Temporal Reasoning in a Meta Constraint Logic Programming Architecture

Evelina Lamma, Michela Milano

DEIS
Università di Bologna
Bologna 40136, Italy

Paola Mello

Istituto di Ingegneria
Università di Ferrara
Ferrara 41100, Italy

Abstract

Constraint Logic Programming (CLP) is a powerful programming paradigm combining the advantages of Logic Programming and the efficiency of constraint solving. However, CLP presents some limitations in dealing with temporal reasoning. First, it uses an "arc consistency" propagation algorithm which cannot be changed by the user and it is too weak in many temporal frameworks. Second, CLP is not able to deal with qualitative temporal constraints.

In this paper, we show how to overcome these limitations. In particular, we present a way of performing a path-consistency check without changing the propagation algorithm of the constraint solver. In addition, we show how to integrate qualitative and quantitative temporal reasoning by using a two module meta CLP architecture. Each module is a finite domain constraint solver (CLP(FD)). The object system (extended with the path-consistency algorithm) performs quantitative reasoning, while the meta-level reasons on constraints of the underlying system thus performing qualitative reasoning.

In this way, we can benefit of the efficiency of the constraint handling mechanism of CLP and the modularity, flexibility and scalability of meta-architectures.

1 Introduction

Temporal reasoning plays a central role in many Artificial Intelligence applications such as planning, scheduling, and natural language understanding. Several frameworks have been proposed in order to deal with this form of reasoning, most notably Allen's Interval Algebra [2], Vilain and Kautz's Point Algebra [29], Dechter, Meiri and Pearl's STP and TCSP [9] and Dean and McDermott's Time Map Management [8]. In addition, some promising approaches have been proposed in order to integrate qualitative and quantitative temporal reasoning [16,21].

In these frameworks the representation of the problem is given in terms of a constraint graph where nodes are the variables of the problem, i.e., points or intervals, and arcs are constraints, i.e., temporal relations between pairs of variables (each n-ary relation among n variables can be represented in terms of binary constraints). Given a constraint graph, we are interested either in determining whether the graph is consistent, or in determining an equivalent minimal representation, or in computing the transitive closure of the network. All the researchers who have worked in the temporal reasoning field have proposed algorithms for solving these problems (see, for example, [2,9,26,29]) but, in general, they did not focus on defining a programming paradigm and architecture for solving these problems.

In this paper, we show how Constraint Logic Programming [14,15] on finite domains (CLP(FD)) can be a flexible and suitable tool for temporal reasoning thanks to its efficient constraint propagation mechanism. However, CLP presents some limitations in the treatment of both quantitative and qualitative temporal constraints. On one hand, CLP has a limited embedded propagation mechanism (*arc-consistency*) which cannot be changed by the user. On the other hand, qualitative constraints can be expressed but not propagated (in order to find, for example, the minimal network) in a CLP(FD) framework. We overcome this drawbacks by integrating both kinds of reasoning (as in the Meiri's Framework [21]) in a flexible and modular architecture.

Our meta architecture contains two finite domain constraint solvers which cooperate and exchange knowledge in order to deal with both kinds of constraints: The object level treats quantitative information, while the meta level handles qualitative information by reasoning on the (passive) constraints of the object level.

The contribution of this paper mainly concerns:

- a way of performing a path-consistency algorithm not by changing the propagation algorithm of the constraint solver as in [6], but by performing an arc-consistency on the so called "arc-variables" and "path-equivalent" constraints;
- a way of handling qualitative temporal constraint propagation by adding a meta constraint solver reasoning on constraints of the underlying system, as proposed by Tsang [25] with *meta-constraint* graphs.

In this way, we can benefit of the efficiency of the constraint handling mechanism provided by CLP, and

0-8186-7528/96 $5.00

the modularity, flexibility and scalability of meta-architectures.

The paper is organized as follows. In Section 2 we will sketch some features of the CLP(FD) paradigm while focusing on its limitations for treating qualitative temporal reasoning. Then, we propose how to overcome these limitations. Section 3 is devoted to the architecture of the general purpose temporal reasoner, i.e., the qualitative and the quantitative modules and their interactions. Section 4 is devoted to an example. Conclusions and a mention of future works follow.

2 Constraint Logic Programming

Constraint Logic Programming (CLP) is a class of programming languages combining the advantages of Logic Programming (LP) ([20]) and the efficiency of constraint solving. In this paper we focus on CLP on finite domains CLP(FD) [15]. In this framework, variables range on a finite domain which is initially set to [0..max]. These domains are gradually reduced by *the propagation of constraints*. *Inconsistency* is detected when a domain becomes empty.

As an example, let us suppose that we have three variables X, Y, Z ranging on the domain [1..10], and the constraint $X < Y$ which produces the propagation: $D_X = [1..9]$ and $D_Y = [2..10]$ and the constraint $Y < Z$ which produces the propagation: $D_Y = [2..9]$ and $D_Z = [3..10]$. The fact that D_Y has changed produces the final propagation of constraints leading to the reduction of the domains of X, Y, Z to: $D_X = [1..8]$, $D_Y = [2..9]$, $D_Z = [3..10]$.

