Semi-Automated Task Planning in Metric Propositional Interval Neighborhood Logic

Laura González-García
Polytechnic University of Cartagena
Cartagena, Spain
Email: lgg2@alu.uptc.es

Guido Sciavicco
Dep. of Information Engineering and Communications
University of Murcia, Murcia, Spain
Email: guido@um.es

***Abstract*—Planning is the process of thinking about and organizing the activities required to achieve a desired goal. It involves the creation and maintenance of a plan. As such, planning is a fundamental property of intelligent behaviour. This process is essential to the creation and refinement of a plan, or integration of it with other plans. In logistics planning, which is a fundamental part of every engineering projects, planning is a usually handcrafted activity, often supported by high-level commercially available software. These techniques are error-prone as they relay on the expertise of the responsible engineer, and suffer of limited reasoning capabilities, being usually based on temporal constraint networks in Allen's style. Recently, interval temporal logics have been studied that allow one to describe temporal situations at a higher level, and yet with a decidable satisfiability problem. We propose here the use of Metric Interval Temporal Neighbourhood Logic, a decidable fragment of Halpern and Shoham's Modal Logic for Time Intervals (HS), as a tool for task planning. The main characteristics of this proposal is that the language, whose syntax heavily restricts HS, and whose proposed applications so far have been limited to theoretical and abstract situations, is still expressive enough to cope with the complexity of a realistic case study.**

***Keywords*-Automated Planning; Interval Temporal Logics.**

I. Introduction

Task management is the process of managing tasks through its life cycle. It involves planning, testing, tracking and reporting. Task management can help either individuals achieve goals, or groups of individuals collaborate and share knowledge for the accomplishment of collective goals [1]. Tasks are also differentiated by complexity, from low to high. Effective task management requires managing all aspects of a task, including its status, priority, time, human and financial resources assignments, recurrence, notifications and so on. These can be lumped together broadly into the basic activities of task management. Managing multiple individual or team tasks may require specialised task management software. Specific software dimensions support common task management activities. These dimensions exist across software products and services and fit different task management initiatives in a number of ways. In fact, many people believe that task management should serve as a foundation for project management activities. Task management may form part of project management and process management and can serve as the foundation for efficient work-flow in an organisation. Project managers adhering to task-oriented management have a detailed and up-to-date project schedule, and are usually good at directing team members and moving the project forward.

As a discipline, task management embraces several key activities. Various conceptual breakdowns exist, and these, at a high-level, always include creative, functional, project, performance and service activities. *Creative* activities pertain to task creation. These should allow for task planning, brainstorming, creation, elaboration, clarification, organization, reduction, targeting and preliminary prioritization. *Functional* activities pertain to personnel, sales, quality or other management areas, for the ultimate purpose of ensuring production of final goods and services for delivery to customers. These should allow for planning, reporting, tracking, prioritizing, configuring, delegating, and managing of tasks. *Project* activities pertain to planning and time/costs reporting. These can encompass multiple functional activities but are always greater and more purposeful than the sum of its parts. Project activities should allow for project task breakdown, task allocation, inventory across projects, and concurrent access to task databases. *Service* activities pertain to client and internal company services provision, including customer relationship management and knowledge management. These should allow for file attachment and links to tasks, document management, access rights management, inventory of client and employee records, orders and calls management, and annotating tasks. *Performance* activities pertain to tracking performance and fulfillment of assigned tasks. Finally, *report* activities pertain to the presentation of information regarding the other five activities listed, including graphical display.

Task management software tools abound in the marketplace [2][3]. Some are free; others exist for enterprise-wide deployment purposes. Some boast enterprise-wide task creation, visualization and notifications capabilities - among others - scalable to smaller, medium and bigger size companies, from individual projects to ongoing corporate task management. Project management and calendaring software also often provide task management software with advanced support for task management activities and corresponding software environment dimensions, reciprocating the myriad project and performance activities built into most good enterprise-level task management software products. Nevertheless, most of such software lack truly intelligent capabilities, as they are based on *algebraic* networks in Allen's style [4][5]. The main limits of algebraic, constraint-based reasoning, compared to *logical* reasoning are discussed in [6], and include the fact that algebraic networks: *(i)* are purely existential, and do not allow,

