, Proc. 4th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR-94), Bonn, Germany, pps 379–390.

[18] R. Kowalski and M. Sergot. (1986) *A Logic-based Calculus of Events*, New Generation Computing, 4, pps 67-95.

[19] P. Ladkin. (1986) *Primitives and units for time specification*, Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86), Vol. 1, pps 354–359, Philadelphia, Pennsylvania, Morgan Kaufmann.

[20] P. Ladkin. (1987) *The Completeness of a Natural System for Reasoning with Time Intervals*, Proceedings of the 10th International Joint Conference on Artificial Intelligence (ed. J. McDermott), Milan, Italy, pps 462–465, Morgan Kaufmann.

[21] P. B Ladkin and A. Reinefeld. (1992) *Effective Solution of Qualitative Interval Constraint Problems*, Artificial Intelligence, Vol. 57, Nr. 1, pps 105–124.

[22] B. Leban, D. McDonald and D. Forster. (1986) *A Representation for Collections of Temporal Intervals*, Proceedings of the 5th National Conference on Artificial Intelligence. Volume 1, pps 367–371, Morgan Kaufmann.

[23] M. Niezette and J. Stevenne. (1992) *An Efficient Symbolic Representation of Periodic Time*, Proc. First International Conference on Information and Knowledge Management, Baltimore, Maryland, Nov. 1992.

[24] E. Schwalb and R. Dechter (1993). *Coping with Disjunctions in Temporal Constraint Satisfaction Problems* In The National Conference on Artificial Intelligence, AAAI-93, Washington, DC, pp. 127-132.

[25] R. T. Snodgrass and M. D. Soo. *Supporting Multiple Calendars*, in: The TSQL2 Temporal Query Language (ed. R.T. Snodgrass), pps 103–121, Kluwer Academic Publishers.

[26] M. Vilain and H. Kautz. (1986) *Constraint Propagation Algorithms for Temporal Reasoning*, Proc. AAAI-1986, Philadelphia, PA, pps 132–144.

[27] X. Y. Wang, S. Jajodia, and V.S. Subrahmanian. Temporal Modules: An Approach Toward Federated Temporal Databases, INFORMATION SCIENCES, Vol. 82, pps 103–128, 1995.