This propagation is named "arc-consistency" and ensures that, for each value of a variable domain D_X, there exists at least one value in each other domain which is consistent with the constraints mentioning the variable X.

2.1 CLP(FD) and its Limitations for Temporal Reasoning

CLP has a very efficient constraint handling mechanism which can be suitably used for representing and reasoning on constraint-based temporal frameworks.

CLP(FD) is able to express directly quantitative temporal reasoning. In fact, we can interpret the constraints:
$(x \in [3..16]) \wedge (y \in [1..19]) \wedge (x - y \geq 7) \wedge (x - y \leq 16)$
in a temporal reasoning context. We can see the variables x and y as temporal points which have a temporal location on the discrete time line (see [3]). In particular, x is a temporal point located somewhere between 3 and 16 time units, y between 1 and 19. The last two constraints link the distance of the time points to range between 7 and 16 time units.

The propagation of constraints leads to the reduction of the variables' domains to[1]:
$(x \in [4..16]) \wedge (y \in [1..13])$
while suspending the two passive constraints
$(x - y \geq 7) \wedge (x - y \leq 16)$.
Therefore, with CLP(FD) we can express quantitative constraints between temporal points, i.e., ranges of absolute locations on the time line and bound on distance constraints (see [9]).

Temporal intervals can be expressed as pairs of points linked by the relation $I_{inf} < I_{sup}$, where I_{inf} and I_{sup} are the starting and ending points of the interval I.

According to the classification of Meiri [21], we can represent augmented Continuous Point Algebra (CPA) and Point Algebra (PA) networks on discrete domains. The augmented CPA networks on discrete domains require an arc consistency algorithm for deciding the consistency and an arc+path-consistency algorithm for computing the minimal domains. CLP(FD) solvers like CHIP [10] or ECL^{i}PS^e [11] provide only an arc consistency algorithm. In section 3.1 we propose a way of performing path consistency not by changing the consistency algorithm of the solver, but by adding what we call "arc-variables" and "path-equivalent" constraints. On the other hand, deciding the consistency of PA networks on discrete domains is a NP-complete problem. Therefore, we can apply a path consistency algorithm just as a preprocessing technique for reducing the search space.

Another limitation of CLP(FD) concerns its capability of representing qualitative constraints such as $x < y$, but its inadequacy of reasoning about them. This inadequacy arises for three reasons that we have identified and called the *composition problem*, the *intersection problem* and the *qualitative-quantitative problem*:

Composition: in the example of Section 2, when the propagation terminates, we are not able to infer information on the relationship that holds between X and Z starting from the domains $D_X = [1..8]$ and $D_Z = [3..10]$. CLP is not able to build the transitive closure of the temporal network. The information that should be inferred from the two constraints $X < Y$ and $Y < Z$ is the relation $X < Z$. This relation should be inferred from the transitivity rule, and, in particular, it is the result of the composition of constraints, see for example [2,29].

Intersection: the second problem of CLP on finite domains is related to the first and concerns the computation of the minimal network. The constraints $X < Y$, $Y < Z$ and $X \leq Z$ are consistent but not minimal. A constraint solver should be able, in this case, to compose the first two constraints and intersect the result with the third constraint leading to $X < Z$.

Qualitative-Quantitative: another limitation of CLP on finite domains is that it deletes values according to constraints, but it does not reduce constraints on the basis of variable domain values. Suppose that we state that $X \leq Y$, X and Y being temporal points ranging on $D_X = [1..5]$ and $D_Y = [6..10]$ respectively. It can be seen immediately that X and Y cannot be equal because there does not exist any pair of values from D_X and from D_Y that "supports" the constraint. Therefore, the tightest constraint that holds between X and Y is $X < Y$. A constraint solver on finite domains propagates constraints and simply concludes that the constraint is satisfied without simplifying it.

[1] These constraints are called active constraints.

2.2 How to Overcome these Problems

In the previous Section, we have underlined that CLP(FD) presents some limitations for qualitative reasoning and a limited consistency algorithm for quantitative reasoning.

Therefore, we propose an approach based on a meta-architecture [1,5] that solves the above mentioned limitations. We use a constraint solver on finite domains for handling quantitative constraints, and a meta-constraint solver for qualitative temporal constraints.

On one hand, we extend the object level solver with the concept of "arc-variables" and "path-equivalent" constraints for reaching path-consistency without modifying the algorithm of the constraint solver.

On the other hand, we solve the qualitative reasoning limitations by building a meta-constraint solver where variables are relations. On meta-level, unary constraints restrict a meta-variable to take its values from a finite domain of relations. For each pair of variables X and Y (nodes of the graph) we have a meta-variable, namely R_{XY}, ranging on a finite domain of relations. This domain contains the relations that hold between X and Y. Suppose that $X \leq Y$, then the domain of the meta-variable R_{XY} is the powerset of the possible constraints holding between X and Y, namely $D_{XY} = [\{<\}, \{=\}, \{<, =\}, \{\}]$.