$\langle A\rangle$	$[i,j]R_A[i',j'] \Leftrightarrow j = i'$
$\langle L\rangle$	$[i,j]R_L[i',j'] \Leftrightarrow j < i'$
$\langle B\rangle$	$[i,j]R_B[i',j'] \Leftrightarrow i = i', j < j'$
$\langle E\rangle$	$[i,j]R_E[i',j'] \Leftrightarrow j = j', i < i'$
$\langle D\rangle$	$[i,j]R_D[i',j'] \Leftrightarrow i < i', j' < j$
$\langle O\rangle$	$[i,j]R_O[i',j'] \Leftrightarrow i < i' < j < j'$

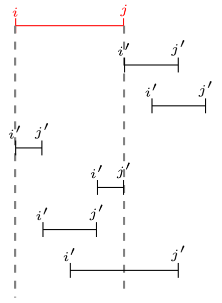

Figure 1: Allen's interval relations and the corresponding HS modalities.

in general, the specification of universal properties; *(ii)* do not allow, in general, the specification of negative information; *(iii)* are not designed to easily integrate and/or compare two or more plans.

The fact that constraint-based networks are usually preferred over logic-based reasoning systems in planning languages and software is explained by the usually bad computational behaviour of interval-based temporal logics. Interval temporal logics provide a natural framework for temporal representation and reasoning on interval structures over linearly ordered domains. They take time intervals as the primitive ontological entities and define truth of formulae with respect to them instead of to time instants. Modal operators of interval temporal logics correspond to binary relations between pairs of intervals In the realm of interval temporal logics, a prominent role is accorded to Halpern and Shoham's modal logic of time intervals (HS) [7], whose modalities make it possible to express all Allen's binary interval relations. Unfortunately, most of them, including HS and the majority of its fragments, turn out to be undecidable (a comprehensive survey on interval logics can be found in [8]; more recent contributions include [9][10]). Focusing our attention to the class of models built on the set of the integers, in [10] it has been shown that there exists exactly 44 expressively different fragments of HS with a decidable satisfiability problem, with complexities from NP-complete to EXPSPACE-complete, and 62 are decidable in the finite case [11]. Among these, a metric extension of the fragment that features $\langle A\rangle$ and $\langle\overline{A}\rangle$ only (known as $\mathsf{A}\overline{\mathsf{A}}$ or PNL) has been developed by Bresolin et al. in [12]. The resulting interval temporal logic, called Metric PNL (MPNL for short), pairs PNL modalities with a family of special proposition letters expressing integer constraints (equalities and inequalities) on the length of the intervals over which they are evaluated. The authors show that the satisfiability problem for MPNL, interpreted over finite linear orders and the natural numbers, is decidable, and, in particular, EXPSPACE-complete.

In this paper, we propose the use of MPNL as a task planning reasoning tool. In the next section, we provide the necessary preliminaries on MPNL and interval temporal logics in general. In Section 3, we consider the problem of representing a plan in MPNL, and in Section 4 we apply our technique to a practical case-study, before concluding.

II. Preliminaries

Let $\mathbb{D} = \langle D, <\rangle$ be a linearly ordered set. An *interval* over $\mathbb{D}$ is an ordered pair $[i,j]$, where $i,j \in D$ and $i < j$ (*strict semantics*). There are 12 different non-trivial ordering relations (excluding equality) between any pair of intervals in a linear order, often called *Allen's relations* [4]: the six relations depicted in Figure **??** and the inverse ones. We interpret interval structures as Kripke structures and Allen's relations as accessibility relations, thus associating a modality $\langle X\rangle$ with each Allen's relation R_X. For each operator $\langle X\rangle$, its *inverse* (or *transpose*), denoted by $\langle\overline{X}\rangle$, corresponds to the inverse relation $R_{\overline{X}}$ of R_X (that is, $R_{\overline{X}} = (R_X)^{-1}$). Halpern and Shoham's logic HS is a multi-modal logic with formulas built on a set $\mathcal{AP}$ of proposition letters, the boolean connectives $\vee$ and $\neg$, and a modality for each Allen's relation. We denote by $\mathsf{X}_1 \ldots \mathsf{X}_k$ the fragment of HS featuring a modality for each Allen's relation in the subset $\{R_{X_1}, \ldots, R_{X_k}\}$. Formulas of $\mathsf{X}_1 \ldots \mathsf{X}_k$ are defined by the grammar:

