Multi-Calendar Appointment Scheduling: Calendar Modeling and Constraint Reasoning

Stephanie Spranger and François Bry

Institute for Informatics, University of Munich, Germany
http://www.pms.ifi.lmu.de/

1 Introduction

This article outlines CaTTS, a language for expressing and solving *multi-calendar appointment scheduling problems* such as planning a phone conference of persons in different time zones and using different calendars. This article complements [1], which is focused on *calendar modeling,* with a presentation of the approach's constraint reasoning.

CaTTS is motivated by the fact that practical scheduling systems should preferably use everyone's calendars, i.e. cultural calendars such as the Gregorian or Hebrew calendars and the plethora of professional calendars, instead of the more abstract temporal specifications commonly used in research.

The 'Calendar and Time Type System' CaTTS consists of a *type definition language* CaTTS-DL and a *constraint language* CaTTS-CL. CaTTS is based on a time model after the set-theoretic 'time granularity' approach [9,10,2,11]. However, in contrast to most time granularity formalisms, CaTTS has no durationless time points. Instead, CaTTS is purely interval-based reflecting a commonsense notion of time: a time point such as "Tuesday, January 3 2006, 9 a.m.", expressed in the time granularity "hour" ("second", resp.) is internally represented in CaTTS by a time interval with a duration of 1 hour ("second", resp.).

2 Multi-Calendar CSPs in CaTTS

In [2] an approach to point-based metric temporal reasoning with time granularities is described. It allows for modeling and reasoning with simple temporal constraints on points and distances between points in a Disjunctive Linear Relations (DLR) Horn framework.[1] In this framework, one can express constraints like *"at time t (in time granularity g), person A is in London"*, but neither *"an event e (in time granularity g) happens during a task t (in time granularity h)"* nor *"an event e (in time granularity g) happens 5 time units (in granularity h) before an event e′ (in time granularity k)"*. Indeed, expressing temporal constraints like the last two mentioned above require not only time points but also time intervals.

[1] DLR [3] represents temporal constraints by disjunctions of linear inequalities and inequations.

E. K. Burke, H. Rudová (Eds.): PATAT 2006, pp. 496–501. ISBN 80-210-3726-1.

A common approach, followed e.g. by [2], represents a time interval by an ordered pair of time points. Expressing that two such intervals *meet without overlapping* requires to consider both (half or both sides) open and closed intervals. Such intervals, however, are rather counter-intuitive for most users. Furthermore, such a 'time point-based approach' makes the modeling of temporal applications significantly more complicated because both, time points and time intervals, have to be considered. Consider an event e taking place sometimes during an interval i which in turn is *during* an interval i': e also takes place during i'. This example shows that relations on intervals and events must be propagated so as to keep reasoning local. It is much more complicated to maintain locality when time intervals are defined by endpoints than when only time intervals (and no time points) are considered.

The approach reported about in this article avoids the problems mentioned above by considering only time intervals and no duration-less time points. In most practical applications, time points, if needed, are conveniently simulated by small intervals (e.g. in scheduling a meeting, intervals with a duration of 1 second may conveniently simulate time points). This makes it easy to represent

1. finite time intervals using finite domains,
2. quantifications such as *"an event e (in time granularity g) happens 5 time units (in granularity h) before an event e′ (in time granularity k)"*, and
3. generalized (i.e. finite and non-convex) as well as (infinite) periodic time intervals

as needed for many applications.

Example 1. A person plans a meeting lasting 3 working days after 20th April 2005 and before May 2005. A colleague's visit of 1 week must overlap with the planned meeting.

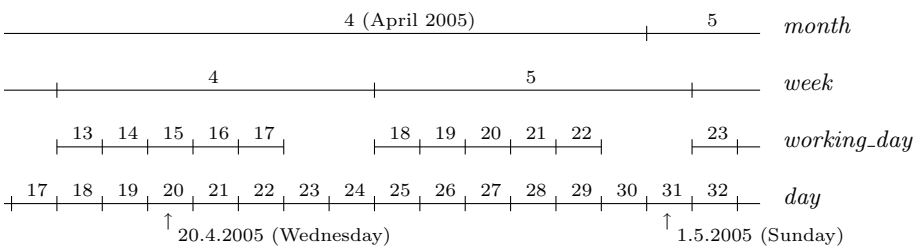


Fig. 1. The calendric types of Example 1.