Intersection and composition are embedded in the meta-constraint solver as primitive operations.

Therefore, the general purpose temporal reasoner is composed by two constraint solvers propagating their "own" constraints, and interacting each other in order to perform a more powerful propagation.

3 A General Purpose Temporal Reasoner based on CLP

In this section, we explain how the two modules propagate qualitative and quantitative constraints, and how they interact. An extended version of this paper including theoretical issues and proofs is available [17].

3.1 Quantitative Constraint Solver

The quantitative module is a CLP(FD) constraint solver. CLP on finite domains can express unary constraints between points such as: $P_i \in I$ and binary constraints between pairs of points such as: $(P_i - P_j) \in I$. The qualitative constraints of a CPA (or PA) network can be translated in bound on distance constraints (see [21]). Therefore, we now concentrate on bound on distance constraints.

Let us see an example: if variable X ranges on the domain $D_X = [10..20]$ and variable Y ranges on the domain $D_Y = [40..50]$ and the constraint $Z = Y - X$ is stated, then the constraint solver is able to associate with the variable Z the domain $D_Z = [20..40]$. Again, if the variable X ranges on the domain $D_X = [10..20]$ and the variable Z on the domain $D_Z = [30..40]$, then the propagation of the constraint $Z = Y - X$ associates with the variable Y the domain $D_Y = [40..60]$ These constraints, after the propagation, are delayed because variables are not yet instantiated and may cause further propagations. This propagation is that performed in the STP framework [9]. In STP we use continuous domains while in this case we use discrete domains. However, the propagation, in this case, can be performed only on upper and lower bounds of each domain. Therefore, the mechanism is the same.

Now we explain how bound on distance constraints are propagated in the temporal reasoner. We refer to an example given by Dechter, Meiri and Pearl in [9]. We have four temporal variables, namely X_1, X_2, X_3, X_4, and a time point X_0 representing the "beginning of the world". The constraints defined by the problem are:

$$10 \leq X_1 - X_0 \leq 20,\ 30 \leq X_2 - X_1 \leq 40,$$
$$10 \leq X_3 - X_2 \leq 20,\ 40 \leq X_4 - X_3 \leq 50,$$
$$60 \leq X_4 - X_0 \leq 70.$$

These constraints can be written as:

```
% The first eleven constraints
% are stated by the problem
X1::10..20, Z21::30..40, Z21=X2-X1,
X4::60..70, Z43::40..50, Z43=X4-X3,
Z23::10..20, Z23=X2-X3, Z42=X4-X2,
Z41=X4-X1,Z31=X3-X1,
% The following constraints perform
% a path-consistency equivalent algorithm
Z31=Z21-Z23, Z31=(-Z43)-(-Z41), Z21=Z23-(-Z31),
Z21=Z24-(-Z41), Z43 =Z41-Z31, Z43=Z42-(-Z23),
Z42=Z43-Z23, Z42=Z41-Z21, Z41=Z43-(-Z31),
Z41=Z42-(-Z21,) Z23=Z21-Z31, Z23=(-Z42)-(-Z43).
```

where the predicate symbol :: associates a variable `Var` with a domain `Domain` in the following way: `Var :: Domain`. The variables Z_{21}, Z_{23}, Z_{43}, Z_{42}, Z_{41} and Z_{31} are called "arc-variables" because they refer to arcs. The first eleven constraints define the structure of the problem. In their paper [9] the authors have shown that the *Floyd-Warshall*'s all-pairs shortest-paths algorithm represents a complete algorithm for determining the consistency of a STP and for computing the minimal network. Moreover, they have shown that for STP this algorithm is equivalent to the path consistency algorithm. On the other hand, a CLP(FD) solver performs a lower consistency check: an arc consistency algorithm which does not check all the paths of the network given the variables of the problem. However, if we add to a program some constraints (the last 12 constraints in the example above, i.e., the "path-equivalent" constraints) describing the paths in the graph and we apply arc-consistency on them, we are able to obtain path consistency.

Theorem 3.1 *Let G=(V,A) be a graph where each node represents a variable (temporal point) of the problem and each arc, linking a pair of variables, is a binary constraint defining the minimal and maximal distance between the two points. Let us consider three variables X, Y and Z linked by the following bound on distance constraints:*

$$a_1 \leq X - Y \leq a_2,$$
$$b_1 \leq X - Z \leq b_2,$$
$$c_1 \leq Z - Y \leq c_2,$$

The path through nodes X, Y *and* Z *is path consistent iff the following unary constraints are arc consistent:*

```
XY :: a1..a2,
XZ :: b1..b2,
ZY :: c1..c2,
XY = XZ + ZY,
```

where XY = X-Y, XZ = X-Z *and* ZY = Z-Y.

Proof.: Refer to [17].
Without modifying the CLP(FD) solver, we are able to reach the same result obtained by [6] which, instead, modifies the structure of the CLP(FD) solver for achieving path-consistency.