$$\varphi ::= p \mid \neg\psi \mid \psi \vee \tau \mid \langle X_1\rangle\psi \mid \ldots \mid \langle X_k\rangle\psi,$$

where $p \in \mathcal{AP}$ is a propositional letter. The other boolean connectives can be viewed as abbreviations, and the dual operators $[X]$ are defined, as usual, as $[X]\varphi \equiv \neg\langle X\rangle\neg p$. The semantics of HS is given in terms of *interval models* $\mathcal{M} = \langle \mathbb{I}(\mathbb{D}), \mathcal{V}\rangle$, where $\mathbb{I}(\mathbb{D})$ is the set of all intervals over $\mathbb{D}$ and $\mathcal{V} : \mathcal{AP} \mapsto 2^{\mathbb{I}(\mathbb{D})}$ is a *valuation function* that assigns to every $p \in \mathcal{AP}$ the set of intervals $\mathcal{V}(p)$ over which p holds. The *truth* of a formula over a given interval $[i,j]$ in an interval model $\mathcal{M}$ is defined by structural induction on formulas:

$$\begin{array}{lll}
\mathcal{M},[i,j] \Vdash p & \text{iff} & [i,j] \in \mathcal{V}(p) \\
\mathcal{M},[i,j] \Vdash \neg\psi & \text{iff} & \mathcal{M},[i,j] \not\Vdash \psi \\
\mathcal{M},[i,j] \Vdash \psi \wedge \tau & \text{iff} & \mathcal{M},[i,j] \Vdash \psi \text{ and } \mathcal{M},[i,j] \Vdash \varphi \\
\mathcal{M},[i,j] \Vdash \langle X_k\rangle\psi & \text{iff} & \mathcal{M},[i',j'] \Vdash \psi \\
 & & \quad\text{for some } [i,j]R_{X_k}[i',j'].
\end{array}$$

Formulae of HS can be interpreted over a class of interval models (built on a given class of linear orders). Among others, we mention the following important classes of (interval models built on important classes of) linear orders: *(i)* the class of *all* linear orders; *(ii)* the class of (all) *dense* linear orders, that is, those in which for every pair of distinct points there exists at least one point in between them; *(iii)* the class of (all) *discrete* linear orders, that is, those in which every element, apart from the greatest element, if it exists, has an immediate successor, and every element, other than the least element, if it exists, has an immediate predecessor *(iv)* the class of (all) *finite* linear orders, that is, those having only finitely many points. In the recent years, a great effort has been devoted to the study of decidability of fragments of HS. Ever since HS was introduced, it was immediately clear that its satisfiability problem is undecidable when interpreted on every interesting class of linearly ordered sets [7], including all of the above mentioned ones. While this sweeping result initially discouraged further research in this direction, recent results showed that the situation is slightly better then it seemed. Given the set of HS modalities that correspond to the set of Allen's relations $\{R_{X_1}, \ldots, R_{X_k}\}$, we call *fragment* $\mathcal{F} = \mathsf{X}_1\mathsf{X}_2 \ldots \mathsf{X}_n$ any subset of such modalities, displayed in alphabetical order. There are 2^{12} such fragments. Some of these are expressively equivalent to each other; in [9] (respectively, [13]) it is possible to find all possible inter-definability in the class of all linearly ordered sets (respectively, all dense linearly ordered sets), giving rise to 1347 (respectively, 966) expressively different fragments. The number of different fragments on other classes of linear orders has not been determined yet, but it is believed that the situation in the finite or discrete case should be similar. Out of these fragments, it has been possible to prove that exactly 62 are decidable in the finite case [11], and 44 in the (strongly) discrete case (and in the case of $\mathbb{Z}$) [10], all of which with complexities that range from NP-complete (in very simple cases) to NEXPTIME-complete, EXPSPACE-complete, to non-primitive recursive.

