Spatio-Temporal Annotated Constraint Logic Programming

Alessandra Raffaetà[1], Thom Frühwirth[2]

[1] Dipartimento di Informatica - Università di Pisa
raffaeta@di.unipi.it
[2] Institut für Informatik - Ludwig-Maximilians-Universität München
fruehwir@informatik.uni-muenchen.de

Abstract. We extend Temporal Annotated Constraint Logic Programming (TACLP) in order to obtain a framework where both temporal and spatial information can be dealt with and reasoned about. This results in a conceptually simple, uniform setting, called STACLP (Spatio-Temporal Annotated Constraint Logic Programming), where temporal and spatial data are represented by means of annotations that label atomic first-order formulae. The expressiveness and conciseness of the approach are illustrated by means of some examples: Definite, periodic and indefinite spatio-temporal information involving time-varying objects and properties can be handled in a natural way.

Keywords: Spatio-temporal reasoning, constraint logic programming, annotated logics.

1 Introduction

The handling of spatio-temporal data has recently begun to attract broader interest [1, 3, 8, 21]. The more computers support humans, the more have they to be capable of dealing with spatio-temporal information. Time and space are closely interconnected: Much information that is referenced to space is also referenced to time and may vary with time. Prominent applications are geographic information systems (GISs), environmental monitoring, and geometric modeling systems (CAD). Other areas in need for spatio-temporal reasoning are cadastral databases, diagrammatic reasoning and scientific applications.

In a logical formalization of spatial and temporal information it is quite natural to think of formulae that are labelled by spatio-temporal information and about proof procedures that take into account these labels. In this perspective, the general framework of annotated constraint logic (ACL) [7] extends first order languages with a distinguished class of predicates, called *constraints*, and a distinguished class of terms forming a lattice, called *annotations*, which are used to label formulae. Semantically, ACL provides *inference rules* for reasoning on annotated formulae and a *constraint theory* defining lattice operations, i.e. constraints, on annotations. One advantage of a language in the ACL family is that its clausal fragment can be efficiently implemented [7]: Given a logic in

this framework, there is a systematic way to make a clausal fragment executable as a constraint logic program [13]. Based on the ACL framework, the family of temporal annotated constraint logic programming (TACLP) languages has been developed and investigated in [7, 12, 16]. TACLP supports qualitative and quantitative (metric) temporal reasoning involving both time points and time periods (time intervals) and their duration. Moreover, it allows one to represent definite, indefinite and periodic temporal information.

Having ACL and TACLP, a natural further step is to consider the addition of spatial information. A first proposal can be found in [12]. In such an approach an object is modeled by a predicate and its spatial extent is represented by adding variables denoting the spatial coordinates as arguments and by placing constraints on such variables. Even if this kind of spatial representation is common [1, 4, 8], it is somewhat awkward in the context of TACLP: While temporal information is represented by annotations, spatial information is encoded into the formulae. The result is a mismatch of conceptual levels and a loss of simplicity. Consequently, in this paper we present STACLP, a framework resulting from the introduction of spatial annotations in TACLP. The pieces of spatio-temporal information are given by couples of annotations which specify the spatial extent of an object or of a property at a certain time period. The use of annotations makes time and space explicit but avoids the proliferation of spatial and temporal variables and quantifiers. Moreover, it supports both definite and indefinite spatial and temporal information, and it allows to establish a dependency between space and time, thus permitting to model continuously moving points and regions.

While a lot of effort has been spent in developing extensions of logic programming languages capable to manage time [14], the logic based languages for the handling of spatial information only deal with the qualitative representation and reasoning about space (see e.g. [17]). And also the few attempts to manipulate time and space have led to languages for qualitative spatio-temporal representation and reasoning [20]. On the other hand temporal [19, 6] and spatial [9, 15] database technologies are relatively mature, although, also in the database area, their combination is far from straightforward [3]. In this context, the constraint database approach [10] appears to be very promising.

Our spatio-temporal language is close to the approaches based on constraint databases [1, 4, 8]. From a database point of view, logic programs can represent deductive databases, i.e. relational databases enriched with intensional rules, constraint logic programs can represent constraint databases [10], and thus STACLP can represent spatio-temporal constraint databases. The spatio-temporal proposals in [1, 8] are extensions of languages originally developed to express only spatial data. Thus the high-level mechanisms they offer are more oriented to query spatial data than temporal information. In fact, they can model only definite temporal information and there is no support for periodic, indefinite temporal data as we will stress in the comparison with [8] in Section 5.1. On the contrary STACLP provides several facilities to reason on temporal data and to establish spatio-temporal correlations. For instance, it allows one to describe

continuous change in time as well as [4] does, whereas both [1] and [8] can represent only *discrete* changes. Also indefinite spatial and temporal information can be expressed in STACLP, a feature supported only by the approach in [11].

Furthermore, STACLP takes advantage from the deductive power supplied by constraint logic programming. For instance, recursive predicates can be exploited to compute the transitive closure of relations, an ability not provided by the traditional approaches in the database field. In Section 5.4 we will illustrate how such an ability is relevant in network analysis, where it may be used to search for connections between objects. More generally, STACLP does not represent only data as in constraint databases but also rules. This extra feature makes the difference if one wants to use the language as a specification and/or analysis language.

Overview of the paper. In Section 2, we shortly introduce the Temporal Annotated Constraint Logic Programming framework (TACLP). In Section 3, we present STACLP which extends TACLP by spatial annotations, and in Section 4 we define its semantics using a meta-interpreter. In Section 5 we give several non-trivial examples aimed at illustrating the expressiveness of our approach. Finally, in Section 6, we draw some conclusions and possible directions of future work.