The result of the computation of the above-mentioned example consists of delayed constraints in which each variable is associated with the minimal domain:

```
X1::10..20, X2::40..50, X3::20..30,
X4::60..70, Z21::30..40, Z31::10..20,
Z41::50..60, Z23::10..20, Z42::20..30,
Z43::40..50.
```

Therefore, with CLP(FD) we are able to solve a STP, to determine its consistency and to compute the minimal network associated with, provided that we add some "path-equivalent" constraints. We are currently studying a way of performing whatever kind of consistency by adding constraint solvers as meta-levels, see [18].

As concerns the complexity of the above mentioned computation, we have proved [17] that it is equal to the complexity of a path consistency algorithm augmented by the overhead of creating arc variables and path equivalent constraints. This result is mainly due to the fact that in the STP framework variables ranges on convex intervals, and arc consistency can modify the lower and upper bound of each domain, thus avoiding the domain values enumeration.

3.2 Qualitative Constraint Solver

The qualitative module is a meta-constraint solver on finite domains of relations. A meta-variable R_{XY} is automatically generated each time a new qualitative constraint between variables X and Y is introduced. Qualitative constraints are disjunctions of primitive constraints between variables. If X and Y are points, primitive constraints are point algebra relations [29] (with or without the constraint $\neq$). If X and Y are intervals the primitive constraints are interval algebra relations [2]. The intervals will be translated in their starting and ending points.

Therefore, if between variables X and Y we have a relation of the kind: $(X\ r_1\ Y) \vee \ldots \vee (X\ r_n\ Y)$ which can be written as: $X\{r_1, \ldots, r_n\}Y$ where r_is are primitive qualitative relations, then the system automatically creates a meta-variable R_{XY} whose domain D_{XY} is the powerset of $\{r_1, \ldots, r_n\}$.

The meta constraint solver has two operations: intersection and composition of qualitative constraints. The intersection of two constraints, linking the same pair of variables $X\{r_1, \ldots, r_n\}Y$ and $X\{r'_1, \ldots, r'_n\}Y$, whose symbol is $=_m$, is the powerset of the set theoretic intersection between the two vectors $\{r_1, \ldots, r_n\}$ and $\{r'_1, \ldots, r'_n\}$. A meta-variable representing the possible relations between X and Y is associated with each constraint. Therefore, the intersection of meta-variables $R_{XY} =_m R'_{XY}$ (corresponding to the two constraints) imposes the meta-variables to be equal by intersecting their domains.

For example, if the meta-variable R_{XY} has an associated domain: $D_{XY} = [\{<\}, \{=\}, \{<, =\}, \{\}]$ because of the existence of the qualitative constraint $X \leq Y$[2], and the meta-variable R'_{XY} has an associated domain $D'_{XY} = [\{>\}, \{>, =\}, \{=\}, \{\}]$ because of the existence of the qualitative constraint $X \geq Y$, then the intersection between the two constraints is performed by a meta-unification $R_{XY} =_m R'_{XY}$ that leads to the following propagation of constraints: $D_{XY} = D'_{XY} = [\{=\}, \{\}]$ corresponding to the qualitative domain $X = Y$ in the finite domain constraint solver.

The composition should be defined directly in the constraint solver via a transitivity table (see [2,29]). The composition is represented by the symbol $+_m$ and it is defined as following. The composition of two meta-variables R_{XZ} and R_{ZY} admits only those values r_{xy} for which there exists $r_{xz} \in D_{XZ}$ and $r_{zy} \in D_{ZY}$ such that $r_{xy} = r_{xz} \otimes r_{zy}$ where the symbol $\otimes$ represents the composition of primitive constraints and is defined by the transitivity table on the constraints of D, see [2,29].

The "tighter" relation is represented by the symbol $<_m$. A set of constraints C' is tighter than another set of constraints C" if each pair of values allowed by C' is also allowed by C". In our case, the relation tighter can be translated into the set inclusion.

It is worth noting that, as in the quantitative solver, we should perform an arc consistency on the meta-variables representing arcs of the object level graph. In fact, the constraint $R_{XY} =_m R_{XZ} +_m R_{ZY}$ performs a composition of R_{XZ} and R_{ZY} in the way defined above, and intersects the computed domain with the domain of R_{XY}. This constraint behaves in the same way of the constraint $XY = XZ + ZY$ described in Section 3.1.

Notice that the qualitative constraint solver can work independently from the quantitative part. For example the following qualitative program:

```
qualitative:-
                RXY:: [{<},{=},{<,=},{}],
                RYZ:: [{<},{}],
                RXZ =m RXY+m RYZ,
                RKH:: [{=},{}],
                RKH <m RXY.
```

contains unary constraints which link the variables to take their values from a set of relations, and binary

[2]The symbol {} is useful for detecting inconsistencies in qualitative reasoning.

constraints containing operations on meta-variables, i.e., intersection and composition, or meta-relations such as the tighter relation. Therefore, in the meta-level we can express the Vilain and Kautz's Point Algebra [29] which deals with qualitative relations between temporal points.