Motivated by the (potential) applicability of these logics, in the discrete and the finite case the possibility of adding *length constraints* has been studied. Following [14] we introduce a set of pre-interpreted atomic propositions referring to the length of the current interval. Given a *distance* function $\delta : \mathbb{Z} \times \mathbb{Z} \to \mathbb{N}$, defined as $\delta(i, j) = |i - j|$, for each $\sim \in \{<, \leq, =, \geq, >\}$, we introduce the length constraint $\mathsf{len}_{\sim\mathsf{k}}$, with the following semantics:

$$\mathcal{M}, [i, j] \Vdash \mathsf{len}_{\sim\mathsf{k}} \text{ iff } \delta(i, j) \sim k.$$

As studied in [14][12], the language of A$\overline{\text{A}}$ can be extended with length constraints when interpreted over $\mathbb{N}$, $\mathbb{Z}$, or finite models without loosing the decidability of the fragment itself; its complexity, though, worsen from NEXPTIME to EXPSPACE. Equality and inequality constraints are mutually definable, although there is a increase in formula length if we consider, for example, only constraints of form $\mathsf{len}_{=\mathsf{k}}$ as primitive. Length constraints can be expressed in HS in a direct way. The simplest way to achieve this is to make use of $\langle B \rangle$ or $\langle E \rangle$. For example, under the discreteness hypothesis, we have that:

$$\mathcal{M}, [i, j] \Vdash \mathsf{len}_{=\mathsf{k}} \text{ iff } \langle B \rangle^{k-1} \top \wedge [B]^k \bot,$$

which proves that A$\overline{\text{A}}$ plus metric constraints (known as MPNL) is a proper fragment of HS.

Finding an optimal balance between expressive power and computational complexity is a challenge for every knowledge representation and reasoning formalism. Interval temporal logics are not an exception in this respect; in [14] the applicability of MPNL has been advocated. To recall some of the arguments, MPNL has been proved expressive enough to to encode (*metric versions* of) basic operators of point-based linear temporal logic (LTL) as well as interval modalities corresponding to Allen's relations. In addition, it allows one to express limited forms of fuzziness. Limiting ourselves to a few examples, we show that MPNL is expressive enough to encode the strict *sometimes in the future* (respectively, *sometimes in the past*) operator of LTL:

$$\langle A \rangle (\mathsf{len}_{>0} \wedge \langle A \rangle (\mathsf{len}_{=0} \wedge p))$$

Moreover, length constraints allow one to define a metric version of the *until* (respectively, *since*) operator. For instance, the condition: '*p is true at a point in the future at distance k from the current interval and, until that point, q is true (pointwise)*' can be expressed as follows:

$$\langle A \rangle (\mathsf{len}_{=\mathsf{k}} \wedge \langle A \rangle (\mathsf{len}_{=0} \wedge p)) \wedge [A](\mathsf{len}_{<\mathsf{k}} \to \langle A \rangle (\mathsf{len}_{=0} \wedge q)).$$

MPNL can also be used to constrain interval length and to express metric versions of basic interval relations. First, we can constrain the length of the intervals over which a given property holds to be at least (respectively, at most, exactly) k. As an example, the following formula constrains p to hold only over intervals of length l, with $k \leq l \leq k'$:

$$[G](p \to \mathsf{len}_{\geq \mathsf{k}} \wedge \mathsf{len}_{\leq \mathsf{k'}}) \quad (bl)$$

where the *universal modality* $[G]$ (*for all intervals*) is expressible in the language, and its corresponding formula depends on the class of models over which the formula is interpreted. By exploiting such a capability, metric versions of almost all Allen's relations can be expressed (the only exception is the *during* relation). As an example, we can state that: '*p holds only over intervals of length l, with $k \leq l \leq k'$, and any p-interval begins a q-interval*' as follows:

$$(bl) \wedge [G] \bigwedge_{i=k}^{k'} (p \wedge \mathsf{len}_{=\mathsf{i}} \to \langle \overline{A} \rangle \langle A \rangle (\mathsf{len}_{>\mathsf{i}} \wedge q)).$$

