GSTP: A Temporal Reasoning System Supporting Multi-Granularity Temporal Constraints*

Claudio Bettini, Sergio Mascetti, Vincenzo Pupillo
Università di Milano, Italy

1 Introduction

The GSTP system has been developed at the University of Milan with the objective of providing universal access to the implementation of a set of algorithms for multi-granularity temporal constraint satisfaction. The many formalisms and algorithms proposed in the literature for Temporal Constraint Satisfaction Problems (TCSP) have essentially ignored the subtleties involved in the presence of multiple time granularities in the temporal constraints. Examples of simple constraints specified in terms of a time granularity are the following: "package shipment must occur *the next business day* after check clearance" and "package delivery should occur *during working hours*". More technically, the GSTP system allows the user to specify binary constraints of the form $Y - X \in [m, n]\, G$, where m and n are the minimum and maximum values of the distance between X and Y in terms of granularity G. Variables take values in the positive integers, and unary constraints can be applied on their domains. This can be considered the extension of STP [Dechter *et al.*1991] to multiple and arbitrary granularities.

A first issue in the representation and processing of these constraints is the need for a clear semantics for time granularities. *Business days*, for example, may really have different meanings in different countries or even in different companies. In this respect GSTP adopts a formalism, first introduced in [Wang *et al.*1997; Bettini *et al.*1998], which can model arbitrary user-defined time granularities and has a clear set-theoretic semantics. In order to guarantee a finite representation, granularities in GSTP are limited to those that can be defined in terms of periodic sets. Hours, days, weeks, business days, business weeks, fiscal years, and academic semesters are common examples.

A second issue is related to the difficulty to reduce a network of constraints given in terms of different granularities into an equivalent one with all constraints in terms of the same granularity, so that some of the standard algorithms for CSP, like consistency checking through arc- or path-consistency [Bessiere 1994; Dechter *et al.*1991], could be successfully applied. Indeed, any conversion necessarily introduces an approximation; For example, the above constraint imposing delivery to start the next business day may be translated in terms of hours with a minimum of 1 hour and a maximum of 95 hours. (The number 95 takes into account a check clearance at the beginning of a Friday and a shipment at the end of next Monday according to the constraint.) However, if the check is cleared on Monday, the new constraint would allow a shipment on Thursday which is clearly a violation of the original constraint. Approximate conversion algorithms are extensively discussed in [Bettini *et al.*1998]. We have shown that any consistency algorithm adopting these conversions as the only tool to reduce the problem to a standard CSP is inevitably incomplete, and have proposed a different algorithm, called ACG, which has been proved to be complete [Bettini *et al.*2002].

GSTP, in addition to implementing the reasoning algorithms, assists the user in the definition of constraint networks, in their submission to a remote processing service and in the analysis of the output.

2 The Algorithms

The most interesting part of the system is perhaps the implementation of the ACG algorithm which has been recently proposed in [Bettini *et al.*2002]. It is based on arc-consistency, and it is essentially an extension of the AC-3 algorithm [Mackworth *et al.*1985] to deal with possibly infinite (but periodic) domains and with constraints in terms of multiple periodic granularities. This extension is not trivial since it involves the algebraic manipulation of the mathematical characterization of granularities. ACG also derives the *minimal* solution for the constraint network.

```
Repeat
        1. Conversion+PC
        2. ACG
        3. RefineEdgesFromNodes()
Until no change is observed
Return Inconsistent or NewNetwork+solution
```

Figure 1: The main loop of the constraint solver

Despite several optimizations have been introduced in the implementation, ACG greatly benefits from any preprocessing phase that can refine some of the original constraints. This is the main reason why GSTP integrates with ACG an approximate algorithm, proposed in [Bettini *et al.*1998], and based

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

on the conversion of constraints in different granularities followed by path consistency. The interaction of the two algorithms is also used to further refine the original constraints.

Fig. 1 shows the three main steps behind the GSTP constraint solver. In step 1, the original network is decomposed in as many networks as are the granularities appearing in the constraints; each network has the explicit constraints given in terms of one granularity as well as constraints in the same granularity obtained by conversion from others on the same edge, but in terms of different granularities. Then, standard path consistency is applied to each network. The resulting network most likely has refined constraints with respect to the original one. Any inconsistency captured by this processing has the effect of terminating the constraint solver reporting the inconsistency status. However, if this is not the case, the network may still be inconsistent and it will go through ACG, the complete consistency algorithm (step 2). From the node domains returned by ACG, it is possible to further refine some of the constraints (the function doing this job in step 3 is called *RefineEdgesFromNodes()*). The steps are repeated, since path consistency applied to the refined constraints may lead to some changes. Our experiments show that after few iterations of the main loop a fixpoint is always reached.