It is worth noting that qualitative information, except for the relation $\neq$, can be translated into non-disjunctive bound on distance constraints. For example, the constraint $X \leq Y$ can be translated in $-\infty \leq X - Y \leq 0$. However, a CLP(FD) solver does not translate this constraint in this way since it applies an "arc-consistency", and therefore the distance of the two variables does not necessarily range on $-\infty..0$. For example, if we have $X :: [0..10]$ and $Y :: [-6..30]$, $Y \leq X$, $Z = X - Y$, the CLP(FD) solver propagates the constraints thus resulting in $X :: [0..10]$, $Y :: [-6..10]$, $Z :: [-10.. + 16]$. Therefore, we need to change the propagation algorithm if we want to achieve $Z :: [0..16]$. Second, we gain in expressiveness since the user can express the fact that the relation that holds between variables X and Y should be tighter than the relation that holds between, say, Z and K. The user can also query the system by asking a feasible scenario for qualitative temporal information (we will show this feature in the example in section 4). This query can be straightforwardly implemented if we have a finite domain of relations at meta level, also using domain dependent heuristics embedded in the meta level.

3.3 Interactions between Qualitative and Quantitative solvers

One of the most interesting parts of the temporal reasoner concerns the integration of qualitative and quantitative constraint solvers. This integration concerns the propagation of quantitative constraints when qualitative domains are modified and the propagation of qualitative constraints when quantitative domains are modified. For this part, our propagation method is different from the Meiri's propagation technique but it is straightforward in a Constraint Logic Programming paradigm. We have defined two linking rules which ensure that a propagation in the quantitative solver is reflected in the qualitative solver and vice versa. Due to space limitations, we do not describe formally these linking rules (reported in [17]), but we give an informal idea of the propagation adopted.

The *qualitative-quantitative* propagation, denoted in section 4 with `qual_quan`, is straightforward. It is equivalent to the addition in a CLP program of a new qualitative constraint. For example, if variables X and Y are constrained by $X \leq Y$, they have an associated meta-variable R_{XY} ranging on $D_{XY} = [\{<\}, \{=\}, \{<, =\}, \{\}]$. Suppose now that, for the propagation of constraints in the meta constraint solver, the domain D_{XY} is reduced to $D_{XY} = [\{<\}, \{\}]$. In the quantitative solver the propagation is performed by adding the new constraint $X < Y$ to the constraint store. This new constraint may cause further propagations on quantitative domains.

The *quantitative-qualitative* interaction, denoted in section 4 with `quan_qual`, concerns the propagation that should be performed when a change in a quantitative domain influences a qualitative domain. First of all, every time a new constraint is encountered in the object system, its meta-representation should be added to the meta constraint store.

In addition, each time a variable domain changes due to the object system propagation, a linking rule provides a propagation in the meta system. This propagation is much more computational expensive and leads to the deletion of constraints that are not supported by variable domain values.

We should check for each constraint linking two variables X and Y, if there exists at least one couples of values $x \in D_X$ and $y \in D_Y$ that supports the constraints between X and Y, represented by the meta-variable R_{XY}. In other words, this propagation checks if the set:

$$S = \{(x, y) \in D_X \times D_Y \mid x \; R_{XY} \; y\}$$

is not empty. The notation $x \; R_{XY} \; y$ represents the "application" of one of the constraints in the domain of the meta-variable R_{XY} to x and y.

For example, suppose that X has an associated domain $D_X = [3..5]$, Y the domain $D_Y = [6..9]$ and X and Y are related by the constraint $X \leq Y$, which is represented by the meta-variable R_{XY} ranging on the domain $D_{XY} = [\{<\}, \{=\}, \{<, =\}, \{\}]$. Intuitively, the constraint "equal" is no longer entailed by the values of the domains of X and Y. In fact, while the set S is not empty for the value $<$ of R_{XY} because it contains, for example, the couples (3,6), (3,7), (3,8) and so on, the set S is empty for the value $=$ of R_{XY}. Therefore, the *quantitative-qualitative* propagation deletes from the domain of R_{XY} the values containing the constraint $=$.

This process is general, but very heavy handled. If we work in a specific domain we can make use of special purpose techniques for the interaction between qualitative and quantitative constraints, see [17].

4 One Example

In this Section we present one example which applies to this architecture, taken from Meiri's paper [21], which handles both qualitative and quantitative temporal knowledge. In the example, we represent points as CLP domain variables while intervals are represented by their starting and ending points `I-` and `I+` linked by the constraint `I- < I+`. The relations between intervals are the usual Allen's relations [2]: starts (symbol `s`), during (symbol `d`), overlaps (symbol `o`), before (symbol `<`), equal (symbol `=`), finishes (symbol `f`), meets (symbol `m`) and their inverses (`si`, `di`, `oi`, `>`, `=`, `fi`, `mi` respectively).