The *activities* "meeting" and "visit" are represented as finite and convex time intervals that are isomorphic to sets of integers. Activities may refer to different *calendric types* such as "day", "working day", "week", or "working week". In Example 1, "meeting" refers to "working days", "visit" to "weeks", cf. Figure 1.

Using the Description Language CaTTS-DL, the calendar of Example 1 can be expressed as follows:[2]

```
calendar CalendarExample:Sig =
 cal
   type day;
   type week = aggregate 7 day @ day(-3);
   type month = aggregate
     31 day named January,
     alternate month(i)
      |(i div 12) mod 4 == 0 && (i div 12) mod 400 != 100 &&
       (i div 12) mod 400 != 200 && (i div 12) mod 400 != 300
        -> 29 day
      | otherwise -> 28 day
     end named February,
     31 day named March,
     ...
     31 day named December
   @ day(-90);
type working_day = select day(i) where
          relative i in week >= 1 && relative i in week <= 5;
end
```

Using the Constraint Language CaTTS-CL, the scheduling problem of Example 1 can be expressed as follows:

```
program SchedulingExample =
  prog
    use_calendar unqualified CalendarExample;
    use_format unqualified CatalogExample;
    Meeting is 3 working_day && Visit is 1 week &&
    Meeting after "20.04.2005" &&  Meeting before "05.2005" &&
    Visit overlaps Meeting
  end
```

An *answer* to such a CaTTS-CL program is defined as a consistent Multi-Calendar CSP which cannot be further reduced. The answer to the CaTTS-CL program above is as follows:

```
Meeting is 3 working_day &&
(begin_of Meeting) within ["21.04.2005".."22.04.2005"] &&
Visit is 1 week &&
(begin_of Visit) within ["Week04.2005".."Week04.2005"]
```

The programmer might ask for one or all *solution*s to this answer. They are computed from the answer by exhaustive search. The solutions to the CaTTS-CL program above are as follows:

```
(Meeting=["21.04.2005".."25.04.2005"] && Visit="Week04.2005")||
(Meeting=["22.04.2005".."26.04.2005"] && Visit="Week04.2005")
```

[2] '@ day(-3)' expresses that week 1 begins with day -3.

3 Solving of Multi-Calendar CSPs expressed in CaTTS

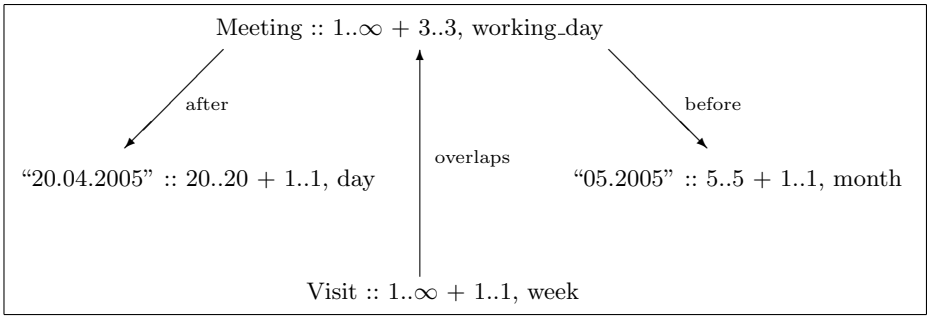


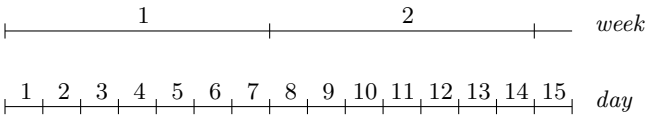
Fig. 2. The scheduling problem of Example 1 as a constraint network.

Multi-Calendar CSPs are seen as *constraint networks,* cf. Figure 2. Since CaTTS makes it possible to use different types expressing temporal granularities such as "days", "weeks" and "months", conversions are needed. CaTTS conversions are formalized by a coercion semantics for subtyping. They are expressed by *calendar conversion constraints.* These conversions follow CaTTs' novel subtyping relations, i.e. *aggregation* (e.g. a week is an aggregation of days) and *selection* (e.g. working days are a selection of days).

In Example 1, the temporal constraint "overlaps" on the variables "Visit" (of type "week") and "Meeting" (of type "working day")", require type conversions, e.g. to the closest type in the subtyping hierarchy, i.e. "day". Indeed, a week is a set of 7 consecutive days and working days are selected among all days.