3 The GSTP Architecture

Fig. 2 shows the general architecture of the GSTP system. There are three main modules: the constraint solver, the web service, which enables external access to the solver, and a user interface that can be used locally or remotely to design and analyze constraint networks.

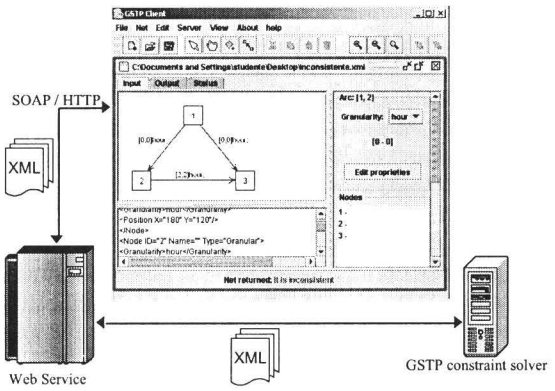

Figure 2: The GSTP Architecture

The constraint solver is the C implementation of the algorithms described above, and it runs on a server machine. The Web Service defines, through a WSDL specification, the parameters that can be passed to the constraint solver, including the XML schema for the constraint network specification; It accepts connections through soap/http from client applications or other web services which require constraint processing, it invokes the solver after validating the parameters, and it passes back the results. The third module is a remote java-based user interface, which is extensively described in the system demo. It allows the user to easily edit constraint networks, to submit them to the constraint solver, and to analyze results. In particular, it is possible to have views in terms of specific granularities, to visualize implicit constraints, to browse descriptions of domains, and to obtain a network solution. Fig. 3 shows a screenshot from the interface.

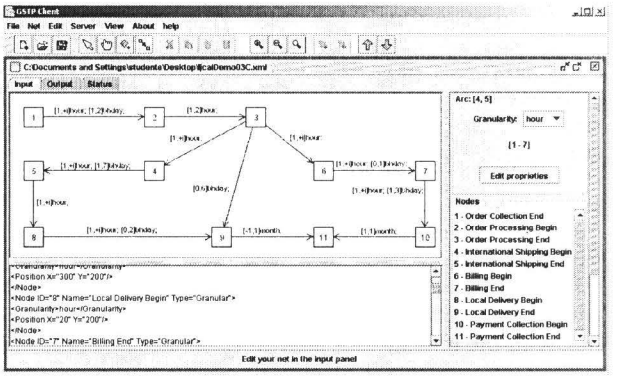

Figure 3: The User Interface

Credits

Many people contributed to the implementation of the GSTP system. R. De Sibi, G. Gabrielli, M. Colombo, S. Ruffini, and V. Pupillo worked at the implementation of the temporal reasoning algorithms; S. Gotta and S. Mascetti worked at the graphical interface; C. Cestana worked at the web service architecture. The theory underlying GSTP is due to the authors of [Bettini *et al.*2002]. S. Wang gave valuable suggestions on implementation issues. C. Bettini coordinated the system implementation and worked closely with all of the above.

References

[Bessiere 1994] C. Bessière. Arc-Consistency and Arc-Consistency Again. *Artificial Intelligence* 65(1):179–190, 1994.

[Bettini *et al.*1998] C. Bettini, X. Wang, S. Jajodia. A General Framework for Time Granularity and its Application to Temporal Reasoning. *Annals of Mathematics and Artificial Intelligence* 22(1,2):29–58, 1998.

[Bettini *et al.*2002] C. Bettini, X. Wang, S. Jajodia, Solving Multi-Granularity constraint networks, *Artificial Intelligence*, 140(1-2):107–152, 2002.

[Dechter *et al.*1991] R. Dechter, I. Meiri, J. Pearl. Temporal constraint networks. *Artificial Intelligence* 49:61–95, 1991.

[Mackworth *et al.*1985] A. Mackworth, E. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence* 25:65–74, 1985.

[Wang *et al.*1997] X. Wang, C. Bettini, A. Brodsky, S. Jajodia. Logical design for temporal databases with multiple granularities. *ACM Transactions on Database Systems* 22(2):115–170, 1997.