Finally, MPNL makes it possible to express some forms of 'fuzziness'. As an example, the condition: '*p is true over the current interval and q is true over some interval close to it*', where by 'close' we mean that the right endpoint of the p-interval is at distance at most k from the left endpoint of the q-interval, can be expressed as follows:

$$p \wedge (\langle A \rangle \langle \overline{A} \rangle (\mathsf{len}_{<\mathsf{k}} \wedge \langle \overline{A} \rangle \langle A \rangle q) \vee \langle A \rangle (\mathsf{len}_{<\mathsf{k}} \wedge \langle A \rangle q)).$$

III. Task Planning in MPNL

It is generally accepted that *task planning* in Engineering is a fundamental phase of the design, organization, and control of any realistic work organization plan. In its simplest version, it includes, at least:

1) A list of each *atomic* task, along with its properties (including its temporal duration);
2) A set of *precedence* relations among tasks.

The purpose of a systematic organization of such set of task is to answer the following question: *Is the plan possible, and, if so, what is its minimal temporal duration?* In view of these considerations, we may define a plan as follows.

Definition 1: A *plan* is a finite collections of *tasks*, each one of which with a finite and univocally determined *duration*, and such that they are placed over a finite temporal line respecting a finite collection of *precedence requirement*.

The typical practical approach to the problem of finding a plan from the collection of its requisite is twofold. On the one side, engineers are trained to organize tasks in a systematic network of precedence (for example, with the so-called *critical path method* [15]), and to compute (by hand) the viability of the entire network. On the other side, commercially available software, such as, for example, Microsoft Project© are used to aid this process. Now, it is easy to observe that:

1) Atomic tasks can be logically treated as propositional letters;
2) The precedence relation can be modeled as Allen's relation *meets*;

3) On a dicrete/finite temporal line, durations are exactly *length constraints*.

These considerations allow us to conclude that MPNL is a suitable logical counterpart of a task planning network. Plans are, by definition, temporally finite, and by seeing the plan the logical conjunction of the (formulas corresponding to) set of all constraints, *plan viability* corresponds to (MPNL-) *formula satisfiability*. Moreover, the notion of *minimal duration* is precisely the notion of *minimal model* of a formula.

From now on, we consider the language of MPNL interpreted in the class of *finite* (and, therefore, discrete) models. Satisfiability of MPNL-formulas in the finite case can be safely restricted to the *initial interval*, which we can denote as $[-1, 0]$. In fact, given any MPNL-formula φ, the latter is satisfiable if and only if the MPNL-formula $\langle A\rangle\varphi$ is initially satisfiable. By applying such a small technical modification, we obtain a task, represented as a propositional letter, will be placed on the interval $[i, j]$ exactly when the plan sets it to start at the moment i and to finish at the moment j. In this context, the *universal modalitity*, introduced in the previous section, can be expressed as follows:

$$[G]p \equiv [A]p \wedge [A][A]p.$$

Let us assume, now, that tasks are represented by the propositional letters $T_1, T_2, \ldots \in \mathcal{T}$, where $\mathcal{T}$ is a finite set of tasks. Similarly, it is convenient to assume that task indexes are collected in a subset of natural numbers $\mathcal{I}$. Therefore, by expressing:

$$\langle A\rangle T_l \vee \langle A\rangle\langle A\rangle T_l,$$

where $l \in \mathcal{I}$, we force the task T_1 to be part of a plan. Similarly, by adding:

$$[G](T_l \rightarrow \mathsf{len}_{=\mathsf{k}}),$$

we force T_l to have the duration of k units. The *precedence relation* can be then expressed as follows. Given a constraint of the type: *the task T_l cannot start before the task T_g has finished*, we can set:

$$[G](T_l \rightarrow \langle\overline{A}\rangle(T_g \vee \langle\overline{A}\rangle T_g).$$