2 TACLP: Time, Annotations, Constraints, Clauses

In this section we briefly describe the syntax and semantics of Temporal Annotated Constraint Logic Programming (TACLP) [7], a natural and powerful framework for formalizing temporal information and reasoning. As mentioned in the introduction, TACLP is a constraint logic programming language where formulae can be annotated with temporal labels and where relations between these labels can be expressed by using constraints. In TACLP, the choice of the temporal ontology is free. In this paper, we will consider the instance of TACLP where time points are totally ordered and labels involve convex, non-empty sets of time points. Moreover only atomic formulae can be annotated and clauses are negation free.

2.1 Time

Time can be discrete or dense. Time points are totally ordered by the relation $\leq$. We denote by $\mathcal{T}$ the set of time points and we suppose to have a set of operations (such as the binary operations $+$, $-$) to manage such points. We assume that the time-line is left-bounded by the number 0 and open to the future, with the symbol ∞ used to denote a time point that is later than any other. A *time period* is an interval $[r, s]$ with $r, s \in \mathcal{T}$ and $0 \leq r \leq s \leq \infty$, which represents the convex, non-empty set of time points $\{t \mid r \leq t \leq s\}$[1]. Thus the interval $[0, \infty]$ denotes the whole time line.

[1] The results we present naturally extend to time lines that are bounded in other ways and to time periods that are open on one or both sides.

2.2 Annotations and Annotated formulae

An *annotated formula* is of the form $A\,\alpha$ where A is an atomic formula and α an annotation. There are three kinds of annotations based on time points and on time periods. Let t be a time point and let $J = [r, s]$ be a time period.

(`at`) The annotated formula $A\,\mathtt{at}\,t$ means that A holds at time point t.

(`th`) The annotated formula $A\,\mathtt{th}\,J$ means that A holds *throughout*, i.e., at *every* time point in, the time period J. A `th`-annotated formula is defined in terms of `at` as

$$A\,\mathtt{th}\,J \;\Leftrightarrow\; \forall t \in J.\; A\,\mathtt{at}\,t.$$

(`in`) The annotated formula $A\,\mathtt{in}\,J$ means that A holds at *some* time point(s) - but we do not know exactly which - in the time period J. An `in`-annotated formula is defined in terms of `at` as

$$A\,\mathtt{in}\,J \;\Leftrightarrow\; \exists t \in J.\; A\,\mathtt{at}\,t.$$

The `in` temporal annotation accounts for indefinite temporal information.

The set of annotations is endowed with a partial order relation $\sqsubseteq$ which turns it into a lattice. Given two annotations α and β, the intuition is that $\alpha \sqsubseteq \beta$ if α is "less informative" than β in the sense that for all formulae A, $A\,\beta \Rightarrow A\,\alpha$. More precisely, being an instance of ACL, in addition to Modus Ponens, TACLP has the two inference rules below:

$$\frac{A\,\alpha \qquad \gamma \sqsubseteq \alpha}{A\,\gamma}\;\; \textit{rule}\;(\sqsubseteq) \qquad\qquad \frac{A\,\alpha \qquad A\,\beta \qquad \gamma = \alpha \sqcup \beta}{A\,\gamma}\;\; \textit{rule}\;(\sqcup)$$

The rule ($\sqsubseteq$) states that if a formula holds with some annotation, then it also holds with all annotations that are smaller according to the lattice ordering. The rule ($\sqcup$) says that if a formula holds with some annotation α and the same formula holds with another annotation β then it holds with the least upper bound $\alpha \sqcup \beta$ of the two annotations.

2.3 Constraints

We introduce the *constraint theory for temporal annotations*. Recall that a constraint theory is a non-empty, consistent first order theory that axiomatizes the meaning of constraints. Besides an axiomatization of the total order relation $\leq$ on the set of time points $\mathcal{T}$, the constraint theory includes the following axioms defining the partial order on temporal annotations.

$$\begin{array}{ll}
(\mathtt{at\,th}) & \mathtt{at}\,t = \mathtt{th}\,[t,t] \\
(\mathtt{at\,in}) & \mathtt{at}\,t = \mathtt{in}\,[t,t] \\
(\mathtt{th}\;\sqsubseteq) & \mathtt{th}\,[s_1,s_2] \sqsubseteq \mathtt{th}\,[r_1,r_2] \Leftrightarrow r_1 \leq s_1,\; s_1 \leq s_2,\; s_2 \leq r_2 \\
(\mathtt{in}\;\sqsubseteq) & \mathtt{in}\,[r_1,r_2] \sqsubseteq \mathtt{in}\,[s_1,s_2] \Leftrightarrow r_1 \leq s_1,\; s_1 \leq s_2,\; s_2 \leq r_2
\end{array}$$

The first two axioms state that $\mathtt{th}\,I$ and $\mathtt{in}\,I$ are equivalent to $\mathtt{at}\,t$ when the time period I consists of a single time point t.[2] Next, if a formula holds at every

[2] Especially in dense time, one may disallow singleton periods and drop the two axioms. This restriction has no effects on the results we are presenting.

element of a time period, then it holds at every element in all sub-periods of that period (($\mathtt{th}\ \sqsubseteq$) axiom). On the other hand, if a formula holds at some points of a time period then it holds at some points in all periods that include this period (($\mathtt{in}\ \sqsubseteq$) axiom).

Next we axiomatize the least upper bound $\sqcup$ of temporal annotations over time points and time periods. As explained in [7], it suffices to consider the least upper bound for time periods that produce another *valid* (non-empty) time period. Concretely, it is enough to compute the least upper bound of $\mathtt{th}$ annotations with overlapping time periods:

$$(\mathtt{th}\sqcup) \qquad \mathtt{th}\,[s_1, s_2] \sqcup \mathtt{th}\,[r_1, r_2] = \mathtt{th}\,[s_1, r_2] \;\Leftrightarrow\; s_1 \leq r_1, r_1 \leq s_2, s_2 \leq r_2$$

2.4 Clauses

The clausal fragment of TACLP, which can be used as an efficient temporal programming language, consists of clauses of the following form:

$$A\,\alpha \leftarrow C_1, \ldots, C_n, B_1\,\alpha_1, \ldots, B_m\,\alpha_m \quad (n, m \geq 0)$$

where A is an atom (not a constraint), α and α_i are (optional) temporal annotations, the C_j's are constraints and the B_i's are atomic formulae. Constraints C_j cannot be annotated.