John and Fred usually work at a local office, in which case John takes less than 20 minutes and Fred 15-20 minutes to get to work. Twice a week John works at the main office, in which case he takes at least 60 minutes. Today John left home between 7:05 and 7:10 a.m., and Fred arrived at work between 7:50 and 7:55 a.m. John and Fred met at the traffic light on their way to work.

The representation of the problem should be written in a configuration file. This file contains information about the above-mentioned story:

```
interval(J, (J-,J+)),
interval(F, (F-,F+)),
point(P0),
origin(P0)
arc(J, F, [s,si,o,oi,f,fi,d,di,=]),
distance(J-, P0, D1),
distance(F+, P0, D2),
distance(F+, F-, D3),
distance(J+, J-, D4),
D1::0..5, D2::50..55,
D3::15..20, D4::0..20,60..max.
```

where J is the interval representing the event John going to work, F is the interval representing the event Fred going to work and **origin(P0)** is a predicate which holds iff P0 is the beginning of the world. The result of the propagation is the same as that reported by Meiri, but it is interesting to follow the two solvers' steps in order to understand their behavior.

A pre-processing additional module interval-point translates the relations of the Interval Algebra in relations between their starting and ending points. The interval-interval relation:

```
arc(J, F, [s,si,o,oi,f,fi,d,di,=]),
```

is translated into six point-point relations:

```
arc(J-, F-, [<,=,>]),
arc(J+, F+, [<,=,>]),
arc(J+, F-, [>]),
arc(J-, F+, [<]),
arc(J-, F+, [<]),
arc(J-, J+, [<]),
arc(F-, F+, [<]).
```

Notice that the constraints `J- [<,=,>] F-`, and `J+ [<,=,>] F+`, are not added to the quantitative program because they do not restrict the variable domains.

These relations are automatically transformed by **quan_qual** in meta-variables of the meta-constraint solver:

$$
\begin{aligned}
&R_{J-F-} :: [\{<\},\{=\},\{>\},\{<,=\},\\
&\{>,=\},\{<,>\},\{<,=,>\},\{\}],\\
&R_{J+F+} :: [\{<\},\{=\},\{>\},\{<,=\},\\
&\{>,=\},\{<,>\},\{<,=,>\},\{\}],\\
&R_{J+F-} :: :: [\{>\},\{\}],\\
&R_{J-F+} :: :: [\{<\},\{\}],\\
&R_{J-J+} :: :: [\{<\},\{\}],\\
&R_{F-F+} :: :: [\{<\},\{\}],
\end{aligned}
$$

linked by meta-constraints:

$$
\begin{aligned}
&R_{J-F-} =_m R_{J-F+} +_m R_{F+F-},\\
&R_{J-F-} =_m R_{J-J+} +_m R_{J+F-},\\
&R_{J+F+} =_m R_{J+F-} +_m R_{F-F+},\\
&R_{J+F+} =_m R_{J-J+} +_m R_{J-F+}.
\end{aligned}
$$

In this case, the propagation of meta-constraints does not change the meta variable domains.

The quantitative solver transforms the distance constraints in quantitative relations for the CLP(FD) solver. Moreover, it generates all the possible paths of the graph (see section 3.1) and performs arc consistency on them. Therefore, the quantitative level constraints are the following:

```
J-::0..5, F+::50..55,
D3::15..20, D4::0..20,60..max,
D3 = F+ - F-, D4 = J+ - J-, D5 = F- - J-,
D6 = F+ - J+, D7 = F+ - J-, D8 = F- - J+,
D5=D8+D4, D5=D7-D3, D6=D7-D4, D6=D3+D8.
```

The propagation of object level constraints leads to the following domains:

```
J-::0..5, F+::50..55, J+::60..max,
F-::30..40, D3::15..20, D4::60..max,
D5::25..40, D6::-max..-5, D7::45..55,
D8::-max..-20
```

The fourth step concerns the qualitative-quantitative and quantitative-qualitative propagations. The **qual_quan** interaction (see section 3.3) adds the constraints:

```
J+ > F-, J- < F+, J- < J+, F- < F+
```

which do not alter the variable domains because they are already consistent with the constraints.

On the other hand, the propagation **quan_qual** (see section 3.3) alters the meta-variable domains and, in particular, the meta-variables representing the following constraints

```
J- [<,=,>] F-, J+ [<,=,>] F+.
```

In fact, the domain of `J-::0..5` and the domain of `F-::30..40` entail only the constraint `J- < F-` while the domain of `J+::60..max` and the domain of `F+::50..55` entail only the constraint `J+ > F+`. The propagation of constraints reduces the meta-variables domains. This reduction leads to another propagation in the meta-constraint solver but this latest propagation does not alter any other meta variable domain (the meta-constraint graph reaches the *quiescence*).

In the last step the module interval-point translates the relations between starting and ending points in relations of the Interval Algebra. In particular, the only relation entailed by the starting and ending point constraints is `F during J`.

After having determined the minimal network representing the above-mentioned problem, we can be interested in finding a feasible scenario, i.e., an arrangement of the temporal objects along the time line which is consistent with the given constraints, see [27].