To make sure that we can exclude unwanted models, we can add the following constraint that ensures tasks are unique:

$$\bigwedge_{l\in\mathcal{I}}[G]\langle A\rangle(T_l \rightarrow [A]\neg T_l).$$

Unlike algebraic networks, MPNL allows one to express more complex requirements. First of all, besides single tasks $T_1, T_2, \ldots$, we can express the concept of *task type*, by adding suitable propositions in conjunction with those that denote tasks, and, then, impose *universal* constraints over them. Suppose, for example, that $\mathcal{T}' \subset \mathcal{T}$ collects all and only those tasks of a certain type, for which we have the constraint that: *between any two successive tasks of $\mathcal{T}'$ a temporal distance of at least k units must get by*. We can deal with such a constraint by means of the following technique:

$$\begin{cases} \bigwedge_{T_l\in\mathcal{T}'}[G](T_l \rightarrow P)\wedge \\ [G](P \rightarrow \bigvee_{T_l\in\mathcal{T}'})\wedge \\ [G](P \wedge \langle A\rangle\langle A\rangle P \rightarrow \langle A\rangle(\mathsf{len}_{>\mathsf{k-1}}P)), \end{cases}$$

where by means of the first formula, we make sure that elements of $\mathcal{T}'$ are labeled by an additional proposition P, by the second one we guarantee that P labels only elements of $\mathcal{T}'$, and, finally, by last one we introduce the temporal constraint.

Other types of constraints can be expressed, such as: *the tasks T_l and T_g cannot start at the same time*:

$$[G]((\langle A\rangle T_l \wedge \langle A\rangle T_g) \rightarrow \bot).$$

Finally, it is worth noticing that more complex constraints can be expressed in MPNL. In fact, we can easily identify the *maximal temporal duration $\overline{k}$* of any task in $\mathcal{T}$; by using this information, as we have explained in Section 2, almost every Allen's relation can be expressed over bounded intervals (all tasks are bounded), and they can be used in both existential and universal statements.

In more advanced task planning systems, one would like to be able to take into account a certain amount of *finite resources*. Indeed, in real cases, not every temporally sound plan is executable, if it requires, at any given moment of time, more resources than those that are at disposition. It is not difficult to see that this information can be expressed by using only a propositional language, such as MPNL. Let us assume that we measure our resources with a natural number n. The requirements may indicate the resources that are consumed by each task, and the total number of resources that are at the disposition for the entire plan. We assume, for each task T_l, that the propositional letter R_l^n denotes the fact that n resource units are necessary; clearly, if N is the maximum number of units that are necessary for any task, at most $|\mathcal{T}| \cdot N$ different propositional letters must be added to the language. We then have to assign the correct number of units to each task, and, since we can only compare intervals, in order to take into account *overlapping* tasks (and therefore, the combined amount of resources required at any given moment), we collect such information at the finest temporal granularity, that is, unit intervals:

$$\begin{cases} \bigwedge_{T_l\in\mathcal{T}}[G](T_l \rightarrow R_l^n)\wedge & \\ \bigwedge_{s=1,\ldots,M}\bigwedge_{l\in\mathcal{I}}\bigwedge_{k\leq\overline{k}}[G] & ((\langle A\rangle(\mathsf{len}_{=\mathsf{k}} \wedge T_l \wedge R_l^s) \rightarrow \\ & \bigwedge_{k'<k}\langle A\rangle\langle A\rangle(\mathsf{len}_{=1} \wedge R_l^s)). \end{cases}$$

Now, we can easily pre-compute all and only those sequences of indexes $l_1, l_2, \ldots$ in $\mathcal{I}$, such that, for each such sequence σ, there are tasks in $\mathcal{T}$ such that, by summing all resources requested by each of them, we obtain a number greater than the maximum numbers of units available, which we can denote by M. Let Σ be the set of such sequences of indexes. If $\sigma = l_1, l_2, \ldots, l_{|\sigma|} \in \Sigma$, then there exists tasks in $\mathcal{T}$ for which we

Table I: A FRAGMENT OF A REALISTIC CASE STUDY. UPPER SIDE: TASKS. LOWER SIDE: GENERAL REQUIREMENTS.