We conclude the introduction of TACLP with an example taken from [16].

Example 1. In a company, there are managers and a secretary who has to manage their meetings. A manager is busy if he is in a meeting or if he is out.

$$\mathit{busy}(P)\,\mathtt{th}\,[T_1, T_2] \leftarrow \mathit{in\text{-}meeting}(P)\,\mathtt{th}\,[T_1, T_2]$$
$$\mathit{busy}(P)\,\mathtt{th}\,[T_1, T_2] \leftarrow \mathit{out\text{-}of\text{-}office}(P)\,\mathtt{th}\,[T_1, T_2]$$

Suppose the schedule for today to be the following: Smith and Jones have a meeting at 9am and at 9:30am respectively, each lasting one hour. In the afternoon Smith goes out for lunch at 2pm and comes back at 3pm:

$$\mathit{in\text{-}meeting}(\mathit{smith})\,\mathtt{th}\,[9am, 10am]. \qquad \mathit{out\text{-}of\text{-}office}(\mathit{smith})\,\mathtt{th}\,[2pm, 3pm].$$
$$\mathit{in\text{-}meeting}(\mathit{jones})\,\mathtt{th}\,[9{:}30am, 10{:}30am].$$

If the secretary wants to know whether Smith is busy between 9:30am and 10:30am she can ask for $\mathit{busy}(\mathit{smith})\,\mathtt{in}\,[9{:}30am, 10{:}30am]$. Since Smith is in a meeting from 9am till 10am, one can indeed derive that Smith is busy. Notice that this query exploits indefinite information: Since Smith is busy at least in one instant of the period $[9{:}30am, 10{:}30am]$, the secretary cannot schedule an appointment for him for that period. Conversely, $\mathit{busy}(\mathit{smith})\,\mathtt{th}\,[9{:}30am, 10{:}30am]$ does not hold, because Smith is not busy between 10am and 10:30am.

The query ($\mathit{busy}(\mathit{smith})\,\mathtt{th}\,[T_1, T_2]$, $\mathit{busy}(\mathit{jones})\,\mathtt{th}\,[T_1, T_2]$) reveals that both managers are busy throughout the time period $[9{:}30am, 10am]$, because this is the largest interval that is included in the time periods where both managers are busy.

3 STACLP: A spatio-temporal language

In this section we introduce an extension to TACLP where both temporal and spatial information can be dealt with and reasoned about. The resulting framework is called Spatio-Temporal Annotated Constraint Logic Programming (STACLP). It is worth noticing that spatial data can be already modeled in TACLP by using contraints in the style of constraint databases (see [12]). However, this spatial representation is somewhat awkward in the context of TACLP: While temporal information is represented by annotations, spatial information is encoded into the formulae. The result is a mismatch of conceptual levels and a loss of simplicity. In STACLP we overcome this mismatch, by defining a uniform setting where spatial information is represented by means of annotations, so that the advantages of using annotations apply to the spatial dimension as well.

3.1 Spatial annotations and constraints

We consider as spatial regions *rectangles* represented as $[(x_1, x_2), (y_1, y_2)]$, where (x_1, y_1) and (x_2, y_2) represent the lower-left and upper-right vertex of the rectangle, respectively. More precisely, $[(x_1, x_2), (y_1, y_2)]$ is intended to model the region $\{(x, y) \mid x_1 \leq x \leq x_2, y_1 \leq y \leq y_2\}$[3]. Rectangles are the two-dimensional counterpart of convex sets of time points.

Furthermore we define three spatial annotations, which resemble the three temporal annotations, namely `atp` (at point/position), `thr` (all throughout region), `inr` (somewhere in region). The set of spatial annotations is endowed with a partial order relation described by the following constraint theory.

$$
\begin{array}{ll}
(\texttt{atp}\,\texttt{thr}) & \texttt{atp}\,(x, y) = \texttt{thr}\,[(x, x), (y, y)] \\
(\texttt{atp}\,\texttt{inr}) & \texttt{atp}\,(x, y) = \texttt{inr}\,[(x, x), (y, y)] \\
(\texttt{thr}\ \sqsubseteq) & \texttt{thr}\,[(x_1, x_2), (y_1, y_2)] \sqsubseteq \texttt{thr}\,[(x_1', x_2'), (y_1', y_2')] \Leftrightarrow \\
 & \quad x_1' \leq x_1,\ x_1 \leq x_2,\ x_2 \leq x_2', y_1' \leq y_1,\ y_1 \leq y_2,\ y_2 \leq y_2' \\
(\texttt{inr}\ \sqsubseteq) & \texttt{inr}\,[(x_1', x_2'), (y_1', y_2')] \sqsubseteq \texttt{inr}\,[(x_1, x_2), (y_1, y_2)] \Leftrightarrow \\
 & \quad x_1' \leq x_1,\ x_1 \leq x_2,\ x_2 \leq x_2', y_1' \leq y_1,\ y_1 \leq y_2,\ y_2 \leq y_2'
\end{array}
$$

3.2 Combining spatial and temporal annotations

In order to obtain spatio-temporal annotations the spatial and temporal annotations are combined by considering couples of annotations as a new class of annotations. Let us first introduce the general idea of coupling of annotations.