In our architecture this problem can be solved in a straightforward way. In fact, the CLP(FD) primitive `indomain(X)`, when applied to a domain variable `X`, instantiates `X` to a feasible value of its domain. The propagation then eliminates, from other variable domains, those values which are incompatible with `X`. At the end of the propagation, either all the variables are instantiated or another **indomain** occurs, until there are no more unbounded variables. In our example, we can instantiate the starting and ending points of the intervals `I` and `J` in order to obtain a feasible scenario. The clause:

```
scenario([]).
scenario([Var|Others]):-
        indomain(Var),
        scenario(Others).
```

instantiates the temporal points of the problem `[Var|Others]` to feasible values.

Usually, according to the *first fail* principle [28], the variable to be instantiated next is the variable with the smallest domain (the most constrained variable). Therefore, in the implementation we used this heuristics. Other scenarios can be found by using the backtracking mechanism of CLP.

The same problem can be handled in a qualitative network. Each meta-variable represents a set of relations between variables linked by an arc. We could be interested in finding a feasible scenario, i.e., an arrangement of the temporal relations between objects which is consistent with the other constraints, see [25]. In this case, the next variable to be instantiated could be the most constrained one, i.e., the variable with the minimum sized domain. Therefore, the above-mentioned clause **scenario** can be used also for qualitative constraints. In the example above, the consistent scenario has been already found because meta-variables have no disjunctive constraint but each meta-variable is instantiated to a single primitive relation. In fact, we have the following qualitative constraints:

```
J+ > F-, J- < F+, J- < J+,
F- < F+, J- < F-, J+ > F+.
```

In qualitative networks, a very important problem is to decide which value to try next. Usually, the ordering of the values according to their restriction is a domain dependent problem. Tsang [25] proposes some orderings of temporal interval-interval relations, e.g., the more restrictive relations are those imposing a greater number of point-point constraints between their starting and ending points or in planning problems the more restrictive relations are those minimizing the overall duration of the schedules generated. In this case, the "tighter" relations between the constraints of the qualitative constraint solver can be useful in order to decide which value to try next.

It is worth mentioning that in applications like least-commitment planning or scheduling the qualitative temporal constraints should be instantiated before quantitative constraints. The choice of the variable values is committed at the end of the computation, thus preventing failures and reducing the search space.

5 Conclusions and Future Works

We have presented a Constraint Logic Programming two level architecture for both quantitative and qualitative temporal reasoning. The architecture is structured into two communicating modules: a finite domain constraint solver for the quantitative reasoning and a meta constraint solver for handling qualitative reasoning. The object level constraint solver has been extended in order to perform a path consistency check not by changing the solver consistency algorithm but by adding "arc-variables" and "path-equivalent" constraints. The meta constraint solver reasons on the constraints of the quantitative module and is able to perform operations on them such as composition and intersection and to state relations between constraints, e.g., the tighter relation. In this paper we have discussed the structure of the two solvers and the relations and interactions between the two.

This architecture has been implemented by using the finite domain library of ECL^{i}PS^e [11], a CLP language developed by ECRC.

Some related works have been studied and discussed in [17]. Here, we list some of the most relevant: As concerns temporal reasoning, there are many related works. First of all Vilain and Kautz' point algebra, Allen Interval Algebra [2], the Meiri's framework [21] and Tsang work on meta graphs [25] are the starting points for our work.

As far as the integration of CLP(FD) and temporal reasoning is concerned, we have very interesting works. T. Frühwirth in [13] describes how to treat qualitative and quantitative temporal reasoning with Constraint Handling Rules. Another related work is presented in [24] where the integration of qualitative and quantitative reasoning is performed by extending a single constraint solver. A third work, described in [19], concerns the integration in a finite domain constraint solver of the Interval Algebra qualitative constraints. All these approaches are based on a flat architecture while our approach is based on a meta architecture thus gaining in modularity and flexibility.

Finally, as regards the path consistency in CLP(FD), a related work is [6] which embeds in a CLP(FD) solver the path consistency algorithm. We do not change the CLP(FD) propagation algorithm, but we simply add new variables and constraints. We are generalizing this idea by using a several level meta architecture for reaching whatever consistency algorithm, see [18].

Future work is devoted to generalize this framework in order be able to apply it to different domains without changing its structure. A possible instance can be for example a CLP(Intervals) as the object-level system [4] and a qualitative solver performing the Interval Algebra reasoning as the meta system. This is a work in progress.

We intend to apply the system to a real life application of scheduling and planning as the one described in [7].

Acknowledgments

This work has been partially supported by the M.U.R.S.T. Project 60% (Ministero dell'Università della Ricerca Scientifica e Tecnologica). We would like to thank Helene Pokidine who helped us in implementing the system and anonymous referees for very useful comments on the first version of this paper.

References

[1] L.Aiello, C.Cecchi, D.Sartini, "Representation and Use of Metaknowledge", *Proceedings of the IEEE*, Vol. 74, No. 10, pp. 1304-1321, 1986.