Symbol	Name	Type	Duration	Preceding Task(s)	Formulas
T_1	Water well building	–	5	–	$\langle A\rangle T_1 \vee \langle A\rangle\langle A\rangle T_1$
T_2	Water convey construction work	P_1	12	T_1	$\langle A\rangle T_2 \vee \langle A\rangle\langle A\rangle T_2$, $[G](T_2 \rightarrow \mathsf{len}_{=12})$ $[G](T_2 \rightarrow \langle\overline{A}\rangle(T_1 \vee \langle\overline{A}\rangle T_1)$
T_3	Water convey construction work	P_1	14	T_2	$\langle A\rangle T_3 \vee \langle A\rangle\langle A\rangle T_3$, $[G](T_3 \rightarrow \mathsf{len}_{=14})$ $[G](T_3 \rightarrow \langle\overline{A}\rangle(T_2 \vee \langle\overline{A}\rangle T_2)$
T_4	Pumphouse construction	–	15	–	$\langle A\rangle T_4 \vee \langle A\rangle\langle A\rangle T_4$, $[G](T_4 \rightarrow \mathsf{len}_{=15})$
T_5	Pump acquisition,installation and electric connections	–	30	T_1	$\langle A\rangle T_5 \vee \langle A\rangle\langle A\rangle T_5$, $[G](T_5 \rightarrow \mathsf{len}_{=30})$ $[G](T_5 \rightarrow \langle\overline{A}\rangle(T_1 \vee \langle\overline{A}\rangle T_1)$
T_6	Ground filling-up and cleaning	–	4	T_3	$\langle A\rangle T_6 \vee \langle A\rangle\langle A\rangle T_6$, $[G](T_6 \rightarrow \mathsf{len}_{=4})$ $[G](T_6 \rightarrow \langle\overline{A}\rangle(T_3 \vee \langle\overline{A}\rangle T_3)$

General Requirement or Generic Data	Formula or Symbol
Maximal duration of a task	30
Tasks are unique	$\bigwedge_{l=1,\ldots,6}[G]\langle A\rangle(T_l \rightarrow [A]\neg T_l)$
*Between any two convey construction works, a minimum of 5 time units must be given for checking	$[G]((T_2 \vee T_3) \leftrightarrow P_1) \wedge [G](P_1 \wedge \langle A\rangle\langle A\rangle P_1 \rightarrow \langle A\rangle(\mathsf{len}_{>4} P_1))$

have used the propositional letters, corresponding to the need of resources, $R_{l_1}^{f(l_1)}, R_{l_2}^{f(l_2)}, \ldots$, and the previous constraints have placed them somewhere in the model. We need to make sure that no such tuple of propositional letter is true at any given unit interval:

$$\bigwedge_{\sigma\in\Sigma}\ \bigwedge_{l_1,\ldots,l_{|\sigma|}\in\sigma} [G]((\mathsf{len}_{=1} \wedge R_{l_1}^{f(l_1)} \wedge R_{l_2}^{f(l_2)} \wedge \ldots \wedge R_{l_{|\sigma|}}^{f(l_{|\sigma|})}) \rightarrow \bot).$$

Concluding, this section shows the applicability of a logical framework as an effective support tool for plan design and test. The advantages of using a logical tool in substitution of an algebraic one are well-known, and include, among others, *(i)* the possibility of specifying universal properties; *(ii)* the possibility of specifying negative information; *(iii)* the possibility of comparing, under various points of view, two or more plans. Moreover, it is worth recalling that algebraic networks feature only *limited* disjunction capabilities; as for example, the only way to encode a requirement such as *the task T_l precedes the task T_m or the task T_g* in an algebraic network is to compute two entirely separated networks, while such a requirement has an immediate logical counterpart:

$$[G]((T_m \vee T_g) \rightarrow \langle\overline{A}\rangle(T_l \vee \langle\overline{A}\rangle T_l).$$

IV. A REALISTIC CASE STUDY

We present in this section a fragment of a realistic case study that includes a planning phase. The project under analysis is the construction of a drinkable water provision system for various towns. It includes the building of two water wells, one pumphouse, an impulsion system, and one energy transformation unit. A realistic case study features various tens of different requirements, all of which fall into some of the categories explained in the previous section, and classified into different (conceptual) groups.

In order to show the applicability of MPNL as planning support system, we give in Table I an extract from the collection of task requirements and conditions for this project, and we translate it into MPNL, adding the general conditions as explained in the previous section. To the original plan, we added a further condition in order to show the capabilities of this approach. We suppose that, at some point the chief engineer requires a minimum time to check the construction work before continuing (*): instead of re-thinking the entire plan, it is enough to add the corresponding formula and re-run the satisfiability checker.