Definition 1. *Let* $(A, \sqsubseteq_A)$ *and* $(B, \sqsubseteq_B)$ *be two disjoint classes of annotations with their partial order. The* coupling *is the class of annotations* $(A * B, \sqsubseteq_{A*B})$ *defined as follows*

$$A * B = \{\alpha\beta, \beta\alpha \mid\ \alpha \in A,\ \beta \in B\}$$

$$\gamma_1 \sqsubseteq_{A*B} \gamma_2 \Leftrightarrow ((\gamma_1 = \alpha_1\beta_1 \wedge \gamma_2 = \alpha_2\beta_2) \vee (\gamma_1 = \beta_1\alpha_1 \wedge \gamma_2 = \beta_2\alpha_2)) \wedge$$
$$(\alpha_1 \sqsubseteq_A \alpha_2\ \wedge\ \beta_1 \sqsubseteq_B \beta_2)$$

[3] The approach can be easily extended to an arbitrary number of dimensions.

In our case the spatio-temporal annotations are obtained by considering the coupling of spatial and temporal annotations.

Definition 2 (Spatio-temporal annotations). *The class of spatio-temporal annotations is the coupling of the spatial annotations* Spat *built from* `atp`, `thr` *and* `inr` *and of the temporal annotations* Temp, *built from* `at`, `th` *and* `in`, *i.e.* Spat∗Temp.

To clarify the meaning of our spatio-temporal annotations, we present some examples of their formal definition in terms of `at` and `atp`. Let t be a time point, $J = [t_1, t_2]$ be a time period, $s = (x, y)$ be a spatial point and $R = [(x_1, x_2), (y_1, y_2)]$ be a rectangle.

- The equivalent annotated formulae $A\,\mathtt{atp}\,s\,\mathtt{at}\,t$, $A\,\mathtt{at}\,t\,\mathtt{atp}\,s$ mean that A holds at time point t in the spatial point s.
- The annotated formula $A\,\mathtt{thr}\,R\,\mathtt{th}\,J$ means that A holds *throughout* the time period J and at every spatial point in R. The definition of such a formula in terms of `atp` and `at` is:

$$A\,\mathtt{thr}\,R\,\mathtt{th}\,J \;\Leftrightarrow\; \forall t \in J.\ \forall s \in R.\ A\,\mathtt{atp}\,s\,\mathtt{at}\,t.$$

  The formula $A\,\mathtt{th}\,J\,\mathtt{thr}\,R$ is equivalent to the formula above because one can be obtained from the other just by swapping the two universal quantifiers.
- The annotated formula $A\,\mathtt{thr}\,R\,\mathtt{in}\,J$ means that there exist(s) *some* time point(s) in the time period J in which A holds throughout the region R. The definition of such a formula in terms of `atp` and `at` is:

$$A\,\mathtt{thr}\,R\,\mathtt{in}\,J \;\Leftrightarrow\; \exists t \in J.\ \forall s \in R.\ A\,\mathtt{atp}\,s\,\mathtt{at}\,t.$$

  In this case swapping the annotations swaps the universal and existential quantifiers and hence results in a different annotated formula $A\,\mathtt{in}\,J\,\mathtt{thr}\,R$, meaning that for every spatial point in the region R, A holds at some time point(s) in J.

Let us point out how two formulae which are obtained one from the other by exchanging the order of annotations might differ. Consider, for instance, the following annotated formulae.

$$\mathit{water}\,\mathtt{thr}\,R\,\mathtt{in}\,[\mathit{apr}, \mathit{jul}]. \qquad\qquad \mathit{water}\,\mathtt{in}\,[\mathit{apr}, \mathit{jul}]\,\mathtt{thr}\,R.$$

The first one expresses that there is a time period between April and July in which the whole region R is completely covered by the water. On the other hand the second formula states that from April to July each point in the region R will be covered by the water, but different points can be covered in different time instants. Hence there is no certainty that in a time instant the whole region is covered by the water.

Consider now $\mathtt{inr}\,R\,\mathtt{th}\,I$ and $\mathtt{th}\,I\,\mathtt{inr}\,R$. The annotation $A\,\mathtt{inr}\,R\,\mathtt{th}\,I$ means that throughout the time interval I, A holds somewhere in the region R, e.g. A may change its position and move inside R during the time interval I. On

the other hand, $A\,\mathtt{th}\,I\,\mathtt{inr}\,R$ means that there exists a point in space R for which A holds throughout the interval I, so A is fixed during I. For instance the annotated atom $\mathit{does}(\mathit{john}, \mathit{ski})\,\mathtt{inr}\,R\,\mathtt{th}\,[\mathit{12am}, \mathit{4pm}]$ means that John is skiing (thus possibly moving) inside the area R from $12am$ to $4pm$. Whereas $\mathit{exhibition}(\mathit{monet})\,\mathtt{th}\,[\mathit{oct}, \mathit{dec}]\,\mathtt{inr}\,R$ means that in some fixed place in the region R there is the Monet exhibition from October to December.

3.3 Least Upper Bound and its constraint theory

As for TACLP, besides Modus Ponens, two inference rules ($\sqsubseteq$) and ($\sqcup$) are provided. Also for spatio-temporal annotations, we restrict the latter rule only to least upper bounds that produce valid, new annotations, i.e., the resulting regions are rectangles and the temporal components are time periods. Thus we consider the least upper bound in the following cases:

(1) $\mathtt{thr}\,[(x_1,x_2),(y_1,y_2)]\mathtt{th}\,[t_1,t_2] \sqcup \mathtt{thr}\,[(x_1,x_2),(z_1,z_2)]\mathtt{th}\,[t_1,t_2] =$
$\mathtt{thr}\,[(x_1,x_2),(y_1,z_2)]\mathtt{th}\,[t_1,t_2] \Leftrightarrow y_1 \leq z_1, z_1 \leq y_2, y_2 \leq z_2$

(1′) axiom obtained by swapping the annotations in (1).