Conversion constraint defines equivalences between calendar domains and make it possible to solve Multi-Calendar CSPs by reducing them to CSPs over finite domains. For example:

$$X :: 1..8 + 7..7, day \simeq Y :: 1..2 + 1..1, week$$



The time interval $[8, 14]$ of days represented by X corresponds to the time interval $[2, 2]$ of weeks represented by Y.

CaTTS' solver for Multi-Calendar CSPs is *complete* in the sense that if it terminates returning a consistent problem, then the original problem is (globally) consistent. Testing for (global) consistency of a Multi-Calendar CSPs is *linear* in both the number of constraints and the number of variables with respect to the size of the calendar domains. This follows from the facts that

1. testing consistency of classical CSPs over finite domain variables where the domains are represented by intervals is linear [8,6], and

2. the access to a CaTTS-DL conversion function in the conversion constraint can be done in *constant time.*

The search for one or all solutions to a bound consistent Multi-Calendar CSPs is however NP-hard.

4 Conclusion

The language CaTTS describes in this article makes it possible to express temporal constraints referring to different calendars. Temporal constraints referring to different calendars are needed in practice.

Novel aspects of the approach are as follows:

1. Only time intervals but no (duration-less) time points are considered,
2. two subtype relations, *aggregation* and *selection*, and
3. the conversion constraints reducing Multi-Calendar CSPs to CSPs over finite domains.

The approach provides with a solution to the problem of "time granularity conversion" mentioned in [12].

Further investigations of the issue would be desirable. Indeed, the approach described in this article cannot solve complex optimization problems like the scheduling of working shifts that require

1. constraints over infinite periodic time intervals and
2. soft constraints [15] and/or preferences [16].

While periodic time intervals can be specified in CaTTS, CaTTS' solver cannot process them. Soft constraints or preferences cannot yet be expressed in CaTTS.

Acknowledgments

This research has been funded by the German Foundation for Research (DFG) within the PhD Program 'Logic in Computer Science' (GKLI) as well as the European Commission and the Swiss Federal Office for Education and Science within project REWERSE (cf. `http://rewerse.net`). The authors thank the anonymous reviewers, Frank André Rieß, and Arnaud Lallouet for useful suggestions.

References

1. Bry, F., Rieß, F.A., Spranger, S.: CaTTS: Calendar Types and Constraints for Web Applications. In: Proc. 14^{th} Int. World Wide Web Conference, Japan. (2005)
2. Bettini, C., Jajodia, S., Wang, S.: Time Granularities in Databases, Data Mining, and Temporal Reasoning. Springer-Verlag (2000)
3. Jonsson, P., Bäckström, C.: A unifying approach to temporal constraint reasoning. Artificial Intelligence **102** (1998) 143–155

4. Spranger, S.: Calendars as Types – Data Modeling, Constraint Reasoning, and Type Checking with Calendars. PhD Thesis. Herbert Utz Verlag, München (2006)
5. Allen, J.: Maintaining Knowledge about Temporal Intervals. Communications of the ACM **26** (1983) 832–843
6. Frühwirth, T., Abdennadher, S.: Essentials of Constraint Programming. Cognitive Technologies. Springer-Verlag (2003)
7. van Hentenryck, P., Saraswat, V., Deville, Y.: Constraint Processing in cc(FD). Technical Report, unpublished Manuscript (1992)
8. Tsang, E.: Foundations of Constraint Satisfaction. Academic Press (1993)
9. Montanari, A.: Metric and Layered Temporal Logics for Time Granularity. ILLC Dissertation Series 1996-02, University of Amsterdam (1996)
10. Jensen, C., (eds.), C.D.: The consensus glossary of temporal database concepts - February 1998 version. (1998)
11. Euzenat, J.: Granularity in Relational Formalisms with Applications to Time and Space Representation. Computational Intelligence **17** (2001) 703–737
12. Franceschet, M., Montanari, A.: A Combined Approach to Temporal Logics for Time Granularity. In: Workshop on Methods for Modalities. (2001)
13. Vilain, M.: A System for Reasoning about Time. In: Proceedings of the 2^{nd} National (US) Conference on Artificial Intelligence, AAAI Press (1982) 197–201
14. Meiri, I.: Combining Qualitative and Quantitative Constraints in Temporal Reasoning. Artificial Intelligence **87** (1996) 343–385
15. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based Constraint Satisfaction and Optimization. Journal of the ACM **44** (1997) 201–236
16. Prestwich, S., Rossi, F., K.B.Venable, Walsh, T.: Constrained CP-Nets. In: Italian Conference on Computational Logic. (2004)