V. CONCLUSIONS

The purpose of this paper was twofold. On the one side, we present a novel technique to solve a well-known problem, that is, plan design and checking in Engineering. This problem is usually solved by means of simple algebraic methods, well-established in the field, but that suffer of intrinsic limitations. On the other side, we give a clear and simple application of a recently studied temporal logic for time intervals, that is, MPNL. Algebraic networks are historically preferred over logical formalism for planning applications; this is due to many reasons, among which we mention computational properties of the formalisms and simplicity of the approach. The recent discover of decidable, pure temporal logic for time intervals may change this perspective, allowing one to use more power formalisms without giving up the decidability and therefore the possibility of computer-assisted design. Moreover, while it is true that algebraic networks present, usually, a satisfiability problem with (non-deterministic) polynomial complexity, the plan designing and checking phase needs not to be real-time (it is usually a off-line procedure), and one can afford longer computation times.

ACKNOWLEDGMENTS

The authors acknowledge the support from the Spanish fellowship program *'Ramon y Cajal' RYC-2011-07821* (G. Sciavicco).

REFERENCES

[1] W. Bibel, "Let's plan it deductively," in *Proc. of the 15th Int. Joint Conference on Artificial Intelligence (IJCAI)*, 1997, pp. 1549–1562.

[2] U. Riss, A. Rickayzen, H. Maus, and W. van der Aalst, "Challenges for business process and task management," *Journal of Universal Knowledge Management*, vol. 0, no. 2, pp. 77–100, 2002.

[3] IBM. (2014, Jan.) Life cycle of human tasks. [Online]. Available: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.bpc.610.doc/doc/bpc/ctasklifecycle.html

[4] J. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[5] J. F. Allen and P. J. Hayes, "A common-sense theory of time," in *Proc. of the 9th International Joint Conference on Artificial Intelligence (IJCAI-85)*, Los Angeles, CA, USA, 1985, pp. 528–531.

[6] G. Sciavicco, "Reasoning with time intervals: A logical and computational perspective," *ISRN Artificial Intelligence*, vol. 2012, 2002, available online. Article ID 616087.

[7] J. Halpern and Y. Shoham, "A propositional modal logic of time intervals," *J. of the ACM*, vol. 38, no. 4, pp. 935–962, 1991.

[8] V. Goranko, A. Montanari, and G. Sciavicco, "A road map of interval temporal logics and duration calculi," *J. of Applied Non-Classical Logics*, vol. 14, no. 1–2, pp. 9–54, 2004.

[9] D. Della Monica, V. Goranko, A. Montanari, and G. Sciavicco, "Expressiveness of the interval logics of allen's relations on the class of all linear orders: Complete classification," in *Proceedings of the 22th International Joint Conference Artificial Intelligence (IJCAI)*, 2011, pp. 845–850.

[10] D. Bresolin, D. Della Monica, A. Montanari, P. Sala, and G. Sciavicco, "Interval temporal logics over strongly discrete linear orders: the complete picture," in *Proc. of the 3rd International Symposium on Games, Automata, Logics and Formal Verication (GANDALF)*, 2012, pp. 155–168.

[11] D. Bresolin, D. Della Monica, A. Montanari, P. Sala, and G. Sciavicco, "Interval temporal logics over finite linear orders: the complete picture," in *Proc. of the 20th ECAI*, 2012, pp. 199–204.

[12] D. Bresolin, A. Montanari, P. Sala, and G. Sciavicco, "Optimal decision procedures for mpnl over finite structures, the natural numbers, and the integers," *Theoretical Computer Science*, no. 493, pp. 98–115, 2013.

[13] A. I. A. M. L. Aceto, D. Della Monica and G. Sciavicco, "Complete classification of the expressiveness of fragments of halpern-shoham logic over dense linear orders," in *20th International Symposium on Temporal Representation and Reasoning (TIME)*, 2013, pp. 65–72.

[14] D. Bresolin, D. D. Monica, V. Goranko, A. Montanari, and G. Sciavicco, "Metric propositional neighborhood logics on natural numbers," *Software and System Modeling*, vol. 12, no. 2, pp. 245–264, 2013.

[15] S. Shaheen, *Practical Project Management*. Wiley, 1986.