(2) $\mathtt{thr}\,[(x_1,x_2),(y_1,y_2)]\mathtt{th}\,[t_1,t_2] \sqcup \mathtt{thr}\,[(z_1,z_2),(y_1,y_2)]\mathtt{th}\,[t_1,t_2] =$
$\mathtt{thr}\,[(x_1,z_2),(y_1,y_2)]\mathtt{th}\,[t_1,t_2] \Leftrightarrow x_1 \leq z_1, z_1 \leq x_2, x_2 \leq z_2$

(2′) axiom obtained by swapping the annotations in (2).

(3) $\mathtt{thr}\,[(x_1,x_2),(y_1,y_2)]\mathtt{th}\,[s_1,s_2] \sqcup \mathtt{thr}\,[(x_1,x_2),(y_1,y_2)]\mathtt{th}\,[r_1,r_2] =$
$\mathtt{thr}\,[(x_1,x_2),(y_1,y_2)]\mathtt{th}\,[s_1,r_2] \Leftrightarrow s_1 \leq r_1, r_1 \leq s_2, s_2 \leq r_2$

(3′) axiom obtained by swapping the annotations in (3).

(4) $\mathtt{inr}\,[(x_1,x_2),(y_1,y_2)]\mathtt{th}\,[s_1,s_2] \sqcup \mathtt{inr}\,[(x_1,x_2),(y_1,y_2)]\mathtt{th}\,[r_1,r_2] =$
$\mathtt{inr}\,[(x_1,x_2),(y_1,y_2)]\mathtt{th}\,[s_1,r_2] \Leftrightarrow s_1 \leq r_1, r_1 \leq s_2, s_2 \leq r_2$

(5) $\mathtt{in}\,[t_1,t_2]\mathtt{thr}\,[(x_1,x_2),(y_1,y_2)] \sqcup \mathtt{in}\,[t_1,t_2]\mathtt{thr}\,[(x_1,x_2),(z_1,z_2)] =$
$\mathtt{in}\,[t_1,t_2]\mathtt{thr}\,[(x_1,x_2),(y_1,z_2)] \Leftrightarrow y_1 \leq z_1, z_1 \leq y_2, y_2 \leq z_2$

(6) $\mathtt{in}\,[t_1,t_2]\mathtt{thr}\,[(x_1,x_2),(y_1,y_2)] \sqcup \mathtt{in}\,[t_1,t_2]\mathtt{thr}\,[(z_1,z_2),(y_1,y_2)] =$
$\mathtt{in}\,[t_1,t_2]\mathtt{thr}\,[(x_1,z_2),(y_1,y_2)] \Leftrightarrow x_1 \leq z_1, z_1 \leq x_2, x_2 \leq z_2$

Axioms (1), (1′), (2) and (2′) allow one to enlarge the region in which a property holds in a certain interval. If a property A holds both throughout a region R_1 and throughout a region R_2 in every point of the time period I then it holds throughout the region which is the union of R_1 and R_2, throughout I. Notice that the constraints on the spatial variables ensure that the resulting region is still a rectangle. Axiom (3) and (3′) concern the temporal dimension: If a property A holds throughout a region R and in every point of the time periods I_1 and I_2 then A holds throughout the region R in the time period which is the union of I_1 and I_2, provided that I_1 and I_2 are overlapping. By using axiom (4) we can prove that if a property A holds in some point(s) of region R throughout the time periods I_1 and I_2 then A holds in some point(s) of region R throughout the union of I_1 and I_2, provided that such intervals are overlapping. Finally, the last two axioms allow to enlarge the region R in which a property holds in the presence of an `in` temporal annotation.

3.4 Clauses

The clausal fragment of STACLP differs from that of TACLP only for the fact that now atoms can be labelled with spatial and/or temporal annotations.

Definition 3. *A* STACLP clause *is of the form:*

$$A\,\alpha\beta \leftarrow C_1, \ldots, C_n, B_1\,\alpha_1\beta_1, \ldots, B_m\,\alpha_m\beta_m \quad (n, m \geq 0)$$

where A is an atom (not a constraint), α, α_i, β, β_i are (optional) temporal and spatial annotations, the C_j's are constraints and the B_i's are atomic formulae. Constraints C_j cannot be annotated.

A STACLP program *is a finite set of STACLP clauses.*

In our setting, a complex region can be represented (possibly in an approximated way) as union of rectangles. A region *idreg*, divided into n rectangles $\{[(x_1^i, x_2^i), (y_1^i, y_2^i)] \mid i = 1, \ldots, n\}$, is modeled by a collection of unit clauses as follows:

$$resort(idreg)\ \mathtt{thr}\,[(x_1^1, x_2^1), (y_1^1, y_2^1)]. \quad \ldots \quad resort(idreg)\ \mathtt{thr}\,[(x_1^n, x_2^n), (y_1^n, y_2^n)].$$

4 Semantics of STACLP

In the definition of the semantics, without loss of generality, we assume all atoms to be annotated with `th`, `in`, `thr` or `inr` labels. In fact, `at` t and `atp` (x, y) annotations can be replaced with `th` $[t, t]$ and `thr` $[(x, x), (y, y)]$ respectively by exploiting the (`at th`) and (`atp thr`) axioms. Moreover, each atom in the object level program which is not two-annotated, i.e., which is labelled by at most one kind of annotation, is intended to be true throughout the whole lacking dimension(s). For instance an atom $A\ \mathtt{thr}\ R$ is transformed into the two-annotated atom $A\ \mathtt{thr}\ R\ \mathtt{th}\,[0, \infty]$. Constraints remain unchanged.