[2] J.F.Allen, "Maintaining Knowledge About Temporal Intervals", *Communications of the ACM*, Vol. 26, pp. 832-843, 1983.

[3] J.F.Allen, P.J.Hayes, "A Common-Sense Theory of Action and Time", *Proceedings of IJCAI85*, pp. 528-531, 1985.

[4] F.Benhamon, D.McAllester, P.Van Hentenryck, "CLP(Intervals) Revisited", *Tech. Report CS-94-18 Computer Science Department*, Brown University, 1994.

[5] K.A.Bowen, R.A. Kowalski, "Amalgamating Language and Metalanguage in Logic Programming" in *Logic Programming*, K.Clark and S.Tarnlund Eds., Accademic Press NY, pp. 153-173, 1982.

[6] P.Codognet, G.Nardiello, "Path Consistency in clp(FD)", in *Proceedings of the First International Conference Constraint in Computational Logics CCL94*, pp. 201-216, 1994.

[7] A.Dalfiume, E.Lamma, P.Mello, M.Milano, "A Constraint Logic Programming Application to a Distributed Train Scheduling Problem", *Proceedings of the Conference on Practical Application of Prolog*, pp.163–182, 1995.

[8] T.L.Dean, D.W.McDermott, "Temporal Data Base Management", *Artificial Intelligence*, Vol. 32, pp. 1-55, 1987.

[9] R.Dechter, I.Meiri, J.Pearl, "Temporal Constraint Networks", *Artificial Intelligence*, Vol. 49, pp. 61-95, 1991.

[10] M. Dincbas, P. Van Hentenryck, M. Simonis, A. Aggoun, T. Graf, F. Berthier, "The Constraint Logic Programming Language CHIP", *Proceedings of the International Conference on Fifth Generation Computer System (FGCS88)*, pp.693-702,1988.

[11] ECL^{i}PS^e *User Manual Release 3.3*, ECRC, 1992.

[12] E.C. Freuder, "Synthesizing Constraint Expressions", *Communication of the ACM*, vol. 21, No11, 1978, pp. 958–966.

[13] T. Frühwirth, "Temporal reasoning with constraint handling rules", *Tech. Report ECRC-94-05*, ECRC, 1994.

[14] J.Jaffar, J.L.Lassez, "Constraint Logic Programming", *Proceedings of the Conference on Principle of Programming Languages*, 1987.

[15] J.Jaffar, M.J.Maher, "Constraint Logic Programming: a Survey", *Journal of Logic Programming on 10 years of Logic Programming*, Vol.19/20, pp. 503–582, 1994.

[16] H.A.Kautz, P.B.Ladkin, "Integrating Metric and Qualitative Temporal Reasoning", *Proceedings AAAI91*, pp.241-246, 1991.

[17] E.Lamma, P.Mello, M.Milano, "A Meta Constraint Logic Programming Architecture for Qualitative and Quantitative Temporal Reasoning", *Tech.Report DEIS-LIA-95-001*, 1995, `www-lia.unibo.it/Research/TechReport.html`.

[18] E.Lamma, P.Mello, M.Milano, "A Multi-Level CLP Architecture for Consistency Techniques", *Submitted for pubblication.*

[19] J.Lever, B.Richards, R.Hirsh, "Temporal Reasoning and Constraint Solving", *Deliverable CHIC, ESPRIT Project EP5291*, IC-Park, 1992.

[20] J.W.Lloyd, *Foundation of Logic Programming"*, Second Extended Edition, Springer-Verlag, 1987.

[21] I.Meiri, "Combining Qualitative and Quantitative Constraints in Temporal Reasoning", *Proceedings of AAAI91*, pp.260-267, 1991.

[22] A.K. Mackworth, "Consistency in Networks of Relations", *Artificial Intelligence*, Vol.8, pp. 99-118, 1977.

[23] R.Mohr, T.C.Henderson, "Arc and Path Consistency Revisited", *Artificial Intelligence*, Vol.28, pp. 225-233, 1986.

[24] F.A.Barber, R.Berlanga, R.Fordellas, F. Ibañez, G.Martin, F.Toledo, "Integration of Metric and Symbolic Temporal Constraints in CLP", *unpublished manuscript.*

[25] E.P.K.Tsang, "The Consistent Labeling Problem in Temporal Reasoning", *Proceedings of AAAI87*, pp.251-255, 1987.

[26] P.VanBeek, R.Cohen, "Exact and Approximate Reasoning about Temporal Relations", *Computational Intelligence*, Vol.6, pp.132-144, 1990.

[27] P.VanBeek, "Reasoning about Qualitative Temporal Information", in *Artificial Intelligence*, Vol.58, pp. 297-326, 1992.

[28] P.Van Hentenryck, *Constraint Satisfaction in Logic Programming*, MIT Press, 1989.

[29] M.B.Vilain, H.Kautz, "Constraint Propagation Algorithms for Temporal Reasoning" *Proceedings of AAAI89*, pp. 377–382, 1989.