The meta-interpreter for STACLP is defined by the following clauses:

$$demo(empty). \tag{1}$$

$$demo((B_1, B_2)) \leftarrow demo(B_1), demo(B_2) \tag{2}$$

$$demo(A\ \alpha\beta) \leftarrow \alpha \sqsubseteq \delta, \beta \sqsubseteq \gamma, clause(A\ \delta\gamma, B), demo(B) \tag{3}$$

$$\begin{aligned} demo(A\ \alpha'\beta') \leftarrow\ & \alpha_1\beta_1 \sqcup \alpha_2\beta_2 = \alpha\beta, \alpha' \sqsubseteq \alpha, \beta' \sqsubseteq \beta, \\ & clause(A\ \alpha_1\beta_1, B), demo(B), \\ & demo(A\ \alpha_2\beta_2) \end{aligned} \tag{4}$$

$$demo(C) \leftarrow constraint(C), C \tag{5}$$

A clause $A\,\alpha\beta \leftarrow B$ of a STACLP program is represented at the meta-level by

$$clause(A\,\alpha\beta, B) \leftarrow valid(\alpha), valid(\beta). \tag{6}$$

where *valid* is a predicate that checks whether the interval or the region in the annotation is not empty.

The first two clauses are the ordinary ones to solve the empty goal and a conjunction of goals. The resolution rule (clause (3)) implements both the Modus Ponens rule and the rule ($\sqsubseteq$). It contains two relational constraints on annotations, which are processed by the constraint solver using the constraint theory for temporal and spatial annotations presented in Sections 2.3 and 3.1. Clause (4) implements the rule ($\sqcup$) (combined with Modus Ponens and rule ($\sqsubseteq$)). The constraint $\alpha_1\beta_1 \sqcup \alpha_2\beta_2 = \alpha\beta$ in such a clause is solved by means of the axioms defining the least upper bound introduced in Section 3.3. Clause (5) manages constraints by passing them directly to the constraint solver.

5 Examples

In this section we present some examples which illustrate the expressiveness and the conciseness of STACLP. The first example shows how spatial data can be modeled by annotations and integrated with temporal information. This example is taken from [8], where objects with time-varying activities are modeled in the system *DEDALE*, a generalization of the constraint database model of [10] which relies on a logical model based on linear constraints. Example 2 points out some further peculiarities of our approach that cannot be modeled in [8]. Example 3 and Example 4 describe how it is possible to define moving objects in STACLP. Finally, the last example highlights how the features offered by constraint logic programming improves the reasoning ability in STACLP.

5.1 Ski Tourism

Assume that a person is described by his/her name, the job, the activity and the spatial position(s) in a *certain time interval*. For instance, John is a tourist and from 1am to 10am he sleeps, from 11am to 12am he has breakfast and then in the afternoon he goes skiing up to 4pm, while Monica is a journalist and she skies from noon to 4pm. This can be expressed by means of the following clauses.

$person(john, tourist)$.
$does(john, sleep)$ `atp` $(2, 12)$ `th` $[1am, 10am]$.
$does(john, eat)$ `atp` $(2, 6)$ `th` $[11am, 12am]$.
$does(john, ski)$ `inr` $[(500, 2000), (1000, 2000)]$ `th` $[12am, 4pm]$.

$person(monica, journalist)$.
$does(monica, ski)$ `inr` $[(500, 2000), (1000, 1500)]$ `th` $[1pm, 4pm]$.

The temporal information is represented by a `th` annotation because the property holds throughout the time period. Instead the spatial location is expressed by using an `atp` annotation when the exact position is known, or by an `inr` annotation if we can only delimit the area where the person can be found.

Furthermore, a resort can be described by its name and its area represented by a `thr` annotation.

$resort(terrace)$ `thr` $[(3,5),(8,10)]$.
$resort(ski)$ `thr` $[(50,2000),(1000,2000)]$.

Below we show how some queries from [8], involving the spatial and/or temporal knowledge, can be formulated in our language.

1. Where is John between 12am and 2pm?
   $does(john,\_)$ `inr` R `in` $[12am, 2pm]$
   The answer to this query consists of (possibly different) regions where John stays during that time period. We use the `in` annotation because we want to know all the different positions of John between 12am and 2pm while the `inr` annotation allows one to know the region John is in during that time period, even if his exact position is unknown.
   If we asked for $does(john,\_)$ `atp` R `th` $[12am, 2pm]$ then we would have constrained John to stay in only one place for the whole time period.
   The query $does(john,\_)$ `atp` R `in` $[12am, 2pm]$ asks for definite positions of John sometime in $[12am, 2pm]$.

2. When does Monica stay at the terrace?
   $does(monica,\_)$ `inr` R `th` I, $resort(terrace)$ `thr` R
   The result is the time interval I in which Monica's position is somewhere in the terrace area.

3. Where is John while Monica is at the terrace?
   $does(john,\_)$ `inr` R `th` I, $does(monica,\_)$ `inr` $R1$ `th` I, $resort(terrace)$ `thr` $R1$
   This query is a composition of a spatial join and a temporal join.

4. In which places does Monica sleep?
   $does(monica, sleep)$ `thr` R `in` $[0am, 12pm]$

5. Where did Monica and John meet?
   $does(john,\_)$ `atp` P `at` T, $does(monica,\_)$ `atp` P `at` T
   This query exploits the fact that two people meet if they are in the same place at the same time.

6. Who ate in the skiing area, and when?
   $does(X, eat)$ `inr` R `th` I, $resort(ski)$ `thr` R

Grumbach et al. [8] represent only definite spatial and temporal data, corresponding to our `thr` and `th` annotations. Indeed, they model the trajectory of a person as a set of regions associated with time periods where the person can be found, a sort of *indefinite spatial* information. Thus, in Grumbach et al. the difference between definite and imprecise information is blurred, whereas in our framework it can be captured by resorting to `inr` annotations for indefinite spatial data (see e.g., $does(monica, ski)$ `inr` $[(500,2000),(1000,1500)]$ `th` $[1pm, 4pm]$).

Furthermore, in [8] time and space are associated with attributes of relations. The temporal and the spatial dimensions are independent, thus it is not possible

to express spatial relations parametric with respect to time. Hence, their concept of moving objects captures only *discrete* changes. We will see in Section 5.3 that, instead, in STACLP continuous changes can be naturally modeled.

An advantage of [8] is that it allows for a more compact representation of the information by means of disjunctive constraints. On the other hand, here we use only conjunctive constraints, and we model disjunction by defining different clauses, one for each disjunct. Observe also that, for the absence of negation, STACLP does not allow us to express those database operations requiring negation, like difference between relations. The treatment of negation is not straightforward and represents an interesting topic of future research.

5.2 Periodicity and Indefiniteness

In STACLP one can express also periodic temporal information and indefinite data both in time and/or space.

Example 2. Suppose that Frank works somewhere in Florence (indefinite spatial information) on Wednesdays (periodic temporal information). Such situation can be simply expressed by the clause

$$does(frank, work)\ \texttt{inr}\ R\ \texttt{at}\ T \leftarrow wed\ \texttt{at}\ T, resort(florence)\ \texttt{thr}\ R$$

The predicate *wed* is defined as

$$wed\ \texttt{at}\ w. \qquad\qquad wed\ \texttt{at}\ T + 7 \leftarrow wed\ \texttt{at}\ T$$

where w is the date of a Wednesday. The predicate *wed* holds if T is Wednesday and $resort(florence)\ \texttt{thr}\ R$ binds R to the area of the town of Florence (represented by a set of rectangles).

Moreover, to express the fact that Frank has a break some time between 5pm and 6pm (indefinite temporal information) at the terrace we can use the clause

$$does(frank, break)\ \texttt{inr}\ R\ \texttt{in}\ [5pm, 6pm] \leftarrow resort(terrace)\ \texttt{thr}\ R$$

5.3 Moving objects

The STACLP framework allows one to describe spatial relations which may be parametric with respect to time, i.e., with respect to their evolution, for instance time-varying areas and moving points. In the following examples we assume that time and space are interpreted over "compatible" domains, for instance if time is dense then also the spatial domain has to be dense.

Example 3. Suppose to have a rectangular area on a shore, where a tide is coming in. The front edge of the tide water is a linear function of time. At 1:00am the area flooded by the water will be a line, then it will start being a rectangle with a linearly growing area. We can represent such a phenomenon by the following clause:

$$floodedarea\ \texttt{thr}\ [(2, 8), (2, 1 + T)]\ \texttt{at}\ T \leftarrow 1 \leq T \leq 6$$

The spatial annotation is parametric with respect to time and this allows an interaction between the spatial and temporal attributes. The idea is similar to the Parametric 2-spaghetti model proposed by Chomicki and Revesz in [4]. In such an approach, which generalizes the 2-spaghetti Model, objects are triangulated and the vertex coordinates of the triangles can be linear functions of time.

Example 4. A moving point can be modeled easily by using a clause of the form:

$$\mathit{moving\_point}\ \mathtt{atp}\ (X, Y)\ \mathtt{at}\ T \leftarrow \mathit{constraint}(X, Y, T)$$

For instance consider a car moving on a straight road with speed v and assume that its initial position at time t_0 is (x_0, y_0). The position (X, Y) of the car at T can be computed as follows:

$$\mathit{car\_position}\ \mathtt{atp}\ (X, Y)\ \mathtt{at}\ T \leftarrow X = x_0 + v(T - t_0), Y = y_0 + v(T - t_0)$$

In a similar way we can represent regions which move continuously in the plane.

5.4 Transitive Closure

Suppose that we want to describe towns and roads of a region and inquire the system about the connections among them. This is a typical example of network analysis, that may find applications in many areas (see e.g. [21]). To do this kind of analysis inside our framework, we can exploit the deductive power and the possibility of defining recursive predicates of constraint logic programming.

First of all we give the definition of a general predicate *path*, that given the identifiers of two objects, O_1 and O_2, a list of properties, *LProp*, and a time period $[T_1, T_2]$, returns a possible way to reach O_2 from O_1 crossing areas satisfying properties in *LProp* during the given time period. The temporal component is associated with the properties, since it is very common that properties vary in time. For instance during Winter some roads may not be available because of the snow, and thus, in order to find a right path, temporal information must be taken into account.

$$\begin{aligned}
&\mathit{path}(O_1, O_2, \mathit{LProp}, \mathit{Acc}, [O_2|\mathit{Acc}])\ \mathtt{th}\ [T_1, T_2] \ \leftarrow \mathit{obj}(O_1)\ \mathtt{thr}\ R, \mathit{obj}(O_2)\ \mathtt{thr}\ R \\
&\mathit{path}(O_1, O_2, \mathit{LProp}, \mathit{Acc}, L)\ \mathtt{th}\ [T_1, T_2] \ \leftarrow \\
&\qquad O_1 \neq O_2, \mathit{hasProp}(O, \mathit{Prop})\ \mathtt{th}\ [T_1, T_2], \\
&\qquad \mathit{member}(\mathit{Prop}, \mathit{LProp}), \mathit{nonMember}(O, \mathit{Acc}), O \neq O_2, \\
&\qquad \mathit{obj}(O)\ \mathtt{thr}\ R, \mathit{obj}(O_1)\ \mathtt{thr}\ R, \mathit{path}(O, O_2, \mathit{LProp}, [O|\mathit{Acc}], L)
\end{aligned}$$

The predicates *member* and *nonMember* check whether an object does or does not belong to a list, respectively. The fourth argument of *path* is an accumulator in which we collect the objects we have already selected during the computation, and the fifth argument is the list, in inverse order, of the objects crossed to reach O_2 from O_1.

The meaning of the two clauses is straightforward: The first one states that if the two objects intersect then our search for the path is finished. Otherwise, we look for an object O, different from O_2, for which one of the properties in

LProp holds in the time period $[T_1, T_2]$, which has not been selected yet, and which intersects the object O_1. Finally, we make a recursive call to find a path between O and O_2.

Now we can easily find a route to go from a town to another during a certain time period, if this route exists, by using roads.

$$route(Town_1, Town_2, L)\ \mathtt{th}\,[T_1, T_2] \leftarrow path(O_1, O_2, [road], [O_1], L)\ \mathtt{th}\,[T_1, T_2]$$

The definition of the predicate *route* consists in specifying *road* as property for the objects we want to use to build a path from $Town_1$ to $Town_2$.

The main advantage of this approach is at the specification level: We declaratively state what is a route without having a fixed network. In other words, the only information we employ is the area of the represented objects, we do not use predefined nodes and links between them to move from one position to another. This leaves a larger freedom to specify conditions that the route has to satisfy and it makes the approach general, and application independent.

6 Conclusions

We have extended the Temporal Annotated Constraint Logic Programming framework by adding spatial annotations, resulting in STACLP. As for TACLP, the approach is conceptually simple and expressive.

The clausal fragment of STACLP can be implemented by encoding the inference rules directly in a constraint logic programming language. We are currently developing an implementation of the constraint theory needed to perform the lattice operations on spatio-temporal annotations, by using the constraint handling rules (CHR) library of Sicstus Prolog [18]. Such an implementation is a straightforward extension of the already implemented TACLP framework, which has been proved to be tractable and efficient [7].

For some application areas like spatial databases and GISs, the STACLP framework is still not expressive enough: It lacks some set-based operations on geometric objects, such as set-difference and complement which would require the presence of negation in the language. The treatment of negation in STACLP is an interesting topic for future research. Furthermore, STACLP does not provide operations for expressing topological relations. In the spirit of Egenhofer's 9-intersection model [5], including the definition of the boundary and of the interior of a spatial object should allow to find the topological relations existing between two convex objects. More work is needed in this direction.

Finally, it would be interesting to cope with different granularities in space and time, a capability which is particularly relevant to support interoperability among systems [2, 4]. For instance, since spatial and temporal information can be expressed in diverse measurement units (meters, feet, seconds, days), one could think of introducing in STACLP a notion of unit and a set of conversion predicates.

Acknowledgments: We thank Paolo Baldan for his useful comments. This work has been partially supported by Esprit Working Group 28115 - DeduGIS and Programma Vigoni 2000.

References

1. A. Belussi, E. Bertino, and B. Catania. An extended algebra for constraint databases. *IEEE TKDE*, 10(5):686–705, 1998.
2. C. Bettini, X. S. Wang, and S. Jajodia. An architecture for supporting interoperability among temporal databases. In *[6]*, pages 36–55. 1998.
3. M.H. Böhlen, C.S. Jensen, and M.O. Scholl, editors. *Spatio-Temporal Database Management*, volume 1678 of *LNCS*. Springer, 1999.
4. J. Chomicki and P.Z. Revesz. Constraint-Based Interoperability of Spatiotemporal Database. *Geoinformatica*, 3(3):211–243, 1999.
5. M. J. Egenhofer. Reasoning about binary topological relations. In *Advances in Spatial Databases*, volume 525 of *LNCS*, pages 143–160. Springer, 1991.
6. O. Etzion, S. Jajodia, and S. Sripada, editors. *Temporal Databases: Research and Practice*, volume 1399 of *LNCS*. Springer, 1998.
7. T. Frühwirth. Temporal Annotated Constraint Logic Programming. *Journal of Symbolic Computation*, 22:555–583, 1996.
8. S. Grumbach, P. Rigaux, and L. Segoufin. Spatio-Temporal Data Handling with Constraints. In *ACM GIS*, pages 106–111. ACM Press, 1998.
9. R.H. Güting. An Introduction to Spatial Database Systems. *VLDB Journal*, 3(4):357–400, 1994.
10. P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26–52, 1995.
11. M. Koubarakis and S. Skiadopoulos. Tractable Query Answering in Indefinite Constraint Databases: Basic Results and Applications to Querying Spatiotemporal Information. In *[3]*, pages 204–223, 1999.
12. P. Mancarella, G. Nerbini, A. Raffaetà, and F. Turini. MuTACLP: A Language for Declarative GIS Analysis. In *DOOD2000*, volume 1861 of *LNAI*, pages 1002–1016. Springer, 2000.
13. K. Marriott and P. J. Stuckey. *Programming with Constraints.* MIT Press, 1998.
14. M. A. Orgun and W. Ma. An Overview of Temporal and Modal Logic Programming. In *ICTL'94*, volume 827 of *LNAI*, pages 445–479. Springer, 1994.
15. J. Paredaens. Spatial databases, the final frontier. In *ICDT'95*, volume 893 of *LNCS*, pages 14–32. Springer, 1995.
16. A. Raffaetà and T. Frühwirth. Semantics for Temporal Annotated Constraint Logic Programming. In *Labelled Deduction*, volume 17 of *Applied Logic Series*, pages 215–243. Kluwer Academic, 2000.
17. D. Randell, Z. Cui, and A. Cohn. A Spatial Logic based on Regions and Connection. In *KR1992*, pages 165–176. Morgan Kaufmann, 1992.
18. SICS. *Sicstus Prolog User's Guide*, 1995.
19. A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass editors. *Temporal Databases: Theory, Design, and Implementation.* Benjamin/Cummings, 1993.
20. F. Wolter and M. Zakharyaschev. Spatio-temporal representation and reasoning based on RCC-8. In *KR2000*, pages 3–14. Morgan Kaufmann, 2000.
21. M. F. Worboys. *GIS - A Computing Perspective.* Taylor & Francis, 1995.